Modular and Parameter-efficient Fine-tuning of Language Models

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades Dr.-Ing.

vorgelegt von
Jonas Korbinian Julian Pfeiffer
geboren in München, Deutschland

Tag der Einreichung: 21.Februar 2023
Tag der Disputation: 21. April 2023

Referenten: Prof. Dr. Iryna Gurevych, Darmstadt, Germany
Dr. Ivan Vulić, Cambridge, United Kingdom
Prof. Dr. Goran Glavaš, Würzburg, Germany

Darmstadt 2023
D17

Pfeiffer, Jonas: Modular and Parameter-efficient Fine-tuning of Language Models
Darmstadt, Technische Universität Darmstadt
Year thesis published in TUprints: 2023
Day of the viva voce: 21. April 2023
Please cite this document as
URN: urn:nbn:de:tuda-tuprints-245651
URL: https://tuprints.ulb.tu-darmstadt.de/id/eprint/24565

Ehrenwörtliche Erklärung[1]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades "Dr.-Ing." mit dem Titel "Modular and Parameter-efficient Fine-tuning of Language Models" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Zürich, Schweiz, den 6. Oktober 2023

Jonas Korbinian Julian Pfeiffer

[1] Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Wissenschaftlicher Werdegang des Verfassers[2]

10/12 – 02/15 Bachelor of Science (B.Sc.) in Wirtschaftsinformatik, Universität Augsburg.

09/15 – 02/18 Master of Science (M.Sc.) in Wirtschaftsinformatik, Universität Mannheim.

01/19 – 01/22 Doktorand, Ubiquitous Knowledge Processing (UKP-Lab), Technische Universität Darmstadt.

[2] Gemäß §8 Abs. 1 lit. a der Promotionsordnung der TU Darmstadt

Oscar et al.

Abstract

Transfer learning has recently become the dominant paradigm of natural language processing. Models pre-trained on unlabeled data can be fine-tuned for downstream tasks based on only a handful of examples. A long-term goal is to develop models that acquire new information at scale without incurring negative transfer and that generalize systematically to new settings. Modular deep learning has emerged as a promising solution to these challenges, by updating parameter-efficient units of computation locally and asynchronously. These units are often implemented as modules that are interlaid between layers, interpolated with pre-trained parameters, or concatenated to the inputs. Conditioned on tasks or examples, information is routed to multiple modules through a fixed or learned function, followed by an aggregation of their outputs. This property enables compositional generalization, by disentangling knowledge and recombining it in new ways.

In this thesis, we provide a unified view of modularity in natural language processing, spanning across four dimensions; specifically, we disentangle modularity into *computation functions*, *routing functions*, *aggregation functions*, and the *training setting*. Along those axes, we propose multiple contributions: a research framework which encompasses all dimensions; a novel attention-based aggregation function which combines the knowledge stored within different modules; routing mechanisms for out of distribution generalization in cross-lingual transfer scenarios; a dataset and modular training strategies for multimodal and multilingual transfer learning; a modular pre-training strategy to tackle catastrophic interference of heterogeneous data.

Acknowledgments

Obtaining my Ph.D. has profoundly impacted my life, and I wish to express my heartfelt gratitude to all those who played a significant role in making this journey incredibly memorable.

Firstly, I would like to thank Iryna Gurevych for giving me the opportunity to pursue a Ph.D. and for supporting me throughout this journey. I would like to thank Aishwarya Kamath, without whom I would never have begun this journey in the first place. Thank you for your endless encouragements and help in (re-)writing those first papers. I would like to thank Samuel Broscheit, for taking so much time during my Master Thesis, you truly laid all the groundwork for me! I would like to thank Goran Glavaš for encouraging me to pursue this path, connecting me, and for pulling me up when I was down. I would like to thank Kyunghyun Cho, for continuously challenging me and for giving me the possibility to visit the New York University for 6 Months. Finally, I would like to thank Ivan Vulić and Sebastian Ruder for your precious guidance and inputs, generous time, and for being there for me along the entire way.

I'm very thankful to all my colleagues from the UKP Lab for the constructive and helpful feedback that I received during various talks and discussions. In particular, I would like to thank my fellow Ph.D. students and PostDocs (random order), Andreas Rücklé, Jan-Martin Steitz, Leonardo Ribeiro, Nils Reimers, Chen Liu, Sukannya Purkayastha, Indraneil Paul, Ilia Kuznetsov, Michael Bugert, Edwin Simpson, Gözde Gül Şahin, Benjamin Schiller, Nafise Sadat Moosavi, Max Glockner, Steffen Eger, Jan-Christoph Klie, Christopher Klamm, Christian Meyer, Prasetya Ajie Utama, Tilman Beck, Ji-Ung Lee, Yevgeniy Puzikov, and Tim Baumgärtner for the deep and insightful conversations during our formal and informal meetings. I especially want to thank the students which I have had the privilege to be working with: Clifton Poth, Phillip Rust, Hannah Sterz, and Gregor Geigle, thank you so much for the inspiring collaborations over the past years, I owe so much to you. I also thank Sue Messenger for all the support, for the great conversations, and for helping with the bureaucracies of daily life. My sincere appreciation goes to the LOEWE initiative (Hesse, Germany) for kindly financing my Ph.D. study within the emergenCITY center.

I would like to express my appreciation to Mikel Artetxe for hosting me for an internship at FAIR - Meta AI in the summer of 2021. Thanks to Mikel Artetxe, Sebastian Riedel, Fabio Petroni, Patrick Lewis, Naman Goyal, Xi Victoria Lin, Xian Li, and James Cross for your inspirational and supportive mentorship and for making this a wonderful internship.

I would like to thank my parents, Claudia Seidl and Jochen Pfeiffer, and sister Isabel Pfeiffer for your love and support. My achievement is undeniably due to you always being by my side.

Finally, and above all, I want to express my deepest gratitude to Clara Muscholl, my life partner and closest friend. I'm immensely thankful for your unwavering

support through hectic weekends, extensive trips, and all-nighters leading up to deadlines. Your enduring love, patience, and constant presence mean the world to me. Thank you for tolerating my absent-mindedness and consistently standing by my side.

Sincerely,
Jonas Pfeiffer

Contents

Part I

Synopsis

Publications and My Contributions

This thesis is based on six scientific publications that I co-authored together with many excellent researchers as well as outstanding students: Iryna Gurevych (Technical University of Darmstadt), Ivan Vulić (University of Cambridge), Sebastian Ruder (Google Research), Aishwarya Kamath (New York University), Kyunghyun Cho (New York University), Clifton Poth (Technical University of Darmstadt / University of Massachusetts - Amherst), Andreas Rücklé (Amazon), Mikel Artetxe (Meta AI), Sebastian Riedel (DeepMind), Naman Goyal (Meta AI), Xi Victoria Lin (Meta AI), Xian Li (Meta AI), James Cross (Meta AI), Gregor Geigle (Universität Würzburg), Jan-Martin Steitz (Technical University of Darmstadt), Stefan Roth (Technical University of Darmstadt). I thank all co-authors for their significant contributions to these pleasant and successful collaborations. In the following, I detail my own contributions to each publication.

Chapter 6 corresponds to the following publication:

> **Jonas Pfeiffer**, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 26-54, Online. Association for Computational Linguistics.

I conceived the original research contributions and performed all initial implementations. Clifton Poth subsequently extended the code and website to be released as a framework. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed the experiments and the paper content with my co-authors, who assisted me in improving the draft.

Chapter 7 corresponds to the following publication:

> **Jonas Pfeiffer**, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487-503, Online. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations. I proposed the AdapterFusion mechanism and conducted analyses on different datasets. I wrote the first draft of the paper and performed the majority of the revisions. I discussed this work regularly with my co-authors, who helped me improve it.

Chapter 8 corresponds to the following publication:

> **Jonas Pfeiffer**, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654-7673, Online. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations. I proposed the stacking of adapters mechanism and conducted analyses on different datasets. I wrote the first draft of the paper and performed the majority of the revisions. I discussed this work regularly with my co-authors, who helped me improve it.

Chapter 9 corresponds to the following publication:

> **Jonas Pfeiffer**, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021c. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186-10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations. I proposed the mechanisms to extend language models to new scripts and conducted analyses on different datasets. I wrote the first draft of the paper and performed the majority of the revisions. I discussed this work regularly with my co-authors, who helped me improve it.

Chapter 10 corresponds to the following publication:

> **Jonas Pfeiffer**, Gregor Geigle, Aishwarya Kamath, Jan-Martin Steitz, Stefan Roth, and Iryna Gurevych. 2022a. xGQA: Cross-Lingual Visual Question Answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2497-2511, Dublin, Ireland. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations with multilingual models; Gregor Geigle did the same for the multimodal models. I collected the xGQA datasets and conducted analyses on it. I wrote the first draft of the paper and performed the majority of the revisions. I discussed this

work regularly with my co-authors, who helped me improve it.

Chapter 11 corresponds to the following publication:

> **Jonas Pfeiffer**, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022b. Lifting the Curse of Multilinguality by Pre-training Modular Transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479-3495, Seattle, United States. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations. I wrote the initial paper draft and executed the majority of the revisions. I discussed this work regularly with my co-authors, who helped me improve it.

During my PhD, I was fortunate to work with great researchers on many topics, some of which did not fit into this thesis. In the interest of completeness, I provide references to these papers:

> **Jonas Pfeiffer**, Aishwarya Kamath, Iryna Gurevych, and Sebastian Ruder. 2019a. What do Deep Networks Like to Read? *arXiv preprint*.

> **Jonas Pfeiffer**, Christian M. Meyer, Claudia Schulz, Jan Kiesewetter, Jan M. Zottmann, Michael Sailer, Elisabeth Bauer, Frank Fischer, Martin R. Fischer, and Iryna Gurevych. 2019b. FAMULUS: Interactive Annotation and Feedback Generation for Teaching Diagnostic Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 73–78, Hong Kong, China. Association for Computational Linguistics.

> **Jonas Pfeiffer**, Edwin Simpson, and Iryna Gurevych. 2019c. Low Resource Multi-Task Sequence Tagging - Revisiting Dynamic Conditional Random Fields. *arXiv preprint*.

> Aishwarya Kamath, **Jonas Pfeiffer**, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019 Specializing Distributional Vectors of All Words for Lexical Entailment. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP 2019*, pages 72–83, Florence, Italy. Association for Computational Linguistics.

> Tariq Alhindi, **Jonas Pfeiffer**, and Smaranda Muresan. 2019. Fine-Tuned Neural Models for Propaganda Detection at the Sentence and Fragment levels. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 98–102, Hong Kong, China. Association for Computational Linguistics.

Andreas Rücklé, **Jonas Pfeiffer**, and Iryna Gurevych. 2020. MultiCQA: Zero-Shot Transfer of Self-Supervised Text Matching Models on a Massive Scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 2471–2486, Online. Association for Computational Linguistics.

Edwin Simpson, **Jonas Pfeiffer**, and Iryna Gurevych. 2020. Low Resource Sequence Tagging with Weak Labels. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pages 8862–8869, New York, NY, USA.

Alan Ansell, Edoardo Maria Ponti, **Jonas Pfeiffer**, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021a. MAD-G: Multilingual Adapter Generation for Efficient Cross-Lingual Transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781 Online. Association for Computational Linguistics.

Clifton Poth, **Jonas Pfeiffer**, Andreas Rücklé, and Iryna Gurevych. 2021. What to Pre-Train on? Efficient Intermediate Task Selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 10585–10605, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, **Jonas Pfeiffer**, Yue Zhang and Iryna Gurevych. 2021. Smelting Gold and Silver for Improved Multilingual AMR-to-Text Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP)*, pages 742–750, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, **Jonas Pfeiffer**, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Phillip Rust, **Jonas Pfeiffer**, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, ACL 2021*, pages 3118–3135, Online. Association for Computational Linguistics.

Gregor Geigle, Jonas Stadtmüller, Wei Zhao, **Jonas Pfeiffer**, and Steffen Eger. 2021. TUDa at WMT21: Sentence-Level Direct Assessment with Adapters. In *Proceedings of the Sixth Conference on Machine Translation*, pages 911–919, Online. Association for Computational Linguistics.

Jan-Martin O. Steitz, **Jonas Pfeiffer**, Iryna Gurevych, and Stefan Roth. 2021. TxT: Crossmodal End-to-End Learning with Transformers. In *Pattern Recognition - 43rd DAGM German Conference, DAGM GCPR 2021*, pages 405-420, Bonn, Germany. Springer.

Tim Baumgärtner, Kexin Wang, Rachneet Sachdeva, Max Eichler, Gregor Geigle, Clifton Poth, Hannah Sterz, Haritz Puerto, Leonardo F. R. Ribeiro, **Jonas Pfeiffer**, Nils Reimers, Gözde Gül Şahin, Iryna Gurevych. 2022. UKP-SQUARE: An Online Platform for Question Answering Research. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.* Dublin, Ireland. Association for Computational Linguistics.

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, **Jonas Pfeiffer**, and Iryna Gurevych. 2022. AdapterHub Playground: Simple and Flexible Few-Shot Learning with Adapters. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022: Systems Demonstrations*, pages 61-75, Dublin, Ireland. Association for Computational Linguistics.

Emanuele Bugliarello, Fangyu Liu, **Jonas Pfeiffer**, Siva Reddy, Desmond Elliott, Edoardo Maria Ponti, and Ivan Vulić. 2022 IGLUE: A benchmark for transfer learning across modalities, tasks, and languages. In *International Conference on Machine Learning, ICML 2022*, pages 2370–2392, Baltimore, Maryland, USA. PMLR.

Gregor Geigle, **Jonas Pfeiffer**, Nils Reimers, Ivan Vulić, and Iryna Gurevych. 2022b. Retrieve Fast, Rerank Smart: Cooperative and Joint Approaches for Improved CrossModal Retrieval. *Transactions of the Association for Computational Linguistics*, 10/503–521. MIT Press.

Chen Liu, **Jonas Pfeiffer**, Anna Korhonen, Ivan Vulić, and Iryna Gurevych. 2022a. Delving deeper into cross-lingual visual question answering. *arXiv preprint.*

Gregor Geigle, Chen Liu,Jonas Pfeiffer, Iryna Gurevych. 2022a. One does not fit all! On the Complementarity of Vision Encoders for Vision and Language Tasks. *arXiv preprint.*

The experimental source code of our work related to the contributions listed above is publicly available at https://github.com/adapter-hub and https://github.com/facebookresearch/fairseq. Details on our strategy to data handling are given in Appendix A.

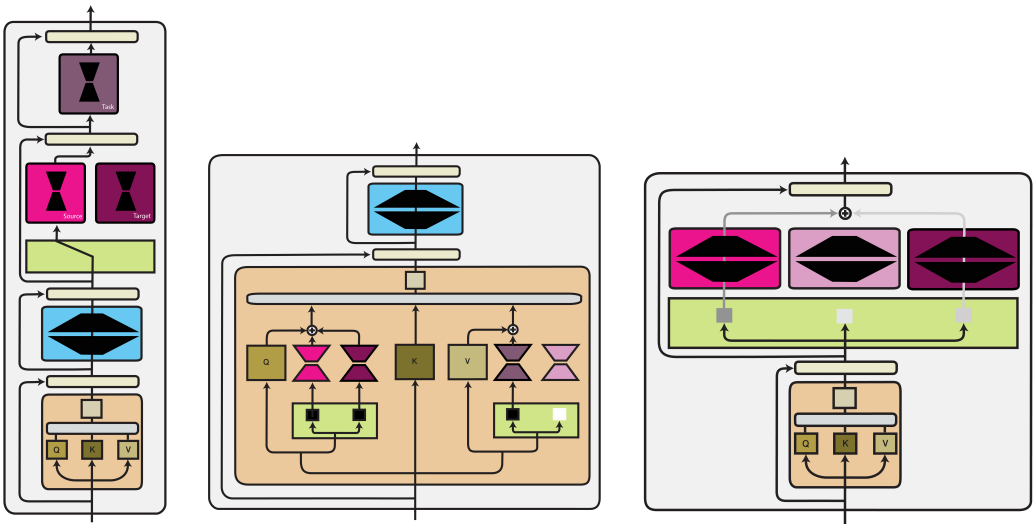
Chapter 1

Introduction

Transfer learning has recently become pervasive in natural language processing (Ruder et al., 2019; Brown et al., 2020). In its most successful incarnation, it consists of pre-training a model on vast amounts of raw data in a self-supervised fashion. Subsequently, this model can be fine-tuned for new tasks based on a small number of labelled examples. Despite its success, this paradigm for transfer learning suffers from a series of limitations. Firstly, in multi-task fine-tuning, the learning signals from different tasks may *(catastrophically) interfere* with each other (McCloskey and Cohen, 1989). Similarly, in a continuous learning setting, adapting to new examples can result in *catastrophic forgetting* of knowledge acquired from previous examples (Sutton, 1986; French, 1999).[1] Secondly, in settings where the training and evaluation distributions are not identical, these models fail in *generalizing systematically* (Lake and Baroni, 2018; Hupkes et al., 2020). This hampers the deployment of these models in real-world applications where distribution shifts are common, as it makes them brittle and inaccurate.

In contrast, many biological and artificial systems do not suffer from these weaknesses by virtue of their *modularity* (Fodor, 1983; Ballard, 1986). Artificial systems, such as programming languages and computer hardware, are similarly designed in a modular fashion (Booch et al., 2008; Baldwin et al., 2000) because this modular design favours consistency, ease of adaptation, and interpretability. Consequently, previous work explored the idea of designing neural networks that are *explicitly* modular (Jacobs et al., 1991a; Rosenbaum et al., 2017; Ponti, 2021). This has the goal of achieving not only functional specialization (Zhang et al., 2022b), but also re-usability and composability. In particular, these methods involve identifying 1) *modules* in a neural network that can be updated locally and asynchronously, without affecting the rest of the parameters; 2) a *routing function* that allocates a subset of modules to each example or task; and 3) an *aggregation function* that aggregates the outputs of the active modules. Each of these three ingredients can be determined *a priori* or learned in an end-to-end fashion. We provide several case studies of different configurations of these components in figure 1.1.

[1] These phenomena have also been referred to as spatial and temporal '*crosstalk*' (Jacobs et al., 1991b).



(a) MAD-X (b) Polytropon (c) MoE

Figure 1.1: Case studies of modular deep learning. 1.1a) MAD-X (Pfeiffer et al., 2020b) uses Adapter layers with fixed routing for zero-shot cross-lingual transfer. 1.1b) Polytropon (Ponti et al., 2022) uses low-rank adapters (LoRA; Hu et al., 2022) with hard learned routing for few-shot task adaptation. 1.1c) Sparse Transformer MoE (Fedus et al., 2021) use Multi-Layer Perceptrons with top-k soft routing, in order to scale to larger model sizes. Green components illustrate different routing functions (see § 3), magenta - purple components illustrate modular parts of the model. The three representative models illustrated here are only a fraction of possible configurations from the 'configuration manifold' that can be created by varying the components surveyed in Chapters 2-5.

The main goals of a modular design of neural architectures are the following: positive transfer, compositionality, and parameter efficiency. *Firstly*, modularity encourages positive transfer by encoding similar functions with the same module. At the same time, it prevents interference and forgetting by allocating distinct functions to different dedicated modules (Jacobs et al., 1991b). For instance, massively multilingual models are known to suffer from a 'curse' of interference (Conneau et al., 2020) due to the conflicting information that the gradient from each language-specific loss carries (Wang et al., 2021). A possible solution is augmenting these entangled, fully shared models with specialised modules responsible for individual languages (Pfeiffer et al., 2020b, 2022c). *Secondly*, modules representing different skills of tasks or features of examples can be composed to *generalize systematically*. This is crucial in two main settings, which correspond to two aspects of compositionality: one is the ability to *re-combine*, i.e. zero-shot transfer to tasks consisting of new subsets of learned skills, or examples consisting of new subsets of observed features (Hupkes et al., 2020). For instance, while modules for the Guaraní language and dependency parsing can only be trained separately due to the lack of annotated data, they can be composed to perform inference on this unobserved task–language combination (Pfeiffer et al., 2020b). *Thirdly*, an additional advantage of modular neural architectures is that they can be *updated locally and asynchronously*, without affecting the rest of the network. As a consequence, fine-tuning a model towards a

specific task only requires storing a parameter-efficient module rather than a separate copy of the entire model. What is more, modules can be added or removed on-the-fly in an incremental manner, adjusting the model capacity according to the task complexity. This ability is known as *conditional computation* (Bengio et al., 2016).

In this thesis, we first offer a unified view of modular deep learning, illustrating how many families of methods can be defined by four key components: **1)** how they implement modules, which constitute the minimum unit of computation; **2)** how they select active modules through a routing function; **3)** how module outputs are aggregated; and **4)** how the modules are trained with the rest of the model. We then discuss the different contributions we have made to each of these axes of modularity.

1.1 Thesis Outline and Taxonomy

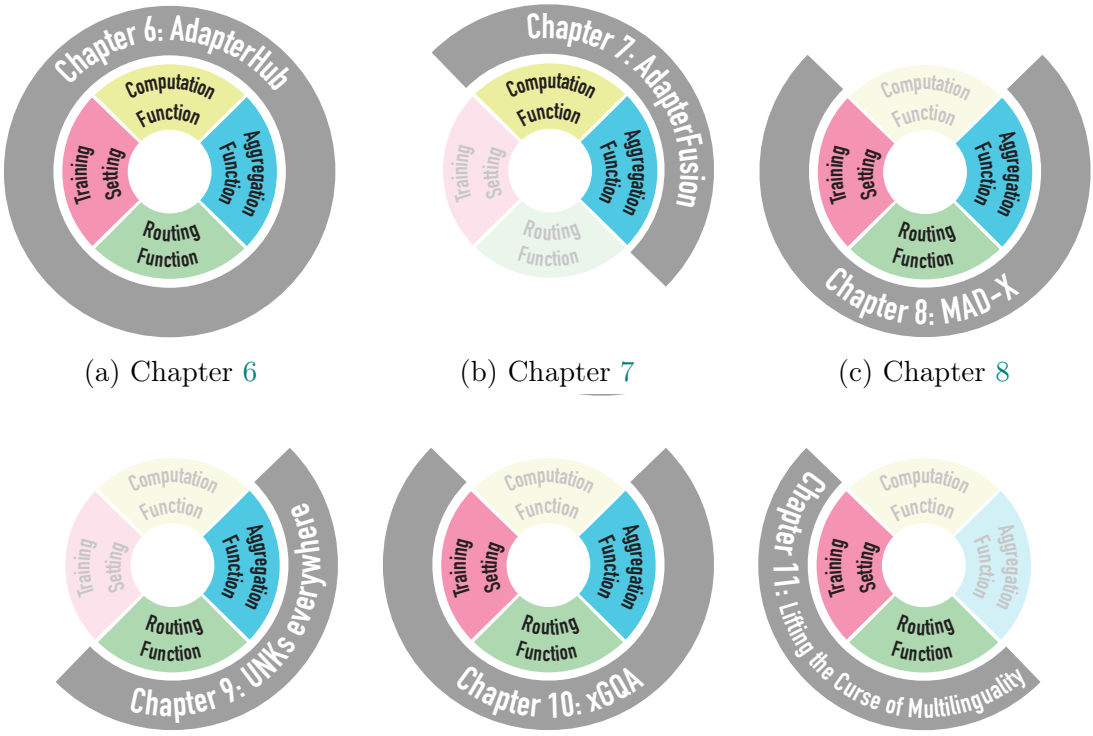
In order to provide a unified view of the varied landscape of modular deep learning, we 'dissect' existing models into four independent dimensions of variation, structured into their respective chapters of this thesis. We classify the individual contributions as part of each of the dimensions, as illustrated in Figure 1.2. We conclude the thesis and provide an outlook on possible future work in Chapter 12.

Chapter 2 - Computation function: *How is each module implemented?* A module may consist of any component of a neural architecture, such as multiple copies of a model (Jacobs et al., 1991a) or one of its layers (Fedus et al., 2021). We distinguish between interpolation of parameters (parameter composition), concatenation with input features (input composition), and function composition by stacking neural modules.

Chapter 3 - Routing function: *How are active modules selected?* Under *fixed routing*, we categorise approaches where the routing function is fixed. This assumes that the information captured by each module as well as the identity of the tasks where a module should be active is known *a priori*. In *learned routing*, the parameters of the routing mechanism are learned as part of the model. In this case, routing is soft if all modules are ranked through a continuous score, or hard if each module is given a binary score (active or inactive).

Chapter 4 - Aggregation function: *How are the outputs of the active modules aggregated?* We differentiate between methods that compose the outputs of the active modules deterministically (e.g., based on a weighted average) from those where the aggregation function is implemented as a learnable neural network.

Chapter 5 - Training setting: *How are the modules trained?* Some methods, such as MoEs, train the modules (and possibly the routing function) jointly with the shared weights of a base model. As an alternative, transfer learning approaches introduce modules after pre-training weights and adapt them during fine-tuning.



(a) Chapter 6 (b) Chapter 7 (c) Chapter 8

(d) Chapter 9 (e) Chapter 10 (f) Chapter 11

Figure 1.2: The organization and contributions of this thesis. We first introduce the four dimension of modularity in Chapter 2 - **Computation Function**, Chapter 3 - **Routing Function**, Chapter 4 - **Aggregation Function**, Chapter 5 - **Training Setting**. The subsequent Chapters 6 - 11 individually contribute to these dimensions, as illustrated above. Finally in Chapter 12 we conclude the thesis and provide an outlook on possible future work.

1.2 Notation

More formally, let a neural network $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ be decomposed into a composition of functions $f_{\theta_1} \odot f_{\theta_2} \odot \cdots \odot f_{\theta_l}$, where $\odot$ stands for function composition. The sub-functions refer to the model's l layers, each with a unique indexed sequence of parameters $\theta_i, i = 1, \ldots, l$. In turn, these can be further decomposed recursively into their constituent functions: for instance, a Transformer layer (Vaswani et al., 2017) includes linear mappings for the query, key, value, and output, as well as a non-linear feed-forward network, and highway connections. We further denote the values of the parameters at initialisation as θ^0, and the parameters after convergence are denoted as $\theta^\star$.

Given a module with parameters ϕ, it can modify the i-th sub-function with input $\boldsymbol{x}$ in different ways:

1. *parameter composition*: $f'_i(\boldsymbol{x}) = f_{\theta_i \oplus \phi}(\boldsymbol{x})$, where $\oplus$ stands for an operation that interpolates the parameters, such as element-wise addition.

2. *input composition*: $f'_i(\boldsymbol{x}) = f_{\theta_i}([\boldsymbol{x}, \phi])$, where $[\cdot, \cdot]$ stands for concatenation.

$\boldsymbol{x} \in \mathcal{X}$	Input data
$\boldsymbol{y} \in \mathcal{Y}$	Output data
$\boldsymbol{h} \in \mathcal{H}$	Hidden representation
$t \in \mathcal{T}$	Task index
$f : \mathcal{X} \cup \mathcal{H} \to \mathcal{Y} \cup \mathcal{H}$	A computation function
θ	Shared parameters
$\mathcal{F} = \{\phi_1, \ldots, \phi_{\lvert\mathcal{F}\rvert}\}$	Set of module parameters
$\alpha \in \mathcal{A}$	Vector of routing scores (for a single task)
$\boldsymbol{A} \in \mathbb{R}^{\lvert\mathcal{T}\rvert \times \lvert\mathcal{F}\rvert}$	Matrix of routing scores (for all tasks)
$r : \mathcal{X} \cup \mathcal{H} \to \mathcal{A}$	A routing function
ρ	Routing parameters
$\oplus$	Element-wise addition
$[\cdot, \cdot]$	Concatenation
$\odot$	Function composition

Table 1.1: Notation and definitions.

3. *function composition*: $f'_i(\boldsymbol{x}) = f_{\theta_i} \odot f_\phi(\boldsymbol{x})$, where the outputs of the first and second function are combined in some way.

For each i-th sub-function, multiple modules from an inventory $\mathcal{F} = f_{\phi_1}, \ldots, f_{\phi_{\lvert\mathcal{F}\rvert}}$ can be selected through a routing function $r(\cdot)$, which returns a score α_i for each module f_{ϕ_i} conditioned on metadata. Metadata can consist of individual tokens x_t, an example $\boldsymbol{x}$, or the current task $t \in \mathcal{T}$. Note that α can be fixed *a priori* through expert knowledge or learned through an appropriate parameterisation $r_\rho(\cdot)$, where ρ refers to (learnable) parameters of the routing function. Often, the routing function takes special forms:

1. In hard routing, $\alpha \in \{0, 1\}^{\lvert\mathcal{F}\rvert}$ is a discrete binary vector.

2. In soft routing, $\alpha \in [0, 1]^{\lvert\mathcal{F}\rvert}$ is a continuous probability distribution, such that $\sum_i \alpha_i = 1$.

3. Finally, $\alpha \in \mathbb{R}^{\lvert\mathcal{F}\rvert}$ can be an unnormalised score vector. This is the case in linear hyper-networks, where α is usually interpreted as a task embedding and the row-wise stacked module parameters $[\phi_1, \ldots, \phi_{\lvert\mathcal{F}\rvert}]$ act as a generating function.

Finally, the output of each module is combined through an aggregation function.[2] For instance, a common deterministic aggregation function is averaging:

$$f'_i(\boldsymbol{x}; \mathcal{F}, \theta_i) = \sum_{\phi_j \in \mathcal{F}} r(\phi_j)\, f(\boldsymbol{x}; \theta_i, \phi_j) \tag{1.1}$$

[2] To avoid clutter in terminology, throughout this work we use the term *composition* to refer to the computation function (Section 2), and the term *aggregation* to refer to different approaches of combining the outputs of different modules (Section 4).

Given shared parameters θ and an inventory of modules $\mathcal{F}$, we first sample a task t, and an input–output pair $(\boldsymbol{x}, \boldsymbol{y})$. We then obtain routing scores α using the routing function r. We now compute the hidden representation $\boldsymbol{h}_i$ of each module ϕ_i and aggregate them (in this case, based on the routing scores). We finally perform a gradient update on the module parameters in $\mathcal{F}$ and the routing parameters ρ. Other settings such as joint training of shared and modular parameter are also common (see chapter 5).

Chapter 2

Computation Function

The computation function determines the design of a module. Modules can take a plethora of shapes, such as layers of MLPs (Rosenbaum et al., 2017; Kirsch et al., 2018; Chang et al., 2018), independent RNNs (Goyal et al., 2021), independent CNNs (Parascandolo et al., 2018), or special-purpose architectures (Andreas et al., 2016). However, modules are most often integrated into a base architecture whose parameters are fully shared. We identify three core types of computation functions that compose the module with the model's sub-functions: parameter composition, input composition, and function composition. We provide example illustrations of the three computation functions (in addition to a hyper-network) as part of a Transformer architecture in figure 2.1.

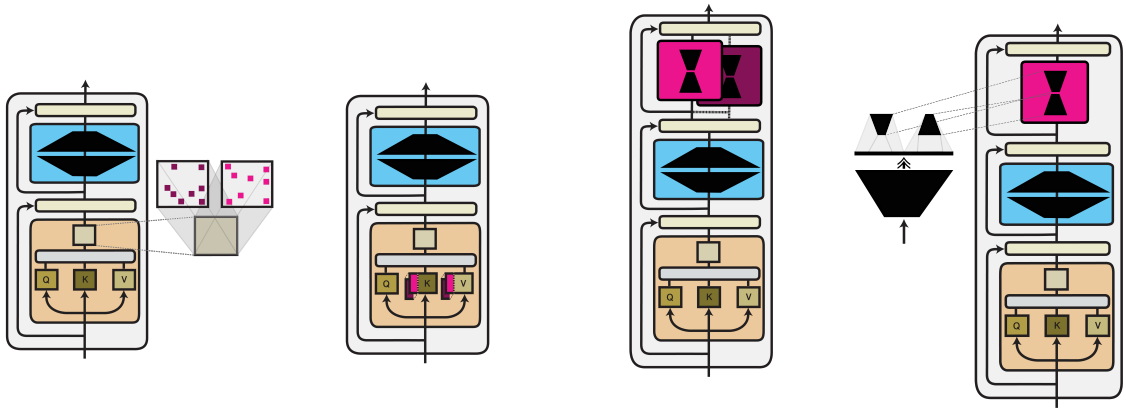
2.1 Parameter Composition

Parameter composition methods augment a base model with sufficient capacity with task-specific components on the level of individual weights.

2.1.1 Sparse Subnetworks

A common inductive bias on the module parameters ϕ is to make them sparse. This is based on the assumption that only a small number of parameters of an over-parameterized model will be relevant for a particular task, and that similar tasks share similar sub-networks. This is the case for language subnetworks (Stanczak et al., 2022; Foroutan et al., 2022) in multilingual language models (Conneau et al., 2020). The most common method to induce sparsity is via pruning. For a general overview of pruning and other sparse methods, we direct the reader to Hoefler et al. (2021).

Pruning can be seen as the application of a binary mask $\boldsymbol{b} \in \{0,1\}^{|\theta|}$ that selectively keeps or removes each connection in a neural network and produces a subnetwork. Weights are commonly pruned based on their magnitude (Han et al., 2016). After training a model, the trained weights are sorted based on their ab-



(a) Parameter Composition (b) Input Composition (c) Function Composition (d) Hyper-Network

Figure 2.1: Different modular designs for Transformer architectures; best viewed in colour. Task-specific modular components are illustrated in magenta and purple, respectively. **(a) Parameter Composition** (§ 2.1): A sparse sub-network in the linear layer as part of multi-head-attention. **(b) Input Composition** (§ 2.2): Prefix-tuning (Li and Liang, 2021) extends the input by prepending embeddings to the key and value matrices in the Transformer layer. **(c) Function Composition** (§ 2.3): Task-specific bottleneck layers are inserted in each layer that transform the hidden representations (Houlsby et al., 2019). **(d) Hyper-Network** (§ 2.4): A small separate neural network generates modular parameters conditioned on meta-data. We show its application to function composition but it is compatible with all computation functions.

solute magnitude and a fraction of the lowest-magnitude weights are selected for removal. As pruning generally leads to a loss in performance due to the change in network connections, the non-pruned weights are typically re-trained. In practice, rather than pruning all weights in a single run, iterative pruning (Han et al., 2015; Frankle and Carbin, 2019) where a model is pruned over multiple iterations often performs better. Pruning with a binary mask $\boldsymbol{b}$ can be seen as adding a task-specific parameter vector ϕ to the parameters of an existing model $f'_\theta = f_{\theta+\phi}$ where $\phi_i = 0$ if $b_i = 0$.[1] Compared to other modular approaches, subnetworks typically do not distinguish between the parameters of different functions f_{θ_i} and compose the module parameters ϕ with the parameters of the entire model θ.

ϕ is typically obtained by pruning the lowest-magnitude weights for each task and re-training the non-pruned components (Mallya and Lazebnik, 2018; Sun et al., 2020a; Lin et al., 2021).[2] If a sparse model is desired at runtime, the pruned weights can be removed entirely by multiplying the binary mask with the existing weights: $f'_\theta = f_{\theta \circ \boldsymbol{b}+\phi}$ where $\circ$ is the element-wise or Hadamard product. Sparse subnetworks that match the performance of full-parameter models are also known as 'winning

[1] For simplicity, we use ϕ_i to refer to the i-th value of ϕ rather than the i-th module.

[2] In practice, this is achieved by masking the gradient based on the binary mask $\boldsymbol{b} \circ \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D})$ where $\mathcal{L}$ is a loss function and $\mathcal{D}$ is a dataset (Ansell et al., 2022).

tickets' (Figure 2.1a; Frankle and Carbin, 2019). Manessi et al. (2018) and Mehta (2019) observe that sparse subnetworks transfer well across computer vision tasks while Chen et al. (2020) find that subnetworks trained on masked language modeling (MLM) transfer to NLP tasks in general.

Pruning a randomly initialized model may be sensitive to the choice of hyper-parameters, so it is more common to prune a pre-trained model (Sun et al., 2020a). First-order (gradient-based) pruning can be used to capture the change from the pre-trained model weights (Molchanov et al., 2017; Sanh et al., 2020). In Lottery Ticket Sparse Fine-Tuning (SFT; Ansell et al., 2022), the weights that are not selected are *frozen* instead of being *pruned* after rewinding the model to its initial state. Thus, the fine-tuned subset of weights creates a sparse vector of difference with respect to the shared model.

Alternatively, the parameters ϕ of the subnetwork itself can be fixed. As the number of possible subnetworks grows combinatorially with the number of model parameters, there are subnetworks that achieve good performance even for randomly initialized, fixed models (Zhou et al., 2019). ϕ in this case consists only of the binary mask $\boldsymbol{b}$: $f'_\theta = f_{\theta \circ \boldsymbol{b}}$. $\boldsymbol{b}$ can be first learned as a real-valued mask and then binarized (Mallya et al., 2018). A fixed underlying model can accommodate a potentially unlimited number of task-specific binary masks that exist in super-position (Wortsman et al., 2020).

2.1.2 Structured Composition

Beyond individual weights, sets of parameters can also be composed. There are many approaches, which modify only certain parts of the model associated with a pre-defined group $\mathcal{G}$: $f'_i = f_{\theta_i + \phi_i} \forall f'_i \in \mathcal{G}$. The most common setting is for each group to correspond to a layer l and to only update the parameters of certain layers such as the last one (Donahue et al., 2014). Groups can also relate to more fine-grained parts of the model. For instance, a group $\mathcal{G}$ consisting of a model's bias parameters is a practical choice as this removes the need to store the model's intermediate activations (Cai et al., 2020; Ben Zaken et al., 2022). Such structure can also be combined with sparse methods by encouraging sharing masks between groups (Guo et al., 2021).

2.1.3 Low-Rank Composition

Similar to sparsity, another useful prior is for the module parameters ϕ_i to lie in a low-dimensional subspace. Li et al. (2018) show that models can be optimised in a low-dimensional, randomly-oriented subspace rather than the full parameter space. In this setting, the module parameters $\phi \in \mathbb{R}^d$ are low-dimensional compared to the model parameters $\theta \in \mathbb{R}^D$ and $d \ll D$. A random matrix $\mathbf{M} \in \mathbb{R}^{d \times D}$ can be used to project from d to D: $f'_\theta = f_{\theta + \phi \mathbf{M}}$. $\mathbf{M}$ is typically obtained via the Fastfood transform (Le et al., 2013) and is factorized as random linear matrices. Specifically, $\mathbf{M} = \mathbf{HG\Pi HB}$ consists of a Hadamard matrix $\mathbf{H}$, a random diagonal matrix with independent standard normal entries $\mathbf{G}$, a random diagonal matrix with

equal probability ±1 entries $\mathbf{B}$, and a random permutation matrix Π.

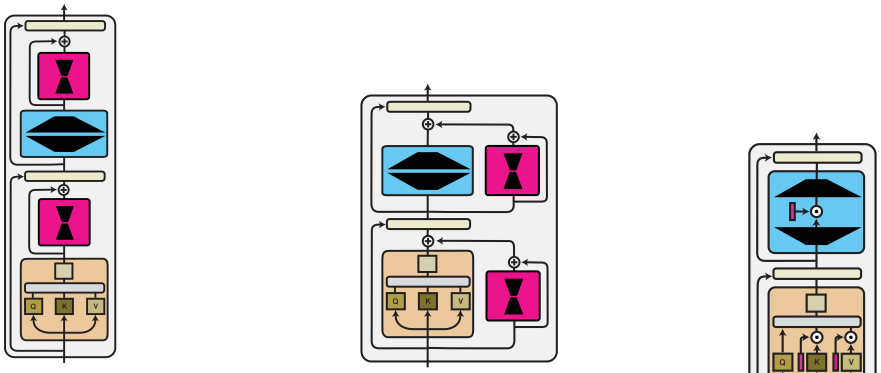
The minimum d that achieves within 90% of the full-parameter model performance is known as the intrinsic dimensionality of a given task. Aghajanyan et al. (2021) investigate the intrinsic dimensionality of various NLP tasks and pre-trained models and identify d as low as 200. However, storing the random matrices results in a substantial memory overhead and is slow to train (Mahabadi et al., 2021a). To save space, a decomposition via low-rank matrices $\boldsymbol{A} \in \mathbb{R}^{r \times k}$ and $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ can be applied to certain groups $\mathcal{G}$: $f_i' = f_{\theta_i + \text{vec}(\boldsymbol{B}_i \boldsymbol{A}_i)} \, \forall f_i' \in \mathcal{G}$ where r is the rank of the matrix, k is the hidden dimensionality, and vec is the vectorization of a matrix. LoRA (Hu et al., 2022) applies this low-rank decomposition to the weight matrices in the self-attention module of the Transformer.

2.2 Input Composition

Input composition methods augment a function's input $\boldsymbol{x}$ by concatenating it with a learnable parameter vector ϕ_i: $f_i'(\boldsymbol{x}) = f_{\theta_i}([\phi_i, \boldsymbol{x}])$. The most common input to augment is the original input fed to the model's first layer f_1. The standard prompting setting for language models (Brown et al., 2020) can be seen as finding a task-specific discrete text prompt—optionally together with few-shot examples—that, when embedded using the model's embedding layer, will yield a vector ϕ that elicits the desired behaviour (Gao et al., 2021; Liu et al., 2021a). Similar methods have been extended to encoder models (Schick and Schütze, 2021a,b). However, models have been shown to be sensitive to the formulation of the prompt as well as to the set and order of few-shot examples (Lu et al., 2022).

Instead, a continuous prompt vector ϕ can be learned directly and concatenated with the input (Lester et al., 2021; Liu et al., 2021b; Zhong et al., 2021; Hambardzumyan et al., 2021). However, as ϕ is only concatenated with the first layer's input, the model has limited capacity to adapt to a specific task. As a result, such continuous (also called *soft*) prompts perform poorly at smaller model sizes and on some harder tasks (Mahabadi et al., 2021a; Liu et al., 2022c). To mitigate this, initialisation via multi-task learning has been proposed (Vu et al., 2022).

As an alternative, module vectors ϕ_i can be learned *for each layer* of the model: $f_i'(\boldsymbol{x}) = f_{\theta_i}([\phi_i, \boldsymbol{x}])$ (Figure 2.1b; Li and Liang, 2021; Liu et al., 2022c). While this increases the number of parameters, it provides the model with more flexibility to adapt to a given task. In practice, module parameters in the form of prefix vectors $\phi_i = \boldsymbol{P}_k^i, \boldsymbol{P}_v^i \in \mathbb{R}^{l \times d}$ are prepended to the keys and values of every multi-head attention layer. Attention is defined as $f_i(\boldsymbol{x}) = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^i, \boldsymbol{C}\boldsymbol{W}_k^i, \boldsymbol{C}\boldsymbol{W}_v^i)$ where $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times d_h}$ are the projections that produce the queries, keys, and values and $\boldsymbol{C} \in \mathbb{R}^{m \times d}$ is a sequence of context vectors. Prefix tuning thus takes the following form: $f_i'(\boldsymbol{x}) = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^i, [\boldsymbol{P}_k^i, \boldsymbol{C}\boldsymbol{W}_k^i], [\boldsymbol{P}_v^i, \boldsymbol{C}\boldsymbol{W}_v^i])$. Overall, input composition methods based on multi-layer prefix tuning (Li and Liang, 2021; Yang and Liu, 2022) can be seen as a generalisation of continuous prompting methods.



(a) Sequential Bottleneck Adapter

(b) Parallel Bottleneck Adapter

(c) IA^3

Figure 2.2: Different approaches of function composition. **a) Sequential Bottleneck Adapter:** The first adapter architecture proposed for transformers which consists of two bottleneck layers placed after the multi-head attention (MHA) and feed-forward (FF) layers (Houlsby et al., 2019). **b) Parallel Bottleneck Adapter:** Bottleneck layers processed in parallel to the MHA and FF layers of the pre-trained transformer components (Rebuffi et al., 2018; Stickland and Murray, 2019; He et al., 2022a). **c) IA^3**: Rescaling operations performed within the MHA and FF layers (Liu et al., 2022b).

2.3 Function Composition

While parameter composition deals with individual weights and input composition methods act only on a function's input, function composition methods are the most general as they augment the model with new task-specific functions (see Figure 2.1c): $f'_i(\boldsymbol{x}) = f_{\theta_i}(\boldsymbol{x}) \odot f_{\phi_i}(\boldsymbol{x})$.

2.3.1 Representation Composition

We can design modules f_{ϕ_i} that are composed with an existing model's functions f_{θ_i}. As the main purpose of such modules is the adaptation of a pre-trained model to new tasks and domains, they are also known as 'adapters'. We provide examples of different adapters in figure 2.2.

The adapter's design and the way it is composed with the model's functions is often modality-specific. In computer vision, the adapter typically consists of a 1×1 convolution, i.e., $f_{\phi_i}(\boldsymbol{x}) = \alpha * \boldsymbol{x}$ where α is a bank of $\mathbf{1} \times 1$ filters and $*$ is the convolution operation (Rebuffi et al., 2017). The module is then inserted between the convolutional blocks of a pre-trained ResNet (He et al., 2016) or another pre-trained model. In reinforcement learning, adapters have been used to reduce the dimensionality of lateral connections (Rusu et al., 2016).

In NLP, a bottleneck architecture has become popular which consists of a down- and up-projection, coupled with an intermediate activation function σ: $f_{\phi_i}(\boldsymbol{x}) = \boldsymbol{W}^D(\sigma(\boldsymbol{W}^U\boldsymbol{x}))$ where $\boldsymbol{W}^D \in \mathbb{R}^{k\times d}$ and $\boldsymbol{W}^U \in \mathbb{R}^{d\times k}$, k is the dimensionality of the input (typically the hidden dimension), and d is the bottleneck dimension. σ is commonly a non-linearity such as a ReLU unit (Figure 2.2a; Houlsby et al., 2019; Pfeiffer et al., 2020b). In a Transformer model, adapters are placed after the multi-head attention and/or feed-forward layers (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2021b).

Other variants for σ such as the identity function, multi-head attention, and multi-head attention with shared projection matrices have also been explored (Stickland and Murray, 2019). Mahabadi et al. (2021a) propose Compacter, a hyper-complex, low-rank adapter that reparameterises $\boldsymbol{W}$ in the adapter as: $\boldsymbol{W} = \sum_{i=1}^{n} \boldsymbol{A}_i \otimes \boldsymbol{B}_i$ where $\boldsymbol{A}_i \in \mathbb{R}^{n\times n}$ is shared across layers (n is a hyper-parameter) and $\boldsymbol{B}_i \in \mathbb{R}^{\frac{k}{n}\times\frac{d}{n}}$ is parameterised as a low-rank matrix $\boldsymbol{B}_i = \boldsymbol{s}_i\boldsymbol{t}_i^\top$ (r is the rank of $\boldsymbol{B}_i$).

Adapters can be routed sequentially or in parallel. Sequential adapters, are inserted between existing functions: $f_i'(\boldsymbol{x}) = f_{\phi_i}(f_{\theta_i}(\boldsymbol{x}))$ (Rebuffi et al., 2017; Houlsby et al., 2019).[3] Parallel adapters are applied in parallel to a model's functions: $f_i'(\boldsymbol{x}) = f_{\theta_i}(\boldsymbol{x}) + f_{\phi_i}(\boldsymbol{x})$ (Figure 2.2b; Rebuffi et al., 2018; Stickland and Murray, 2019; He et al., 2022a).

2.3.2 Rescaling

Instead of learning a transformation function, the representations can also be directly transformed via element-wise multiplication with learned parameters: $f_i'(\boldsymbol{x}) = f_{\theta_i}(\boldsymbol{x}) \circ \phi_i$. In computer vision, such task-specific rescaling is typically applied to batch normalization parameters (Bilen and Vedaldi, 2017). In NLP, similar rescaling can be applied to the layer normalization parameters (Houlsby et al., 2019). IA^3 (Figure 2.2c; Liu et al., 2022b) multiplies learned vectors with the keys and values in self-attention blocks and the intermediate activations in position-wise feedforward networks in the Transformer. Rescaling allows the model to select parameters that are more and less important for a given task and is compatible with other methods such as LoRA (Figure 2.1b; Hu et al., 2022). Multiplication with a binary mask can be seen as a special case of rescaling that incorporates sparsity. Similarly, Strezoski et al. (2019) multiply a task-specific random binary mask $\boldsymbol{b}$ with a function's input $\boldsymbol{x}$ at every layer.

2.4 Module Parameter Generation

In most of the previous approaches, the parameters of different modules $\phi_1, \ldots, \phi_{|\mathcal{F}|}$ are optimised separately. However, particularly when modules are applied to an existing model, the modules may benefit from sharing an underlying structure. Rather than learning ϕ_i directly, a small neural network can be used to generate the module parameters instead. Such a neural network that generates the parameters of another

[3] We omit the residual connection for simplicity.

model is known as a hyper-network (Ha et al., 2017). The hyper-network is usually parameterised as a one-hidden layer MLP.

Hyper-networks are most effective when they generate module parameters conditioned on relevant inputs (Figure 2.1d). For instance, in conditional batch normalisation (De Vries et al., 2017), batch normalisation rescaling parameters are generated based on a question embedding obtained via an LSTM. Feature-wise linear modulation (FiLM; Perez et al., 2018) and self-modulation (Chen et al., 2019) generate an element-wise affine transformation that is applied to image features, conditioned on a question embedding and generator input respectively.

Hyper-networks have been used to generate a diverse set of module parameters ϕ_i, including classifier heads (Ponti et al., 2021), continuous prompts (He et al., 2022b), and adapter layers (Üstün et al., 2020; Ansell et al., 2021b; Mahabadi et al., 2021b), most commonly conditioned on task (Mahabadi et al., 2021b) or language embeddings (Üstün et al., 2020; Baziotis et al., 2022). Such task or language embeddings can themselves be learned directly, or generated, for instance, based on the typological features of a language (Üstün et al., 2020; Ansell et al., 2021b).

To make the hyper-network more parameter-efficient, the same network can be used to generate multiple modules for the same task by conditioning it on the module position in addition to the task index (Mahabadi et al., 2021b). If additional side information is available, this can be integrated into the hyper-network via bi-linear interaction (Chen et al., 2019) or concatenation (Üstün et al., 2022). Conditioning on multiple inputs may allow the hyper-network to generalise to new combinations at inference time. A hyper-network is also helpful when the direct optimisation of module parameters is unstable, as in the case of prefix vectors (Li and Liang, 2021; Liu et al., 2021b).

2.5 Contributions

In what follows, we discuss this thesis' contributions to the *computation function* axis of modularity, as illustrated in Figure 1.2:

- In Pfeiffer et al. (Chapter 6; 2020a) we propose a framework which integrates many different composition functions. The framework incorporates low-rank composition (LoRA; Hu et al., 2022), input composition (Prefix-Tuning Li and Liang, 2021), and representation composition (Bottleneck Adapter, Parallel Adapter, Compacter; Houlsby et al., 2019; He et al., 2022a; Mahabadi et al., 2021a) function. The AdapterHub is continuously extended to new modular architectures, making it an easy-to-use framework for modular and parameter-efficient research.

- In Pfeiffer et al. (Chapter 7; 2021b) we propose a new bottleneck-adapter variant which places the bottleneck layer only after the feed-forward layer of the vanilla transformer architecture. This simplifies the original architecture proposed by Houlsby et al. (2019), which additionally places a layer after the

multi-head-attention component. We demonstrate that this simplified version performs on-par with the original variant.

Chapter 3

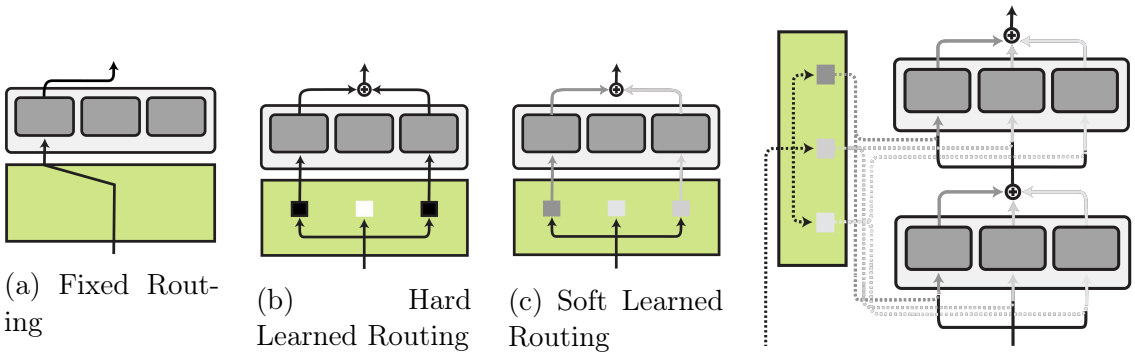
Routing Function

A sub-function $f(\cdot)$ of a modular neural network is parameterised by shared weights θ and module weights ϕ_i. A decision-making process is required to determine which modules are active, conditioned on metadata. This process is implemented through a routing function $r(\cdot)$ that assigns a score α_i to each module from an inventory $\mathcal{F} = \phi_1, \ldots, \phi_{|\mathcal{F}|}$. These scores determine which subset of modules is used, which leads to a sparse architectural design, owing to the fact that inactive modules do not contribute to the computation. This reduces time complexity while augmenting the model capacity (Fedus et al., 2022). We provide an overview of different routing methods in figure 3.1.

When metadata such as expert knowledge about sub-tasks (or skills) involved in a task is available, $r(\cdot)$ can be designed as a *fixed* function, that is, each routing decision is made *a priori* (Figure 3.1a). For instance, performing dialogue generation in Swahili, a task module for dialogue generation and a language module for Swahili can be selected. When no such prior information is available—for instance when modelling heterogeneous unlabelled data—routing of the sampled examples needs to be *learned* (Figures 3.1b-3.1c). In this case, the routing function is conditioned on either the current example $\mathbf{x}$ or metadata, such as the current task index $t \in \mathcal{T}$.[1]

Unfortunately, learned routing significantly increases the difficulty of the learning problem. In fact, in a multi-task or continual learning setting, similar to the rest of the model, routing parameters ρ may incur catastrophic interference and forgetting (McCloskey and Cohen, 1989; French, 1999). Another common failure mode is module collapse (Kirsch et al., 2018), whereby a one-to-one or one-to-all mapping between modules and parameters is established. Learning-to-route is crucially under-constrained, as multiple possible ways of decomposing the function into sub-modules are reasonable (Jacobs et al., 1991a). In the literature, it remains controversial whether learned routing can surpass heuristic fixed routing: Muqeeth et al. (2022) report negative results, whereas Ponti et al. (2022) find that learned routing may surpass expert module selection even in settings where tasks are procedurally

[1] Alternative strategies which aim at increasing the capacity of the model include random routing (Zuo et al., 2022; Wang et al., 2022) or routing based on hash functions (Roller et al., 2021).



(a) Fixed Routing (b) Hard Learned Routing (c) Soft Learned Routing (d) Layer Routing

Figure 3.1: Different routing methods. (a) **Fixed Routing:** Examples are passed to modules based on a pre-defined logic, known a priori. (b) **Hard Learned Routing:** Learned hard selection modules. (c) **Soft Learned Routing:** Soft selection and weighting of modules. (b) **Layer Routing:** Global routing of modules for multiple layers of the model.

constructed to require certain skills.

Another aspect of designing a routing function is its level of granularity. Routing can select modules globally for the entire network, make different allocation decisions per layer, or even hierarchically select sub-routers (Figure 3.1d). This last method is also referred to as 'dispatched routing' by Rosenbaum et al. (2017). Global routing is more challenging as the space of potential architectures grows exponentially as $|\mathcal{F}|^l$, where l is the number of layers or sub-functions of the network. Per-layer routing instead assumes conditional independence among decisions, thus facilitating scaling. Crucially, routing scores are sometimes not only employed to select a subset of modules, but also to aggregate their outputs. This second purpose is addressed in more depth in chapter 4.

3.1 Fixed Routing

When making the routing decision *a priori*—i.e. when we utilise metadata (e.g. task identity t) to make the discrete routing decisions *before* training—we speak of fixed routing (Figure 3.1a). Here the routing function $r(\cdot)$ simplifies to a k-hot selection of the respective modules $\mathcal{K} \subseteq \mathcal{F}$ for the examples with certain metadata:

$$r(\phi_i) = \begin{cases} 1 & \text{if } i \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

This function defines a binary matrix $A \in \{0,1\}^{|\mathcal{T}| \times |\mathcal{F}|}$, where the number of rows corresponds to possible tasks and the number of columns corresponds to the size of the module inventory.

One simple example of fixed routing in multi-task learning is when all parameters, except the final classification layer, are shared among all tasks (Ruder, 2017).

Independently from the task identity, the examples are passed through the same network until the penultimate layer. The penultimate layer's representations are then *routed* to their respective final classification layer according to the task identity. This boils down to setting $|\mathcal{K}| = 1$, with the additional constraint that tasks cannot share modules, which results in the allocation matrix being an identity matrix, $A = I$.

While not immediately apparent, methods which adapt pre-trained models towards individual tasks (Rebuffi et al., 2017, 2018; Houlsby et al., 2019; Bapna and Firat, 2019; Li and Liang, 2021; Liu et al., 2022b; Hu et al., 2022; Ansell et al., 2022; Ben Zaken et al., 2022, *inter alia*)–as discussed in §2–deterministically route representations through the newly introduced module f_ϕ. Given that the pre-trained weights are frozen and modules trained on different tasks can be plugged in and out, the components become modular even if they are developed asynchronously and independently of each other (Pfeiffer et al., 2021b). In a sense, community-based hubs of pre-trained adapters such as AdapterHub (Pfeiffer et al., 2020a) can be considered as ever-evolving multi-task models, the development of whose components has been distributed throughout the community.[2] Moreover, since newly introduced weights are encapsulated between frozen (shared) weights, adapted representations of intermediate layers are implicitly aligned as they are passed as input to the same frozen components.

Hampshire and Waibel (1992) were possibly among the first to train independent experts for a series of sub-tasks known *a priori*. In this case, the subset of experts $\mathcal{K}$ associated with each task t is assumed as given, resulting in the rows of A being k-hot vectors. In cross-lingual transfer, any problem can be decomposed into a task and language variety. Fixed routing can select separate language and task components, and facilitate generalisation to new, unobserved combinations of tasks and languages at inference time (Pfeiffer et al., 2020b; Ponti et al., 2021; Üstün et al., 2022). In this case, $|\mathcal{K}| = 2$.

Beyond task identity, routing can be performed based on other metadata such as language, domain, or modality information. Fan et al. (2021) route deterministically for multilingual machine translation where the language family is used to route examples from different languages, each language family having a specified module/expert. (Pfeiffer et al., 2022c) add adapters for each language to a multilingual language model during pre-training on unlabelled text. In a similar vein, (Gururangan et al., 2022) add domain-specific adapters to language models, deterministically routing based on the source domain. This concept was further extended by Li et al. (2022), who proposed the branch-train-merge method: copies of the same model are trained on different domains, and subsequently their weights are merged *post hoc*. Finally, modality can also inform fixed routing, such as in vision-and-language models: (Pfeiffer et al., 2022a). This allows for adapting the encoders of different modality streams.

[2] In contrast, combining entire models stored in model repositories via distillation (Khanuja and Johnson, 2021) or averaging (Matena and Raffel, 2022) is much less efficient and subject to other limitations (see chapter 4).

3.2 Learned Routing

When the routing of examples is not decided a priori, a parameterized routing function $r_\rho(\cdot)$ needs to learn to route examples conditioned on metadata such as the task t or the input example $\mathbf{x}$. While this presentation is agnostic to the family of this function, usually $r_\rho(\cdot)$ is linear. This prevents the function from entirely ignoring the metadata it is conditioned upon.

Learning-to-route can roughly be split into *hard* routing and *soft* routing (Rosenbaum et al., 2019). *Hard* routing methods learn a discrete selection of modules, thereby simulating the fixed routing scheme, where only a subset of modules is selected for each decision-making step (Figure 3.1b). Inference for hard routing systems typically builds on score function estimators (Williams, 1988, 1992) or stochastic re-parameterisation (Jang et al., 2017). On the other hand, *soft* routing methods learn a probability distribution over modules (Jacobs et al., 1991a), which is often interpreted as a score to select the top-k modules (Figure 3.1c; Fedus et al., 2021).

3.2.1 Challenges of Learned Routing

Learned routing introduces a number of challenges, including *training instability*, *module collapse* (Kirsch et al., 2018), and *overfitting*. These were first systematically described by Rosenbaum et al. (2019). We follow a similar taxonomy.

Training Instability This emerges especially in settings where modules are jointly learned from scratch; in this case, the router is required to select a module that has no clear specialisation at the start of training, which can result in routing dynamics that never stabilise. Curriculum learning can be used to mitigate this challenge to some extent (Chang et al., 2018). As an alternative, the learning rate of the router parameters can be lowered (Rosenbaum et al., 2017) or increased (Ponti et al., 2022) compared to the modules. This creates a coarse-to-fine dynamic where necessary skills are determined after or before module specialisation.

Finally, module and router initialisation also likely plays a key role in breaking symmetries at the beginning of training.

Module Collapse This term describes scenarios where the router falls into a local optimum where only a small number of modules (in the extreme case, one) from the available inventory are selected and trained, while not making use of the remaining modules. This excessively favours exploitation over exploration, possibly leading to sub-optimal results. Ways to address this problem include encouraging diversity through rewards (Cases et al., 2019), training the router and modules separately (Kirsch et al., 2018), and using high-level text features as metadata to condition the router, e.g. text genre (Cases et al., 2019). The diversity of training tasks also plays an important role in learning viable routing strategies (Chang et al., 2018; Caccia et al., 2022). Ahn et al. (2019) fix the routing for the first few iterations following an ϵ-greedy method for initial exploration of all modules and afterwards switch to learned routing. Dua et al. (2022) propose tricks to stabilise MoE training, including

temperature warm-up for sampling domains with fewer examples in unbalanced distributions.

Overfitting Finally, due to their ability to model specific areas of the input space independently, deep modular networks risk *overfitting* to the noise (Rosenbaum et al., 2019). For instance, routing at the token level was shown to lead to performance drops in out-of-domain generalisation for MoEs (Artetxe et al., 2021). For a similar reason, gains in pre-training do not always translate into gains in fine-tuning for MoEs (Fedus et al., 2021). Increased robustness can be achieved by routing conditioned on higher-level metadata if available (Chang et al., 2018; Cases et al., 2019; Kudugunta et al., 2021). In general, routing should strike the right balance by affording higher expressivity while not overfitting the training data (Rosenbaum et al., 2019). In particular, strategies that favour the combinatorial behaviour of modules yield superior generalisation (Chang et al., 2018; Ponti et al., 2022).

3.2.2 Hard Learned Routing

A model may learn how to select modules through *hard* routing. This implies that the choice of whether a module is active or excluded from the computation graph is binary. Discrete decisions are not amenable to be learned through vanilla gradient descent: since small perturbations of parameters do not affect the outcome, the gradient of the loss with respect to the routing parameters is zero. Consequently, alternative strategies such as Reinforcement Learning (e.g. Routing Networks (Rosenbaum et al., 2017), Modular Networks (Kirsch et al., 2018), Compositional Recursive Learner (CRL; Chang et al., 2018)), Evolutionary Algorithms (e.g. PathNet (Fernando et al., 2017), μNet (Gesmundo and Dean, 2022a,b)), or Stochastic Reparametrization (e.g. AdaShare (Sun et al., 2020b), Polytropon (Ponti et al., 2022; Caccia et al., 2022)) need to be utilized.

3.2.3 Soft Learned Routing

Mixture of Experts To sidestep *discrete* selections of modules, several works propose *soft* routing methods, where *all* modules are selected and aggregated according to a *weighted combination*, i.e. a mixture of experts (MoE; Jacobs et al., 1991b; Jordan and Jacobs, 1994).[3] Here, the router learns a probability distribution over the available modules, i.e. $p(\mathcal{F}) = r_\rho(\cdot)$. Hence, routing and aggregation take place as:

$$f'_i(\boldsymbol{x}) = \sum_{\phi_j \in F} r(\phi_j)\, f(\boldsymbol{x}; \theta_i, \phi_j) \tag{3.2}$$

In contrast to the discrete selection of *hard routing* methods, this setup is fully differentiable, allowing the system to be trained end-to-end. A number of works (Eigen et al., 2014; Meyerson and Miikkulainen, 2018; Wortsman et al., 2020, *inter*

[3] In the following sections we use the term "expert" and "module" interchangeably to reflect common practice in the body of research on MoEs.

alia) train a continuous weighting (i.e. a mixture) of all modules; however, this limits the degree of modularity as parameters updates are not local; instead, they always affect all modules. Additionally, activating *all* modules for each example significantly increases the computational cost for each forward and backward pass through the network. To circumvent this, Shazeer et al. (2017) and Lepikhin et al. (2021) only route to the top-k of $|\mathcal{F}|$ modules, where $1 < k < |\mathcal{F}|$. The output representations of the k *active* modules are averaged according to the respective routing weights, whose sum is re-normalised to 1:

$$f'_i(\boldsymbol{x}) = \sum_{\phi_j \in \text{top}_k[r(\boldsymbol{\phi})]} \frac{r(\phi_j)}{\sum r(\phi)} f(\boldsymbol{x}; \theta_i, \phi_j) \tag{3.3}$$

Fedus et al. (2021) and Clark et al. (2022) take this idea to the extreme and demonstrate that even top-1 routing can achieve competitive results for language modelling. Essentially, this boils down to a differentiable approximation of hard routing as follows:

$$f'_i(\boldsymbol{x}) = \sum_{\phi_j \in \mathcal{F}} f(\boldsymbol{x}; \theta_i, \phi_j)\, \mathbb{1}_j(\text{argmax}[r(\phi)]) \tag{3.4}$$

where $\mathbb{1}$ is the indicator function, which returns 1 if $j = \text{argmax}[r(\phi)]$ and 0 otherwise.

Token-Level Routing MoEs have recently undergone a revival as part of the efforts to scale Transformers. In particular, MoE Transformers route to a subset of Feed-Forward Network (FFN) modules per layer instead of a single FFN. The focus of these works is on computationally efficient training of very large models. This is achieved by splitting the input tokens across different (hardware-) accelerators. The MoE routing algorithm is therefore required to (ideally) uniformly distribute the tokens of all the examples in an input batch across all accelerators, i.e. to *load balance* computation across "experts". The dominating routing strategy is for each *token* to choose the top-k *experts* (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2021; Clark et al., 2022; Yang et al., 2021; Dua et al., 2022; Hazimeh et al., 2021; Rajbhandari et al., 2022; Riquelme et al., 2021; Du et al., 2022; Zoph et al., 2022). However, alternative approaches let each *expert* choose the top-k *tokens* (You et al., 2022; Zhou et al., 2022) or *globally* determine the best routing path (Lewis et al., 2021).[4]

However, since routing is conditioned on the token level, and the *load balancing* restriction limits the system from routing an entire example to a single module, all these methods suffer from a series of weaknesses: **1)** the experts are limited in terms of the functions they can learn, e.g. they cannot be chosen based on the semantics of the entire example (e.g. a sentence); and **2)** the system is potentially required

[4] For more details on load balancing methods we refer to Fedus et al. (2022) Chapter 4.

to relearn similar concepts in multiple modules as load balancing hinders the router from selecting the single best module for longer (e.g., repetitive) sequences. This is investigated further by (Lewis et al., 2021), who find that sparse models route syntactically and semantically similar *words* (in contrast to sentences or phrases) to the same modules. This sheds light on the limited expressiveness of modules which are learned on the *token*-level. Since scaling is the main focus of these works, their goals are *orthogonal* to modular approaches centred on parameter efficiency, transfer–interference trade-offs, and combinatorial generalisation.

Example-Level Routing Nevertheless, one could imagine obtaining the best of both worlds by hybridising sparse MoE Transformers models with deterministic or learned routing strategies from section 3.1 and section 3.2.2. Instead of routing each individual token separately, all tokens of a single example can be routed to the same experts. Kudugunta et al. (2021) experiment with two versions of example-level routing for machine translation: In *sentence-level* routing, they average pool over the token embeddings, and condition the router on the resulting representation. In *task-level* routing a task embedding is trained, based on which the router learns the distribution over modules. In a similar vein, Gupta et al. (2022) and Xi et al. (2022) implement *task-level* routing across modular experts to improve the amount of knowledge sharing during multi-task learning in the NLP and CV domain, respectively.

Since task identity (or other metadata) is not always given, especially in continual learning, it can be inferred through an auxiliary model. Van de Ven and Tolias (2019) refers to this scenario as '*class-incremental learning*'. For instance, the current task can be identified based on the lowest predictive uncertainty or an auxiliary task classifier (Oswald et al., 2020). In these cases, routing can depend on the predicted task identity.

3.3 Contributions

In this thesis, we predominantly focused on fixed routing strategies as a means to disentangle knowledge specific information into designated parts of the model. In what follows, we discuss this thesis' contributions to the *routing function* axis of modularity, as illustrated in Figure 1.2:

- In Pfeiffer et al. (Chapter 6; 2020a) we provide a general framework for fixed routing for multiple different modular architectures. Routing is performed based on an *id*, such as task, or language. The representations are then passed through the designated components at every layer.

- In Pfeiffer et al. (Chapter 8; 2020b) we propose a fixed routing method which passes representations through different components based on their language and task id. By disentangling language from task information through our proposed fixed routing mechanism we demonstrate improvements in cross-lingual transfer.

- In Pfeiffer et al. (Chapter 9; 2021c) we extend the former work by performing an additional routing step through new language-specific word-embedding layers.

- In Pfeiffer et al. (Chapter 10; 2022a) we extend the former two works by adding a vision modality: We not only perform fixed routing through language and task specific components, but additionally route modality specific representations (i.e. text and vision) through designated parts of the model.

- While in the previous works we performed fixed routing in *post-hoc* specialization, i.e. pre-trained weights are largely frozen, in Pfeiffer et al. (Chapter 11; 2022b) we perform language-specific fixed-routing during *pre-training*. We demonstrate that by integrating language specific capacity through modular components we mitigate catastrophic interference.

Chapter 4

Aggregation Function

While in the previous section on *routing* we have covered the topic of how to *select* different modules during training, we will now focus on how we can *aggregate* these functions in order to combine the respective information. It is important to emphasise that for many approaches, routing and aggregation are inseparable; that is the *selection* and *aggregation* of modules are performed simultaneously. On the other hand, the strategies for *aggregating* functions in this section are reminiscent of the taxonomy previously discussed for *composition* functions (see §2); while in the latter we looked into the composition of *shared* components with *modules*, in this section we provide insights into the composition of multiple *modular* components.

In particular, the aggregation of modular components can (again) be realised on the *parameter* level $f'_i(\boldsymbol{x}) = f_{\phi_i^1 \oplus \ldots \oplus \phi_i^{|\mathcal{F}|}}(\boldsymbol{x})$, *representation* level $f'_i(\boldsymbol{x}) = f_{\theta_i}(\boldsymbol{x}) \oplus f_{\phi_i^1}(\boldsymbol{x}), \ldots, f_{\phi_i^{|\mathcal{F}|}}\boldsymbol{x}])$, as well as *functional* level $f'_i(\boldsymbol{x}) = f_{\phi_i^1} \odot \ldots \odot f_{\phi_i^{|\mathcal{F}|}}(\boldsymbol{x})$. We discuss these different strategies in the following sections.

4.1 Parameter Interpolation

A natural strategy to aggregate information from multiple modules is interpolating their weights. However, given that neural *architectures* differ, and representations might not necessarily be equivalent (e.g. under invariance to invertible linear transforms) even if the model architectures are the same (Kornblith et al., 2019), naively aggregating module weights may have catastrophic consequences. However, recent work on *linear mode connectivity* (Frankle et al., 2020) suggests that under certain conditions, it is in fact possible to interpolate between multiple models, which has positive ramifications for modular aggregation methods. To understand these conditions, we first provide a brief introduction to the constraints for which parameter aggregation is permissible.

4.1.1 Mode connectivity

This phenomenon, where the minima found by two networks are connected by a path of non-increasing error, has been the subject of research for many years (Freeman and Bruna, 2017; Draxler et al., 2018; Garipov et al., 2018; Nagarajan and Kolter, 2019). However, most works demonstrate that mode paths are in fact not linear. While (Nagarajan and Kolter, 2019) find linear paths between networks, their experimental setup requires to initialise models with the same set of weights. Frankle et al. (2020) and Neyshabur et al. (2020) demonstrate that this *linear* mode connectivity phenomenon is closely linked to the *Lottery Ticket Hypothesis* (Frankle and Carbin, 2019), which suggests that only a small subset of randomly initialised weights are the main drivers for the final performance of a model—the so-called *winning tickets* (see section 2.1). When interpolating between models trained on different tasks but initialised with the same set of weights, the models tend to stay in the same loss basin, indicated by the lack of a sudden increase in loss when interpolating the weights. Consequently, a common assumption is that the flatness of the basin of the loss landscape translates to better generalisation capabilities of a model. However, Ainsworth et al. (2022) argue that the success of such interpolation is strongly connected to the inherent bias of the optimiser being used, and not the neural network architecture itself.

4.1.2 Weight Interpolation

Building on the findings of interpolating the weights of models, (Ansell et al., 2022) propose Lottery Ticket Sparse Fine-Tuning (LT-SFT), described in **??**. In particular, they identify language-, and task-specific sub-networks ϕ_l and ϕ_t. For instance, they extract task modules ϕ_t using the weight difference of the parameters of an initial model θ_0 and the weights $\theta^\star$ after fine-tuning on a target task: $\phi_t = \theta^\star - \theta_0$. These can be aggregated by simply adding them to the base model, i.e. $\theta' = \theta_0 + \phi_l + \phi_t$. Instead of identifying task adaptations on subsets of model parameters, Ilharco et al. (2022) propose to edit entire models with further arithmetic operations. For example, tasks can include toxic language generation and general language modelling. By performing the arithmetic negation operation $\theta' = \theta_0 + (\phi_{\text{general}} - \phi_{\text{toxic}})$, their new model $f_{\theta'}(\boldsymbol{x})$ generates less toxic text.[1]

4.2 Representation Interpolation

Closely related to parameter interpolation, representation interpolation consists of interpolating the outputs of individual modules. Crucially, both operations are equivalent if the functions are linear: $(\alpha_i \Phi_i + \alpha_j \Phi_j)\boldsymbol{x} = \alpha_i \Phi_i \boldsymbol{x} + \alpha_j \Phi_j \boldsymbol{x}$. However, this does not hold true for non-linear functions, e.g. if the module is an adapter layer (Houlsby et al., 2019). In particular, at the i-th sub-function of the model, where multiple modules $\phi \in \mathcal{F}_i$ exist, the representations are passed through the (active) modules, outputting $|\mathcal{F}_i|$ (latent) representations $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{|\mathcal{F}_i|}$.

[1] The method has been heavily influenced by the word analogy task (i.e., 'word arithmetics') (Mikolov et al., 2013); vec('King') − vec('Man') + vec('Woman') ≈ vec('Queen'), with $vec(\cdot)$ denoting word embeddings of the respective words.

4.2.1 Weighted Representation Averaging

One way of performing aggregation is to learn the weights α to interpolate over the hidden representations:

$$f_i'(\boldsymbol{x}) = \sum_{j}^{|\mathcal{F}_i|} \alpha_j \boldsymbol{h}_j \tag{4.1}$$

with α_j being a module-specific scalar weighting. This aggregation is equivalent to equation (3.2) when interpreting each weight α_j as the output of a router, i.e. $\alpha_j = r(\phi_j)$, as discussed in section 3.2.3 for MoE routing. Consequently, all soft-learned routing approaches (e.g. MoE) that do not perform top_1 routing (see § 3.2.3) also determine how to aggregate the representations of different modules.

As an extension to the traditional MoE aggregation/routing function, Ma et al. (2018) propose to learn one aggregation function per task t in a multi-task setup. Gururangan et al. (2022) pre-train modular components for different textual domains $d \in \mathcal{D}$. When utilising the pre-trained modules on unseen data, they weight the output representations $\mathbf{h}_d$ of the respective domain modules ϕ_d according to the posterior distribution over the input examples, i.e. $p(\mathcal{D} \mid \mathbf{x})$:

$$f_i'(\boldsymbol{x}) = \sum_{d \in \mathcal{D}} p(d \mid \mathbf{x})\, \mathbf{h}_d^{(i)} \tag{4.2}$$

This posterior is inferred through the Bayes rule. This does not require any auxiliary model, and only relies on the original d-conditioned language model. In fixed routing, module representations are often averaged without weighting (Zhang et al., 2022a; Chronopoulou et al., 2022). Similarly, in hard routing methods, the representations of all *active* modules are averaged, such as in Polytropon (Ponti et al., 2022), or summed, as in PathNet (Fernando et al., 2017).[2]

One disadvantage of simply learning gating parameters is that the weights do not depend on the hidden representations. Thus, they do not take into account their information content. This is taken into account by *attention-based aggregation* functions.

4.2.2 Attention-Based Representation Aggregation

Instead of inferring the weighting before a module has performed its transformation on the latent representation, we can make the aggregation decision *after* we can identify whether or not the information added by the respective module is ancillary to the target task.

In AdapterFusion, Pfeiffer et al. (2021b) propose an attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017) over the stacked hidden representations $\boldsymbol{H}_i$

[2] Note that the latter strategy leads to high variance in the norms of hidden representations if the router can select variable-size subsets of modules.

produced by the modules:

$$f_i(\boldsymbol{x}) = \text{Attn}(\boldsymbol{h}_{i-1}\boldsymbol{Q}_i, \boldsymbol{H}_i\boldsymbol{K}_i, \boldsymbol{H}_i\boldsymbol{V}_i) \tag{4.3}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d \times h}$ are the projections that produce the queries, keys, and values, and $\boldsymbol{h}^{i-1}$ is the input representation *to* each of the modules (i.e., the output representation of the previous layer). $\boldsymbol{H}_i \in \mathbb{R}^{|\mathcal{F}| \times d}$ is a matrix consisting of row-wise stacking of the output representations $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{|\mathcal{F}_i|}$ of each module. In other words, the input of each module is interpreted as the query and the output of each module is interpreted as the value and key. The attention mechanism thus learns to attend over the module representations and weigh them dynamically.

Instead of aggregating module outputs into a single representation, Recurrent Independent Mechanisms (Goyal et al., 2021) keep sequences of representations for different modules separate. However, in between the application of computation functions, they exploit an attention mechanism over hidden representations to enable sparse communication among modules.

One major disadvantage of both *weighted* and *attention-based* representation averaging, is that—in combination with soft routing—they require a full forward pass through all modules, even if they contribute only marginally to the final aggregation. Thus, they incur significant increases in time and space complexity. While this can be mitigated by pruning (i.e., dropping) some modules during inference (Rücklé et al., 2021), latency still remains an issue for scalability.

4.3 Function Composition

Finally, aggregation can be achieved on the function level; $f'_i(\boldsymbol{x}) = f_{\phi_1} \odot f_{\phi_2}(\boldsymbol{x})$. Different aggregation methods infer either a sequence or a (tree) structure that determines the order of the aggregation.

4.3.1 Sequential Aggregation

By performing a forward pass through multiple modules, where the input to the next module is the output of the previous one, the respective hidden representations are sequentially transformed: $f'_i(\boldsymbol{x}) = f_{\phi_1}(f_{\phi_2}(\ldots(f_{\phi_{|\mathcal{F}|}}(\boldsymbol{x}))))$. This form of information aggregation is often chosen in conjunction with *fixed routing*, as discussed in § 3.1, given that the routing order is determined by the role of each module (e.g. language and task adapters).

Pfeiffer et al. (2020b, 2021d) propose a two-stage setup where language-specific components are disentangled from task-specific components, in order to perform zero-shot cross-lingual transfer. First, language (adapter) modules $f_{\phi_{l_s}}$ and $f_{\phi_{l_t}}$ are trained on monolingual unlabelled data for the source language s and the target language t, respectively. Then, in the second stage, the language component $f_{\phi_{l_s}}$ is inserted but frozen, and a new (adapter) module is added for a task f_{ϕ_t} and

trained on annotated data for the source language: $f_{\phi_t}(f_{\phi_{l_s}}(\boldsymbol{x}))$. Since this effectively disentangles language from task information, this also enables zero-shot inference on the target language t without annotated data. In particular, $f_{\phi_{l_s}}$ is substituted with $f_{\phi_{l_t}}$, thereby hierarchically aggregating the information from the respective modular components: $f_{\phi_t}(f_{\phi_{l_t}}(\boldsymbol{x}))$.

Parovic et al. (2022) extend this concept through bilingual modules instead of language-specific ones, which facilitate bridging between the source and target languages. Similarly, Stickland et al. (2021) propose to disentangle l language module f_{ϕ_l} and d domain module f_{ϕ_d} = for multilingual multi-domain machine translation: $f_{\phi_d}(f_{\phi_l}(\boldsymbol{x}))$.

4.3.2 Hierarchical Aggregation

The semantic parse induced by Neural Module Networks (Andreas et al., 2016) provides a graphical structure for module aggregation. While all leaf nodes find objects by identifying regions of an image through attention, intermediate nodes either transform or combine these representations (depending on the arity of the node). Finally, the root predicts the label by describing or measuring the attended objects.

4.4 Hypernetworks

Finally, hypernetworks aggregate information from different 'modules', i.e. task embeddings, by combining them before feeding them as input to the parameter generator. For instance, in (Ponti et al., 2021) task and language embeddings are concatenated in the input when training a multilingual multi-task architecture where the encoder is fully shared and the hypernetwork generates the classifier head. By recombining embeddings appropriately, this method allows for inferring the parameters of unseen task–language combinations. Similar combinations of embeddings have been used to generate adapters in multilingual (Üstün et al., 2020; Ansell et al., 2021b) and multi-task settings (Mahabadi et al., 2021b; Pilault et al., 2021). Contrary to modular adapters, hypernetworks facilitate multi-source language and task training as they allow for soft weight sharing while preventing negative interference. Moreover, task embeddings can straightforwardly encapsulate side information (such as the typological features of a language) in the form of initialisation or regularisers (Ansell et al., 2021b).

4.5 Contributions

In this thesis, we predominantly focused on fixed routing strategies as a means to disentangle knowledge specific information into designated parts of the model. In what follows, we discuss this thesis' contributions to the *aggregation function* axis of modularity, as illustrated in Figure 1.2:

- In Pfeiffer et al. (Chapter 6; 2020a) we provide a general framework for

dynamic aggregation of modular components. Arbitrary architectures can be combined in mixtures where representations can be sequentially aggregated.

- As discussed earlier, in Pfeiffer et al. (Chapter 7; 2021b) we propose an attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017) over the stacked hidden representations $\boldsymbol{H}_i$ produced by the modules:
$f_i(\boldsymbol{x}) = \text{Attn}(\boldsymbol{h}_{i-1}\boldsymbol{Q}_i, \boldsymbol{H}_i\boldsymbol{K}_i, \boldsymbol{H}_i\boldsymbol{V}_i)$
where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d \times h}$ are the projections that produce the queries, keys, and values, and $\boldsymbol{h}^{i-1}$ is the input representation *to* each of the modules (i.e., the output representation of the previous layer). $\boldsymbol{H}_i \in \mathbb{R}^{|\mathcal{F}| \times d}$ is a matrix consisting of row-wise stacking of the output representations $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{|\mathcal{F}_i|}$ of each module. In other words, the input of each module is interpreted as the query and the output of each module is interpreted as the value and key. The attention mechanism thus learns to attend over the module representations and weigh them dynamically.

- As discussed earlier, in Pfeiffer et al. (Chapter 8; 2020b) and Pfeiffer et al. (Chapter 9; 2021c) we propose a two-stage setup where language-specific components are disentangled from task-specific components, in order to perform zero-shot cross-lingual transfer. First, language (adapter) modules $f_{\phi_{l_s}}$ and $f_{\phi_{l_t}}$ are trained on monolingual unlabelled data for the source language s and the target language t, respectively. Then, in the second stage, the language component $f_{\phi_{l_s}}$ is inserted but frozen, and a new (adapter) module is added for a task f_{ϕ_t} and trained on annotated data for the source language: $f_{\phi_t}(f_{\phi_{l_s}}(\boldsymbol{x}))$. Since this effectively disentangles language from task information, this also enables zero-shot inference on the target language t without annotated data. In particular, $f_{\phi_{l_s}}$ is substituted with $f_{\phi_{l_t}}$, thereby hierarchically aggregating the information from the respective modular components: $f_{\phi_t}(f_{\phi_{l_t}}(\boldsymbol{x}))$.

- In Pfeiffer et al. (Chapter 10; 2022a) we extend the former two works by adding a vision modality: We not only perform sequential aggregation through language and task specific components, but additionally through modality specific (i.e. text and vision) components.

Chapter 5

Training Setting

The final concept which influences modularity is the imposed training strategy. We can categories these strategies into methods where **1)** all modules are *jointly trained*, and **2)** where modularity is added *post-hoc*. However, it is important to note that these strategies do not necessarily rule each other out, as they can be combined.

5.1 Joint Training

Many multi-task learning works have proposed to integrate task-specific parameterized components into neural network architectures as means to mitigate catastrophic forgetting or interference (McCloskey and Cohen, 1989; French, 1999). Conceptually, in scenarios where the goal is to train one model on multiple tasks, the idea is to train (sub-)modules that are optimized only on single tasks. Here, a number of works propose to integrate task-specific layers to which examples are routed in a fixed (Hampshire and Waibel, 1992; Rajendran et al., 2017, *inter alia*; see § 3.1 for more details) or learned (Jacobs et al., 1991b,a; Shazeer et al., 2017, *inter alia*; see § 3.2.3 for more details) fashion. While in the past these approaches have predominantly focused on mitigating the aforementioned issues, these methods essentially result in modular models, as task-specific information is stored in 'explicit' parts of the architecture. For instance, (Pfeiffer et al., 2022c) add language-specific layers during multilingual pre-training, and demonstrate that this setup not only mitigates catastrophic interference between languages, but also prepares the model to be extended to more languages later; new modular components are added later in the respective slots for new languages. Similarly, (Alet et al., 2018) learn to select a set of modules for each task via meta-learning. Starting with a simple module structure and network initialisation, they sample structures using simulated annealing (Kirkpatrick et al., 1983) and perform a step of gradient descent on the network parameters for a set of meta-training tasks.

Joint training on a subset of tasks can also be useful to warm-up the base model so that new modules may be more easily added to it (Sun et al., 2020a) and may provide a useful initialisation for modular parameters (Vu et al., 2022).

5.2 Post-Hoc Adaptation

Recently, transfer learning has become the dominating strategy for state-of-the-art results on most tasks. Auxiliary self-supervised objectives are utilised to pre-train models on a large amount of data. Subsequently, the model's weights are fine-tuned on the target tasks, which achieves peak results (Howard and Ruder, 2018; Devlin et al., 2019). Updating a small set of parameters of these large models has been demonstrated to perform equally well as full model fine-tuning, leading to an emergence of parameter-efficient fine-tuning strategies.

Most methods discussed in § 2 that are applied to large pre-trained models can be considered as post-hoc adaptation. Modularity can be achieved through **parameter composition** (§ 2.1) using *sparse subnetworks* (Mehta, 2019; Chen et al., 2020), *structured composition* (Donahue et al., 2014; Cai et al., 2020; Ben Zaken et al., 2022; Guo et al., 2021), or *low-rank composition* (Li et al., 2018; Hu et al., 2022), **input composition** (§ 2.2) by augmenting the function's input (Brown et al., 2020; Li and Liang, 2021), and **function composition** (§ 2.3) through *representation composition* (Rebuffi et al., 2017; Houlsby et al., 2019) and *rescaling* (Liu et al., 2022b). Essentially, all of these methods can be realized as hyper-networks (§ 2.4) and are tightly connected as they adapt the weights of a model pre-trained on a auxiliary task, and only update a very small fraction of the entire model's weights. By only fine-tuning task-specific weights which are encapsulated by freezing weights which are shared across multiple tasks, we essentially create modular and 'aggregatable' components (Pfeiffer et al., 2021b).

5.3 Contributions

In this thesis, we predominantly focused on post-hoc adaptation strategies as a means to disentangle knowledge specific information into designated parts of the model. In what follows, we discuss this thesis' contributions to the *training setting* axis of modularity, as illustrated in Figure 1.2:

- In Pfeiffer et al. (Chapter 6; 2020a) we provide a general framework for post-hoc adaptation. Pre-trained transformer-based models can be extended with different modular and parameter-efficient architectures.

- In Pfeiffer et al. (Chapter 8; 2020b) we propose to sequentially extend a multilingual model with new parameters for each dataset we train on. We stack and freeze the parameters of previous iterations, thereby disentangling the new information from information trained earlier. Here the sequence of datasets is intermediate multilingual training and subsequent task adaptation.

- In Pfeiffer et al. (Chapter 10; 2022a) we extend the former work by adding a vision modality: We freeze and unfreeze modality-specific (i.e. text and vision) components to extend multilingual models to become multimodal, and vise-versa, multimodal models to become multilingual.

- In Pfeiffer et al. (Chapter 11; 2022b) we demonstrate that by integrating language specific capacity during multilingual pre-training we mitigate catastrophic interference. We show that it is possible to extend multilingual models to more languages through post-hoc adaptation; i.e. new modular components are added to the model for new languages later. We demonstrate, that this setup allows the extension to more languages, without any drop in performance in comparison to languages that the model was pre-trained on.

Part II

Publications

Chapter 6

AdapterHub: A Framework for Adapting Transformers

AdapterHub: A Framework for Adapting Transformers

Jonas Pfeiffer[*1], **Andreas Rücklé**[*1], **Clifton Poth**[*1], **Aishwarya Kamath**[2], **Ivan Vulić**[4], **Sebastian Ruder**[5], **Kyunghyun Cho**[2,3], **Iryna Gurevych**[1]
[1]Technical University of Darmstadt
[2]New York University [3]CIFAR Associate Fellow
[4]University of Cambridge
[5]DeepMind
AdapterHub.ml

Abstract

The current modus operandi in NLP involves downloading and fine-tuning pre-trained models consisting of hundreds of millions, or even billions of parameters. Storing and sharing such large trained models is expensive, slow, and time-consuming, which impedes progress towards more general and versatile NLP methods that learn from and for many tasks. Adapters—small learnt bottleneck layers inserted within each layer of a pre-trained model— ameliorate this issue by avoiding full fine-tuning of the entire model. However, sharing and integrating adapter layers is not straightforward. We propose AdapterHub, a framework that allows dynamic "stiching-in" of pre-trained adapters for different tasks and languages. The framework, built on top of the popular HuggingFace Transformers library, enables extremely easy and quick adaptations of state-of-the-art pre-trained models (e.g., BERT, RoBERTa, XLM-R) across tasks and languages. Downloading, sharing, and training adapters is as seamless as possible using minimal changes to the training scripts and a specialized infrastructure. Our framework enables scalable and easy access to sharing of task-specific models, particularly in low-resource scenarios. AdapterHub includes all recent adapter architectures and can be found at AdapterHub.ml.

1 Introduction

Recent advances in NLP leverage transformer-based language models (Vaswani et al., 2017), pre-trained on large amounts of text data (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020). These models are fine-tuned on a target task and achieve state-of-the-art (SotA) performance for most natural language understanding tasks. Their performance has been shown to scale with their size (Kaplan et al., 2020) and recent models have reached billions of parameters (Raffel et al., 2019; Brown et al., 2020). While fine-tuning large pre-trained models on target task data can be done fairly efficiently (Howard and Ruder, 2018), training them for multiple tasks and sharing trained models is often prohibitive. This precludes research on more modular architectures (Shazeer et al., 2017), task composition (Andreas et al., 2016), and injecting biases and external information (e.g., world or linguistic knowledge) into large models (Lauscher et al., 2019; Wang et al., 2020).

Adapters (Houlsby et al., 2019) have been introduced as an alternative lightweight fine-tuning strategy that achieves on-par performance to full fine-tuning (Peters et al., 2019) on most tasks. They consist of a small set of additional newly initialized weights at every layer of the transformer. These weights are then trained during fine-tuning, while the pre-trained parameters of the large model are kept frozen/fixed. This enables efficient parameter sharing between tasks by training many task-specific and language-specific adapters for the same model, which can be exchanged and combined post-hoc. Adapters have recently achieved strong results in multi-task and cross-lingual transfer learning (Pfeiffer et al., 2020a,b).

However, reusing and sharing adapters is not straightforward. Adapters are rarely released individually; their architectures differ in subtle yet important ways, and they are model, task, and language dependent. To mitigate these issues and facilitate transfer learning with adapters in a range of settings, we propose AdapterHub, a framework that enables seamless training and sharing of adapters.

AdapterHub is built on top of the popular `transformers` framework by HuggingFace[1] (Wolf et al., 2020), which provides access to state-of-the-art pre-trained language models. We en-

*Equal contribution.

[1]https://github.com/huggingface/transformers

Proceedings of the 2020 EMNLP (Systems Demonstrations), pages 46–54

hance `transformers` with adapter modules that can be combined with existing SotA models with minimal code edits. We additionally provide a website that enables quick and seamless upload, download, and sharing of pre-trained adapters. AdapterHub is available online at: AdapterHub.ml.

AdapterHub for the first time enables NLP researchers and practitioners to easily and efficiently share and obtain access to models that have been trained for particular tasks, domains, and languages. This opens up the possibility of building on and combining information from many more sources than was previously possible, and makes research such as intermediate task training (Pruksachatkun et al., 2020), composing information from many tasks (Pfeiffer et al., 2020a), and training models for very low-resource languages (Pfeiffer et al., 2020b) much more accessible.

Contributions. 1) We propose an easy-to-use and extensible adapter training and sharing framework for transformer-based models such as BERT, RoBERTa, and XLM(-R); 2) we incorporate it into the HuggingFace `transformers` framework, requiring as little as two additional lines of code to train adapters with existing scripts; 3) our framework automatically extracts the adapter weights, storing them separately to the pre-trained transformer model, requiring as little as 1Mb of storage; 4) we provide an open-source framework and website that allows the community to upload their adapter weights, making them easily accessible with only one additional line of code; 5) we incorporate adapter composition as well as adapter stacking out-of-the-box and pave the way for a wide range of other extensions in the future.

2 Adapters

While the predominant methodology for transfer learning is to fine-tune all weights of the pre-trained model, *adapters* have recently been introduced as an alternative approach, with applications in computer vision (Rebuffi et al., 2017) as well as the NLP domain (Houlsby et al., 2019; Bapna and Firat, 2019; Wang et al., 2020; Pfeiffer et al., 2020a,b).

2.1 Adapter Architecture

Adapters are neural modules with a small amount of additional newly introduced parameters Φ within a large pre-trained model with parameters Θ. The parameters Φ are learnt on a target task while keeping Θ fixed; Φ thus learn to encode task-specific representations in intermediate layers of the pre-trained model. Current work predominantly focuses on training adapters for each task separately (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2020a,b), which enables parallel training and subsequent combination of the weights.

In NLP, adapters have been mainly used within deep transformer-based architectures (Vaswani et al., 2017). At each transformer layer l, a set of adapter parameters Φ_l is introduced. The placement and architecture of adapter parameters Φ within a pre-trained model is non-trivial and may impact their efficacy: Houlsby et al. (2019) experiment with different adapter architectures, empirically validating that a two-layer feed-forward neural network with a bottleneck works well. While this down- and up-projection has largely been agreed upon, the actual placement of adapters within each transformer block, as well as the introduction of new LayerNorms[2] (Ba et al., 2016) varies in the literature (Houlsby et al., 2019; Bapna and Firat, 2019; Stickland and Murray, 2019; Pfeiffer et al., 2020a). In order to support standard adapter architectures from the literature, as well as to enable easy extensibility, AdapterHub provides a configuration file where the architecture settings can be defined dynamically. We illustrate the different configuration possibilities in Figure 3, and describe them in more detail in §3.

2.2 Why Adapters?

Adapters provide numerous benefits over fully fine-tuning a model such as scalability, modularity, and composition. We now provide a few use-cases for adapters to illustrate their usefulness in practice.

Task-specific Layer-wise Representation Learning. Prior to the introduction of adapters, in order to achieve SotA performance on downstream tasks, the entire pre-trained transformer model needs to be fine-tuned (Peters et al., 2019). Adapters have been shown to work on-par with full fine-tuning, by adapting the representations at every layer. We present the results of fully fine-tuning the model compared to two different adapter architectures on the GLUE benchmark (Wang et al., 2018) in Table 1. The adapters of Houlsby et al. (2019, Figure 3c) and Pfeiffer et al. (2020a, Figure 3b) comprise two and one down- and up-projection

[2]Layer normalization learns to normalize the inputs across the features. This is usually done by introducing a new set of features for mean and variance.

	Full	Pfeif.	Houl.
RTE (Wang et al., 2018)	66.2	70.8	69.8
MRPC (Dolan and Brockett, 2005)	90.5	89.7	91.5
STS-B (Cer et al., 2017)	88.8	89.0	89.2
CoLA (Warstadt et al., 2019)	59.5	58.9	59.1
SST-2 (Socher et al., 2013)	92.6	92.2	92.8
QNLI (Rajpurkar et al., 2016)	91.3	91.3	91.2
MNLI (Williams et al., 2018)	84.1	84.1	84.1
QQP (Iyer et al., 2017)	91.4	90.5	90.8

Table 1: Mean development scores over 3 runs on GLUE (Wang et al., 2018) leveraging the BERT-Base pre-trained weights. We present the results with full fine-tuning (**Full**) and with the adapter architectures of Pfeiffer et al. (2020a, **Pfeif.**, Figure 3b) and Houlsby et al. (2019, **Houl.**, Figure 3c) both with bottleneck size 48. We show F1 for MRPC, Spearman rank correlation for STS-B, and accuracy for the rest. RTE is a combination of datasets (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007).

	Base		**Large**	
CRate	#Params	Size	#Params	Size
64	0.2M	0.9Mb	0.8M	3.2Mb
16	0.9M	3.5Mb	3.1M	13Mb
2	7.1M	28Mb	25.2M	97Mb

Table 2: Number of additional parameters and compressed storage space of the adapter of Pfeiffer et al. (2020a) in (Ro)BERT(a)-Base and Large transformer architectures. The adapter of Houlsby et al. (2019) requires roughly twice as much space. *CRate* refers to the adapter's compression rate: e.g., a. rate of 64 means that the adapter's bottleneck layer is 64 times smaller than the underlying model's hidden layer size.

within each transformer layer, respectively. The former adapter thus has more capacity at the cost of training and inference speed. We find that for all settings, there is no large difference in terms of performance between the model architectures, verifying that training adapters is a suitable and lightweight alternative to full fine-tuning in order to achieve SotA performance on downstream tasks.

Small, Scalable, Shareable. Transformer-based models are very deep neural networks with millions or billions of weights and large storage requirements, e.g., around 2.2Gb of compressed storage space is needed for XLM-R Large (Conneau et al., 2020). Fully fine-tuning these models for each task separately requires storing a copy of the fine-tuned model for each task. This impedes both iterating and parallelizing training, particularly in storage-restricted environments.

Adapters mitigate this problem. Depending on the model size and the adapter bottleneck size, a single task requires as little as 0.9Mb storage space. We present the storage requirements in Table 2. This highlights that $> 99\%$ of the parameters required for each target task are fixed during training and can be shared across all models for inference. For instance, for the popular Bert-Base model with a size of 440Mb, storing 2 fully fine-tuned models amounts to the same storage space required by 125 models with adapters, when using a bottleneck size of 48 and adapters of Pfeiffer et al. (2020a). Moreover, when performing inference on a mobile device, adapters can be leveraged to save a significant amount of storage space, while supporting a large number of target tasks. Additionally, due to the small size of the adapter modules—which in many cases do not exceed the file size of an image—new tasks can be added on-the-fly. Overall, these factors make adapters a much more computationally—and ecologically (Strubell et al., 2019)—viable option compared to updating entire models (Rücklé et al., 2020). Easy access to fine-tuned models may also improve reproducibility as researchers will be able to easily rerun and evaluate trained models of previous work.

Modularity of Representations. Adapters learn to encode information of a task within designated parameters. Due to the encapsulated placement of adapters, wherein the surrounding parameters are fixed, at each layer an adapter is forced to learn an output representation compatible with the subsequent layer of the transformer model. This setting allows for modularity of components such that adapters can be stacked on top of each other, or replaced dynamically. In a recent example, Pfeiffer et al. (2020b) successfully combine adapters that have been independently trained for specific tasks and languages. This demonstrates that adapters are modular and that output representations of different adapters are compatible. As NLP tasks become more complex and require knowledge that is not directly accessible in a single monolithic pre-trained model (Ruder et al., 2019), adapters will provide NLP researchers and practitioners with many more sources of relevant information that can be easily combined in an efficient and modular way.

Non-Interfering Composition of Information. Sharing information across tasks has a long-standing history in machine learning (Ruder, 2017). Multi-task learning (MTL), which shares a set of parameters between tasks, has arguably received

the most attention. However, MTL suffers from problems such as catastrophic forgetting where information learned during earlier stages of training is "overwritten" (de Masson d'Autume et al., 2019), catastrophic interference where the performance of a set of tasks deteriorates when adding new tasks (Hashimoto et al., 2017), and intricate task weighting for tasks with different distributions (Sanh et al., 2019).

The encapsulation of adapters forces them to learn output representations that are compatible across tasks. When training adapters on different downstream tasks, they store the respective information in their designated parameters. Multiple adapters can then be combined, e.g., with attention (Pfeiffer et al., 2020a). Because the respective adapters are trained separately, the necessity of sampling heuristics due to skewed data set sizes no longer arises. By separating knowledge extraction and composition, adapters mitigate the two most common pitfalls of multi-task learning, catastrophic forgetting and catastrophic interference.

Overcoming these problems together with the availability of readily available trained task-specific adapters enables researchers and practitioners to leverage information from specific tasks, domains, or languages that is often more relevant for a specific application—rather than more general pre-trained counterparts. Recent work (Howard and Ruder, 2018; Phang et al., 2018; Pruksachatkun et al., 2020; Gururangan et al., 2020) has shown the benefits of such information, which was previously only available by fully fine-tuning a model on the data of interest prior to task-specific fine-tuning.

3 AdapterHub

AdapterHub consists of two core components: **1)** A library built on top of HuggingFace `transformers`, and **2)** a website that dynamically provides analysis and filtering of pre-trained adapters. AdapterHub provides tools for the entire life-cycle of adapters, illustrated in Figure 1 and discussed in what follows: ① introducing new adapter weights Φ into pre-trained transformer weights Θ; ② training adapter weights Φ on a downstream task (while keeping Θ frozen); ③ automatic extraction of the trained adapter weights Φ' and open-sourcing the adapters; ④ automatic visualization of the adapters with configuration filters; ⑤ on-the-fly downloading/caching the pre-trained adapter weights Φ' and stitching the adapter into the pre-trained transformer model Θ; ⑥ performing inference with the trained adapter transformer model.

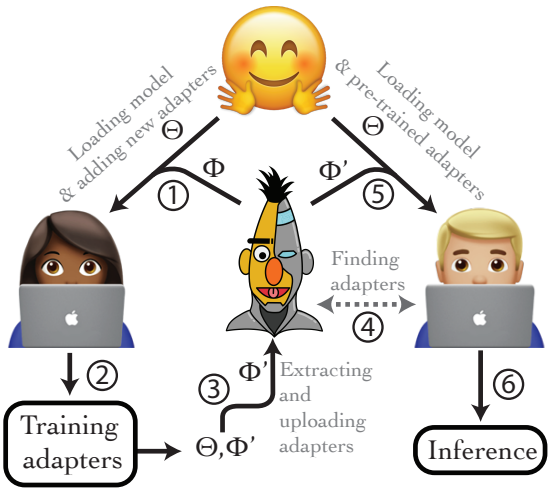


Figure 1: The AdapterHub Process graph. Adapters Φ are introduced into a pre-trained transformer Θ (step ①) and are trained (②). They can then be extracted and open-sourced (③) and visualized (④). Pre-trained adapters are downloaded on-the-fly (⑤) and stitched into a model that is used for inference (⑥).

① Adapters in Transformer Layers

We minimize the required changes to existing HuggingFace training scripts, resulting in only two additional lines of code. In Figure 2 we present the required code to add adapter weights (line 3) and freeze all the transformer weights Θ (line 4). In this example, the model is prepared to train a task adapter on the binary version of the Stanford Sentiment Treebank (SST; Socher et al., 2013) using the adapter architecture of Pfeiffer et al. (2020a). Similarly, language adapters can be added by setting the type parameter to `AdapterType.text_language`, and other adapter architectures can be chosen accordingly.

While we provide ready-made configuration files for well-known architectures in the current literature, adapters are dynamically configurable, which makes it possible to define a multitude of architectures. We illustrate the configurable components as dashed lines and objects in Figure 3. The configurable components are placements of new weights, residual connections as well as placements of LayerNorm layers (Ba et al., 2016).

The code changes within the HuggingFace `transformers` framework are realized through `MixIns`, which are inherited by the respective transformer classes. This minimizes the amount of code changes of our proposed extensions and en-

```
1 from transformers import AutoModelForSequenceClassification, AdapterType
2 model = AutoModelForSequenceClassification.from_pretrained("roberta-base")
3 model.add_adapter("sst-2", AdapterType.text_task, config="pfeiffer")
4 model.train_adapter(["sst-2"])
5 # Train model ...
6 model.save_adapter("adapters/text-task/sst-2/", "sst-2")
7 # Push link to zip file to AdapterHub ...
```

Figure 2: ① Adding new adapter weights Φ to pre-trained RoBERTa-Base weights Θ (line 3), and freezing Θ (line 4). ③ Extracting and storing the trained adapter weights Φ' (line 7).

capsulates adapters as designated classes. It further increases readability as adapters are clearly separated from the main `transformers` code base, which makes it easy to keep both repositories in sync as well as to extend AdapterHub.

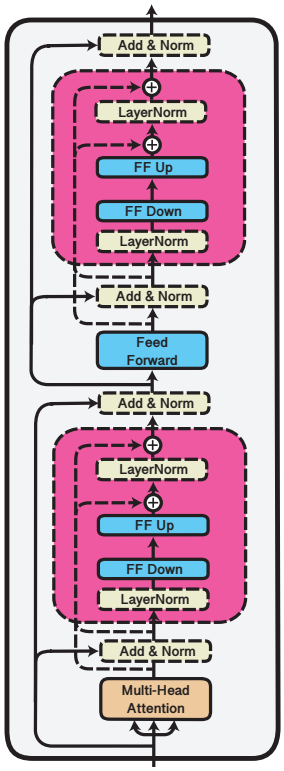
② Training Adapters

Adapters are trained in the same manner as full fine-tuning of the model. The information is passed through the different layers of the transformer where additionally to the pre-trained weights at every layer the representations are additionally passed through the adapter parameters. However, in contrast to full fine-tuning, the pre-trained weights Θ are fixed and only the adapter weights Φ and the prediction head are trained. Because Θ is fixed, the adapter weights Φ are encapsuled within the transformer weights, forcing them to learn compatible representations across tasks.

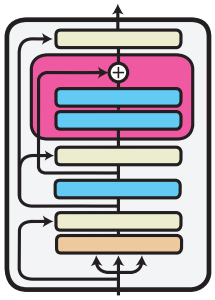
③ Extracting and Open-Sourcing Adapters

When training adapters instead of full fine-tuning, it is no longer necessary to store checkpoints of the entire model. Instead, only the adapter weights Φ', as well as the prediction head need to be stored, as the base model's weights Θ remain the same. This is integrated automatically as soon as adapters are trained, which significantly reduces the required storage space during training and enables storing a large number of checkpoints simultaneously.

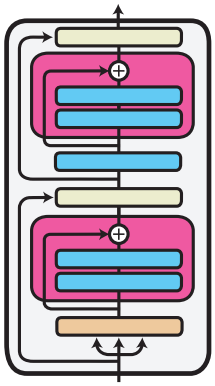
When adapter training has completed, the parameter file together with the corresponding adapter configuration file are zipped and uploaded to a public server. The user then enters the metadata (e.g., URL to weights, user info, description of training procedure, data set used, adapter architecture, GitHub handle, Twitter handle) into a designated YAML file and issues a pull request to the AdapterHub GitHub repository. When all automatic checks pass, the AdapterHub.ml website is automatically regenerated with the newly available adapter, which makes it possible for users to immediately find and use these new weights described by the metadata. We hope that the ease of sharing pre-trained adapters will further facilitate and speed up new developments in transfer learning in NLP.



(a) Configuration Possibilities



(b) Pfeiffer Architecture



(c) Houlsby Architecture

Figure 3: Dynamic customization possibilities where dashed lines in (a) show the current configuration options. These options include the placements of new weights Φ (including down and up projections as well as new LayerNorms), residual connections, bottleneck sizes as well as activation functions. All new weights Φ are illustrated within the pink boxes, everything outside belongs to the pre-trained weights Θ. In addition, we provide pre-set configuration files for architectures in the literature. The resulting configurations for the architecture proposed by Pfeiffer et al. (2020a) and Houlsby et al. (2019) are illustrated in (b) and (c) respectively. We also provide a configuration file for the architecture proposed by Bapna and Firat (2019), not shown here.

```
1 from transformers import AutoModelForSequenceClassification, AdapterType
2 model = AutoModelForSequenceClassification.from_pretrained("roberta-base")
3 model.load_adapter("sst-2", config="pfeiffer")
```

Figure 4: ⑤ After the correct adapter has been identified by the user on the explore page of AdapterHub.ml, they can load and stitch the pre-trained adapter weights Φ' into the transformer Θ (line 3).

④ Finding Pre-Trained Adapters

The website AdapterHub.ml provides a dynamic overview of the currently available pre-trained adapters. Due to the large number of tasks in many different languages as well as different transformer models, we provide an intuitively understandable hierarchical structure, as well as search options. This makes it easy for users to find adapters that are suitable for their use-case. Namely, AdapterHub's explore page is structured into three hierarchical levels. At the *first* level, adapters can be viewed by task or language. The *second* level allows for a more fine-grained distinction separating adapters into data sets of higher-level NLP tasks following a categorization similar to paperswithcode.com. For languages, the second level distinguishes the adapters by the language they were trained on. The *third* level separates adapters into individual datasets or domains such as SST for sentiment analysis or Wikipedia for Swahili.

When a specific dataset has been selected, the user can see the available pre-trained adapters for this setting. Adapters depend on the transformer model they were trained on and are otherwise *not* compatible.[3] The user selects the model architecture and certain hyper-parameters and is shown the compatible adapters. When selecting one of the adapters, the user is provided with additional information about the adapter, which is available in the metadata (see ③ again for more information).

⑤ Stitching-In Pre-Trained Adapters

Pre-trained adapters can be stitched into the large transformer model as easily as adding randomly initialized weights; this requires a single line of code, see Figure 4, line 3. When selecting an adapter on the website (see ④ again) the user is provided with sample code, which corresponds to the configuration necessary to include the specific weights.[4]

[3] We plan to look into mapping adapters between different models as future work.

[4] When selecting an adapter based on a name, we allow for string matching as long as there is no ambiguity.

⑥ Inference with Adapters

Inference with a pre-trained model that relies on adapters is in line with the standard inference practice based on full fine-tuning. Similar to *training* adapters, during inference the active adapter name is passed into the model together with the text tokens. At every transformer layer the information is passed through the transformer layers and the corresponding adapter parameters.

The adapters can be used for inference in the designated task they were trained on. To this end, we provide an option to upload the prediction heads together with the adapter weights. In addition, they can be used for further research such as transferring the adapter to a new task, stacking multiple adapters, fusing the information from diverse adapters, or enriching AdapterHub with adapters for other modalities, among many other possible modes of usage and future directions.

4 Conclusion and Future Work

We have introduced AdapterHub, a novel easy-to-use framework that enables simple and effective transfer learning via training and community sharing of *adapters*. Adapters are small neural modules that can be stitched into large pre-trained transformer models to facilitate, simplify, and speed up transfer learning across a range of languages and tasks. AdapterHub is built on top of the commonly used HuggingFace `transformers`, and it requires only adding as little as two lines of code to existing training scripts. Using adapters in AdapterHub has numerous benefits such as improved reproducibility, much better efficiency compared to full fine-tuning, easy extensibility to new models and new tasks, and easy access to trained models.

With AdapterHub, we hope to provide a suitable and stable framework for the community to train, search, and use adapters. We plan to continuously improve the framework, extend the composition and modularity possibilities, and support other transformer models, even the ones yet to come.

Acknowledgments

Jonas Pfeiffer is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. Andreas Rücklé is supported by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the German Research Foundation under grant EC 503/1-1 and GU 798/21-1. Aishwarya Kamath is supported in part by a DeepMind PhD Fellowship. The work of Ivan Vulić is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). Kyunghyun Cho is supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Research (Improving Deep Learning using Latent Structure).

We would like to thank Isabel Pfeiffer for the illustrations.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1545–1554.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the PASCAL@ACL 2006*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SemEval-2017*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 8440–8451.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL@ACL 2007*.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1923–1933.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs [online]. 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint*.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Specializing unsupervised pretraining models for word-level semantic similarity. *arXiv preprint*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13122–13131.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 7–14.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Virtual Conference*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint*.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5231–5247.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. AdapterDrop: On the Efficiency of Adapters in Transformers. *arXiv preprint*.

Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint*.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Tutorials*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6949–6956.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5986–5995.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi andArt Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Virtual Conference, 2020 Proceedings of EMNLP: Systems Demonstrations*.

Chapter 7

AdapterFusion: Non-Destructive Task Composition for Transfer Learning

AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Jonas Pfeiffer[1], **Aishwarya Kamath**[2], **Andreas Rücklé**[1],
Kyunghyun Cho[2,3], **Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab (UKP Lab), Technical University of Darmstadt
[2]New York University [3]CIFAR Associate Fellow
pfeiffer@ukp.tu-darmstadt.de

Abstract

Sequential fine-tuning and multi-task learning are methods aiming to incorporate knowledge from multiple tasks; however, they suffer from catastrophic forgetting and difficulties in dataset balancing. To address these shortcomings, we propose *AdapterFusion*, a new two stage learning algorithm that leverages knowledge from multiple tasks. First, in the *knowledge extraction* stage we learn task specific parameters called *adapters*, that encapsulate the task-specific information. We then combine the adapters in a separate *knowledge composition* step. We show that by separating the two stages, i.e., knowledge extraction and knowledge composition, the classifier can effectively exploit the representations learned from multiple tasks in a non-destructive manner. We empirically evaluate AdapterFusion on 16 diverse NLU tasks, and find that it effectively combines various types of knowledge at different layers of the model. We show that our approach outperforms traditional strategies such as full fine-tuning as well as multi-task learning. Our code and adapters are available at AdapterHub.ml.

1 Introduction

The most commonly used method for solving NLU tasks is to leverage pretrained models, with the dominant architecture being a transformer (Vaswani et al., 2017), typically trained with a language modelling objective (Devlin et al., 2019; Radford et al., 2018; Liu et al., 2019b). Transfer to a task of interest is achieved by fine-tuning all the weights of the pretrained model on that *single task*, often yielding state-of-the-art results (Zhang and Yang, 2017; Ruder, 2017; Howard and Ruder, 2018; Peters et al., 2019). However, each task of interest requires all the parameters of the network to be fine-tuned, which results in a specialized model for each task.

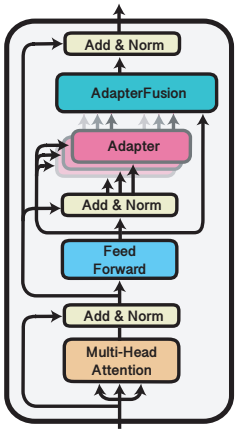


Figure 1: AdapterFusion architecture inside a transformer (Vaswani et al., 2017). The AdapterFusion component takes as input the representations of multiple adapters trained on different tasks and learns a parameterized mixer of the encoded information.

There are two approaches for sharing information across multiple tasks. The first consists of starting from the pretrained language model and sequentially fine-tuning on each of the tasks one by one (Phang et al., 2018). However, as we subsequently fine-tune the model weights on new tasks, the problem of catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999) can arise, which results in loss of knowledge already learned from all previous tasks. This, together with the non-trivial decision of the order of tasks in which to fine-tune the model, hinders the effective transfer of knowledge. Multi-task learning (Caruana, 1997; Zhang and Yang, 2017; Liu et al., 2019a) is another approach for sharing information across multiple tasks. This involves fine-tuning the weights of a pretrained language model using a weighted sum of the objective function of each target task simultaneously. Using this approach, the network captures the common structure underlying all the target tasks. However, multi-task learning requires simul-

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 487–503
April 19 - 23, 2021.

taneous access to all tasks during training. Adding new tasks thus requires complete joint retraining. Further, it is difficult to balance multiple tasks and train a model that solves each task equally well. As has been shown in Lee et al. (2017), these models often overfit on low resource tasks and underfit on high resource tasks. This makes it difficult to effectively transfer knowledge across tasks with all the tasks being solved equally well (Pfeiffer et al., 2020b), thus considerably limiting the applicability of multi-task learning in many scenarios.

Recently, *adapters* (Rebuffi et al., 2017; Houlsby et al., 2019) have emerged as an alternative training strategy. Adapters do not require fine-tuning of all parameters of the pretrained model, and instead introduce a small number of task specific parameters — while keeping the underlying pretrained language model fixed. Thus, we can separately and simultaneously train adapters for multiple tasks, which all share the same underlying pretrained parameters. However, to date, there exists no method for using *multiple* adapters to maximize the transfer of knowledge across tasks without suffering from the same problems as sequential fine-tuning and multi-task learning. For instance, Stickland and Murray (2019) propose a multi-task approach for training adapters, which still suffers from the difficulty of balancing the various target tasks and requiring simultaneous access to all target tasks.

In this paper we address these limitations and propose a new variant of adapters called *AdapterFusion*. We further propose a novel two stage learning algorithm that allows us to effectively share knowledge across multiple tasks while avoiding the issues of catastrophic forgetting and balancing of different tasks. Our AdapterFusion architecture, illustrated in Figure 1, has two components. The first component is an adapter trained on a task without changing the weights of the underlying language model. The second component — our novel Fusion layer — combines the representations from several such task adapters in order to improve the performance on the target task.

Contributions Our main contributions are: (1) We introduce a novel two-stage transfer learning strategy, termed *AdapterFusion*, which combines the knowledge from multiple source tasks to perform better on a target task. (2) We empirically evaluate our proposed approach on a set of 16 diverse NLU tasks such as sentiment analysis, commonsense reasoning, paraphrase detection, and recognizing textual entailment. (3) We compare our approach with Stickland and Murray (2019) where adapters are trained for all tasks in a multi-task manner, finding that AdapterFusion is able to improve this method, even though the model has simultaneous access to all tasks during pretraining. (4) We show that our proposed approach outperforms fully fine-tuning the transformer model on a single target task. Our approach additionally outperforms adapter based models trained both in a Single-Task, as well as Multi-Task setup.

The code of this work is integrated into the AdapterHub.ml (Pfeiffer et al., 2020a).

2 Background

In this section, we formalize our goal of transfer learning (Pan and Yang, 2010; Torrey and Shavlik, 2010; Ruder, 2019), highlight its key challenges, and provide a brief overview of common methods that can be used to address them. This is followed by an introduction to *adapters* (Rebuffi et al., 2017) and a brief formalism of the two approaches to training adapters.

Task Definition. We are given a model that is pretrained on a task with training data D_0 and a loss function L_0. The weights Θ_0 of this model are learned as follows:

$$
\begin{aligned}
D_0 &:= \text{Large corpus of unlabelled text} \\
L_0 &:= \text{Masked language modelling loss} \\
\Theta_0 &\leftarrow \underset{\Theta}{\operatorname{argmin}}\ L_0(D_0; \Theta)
\end{aligned}
$$

In the remainder of this paper, we refer to this pretrained model by the tuple (D_0, L_0).

We define C as the set of N classification tasks having labelled data of varying sizes and different loss functions:

$$C = \{(D_1, L_1), \ldots, (D_N, L_N)\}$$

The aim is to be able to leverage a set of N tasks to improve on a target task m with $C_m = (D_m, L_m)$. In this work we focus on the setting where $m \in \{1, \ldots, N\}$.

Desiderata. We wish to learn a parameterization Θ_m that is defined as follows:

$$\Theta_m \leftarrow \underset{\Theta'}{\operatorname{argmin}}\ L_m(D_m; \Theta')$$

where Θ' is expected to have encapsulated relevant information from all the N tasks. The target model

for task m is initialized with Θ' for which we learn the optimal parameters Θ_m through minimizing the task's loss on its training data.

2.1 Current Approaches to Transfer Learning

There are two predominant approaches to achieve sharing of information from one task to another.

2.1.1 Sequential Fine-Tuning

This involves sequentially updating all the weights of the model on each task. For a set of N tasks, the order of fine-tuning is defined and at each step the model is initialized with the parameters learned through the previous step. However, this approach does not perform well beyond two sequential tasks (Phang et al., 2018; Pruksachatkun et al., 2020) due to catastrophic forgetting.

2.1.2 Multi-Task Learning (MTL)

All tasks are trained simultaneously with the aim of learning a shared representation that will enable the model to generalize better on each task (Caruana, 1997; Collobert and Weston, 2008; Nam et al., 2014; Liu et al., 2016, 2017; Zhang and Yang, 2017; Ruder, 2017; Ruder et al., 2019; Sanh et al., 2019; Pfeiffer et al., 2020b, *inter alia*).

$$\Theta_{0\rightarrow\{1,...,N\}} \leftarrow \underset{\Theta}{\operatorname{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0) \right)$$

Where $\Theta_{0\rightarrow\{1,...,N\}}$ indicates that we start with Θ_0 and fine-tune on a set of tasks $\{1, ..., N\}$.

However, MTL requires simultaneous access to all tasks, making it difficult to add more tasks on the fly. As the different tasks have varying sizes as well as loss functions, effectively combining them during training is very challenging and requires heuristic approaches as proposed in Stickland and Murray (2019).

2.2 Adapters

While the predominant methodology for transfer learning is to fine-tune all weights of the pretrained model, *adapters* (Houlsby et al., 2019) have recently been introduced as an alternative approach with applications in domain transfer (Rücklé et al., 2020b), machine translation (Bapna and Firat, 2019; Philip et al., 2020) transfer learning (Stickland and Murray, 2019; Wang et al., 2020; Lauscher et al., 2020), and cross-lingual transfer (Pfeiffer et al., 2020c,d; Üstün et al., 2020; Vidoni et al., 2020). Adapters share a large set of parameters Θ across all tasks and introduce a small number of task-specific parameters Φ_n. While Θ represents the weights of a pretrained model (e.g., a transformer), the parameters Φ_n, where $n \in \{1, \ldots, N\}$, are used to encode task-specific representations in intermediate layers of the shared model. Current work on adapters focuses either on training adapters for each task separately (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2020a) or training them in a multi-task setting to leverage shared representations (Stickland and Murray, 2019). We discuss both variants below.

2.2.1 Single-Task Adapters (ST-A)

For each of the N tasks, the model is initialized with parameters Θ_0. In addition, a set of new and randomly initialized adapter parameters Φ_n are introduced.

The parameters Θ_0 are fixed and only the parameters Φ_n are trained. This makes it possible to efficiently parallelize the training of adapters for all N tasks, and store the corresponding knowledge in designated parts of the model. The objective for each task $n \in \{1, \ldots, N\}$ is of the form:

$$\Phi_n \leftarrow \underset{\Phi}{\operatorname{argmin}}\ L_n(D_n; \Theta_0, \Phi)$$

For common adapter architectures, Φ contains considerably fewer parameters than Θ, e.g., only 3.6% of the parameters of the pretrained model in Houlsby et al. (2019).

2.2.2 Multi-Task Adapters (MT-A)

Stickland and Murray (2019) propose to train adapters for N tasks in parallel with a multi-task objective. The underlying parameters Θ_0 are fine-tuned along with the task-specific parameters in Φ_n. The training objective can be defined as:

$$\mathbf{\Theta} \leftarrow \underset{\Theta,\Phi}{\operatorname{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0, \Phi_n) \right)$$

where

$$\mathbf{\Theta} = \Theta_{0\rightarrow\{1,...,N\}}, \Phi_1, \ldots, \Phi_N.$$

2.2.3 Adapters in Practice

Introducing new adapter parameters in different layers of an otherwise fixed pretrained model has been shown to perform on-par with, or only slightly below, full model fine-tuning (Houlsby et al., 2019; Stickland and Murray, 2019; Pfeiffer et al., 2020a).

For NLP tasks, adapters have been introduced for the transformer architecture (Vaswani et al., 2017). At each transformer layer l, a set of adapter parameters Φ_l is introduced. The placement and architecture of adapter parameters Φ within a pretrained model is non-trivial. Houlsby et al. (2019) experiment with different architectures, finding that a two-layer feed-foward neural network with a bottleneck works well. They place two of these components within one layer, one after the multi-head attention (further referred to as *bottom*) and one after the feed-forward layers of the transformer (further referred to as *top*).[1] Bapna and Firat (2019) and Stickland and Murray (2019) only introduce one of these components at the *top* position, however, Bapna and Firat (2019) include an additional *layer norm* (Ba et al., 2016).

Adapters trained in both single-task (ST-A) or multi-task (MT-A) setups have learned the idiosyncratic knowledge of the respective tasks' training data, encapsulated in their designated parameters. This results in a compression of information, which requires less space to store task-specific knowledge. However, the distinct weights of adapters prevent a downstream task from being able to use multiple sources of extracted information. In the next section we describe our two stage algorithm which tackles the sharing of information stored in adapters trained on different tasks.

3 AdapterFusion

Adapters avoid catastrophic forgetting by introducing task-specific parameters; however, current adapter approaches do not allow sharing of information between tasks. To mitigate this we propose AdapterFusion.

3.1 Learning algorithm

In the first stage of our learning algorithm, we train either ST-A or MT-A for each of the N tasks.

In the second stage, we then combine the set of N adapters by using AdapterFusion. While fixing both the parameters Θ as well as all adapters Φ, we introduce parameters Ψ that learn to combine the N task adapters to solve the target task.

$$\Psi_m \leftarrow \underset{\Psi}{\operatorname{argmin}}\ L_m(D_m; \Theta, \Phi_1, \ldots, \Phi_N, \Psi)$$

Ψ_m are the newly learned AdapterFusion parameters for task m. Θ refers to Θ_0 in the ST-A setting or $\Theta_{0\rightarrow\{1,\ldots,N,m\}}$ in the MT-A setup. In our experiments we focus on the setting where $m \in \{1, ..., N\}$, which means that the training dataset of m is used twice: *once* for training the adapters Φ_m and *again* for training Fusion parameters Ψ_m, which learn to compose the information stored in the N task adapters.

By separating the two stages — knowledge extraction in the adapters, and knowledge composition with AdapterFusion — we address the issues of catastrophic forgetting, interference between tasks and training instabilities.

[1] We illustrate these placements in Appendix Figure 5 (left).

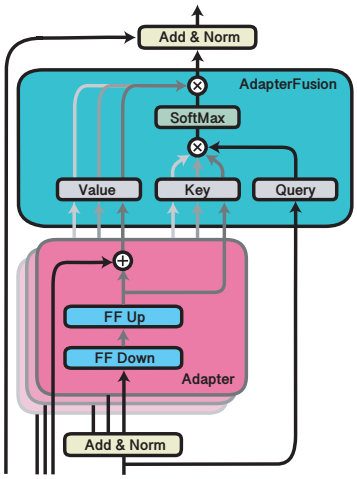


Figure 2: Our AdapterFusion architecture. This includes learnable weights *Query*, *Key*, and *Value*. *Query* takes as input the output of the pretrained transformer weights. Both *Key* and *Value* take as input the output of the respective adapters. The dot product of the *query* with all the *keys* is passed into a softmax function, which learns to weight the adapters with respect to the context.

3.2 Components

AdapterFusion learns to compose the N task adapters Φ_n and the shared pretrained model Θ, by introducing a new set of weights Ψ. These parameters learn to combine the adapters as a dynamic function of the target task data.

As illustrated in Figure 2, we define the AdapterFusion parameters Ψ to consist of *Key, Value* and *Query* matrices at each layer l, denoted by $\mathbf{K}_l$, $\mathbf{V}_l$ and $\mathbf{Q}_l$ respectively. At *each layer* l of the transformer and *each time-step* t, the output of the feed-forward sub-layer of layer l is taken as the query vector. The output of each adapter $\mathbf{z}_{l,t}$ is used as input to both the *value* and *key* transformations. Similar to attention (Bahdanau et al., 2015; Vaswani et al., 2017), we learn a contextual activation of

each adapter n using

$$\mathbf{s}_{l,t} = \text{softmax}(\mathbf{h}_{l,t}^{\top}\mathbf{Q}_l \otimes \mathbf{z}_{l,t,n}^{\top}\mathbf{K}_l), n \in \{1, ..., N\}$$
$$\mathbf{z}'_{l,t,n} = \mathbf{z}_{l,t,n}^{\top}\mathbf{V}_l, n \in \{1, ..., N\}$$
$$\mathbf{Z}'_{l,t} = [\mathbf{z}'_{l,t,0}, ..., \mathbf{z}'_{l,t,N}]$$
$$\mathbf{o}_{l,t} = \mathbf{s}_{l,t}^{\top}\mathbf{Z}'_{l,t}$$

Where $\otimes$ represents the dot product and $[\cdot, \cdot]$ indicates the concatenation of vectors.

Given the context, AdapterFusion learns a parameterized mixer of the available trained adapters. It learns to identify and activate the most useful adapter for a given input.

4 Experiments

In this section we evaluate how effective AdapterFusion is in overcoming the issues faced by other transfer learning methods. We provide a brief description of the 16 diverse datasets that we use for our study, each of which uses accuracy as the scoring metric.

4.1 Experimental Setup

In order to investigate our model's ability to overcome catastrophic forgetting, we compare Fusion using ST-A to only the ST-A for the task. We also compare Fusion using ST-A to MT-A for the task to test whether our two-stage procedure alleviates the problems of interference between tasks. Finally, our experiments to compare MT-A with and without Fusion let us investigate the versatility of our approach. Gains in this setting would show that AdapterFusion is useful even when the base adapters have already been trained jointly.

In all experiments, we use BERT-base-uncased (Devlin et al., 2019) as the pretrained language model. We train ST-A, described in Appendix A.2 and illustrated in Figure 5, for all datasets described in §4.2. We train them with reduction factors[2] $\{2, 16, 64\}$ and learning rate 0.0001 with AdamW and a linear learning rate decay. We train for a maximum of 30 epochs with early stopping. We follow the setup used in Stickland and Murray (2019) for training the MT-A. We use the default hyperparameters[3], and train a MT-A model on all datasets simultaneously.

For AdapterFusion, we empirically find that a learning rate of $5e-5$ works well, and use this in all experiments.[4] We train for a maximum of 10 epochs with early stopping. While we initialize $\mathbf{Q}$ and $\mathbf{K}$ randomly, we initialize $\mathbf{V}$ with a diagonal of ones and the rest of the matrix with random weights having a small norm ($1e-6$). Multiplying the adapter output with this value matrix $\mathbf{V}$ initially adds small amounts of noise, but retains the overall representation. We continue to regularize the *Value* matrix using l_2-norm to avoid introducing additional capacity.

4.2 Tasks and Datasets

We briefly summarize the different types of *tasks* that we include in our experiments, and reference the related datasets accordingly. A detailed descriptions can be found in Appendix A.1.

Commonsense reasoning is used to gauge whether the model can perform basic reasoning skills: *Hellaswag* (Zellers et al., 2018, 2019), *Winogrande* (Sakaguchi et al., 2020), *CosmosQA* (Huang et al., 2019), *CSQA* (Talmor et al., 2019), *SocialIQA* (Sap et al., 2019). **Sentiment analysis** predicts whether a given text has a positive or negative sentiment: *IMDb* (Maas et al., 2011), *SST* (Socher et al., 2013). **Natural language inference** predicts whether one sentence entails, contradicts, or is neutral to another: *MNLI* (Williams et al., 2018), *SciTail* (Khot et al., 2018), *SICK* (Marelli et al., 2014), *RTE* (as combined by Wang et al. (2018)), *CB* (De Marneffe et al., 2019). **Sentence relatedness** captures whether two sentences include similar content: *MRPC* (Dolan and Brockett, 2005), *QQP*[5]. We also use an argument mining *Argument* (Stab et al., 2018) and reading comprehension *BoolQ* (Clark et al., 2019) dataset.

5 Results

We present results for all 16 datasets in Table 1. For reference, we also include the adapter architecture of Houlsby et al. (2019), ST-A$^{\text{Houlsby}}$, which has twice as many parameters compared to ST-A. To provide a fair comparison to Stickland and Murray (2019) we primarily experiment with BERT-base-uncased. We additionally validate our best model configurations — ST-A and Fusion with ST-A — with RoBERTa-base, for which we present our results in Appendix Table 4.

[2] A reduction factor indicates the factor by which the hidden size is reduced such that the bottle-neck size for BERT Base with factor 64 is reduced to 12 ($768/64 = 12$).

[3] We additionally test out batch sizes 16 and 32.

[4] We have experimented with learning rates $\{6e-6, 5e-5, 1e-4, 2e-4\}$

[5] data.quora.com/First-Quora-DatasetReleaseQuestion-Pairs

Dataset	Head	Full	ST-A	MT-A	F. w/ ST-A	F. w/ MT-A	ST-A$^{\text{Houlsby}}$
MNLI	54.59	84.10	**84.32**	82.49 ±0.49	84.28	83.05	84.13
QQP	76.79	**90.87**	90.59	89.47 ±0.60	90.71	90.58	90.63
SST	85.17 ±0.45	92.39 ±0.22	91.85 ±0.41	92.27 ±0.71	92.20 ±0.18	**93.00** ±0.20	92.75 ±0.37
WGrande	51.92 ±0.35	60.01 ±0.08	**61.09** ±0.11	57.70 ±1.40	60.23 ±0.31	59.32 ±0.30	59.32 ±1.33
IMDB	85.05 ±0.22	**94.05** ±0.21	93.85 ±0.07	92.56 ±0.54	93.82 ±0.39	92.66 ±0.32	93.96 ±0.22
HSwag	34.17 ±0.27	**39.25** ±0.76	38.11 ±0.14	36.47 ±0.98	37.98 ±0.01	37.36 ±0.10	38.65 ±0.25
SocIQA	50.33 ±2.50	62.05 ±0.04	62.41 ±0.11	61.21 ±0.89	**63.16** ±0.24	62.56 ±0.10	62.73 ±0.53
CosQA	50.06 ±0.51	60.28 ±0.40	60.01 ±0.02	61.25 ±0.90	60.65 ±0.55	**62.78** ±0.07	61.37 ±0.35
SciTail	85.30 ±2.44	94.32 ±0.11	93.90 ±0.16	94.53 ±0.43	94.04 ±0.23	**94.79** ±0.17	94.07 ±0.39
Argument	70.61 ±0.59	76.87 ±0.32	**77.65** ±0.34	75.70 ±0.60	**77.65** ±0.21	76.08 ±0.27	77.44 ±0.62
CSQA	41.09 ±0.27	58.88 ±0.40	58.91 ±0.57	53.30 ±2.19	59.73 ±0.54	56.73 ±0.14	**60.05** ±0.36
BoolQ	63.07 ±1.27	74.84 ±0.24	75.66 ±1.25	78.76 ±0.76	76.25 ±0.19	**79.18** ±0.45	76.02 ±1.13
MRPC	71.91 ±0.13	85.14 ±0.45	85.16 ±0.52	81.86 ±0.99	**90.29** ±0.84	84.68 ±0.32	86.66 ±0.81
SICK	76.30 ±0.71	87.30 ±0.42	86.20 ±0.00	88.61 ±1.06	87.28 ±0.99	**90.43** ±0.30	86.12 ±0.54
RTE	61.37 ±1.17	65.41 ±0.90	71.04 ±1.62	77.61 ±3.21	76.82 ±1.68	**79.96** ±0.76	69.67 ±1.96
CB	68.93 ±4.82	82.49 ±2.33	86.07 ±3.87	89.09 ±1.15	**92.14** ±0.97	89.81 ±0.99	87.50 ±4.72
Mean	64.17	75.51	76.05	75.80	**77.33**	77.06	76.32

Table 1: Mean and standard deviation results (development sets) for each of the 16 datasets and the different architectural setups. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separate datasizes $\{> 40k, > 10k, > 5k\}$, respectively. Each model is initialized with BERT-base (Devlin et al., 2019) weights. **Head** indicates training only a classification head on top of fixed BERT weights. For **Full** training we fine-tune all weights of BERT. Single-Task Adapters (**ST-A**) is the training of independently trained adapters for each task, using the architecture illustrated in Figure 5. Multi-Task Adapters (**MT-A**) shows results of jointly trained adapters using the default settings of Stickland and Murray (2019). **Fusion w/ ST-A** and **Fusion w/ MT-A** show the results of AdapterFusion using the respective pre-trained Adapters. **ST-A$^{\text{Houlsby}}$** shows the results of ST-Adapters with the architecture proposed by Houlsby et al. (2019). Reported results are accuracy scores.

5.1 Adapters

Training only a prediction-head on the output of a pretrained model can also be considered an adapter. This procedure, commonly referred to as training only the *Head*, performs considerably worse than fine-tuning all weights (Howard and Ruder, 2018; Peters et al., 2019). We show that the performance of only fine-tuning the *Head* compared to *Full* fine-tuning causes on average a drop of 10 points in accuracy. This demonstrates the need for more complex adaptation approaches.

In Table 1 we show the results for MT-A and ST-A with a reduction factor 16 (see the appendix Table 3 for more results) which we find has a good trade-off between the number of newly introduced parameters and the task performance. Interestingly, the ST-A have a regularization effect on some datasets, resulting in better performance on average for certain tasks, even though a much small proportion of weights is trained. On average, we improve 0.66% by training ST-A instead of the *Full* model.

For MT-A we find that there are considerable performance drops of more than 2% for *CSQA* and *MRPC*, despite the heuristic strategies for sampling from the different datasets (Stickland and Murray, 2019). This indicates that these heuristics only partially address common problems of multi-task learning such as catastrophic interference. It also shows that learning a shared representation jointly does not guarantee the best results for all tasks. On average, however, we do see a performance increase of 0.4% using MT-A over *Full* fine-tuning on each task separately, which demonstrates that there are advantages in leveraging information from other tasks with multi-task learning.

5.2 AdapterFusion

AdapterFusion aims to improve performance on a given target task m by transferring task specific knowledge from the set of all N task adapters, where $m \in \{1, \ldots, \text{N}\}$. We hypothesize that if there exists at least one task that supports the target task, AdapterFusion should lead to performance gains. If no such task exists, then the performance should remain the same.

Dependence on the size of training data. In Table 1 we notice that having access to relevant tasks considerably improves the performance for the target task when using AdapterFusion. While datasets with more than 40k training instances perform well without Fusion, smaller datasets with fewer training instances benefit more from our approach. We

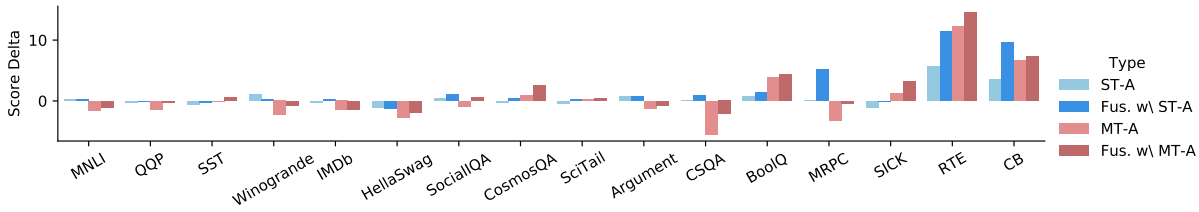


Figure 3: Relative performance difference of the two adapter architectures and the AdapterFusion models over fully fine-tuned BERT. Fusion improves over its corresponding adapters (ST-A and MT-A) for most tasks.

	Fus. w/ ST-A		**Fus. w/ MT-A**	
compared to	ST-A	MT-A	ST-A	MT-A
MNLI	→	↗	↘	↗
QQP	→	↗	→	↗
SST	↗	→	↗	↗
Winogrande	↘	↗	↘	↗
IMDB	↗	↗	↘	→
HellaSwag	→	↗	↘	↗
SocialIQA	↗	↗	→	↗
CosmosQA	↗	↘	↗	↗
SciTail	→	↗	↗	→
Argument	→	↗	↘	↗
CSQA	↗	↗	↘	↗
BoolQ	↗	↘	↗	↗
MRPC	↗	↗	↘	↗
SICK	↗	↘	↗	↗
RTE	↗	↘	↗	↗
CB	↗	↗	↗	↗
Improved	10/16	11/16	7/16	14/16

Table 2: Performance changes of AdapterFusion compared to ST-A and MT-A. Arrows indicate whether there has been an improvement ↗ (> 0.3), decrease ↘ (< -0.3), or whether the results have stayed the same → $[-0.3, 0.3]$.

observe particularly large performance gains for datasets with less than 5k training instances. For example, Fusion with ST-A achieves substantial improvements of 6.5 % for *RTE* and 5.64 % for *MRPC*. In addition, we also see performance gains for moderately sized datasets such as the commonsense tasks *CosmosQA* and *CSQA*. Fusion with MT-A achieves smaller improvements, as the model already includes a shared set of parameters. However, we do see performance gains for *SICK*, *SocialIQA*, *Winogrande* and *MRPC*. On average, we observe improvements of 1.27% and 1.25% when using Fusion with ST-A and MT-A, respectively.

Mitigating catastrophic interference. In order to identify whether our approach is able to mitigate problems faced by multi-task learning, we present the performance differences of adapters and AdapterFusion compared to the fully fine-tuned model in Figure 3. In Table 2, we compare AdapterFusion to ST-A and MT-A. The arrows indicate whether there is an improvement ↗, decrease ↘, or if the the results remain the same →. We compare the performance of both, Fusion with ST-A and Fusion with MT-A, to ST-A and MT-A. We summarize our four most important findings below.

(1) In the case of Fusion with ST-A, for $15/16$ tasks, the performance remains the same or improves as compared to the task's pretrained adapter. For $10/16$ tasks we see performance gains. This shows that having access to adapters from other tasks is beneficial and in the majority of cases leads to better results on the target task. **(2)** We find that for $11/16$ tasks, Fusion with ST-A improves the performance compared to MT-A. This demonstrates the ability of Fusion with ST-A to share information between tasks while avoiding the interference that multi-task training suffers from. **(3)** For only $7/16$ tasks, we see an improvement of Fusion with MT-A over the ST-A. Training of MT-A in the first stage of our algorithm suffers from all the problems of multi-task learning and results in less effective adapters than our ST-A on average. Fusion helps bridge some of this gap but is not able to mitigate the entire performance drop. **(4)** In the case of AdapterFusion with MT-A, we see that the performances on *all 16 tasks* improves or stays the same. This demonstrates that AdapterFusion can successfully combine the specific adapter weights, even if the adapters were trained in a multi-task setting, confirming that our method is versatile.

Summary. Our findings demonstrate that Fusion with ST-A is the most promising approach to sharing information across tasks. Our approach allows us to train adapters in parallel and it requires no heuristic sampling strategies to deal with imbalanced datasets. It also allows researchers to easily add more tasks as they become available, without requiring complete model retraining.

While Fusion with MT-A does provide gains over simply using MT-A, the effort required to train

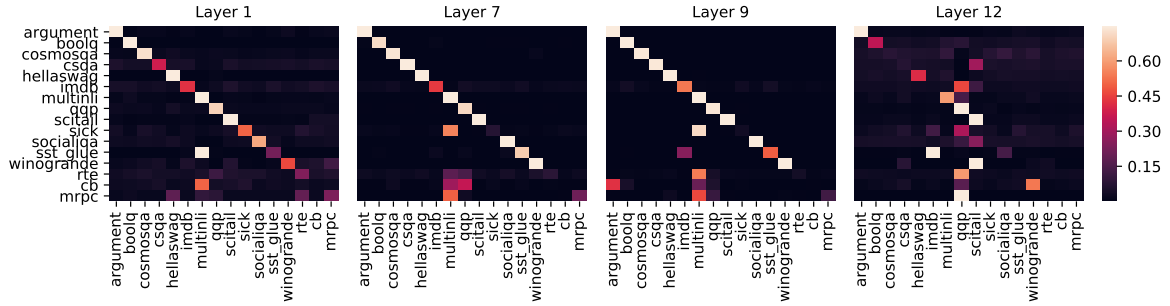


Figure 4: AdapterFusion activations of pretrained **ST-Adapters**. Rows indicate the target task m, columns indicate adapters n. We assume that the softmax activation for $\Phi_{n,l}$ is high if the information of adapter n is useful for task m. For our analysis, we calculate the softmax activation for each adapter $\Phi_{n,l}$, where $n \in \{1, \ldots, N\}$, and average over all activations within the same layer l calculated over all instances in the development set.

these in a multi-task setting followed by the Fusion step are not warranted by the limited gains in performance. On the other hand, we find that Fusion with ST-A is an efficient and versatile approach to transfer learning.

6 Analysis of Fusion Activation

We analyze the weighting patterns that are learned by AdapterFusion to better understand which tasks impact the model predictions, and whether there exist differences across BERT layers.

We plot the results for layers 1, 7, 9, and 12 and ST-A in Figure 4 (see Appendix Figure 6 for the remaining layers). We find that tasks which do not benefit from AdapterFusion tend to more strongly activate their own adapter at every layer (e.g. *Argument, HellaSwag, MNLI, QQP, SciTail*). This confirms that AdapterFusion only extracts information from adapters if they are beneficial for the target task m. We further find that *MNLI* is a useful intermediate task that benefits a large number of target tasks, e.g. *BoolQ, SICK, CSQA, SST-2, CB, MRPC, RTE*, which is in line with previous work (Phang et al., 2018; Conneau and Kiela, 2018; Reimers and Gurevych, 2019). Similarly, *QQP* is utilized by a large number of tasks, e.g. *SICK, IMDB, RTE, CB, MRPC, SST-2*. Most importantly, tasks with small datasets such as *CB, RTE,* and *MRPC* often strongly rely on adapters trained on large datasets such as *MNLI* and *QQP*.

Interestingly, we find that the activations in layer 12 are considerably more distributed across multiple tasks than adapters in earlier layers. The potential reason for this is that the last adapters are not encapsulated between frozen pretrained layers, and can thus be considered as an extension of the prediction head. The representations of the adapters in the 12$^{\text{th}}$ layer might thus not be as comparable, resulting in more distributed activations. This is in line with Pfeiffer et al. (2020d) who are able to improve zero-shot cross-lingual performance considerably by dropping the adapters in the last layer.

7 Contemporary Work

In contemporaneous work, other approaches for parameter efficient fine-tuning have been proposed. Guo et al. (2020) train sparse "diff" vectors which are applied on top of pretrained frozen parameter vectors. Ravfogel and Goldberg (2021) only fine-tune bias terms of the pretrained language models, achieving similar results as full model fine-tuning. Li and Liang (2021) propose prefix-tuning for natural language generation tasks. Here, continuous task-specific vectors are trained while the remaining model is kept frozen. These alternative, parameter-efficient fine-tuning strategies all encapsulate the idiosyncratic task-specific information in designated parameters, creating the potential for new composition approaches of multiple tasks.

Rücklé et al. (2020a) analyse the training and inference efficiency of adapters and AdapterFusion. For AdapterFusion, they find that adding more tasks to the set of adapters results in a linear increase of computational cost, both for training and inference. They further propose approaches to mitigate this overhead.

8 Conclusion and Outlook

8.1 Conclusion

We propose a novel approach to transfer learning called AdapterFusion which provides a simple and effective way to combine information from several

tasks. By separating the extraction of knowledge from its composition, we are able to effectively avoid the common pitfalls of multi-task learning, such as catastrophic forgetting and interference between tasks. Further, AdapterFusion mitigates the problem of traditional multi-task learning in which complete re-training is required, when new tasks are added to the pool of datasets.

We have shown that AdapterFusion is compatible with adapters trained in both single-task as well as multi-task setups. AdapterFusion consistently outperforms fully fine-tuned models on the target task, demonstrating the value in having access to information from other tasks. While we observe gains using both ST-A as well as MT-A, we find that composing ST-A using AdapterFusion is the more efficient strategy, as adapters can be trained in parallel and re-used.

Finally, we analyze the weighting patterns of individual adapters in AdapterFusion which reveal that tasks with small datasets more often rely on information from tasks with large datasets, thereby achieving the largest performance gains in our experiments. We show that AdapterFusion is able to identify and select adapters that contain knowledge relevant to task of interest, while ignoring the remaining ones. This provides an implicit no-op option and makes AdapterFusion a suitable and versatile transfer learning approach for any NLU setting.

8.2 Outlook

Rücklé et al. (2020a) have studied pruning a large portion of adapters after Fusion training. Their results show that removing the less activated adapters results in almost no performance drop at inference time while considerably improving the inference speed. They also provide some initial evidence that it is possible to train Fusion with a subset of the available adapters in each minibatch, potentially enabling us to scale our approach to large adapter sets — which would otherwise be computationally infeasible. We believe that such extensions are a promising direction for future work.

Pfeiffer et al. (2020d) have achieved considerable improvements in the zero-shot cross-lingual transfer performance by dropping the adapters in the last layer. In preliminary results, we have observed similar trends with AdapterFusion when the adapters in the last layer are not used. We will investigate this further in future work.

Acknowledgments

Jonas is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. Aishwarya was supported in part by a DeepMind PhD Fellowship during the time which this project was carried out. Andreas is supported by the German Research Foundation within the project "Open Argument Mining" (GU 798/25-1), associated with the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999). This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Research (Improving Deep Learning using Latent Structure). Kyunghyun was a research scientist at Facebook AI Research part-time during which this project was carried out.

We thank Sebastian Ruder, Max Glockner, Jason Phang, Alex Wang, Katrina Evtimova and Sam Bowman for insightful feedback and suggestions on drafts of this paper.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2391–2401.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5189–5197.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common Sense or World Knowledge? Investigating Adapter-Based Knowledge Injection into Pretrained Transformers. *arXiv preprint*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics 2017*, 5:365–378.

Hector J. Levesque. 2011. The winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4487–4496.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Jinseok Nam, Jungi Kim, Eneldo Loza Menc'ia, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification - revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 437–452.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 7–14.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Edwin Simpson, and Iryna Gurevych. 2020b. Low resource multi-task sequence tagging - revisiting dynamic conditional random fields. *arXiv preprint*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020c. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020d. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. *arXiv preprint*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint*.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4465–4470.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Elad Ben-Zaken1 Shauli Ravfogel and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint*.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990, Hong Kong, China. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020a. AdapterDrop: On the Efficiency of Adapters in Transformers. *arXiv preprint*.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020b. MultiCQA: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2486, Online. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint*.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4822–4829.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6949–6956.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4462–4472.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3664–3674.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5986–5995.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

M. Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. In *arXiv preprint*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint*.

A Appendices

A.1 Datasets

Commonsense Reasoning We work with a large number of datasets, all of which have emerged recently in this domain, ranging from sentence level and document level classification to multiple choice questions. The next sentence prediction task *HellaSWAG* (Zellers et al., 2019) is a more difficult version of the previously released *SWAG* dataset (Zellers et al., 2018). *Winogrande* (Sakaguchi et al., 2020) is a large scale and adversarially filtered (Zellers et al., 2018) adaptation of the *Winograd Schema Challenge* (Levesque, 2011). *Cosmos QA* (Huang et al., 2019) is a commonsense reading comprehension dataset which requires reasoning over larger text passages. *Social IQA* (Sap et al., 2019) is a multiple choice dataset which requires reasoning over social interactions between humans. *Commonsense QA* (Talmor et al., 2019) is a multiple choice dataset based on ConceptNet (Speer et al., 2017), which requires reasoning over general knowledge.

Sentiment Analysis We conduct experiments on two binary sentiment classification tasks on long and short text passages. *IMDb* (Maas et al., 2011) consists of long movie reviews and *SST-2* (Socher et al., 2013) consists of short movie reviews from Rotten Tomatoes[6].

Natural Language Inference (NLI) The goal is to classify whether two sentences entail, contradict, or are neutral to each other. For this we conduct experiments on *MultiNLI* (Williams et al., 2018), a multi-genre dataset, *SciTail* (Khot et al., 2018) a NLI dataset on scientific text, *SICK* (Marelli et al., 2014) a NLI dataset with relatedness scores, the composition of *Recognizing Textual Entailment (RTE)* datasets provided by Wang, Singh, Michael, Hill, Levy, and Bowman (2018), as well as the *Commitment Bank (CB)* (De Marneffe et al., 2019) three-class textual entailment dataset.

Sentence Relatedness We include two semantic relatedness datasets which capture whether or not two text samples include similar content. *Microsoft Research Paraphrase Corpus (MRPC)* (Dolan and Brockett, 2005) consists of sentence pairs which capture a paraphrase/semantic equivalence relationship. *Quora Question Pairs (QQP)* targets duplicate question detection.[7]

Misc The Argument Aspect corpus (Stab et al., 2018) is a three-way classification task to predict whether a document provides arguments *for*, *against* or *none* for a given topic (Nuclear Energy, Abortion, Gun-Control, etc). BoolQ (Clark et al., 2019) is a binary reading comprehension classification task for simple *yes, no* questions.

A.2 What Is The Best Adapter Setup?

As described in §2.2.3, the placement of adapter parameters Φ within a pretrained model is non-trivial, and thus requires extensive experiments. In order to identify the best ST-A setting, we run an exhaustive architecture search on the hyperparameters — including the position and number of adapters in each transformer layer, the position and number of pretrained or task dependent layer norms, the position of residual connections, the bottleneck reduction factors $\{2, 8, 16, 64\}$, and the non linearity {ReLU, LeakyReLU, Swish} used within the adapter. We illustrate this in Figure 5. This grid search includes the settings introduced by Houlsby et al. (2019) and Bapna and Firat (2019). We perform this search on three diverse tasks[8] and find

[6] www.rottentomatoes.com

[7] data.quora.com/First-Quora-DatasetReleaseQuestion-Pairs

[8] SST-2, Commonsense QA, and Argument.

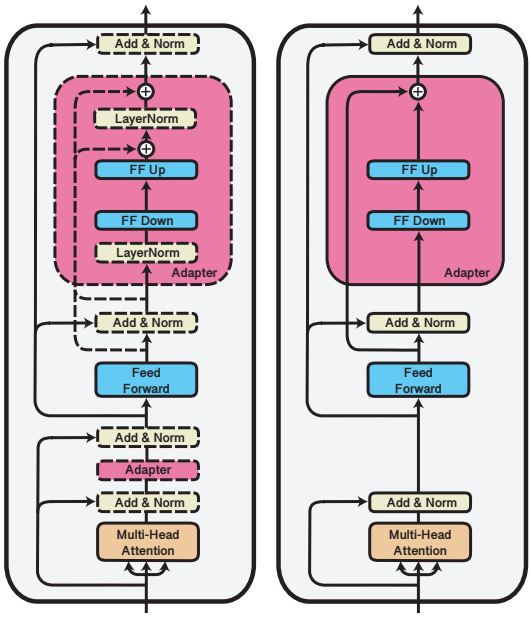


Figure 5: Different architectural components of the adapter. On the left, we show all components for which we conduct an exhaustive search (dashed lines). On the right, we show the adapter architecture that performs the best across all our tasks.

that across all three tasks, the same setup obtains best results. We present our results on the SST-2, Argument, and CSQA datasets in Figures 7, 8, and 9 respectively, at different granularity levels. We find that in contrast to Houlsby et al. (2019), but in line with Bapna and Firat (2019), a single adapter after the feed-forward layer outperforms other settings. While we find that this setting performs on-par with that of Houlsby et al. (2019), it requires only half the number of newly introduced adapters as compared to them, resulting in a more efficient setting in terms of number of operations.

For the single-task adapter setting, we thus perform all subsequent experiments with the best architecture illustrated in Figure 5 on the right and a learning rate of $1e-4$. In order to reproduce the multi-task results in Stickland and Murray (2019) and build upon them, for experiments involving multi-task training, we adopt their architecture as described in §2.2.3.

A.3 AdapterFusion Activations of all Layers

We present the cross-product of activations of AdapterFusion of all layers for BERT-Base and ST-A_{16} in Figure 6, as an extension to Figure 4.

A.4 BERT-base ST-A with Reduction Factors $\{2, 16, 64\}$

We present the ST-A results with different capacity leveraging BERT-base weights in Table 3. Reduction factors 2, 16, and 64 amount to dense adapter dimensions 384, 48, and 12 respectively.

A.5 ST-A and Fusion with ST-A Results with RoBERTa-base

In order to validate our findings of our best setup—ST-A—we re-evaluate our results leveraging RoBERTa-base weights. We present our results in Table 4. Similar to our findigs with BERT-base, especially datasets with less data profit from AdapterFusion. We find that, in contrast to BERT-base, RoBERTa-base does not perform well with high capacity adapters with reduction factor 2.

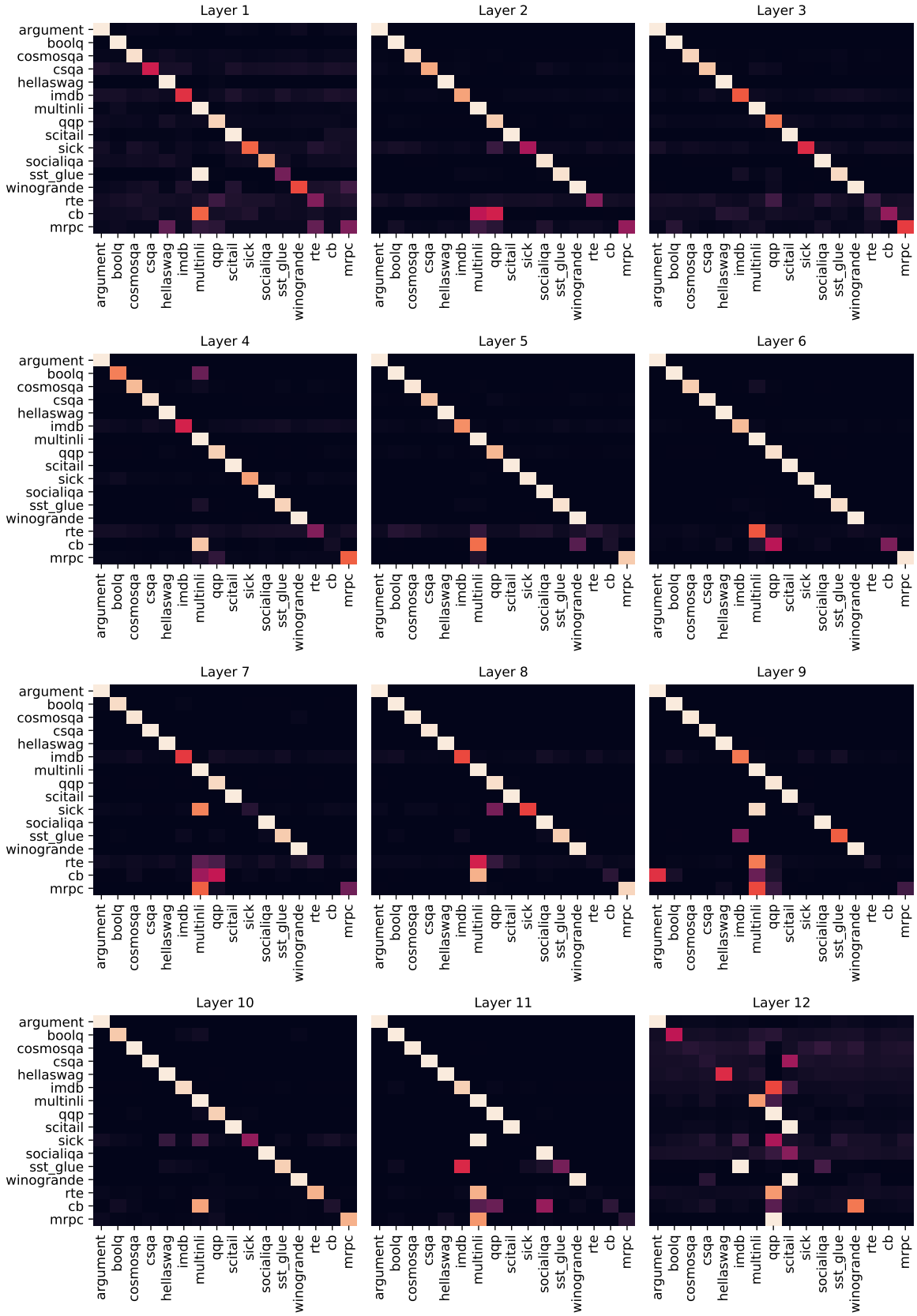
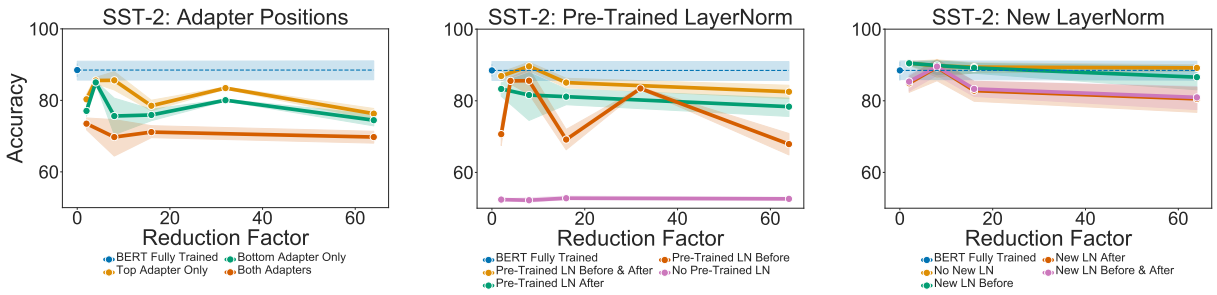


Figure 6: AdapterFusion activations in the 12 BERT-base layers. Target tasks are presented in rows, whereas the set of adapters are displayed in columns. Black squares indicate that an adapter has not been activated, whereas white cells indicate full activation.

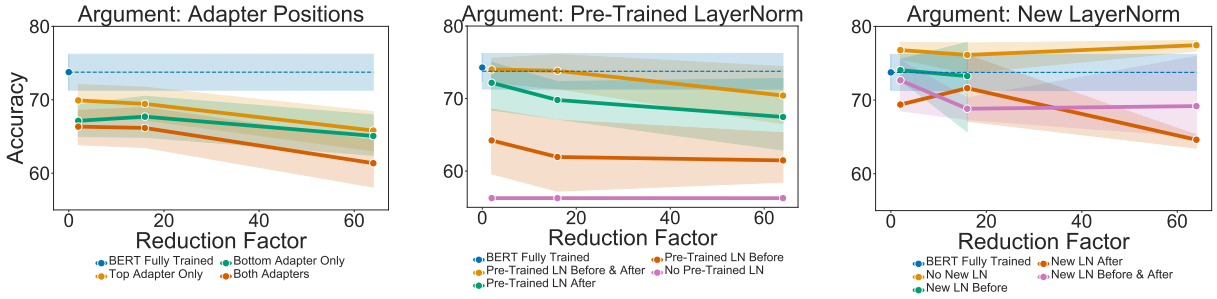


(a) Adapter Positions in Layer

(b) Position of pretrained LayerNorm

(c) Position of newly trained LayerNorm

Figure 7: Results of the grid search on the SST-2 dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.

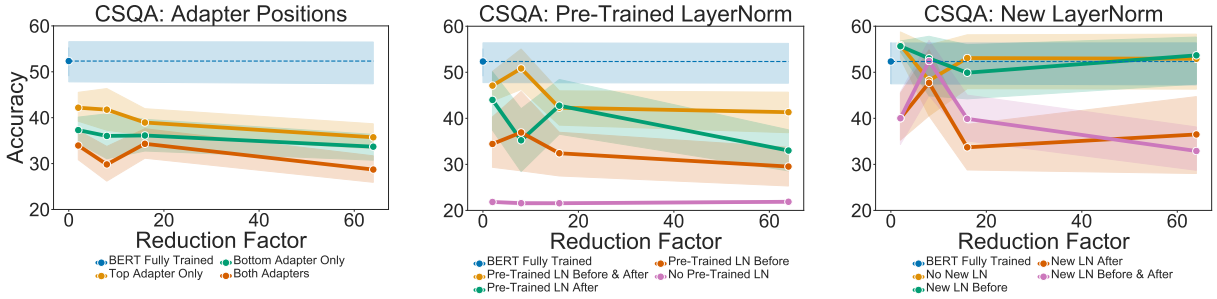


(a) Adapter Positions in Layer

(b) Position of Pretrained LayerNorm

(c) Position of newly trained LayerNorm

Figure 8: Results of the grid search on the Argument dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.



(a) Adapter Positions in Layer

(b) Position of Pretrained LayerNorm

(c) Position of newly trained LayerNorm

Figure 9: Results of the grid search on the CSQA dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.

Dataset	ST-A$_2$	ST-A$_{16}$	ST-A$_{64}$
MultiNLI	84.60	84.32	84.08
QQP	90.57	90.59	89.73
SST	92.66 ±0.32	91.85 ±0.41	92.01 ±0.33
Winogrande	62.11 ±0.09	61.09 ±0.11	59.70 ±0.06
IMDB	94.20 ±0.28	93.85 ±0.07	93.90 ±0.14
HellaSwag	39.45 ±0.20	38.11 ±0.14	38.28 ±0.37
SocialIQA	60.95 ±0.15	62.41 ±0.11	62.23 ±0.73
CosmosQA	59.32 ±0.24	60.01 ±0.02	60.65 ±0.34
SciTail	94.44 ±0.81	93.90 ±0.16	93.82 ±0.49
Argument	76.83 ±0.21	77.65 ±0.34	77.64 ±0.56
CSQA	57.83 ±0.23	58.91 ±0.57	58.88 ±0.40
BoolQ	77.14 ±1.10	75.66 ±1.25	76.07 ±0.54
MRPC	86.13 ±1.59	85.16 ±0.52	85.58 ±0.32
SICK	87.50 ±0.14	86.20 ±0.00	85.70 ±0.42
RTE	70.68 ±4.57	71.04 ±1.62	69.16 ±1.59
CB	87.85 ±2.94	86.07 ±3.87	84.28 ±4.79
Mean	76.39	76.05	75.73

Table 3: Mean and standard deviation results (development sets) for each of the 16 datasets and reduction factors {2, 16, 64} for ST-A. Each model is initialized with BERT-base (Devlin et al., 2019) weights. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separates datasizes $\{> 40k, > 10k, > 5k\}$ respectively.

Dataset	Head	Full	ST-A$_2$	ST-A$_{16}$	ST-A$_{64}$	F. w/ ST-A$_{16}$	ST-A$_{16}^{\text{Houlsby}}$
MultiNLI	56.84	86.42	85.56	86.06	85.86	86.20	86.57
QQP	71.40	91.07	90.88 ±0.07	90.27	89.39 ±0.63	90.28	90.66
SST	81.86 ±0.21	94.29 ±0.22	93.71 ±0.29	93.80 ±0.23	93.35 ±0.43	93.67 ±0.13	94.17 ±0.15
Winogrande	51.93	66.77	51.27 ±0.78	65.58 ±0.53	62.43	66.01 ±0.47	63.46 ±6.38
IMDB	85.40	96.00	95.70	95.78 ±0.13	95.80	95.78 ±0.19	95.68 ±0.26
HellaSwag	41.16	63.53	61.09 ±0.08	61.57 ±0.14	61.18 ±0.21	61.52 ±0.07	61.21 ±0.37
SocialIQA	46.87	69.44	69.24	70.14 ±0.40	70.21	70.13 ±0.11	70.78 ±0.17
CosmosQA	41.88 ±0.29	68.52 ±0.49	68.01 ±0.94	68.76 ±0.53	68.62 ±0.55	68.64 ±0.04	69.18 ±0.34
SciTail	49.57	94.47	94.24	94.59 ±0.64	94.32	94.44 ±0.09	94.09 ±0.39
Argument	66.22 ±0.62	78.04 ±0.42	78.60 ±0.34	78.50 ±0.45	78.53 ±0.59	77.98 ±0.24	78.42 ±0.44
CSQA	41.37 ±0.34	65.81 ±0.59	66.11 ±0.60	66.30 ±0.38	64.03 ±0.27	66.52 ±0.18	67.53 ±0.70
BoolQ	62.17	81.89	80.86 ±0.86	80.83 ±0.27	80.17 ±0.25	80.86 ±0.15	81.11 ±0.54
MRPC	68.38 ±0.00	89.11 ±0.93	89.11 ±0.51	88.72 ±0.71	87.10 ±1.67	89.65 ±0.50	89.17 ±1.06
SICK	56.40	86.60	84.80	85.40 ±0.32	85.40	85.76 ±0.26	85.88 ±0.46
RTE	55.81 ±2.92	72.34 ±11.02	61.80 ±12.47	75.30 ±0.61	73.86 ±1.55	78.79 ±1.12	78.56 ±1.54
CB	59.64 ±11.05	90.00 ±1.60	87.14 ±6.85	89.28 ±2.82	81.07 ±4.82	92.86 ±3.79	89.64 ±3.87
Mean	58.05	81.08	78.63	80.83	79.52	**81.41**	81.18

Table 4: Mean and standard deviation results of models initialized with RoBERTa-base (Liu et al., 2019b) weights. Performances are measured on the development sets of the 16 datasets for the different architectural setups. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separate datasizes $\{> 40k, > 10k, > 5k\}$ respectively. **Head** indicates training only a classification head on top of fixed RoBERTa weights. For **Full** training we fine-tune all weights of RoBERTa. Single-Task adapters (**ST-A**) is the training of independently trained adapters for each task, using the architecture illustrated in Figure 5, indices {2, 16, 64} indicate the reduction factor. **Fusion w/ ST-A** show the results of AdapterFusion using the respective pretrained adapters. **ST-A$_{16}^{\text{Houlsby}}$** shows the results of ST-A with with architecture proposed by Houlsby et al. (2019).

Chapter 8

MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer

MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer

Jonas Pfeiffer[1], **Ivan Vulić**[2], **Iryna Gurevych**[1], **Sebastian Ruder**[3]
[1]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[2]Language Technology Lab, University of Cambridge
[3]DeepMind
pfeiffer@ukp.tu-darmstadt.de

Abstract

The main goal behind state-of-the-art pretrained multilingual models such as multilingual BERT and XLM-R is enabling and bootstrapping NLP applications in *low-resource languages* through zero-shot or few-shot cross-lingual transfer. However, due to limited model capacity, their transfer performance is the weakest exactly on such low-resource languages and languages *unseen* during pretraining. We propose **MAD-X**, an adapter-based framework that enables high portability and parameter-efficient transfer to arbitrary tasks and languages by learning modular language and task representations. In addition, we introduce a novel invertible adapter architecture and a strong baseline method for adapting a pretrained multilingual model to a new language. MAD-X outperforms the state of the art in cross-lingual transfer across a representative set of typologically diverse languages on named entity recognition and causal commonsense reasoning, and achieves competitive results on question answering. Our code and adapters are available at AdapterHub.ml.

1 Introduction

Current deep pretrained multilingual models (Devlin et al., 2019; Conneau and Lample, 2019) achieve state-of-the-art results on cross-lingual transfer, but do not have enough capacity to represent all languages. Evidence for this is the importance of the vocabulary size (Artetxe et al., 2020) and *the curse of multilinguality* (Conneau et al., 2020), a trade-off between language coverage and model capacity. Scaling up a model to cover all of the world's 7,000+ languages is prohibitive. At the same time, limited capacity is an issue even for high-resource languages where state-of-the-art multilingual models underperform their monolingual variants (Eisenschlos et al., 2019; Virtanen et al., 2019; Nozza et al., 2020), and performance decreases further with lower-resource languages covered by the pretrained models. Moreover, the model capacity issue is arguably most severe for languages that were not included in the training data at all, and pretrained models perform poorly on those languages (Ponti et al., 2020b).

In this paper, we propose Multiple ADapters for Cross-lingual transfer (**MAD-X**), a modular framework that leverages a small number of extra parameters to address the fundamental capacity issue that limits pretrained multilingual models. Using a state-of-the-art multilingual model as the foundation, we adapt the model to arbitrary tasks and languages by learning modular language- and task-specific representations *via adapters* (Rebuffi et al., 2017; Houlsby et al., 2019), small bottleneck layers inserted between a model's weights.

In particular, using a recent efficient adapter variant (Pfeiffer et al., 2020a; Rücklé et al., 2020), we train **1)** *language-specific adapter modules* via masked language modelling (MLM) on unlabelled target language data, and **2)** *task-specific adapter modules* via optimising a target task on labelled data in any source language. Task and language adapters are stacked as in Figure 1, allowing us to adapt the pretrained multilingual model also to languages that are not covered in the model's (pre)training data by substituting the target language adapter at inference.

In order to deal with a mismatch between the shared multilingual vocabulary and target language vocabulary, we propose *invertible adapters*, a new type of adapter that is well suited to performing MLM in another language. Our framework goes beyond prior work on using adapters for cross-lingual transfer (Bapna and Firat, 2019; Artetxe et al., 2020) by enabling adaptation to languages *unseen* during pretraining and without learning expensive language-specific token-level embeddings.

We compare MAD-X against state-of-the-art

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 7654–7673,
November 16–20, 2020.

cross-lingual transfer methods on the standard WikiANN NER dataset (Pan et al., 2017; Rahimi et al., 2019) and the XCOPA dataset (Ponti et al., 2020a) for causal commonsense reasoning, relying on a representative set of typologically diverse languages which includes high-resource, low-resource, as well as languages unseen by the pretrained model. MAD-X outperforms the baselines on seen and unseen high-resource and low-resource languages. On the high-resource languages of the challenging XQuAD QA dataset (Artetxe et al., 2020), our framework achieves competitive performance while being more parameter-efficient.

Another contribution of our work is a simple method of adapting a pretrained multilingual model to a new language, which outperforms the standard setting of transferring a model only from labelled source language data.

In sum, our contributions are as follows. **1)** We propose MAD-X, a modular framework that mitigates the curse of multilinguality and adapts a multilingual model to arbitrary tasks and languages. Both code and adapter weights are integrated into the AdapterHub.ml repository (Pfeiffer et al., 2020b).[1] **2)** We propose invertible adapters, a new adapter variant for cross-lingual MLM. **3)** We demonstrate strong performance and robustness of MAD-X across diverse languages and tasks. **4)** We propose a simple and more effective baseline method for adapting a pretrained multilingual model to target languages. **5)** We shed light on the behaviour of current methods on languages that are unseen during multilingual pretraining.

2 Related Work

Cross-lingual Representations Research in modern cross-lingual NLP is increasingly focused on learning general-purpose cross-lingual representations that can be applied to many tasks, first on the word level (Mikolov et al., 2013; Gouws et al., 2015; Glavaš et al., 2019; Ruder et al., 2019; Wang et al., 2020) and later on the full-sentence level (Devlin et al., 2019; Conneau and Lample, 2019; Cao et al., 2020). More recent models such as multilingual BERT (Devlin et al., 2019)—large Transformer (Vaswani et al., 2017) models pretrained on large amounts of multilingual data—have been observed to perform surprisingly well when transferring to other languages (Pires et al., 2019; Wu and Dredze, 2019; Wu et al., 2020) and the current state-of-the-art model, XLM-R is competitive with the performance of monolingual models on the GLUE benchmark (Conneau et al., 2020). Recent studies (Hu et al., 2020), however, indicate that state-of-the-art models such as XLM-R still perform poorly on cross-lingual transfer across many language pairs. The main reason behind such poor performance is the current lack of capacity in the model to represent all languages equally in the vocabulary and representation space (Bapna and Firat, 2019; Artetxe et al., 2020; Conneau et al., 2020).

Adapters Adapter modules have been originally studied in computer vision tasks where they have been restricted to convolutions and used to adapt a model for multiple domains (Rebuffi et al., 2017, 2018). In NLP, adapters have been mainly used for parameter-efficient and quick fine-tuning of a base pretrained Transformer model to new tasks (Houlsby et al., 2019; Stickland and Murray, 2019) and new domains (Bapna and Firat, 2019), avoiding catastrophic forgetting (McCloskey and Cohen, 1989; Santoro et al., 2016). Bapna and Firat (2019) also use adapters to fine-tune and recover performance of a multilingual NMT model on high-resource languages, but their approach cannot be applied to languages that were not seen during pretraining. Artetxe et al. (2020) employ adapters to transfer a pretrained monolingual model to an unseen language but rely on learning new token-level embeddings, which do not scale to a large number of languages. Pfeiffer et al. (2020a) combine the information stored in multiple adapters for more robust transfer learning between monolingual tasks. In their contemporaneous work, Üstün et al. (2020) generate adapter parameters from language embeddings for multilingual dependency parsing.

3 Multilingual Model Adaptation for Cross-lingual Transfer

Standard Transfer Setup The standard way of performing cross-lingual transfer with a state-of-the-art large multilingual model such as multilingual BERT or XLM-R is 1) to fine-tune it on labelled data of a downstream task in a source language and then 2) apply it directly to perform inference in a target language (Hu et al., 2020). A downside of this setting is that the multilingual initialisation balances *many* languages. It is thus not suited to excel at a specific language at inference time. We propose a simple method to ameliorate this issue by allowing the model to additionally

[1] https://github.com/Adapter-Hub/adapter-transformers

adapt to the particular target language.

Target Language Adaptation Similar to fine-tuning monolingual models on the task domain (Howard and Ruder, 2018), we propose to fine-tune a pretrained multilingual model via MLM on unlabelled data of the target language prior to task-specific fine-tuning in the source language. A disadvantage of this approach is that it no longer allows us to evaluate the same model on multiple target languages as it biases the model to a specific target language. However, this approach might be preferable if we only care about performance in a specific (i.e., fixed) target language. We find that target language adaptation results in improved cross-lingual transfer performance over the standard setting (§6). In other words, it does not result in catastrophic forgetting of the multilingual knowledge already available in the pretrained model that enables the model to transfer to other languages. In fact, experimenting with methods that explicitly try to prevent catastrophic forgetting (Wiese et al., 2017) led to worse performance in our experiments.

Nevertheless, the proposed simple adaptation method inherits the fundamental limitation of the pretrained multilingual model and the standard transfer setup: the model's limited capacity hinders effective adaptation to low-resource and unseen languages. In addition, fine-tuning the full model does not scale well to many tasks or languages.

4 Adapters for Cross-lingual Transfer

Our MAD-X framework addresses these deficiencies and can be used to effectively adapt an existing pretrained multilingual model to other languages. The framework comprises three types of adapters: language, task, and invertible adapters. As in previous work (Rebuffi et al., 2017; Houlsby et al., 2019), adapters are trained while keeping the parameters of the pretrained multilingual model fixed. Our framework thus enables learning language and task-specific transformations in a modular and parameter-efficient way. We show the full framework as part of a standard Transformer model in Figure 1 and describe the three adapter types.

4.1 Language Adapters

For learning language-specific transformations, we employ a recent efficient adapter architecture proposed by Pfeiffer et al. (2020a). Following Houlsby et al. (2019) they define the interior of the adapter to be a simple down- and up-projection combined with a residual connection.[2] The language adapter LA_l at layer l consists of a down-projection $\mathbf{D} \in \mathbb{R}^{h \times d}$ where h is the hidden size of the Transformer model and d is the dimension of the adapter, followed by a ReLU activation and an up-projection $\mathbf{U} \in \mathbb{R}^{d \times h}$ at every layer l:

$$\mathsf{LA}_l(\mathbf{h}_l, \mathbf{r}_l) = \mathbf{U}_l(\mathsf{ReLU}(\mathbf{D}_l(\mathbf{h}_l))) + \mathbf{r}_l \quad (1)$$

where $\mathbf{h}_l$ and $\mathbf{r}_l$ are the Transformer hidden state and the residual at layer l, respectively. The residual connection $\mathbf{r}_l$ is the output of the Transformer's feed-forward layer whereas $\mathbf{h}_l$ is the output of the subsequent layer normalisation (see Figure 1).

We train language adapters on unlabelled data of a language using MLM, which encourages them to learn transformations that make the pretrained multilingual model more suitable for a specific language. During task-specific training with labelled data, we use the language adapter of the corresponding source language, which is kept fixed. In order to perform zero-shot transfer to another language, we

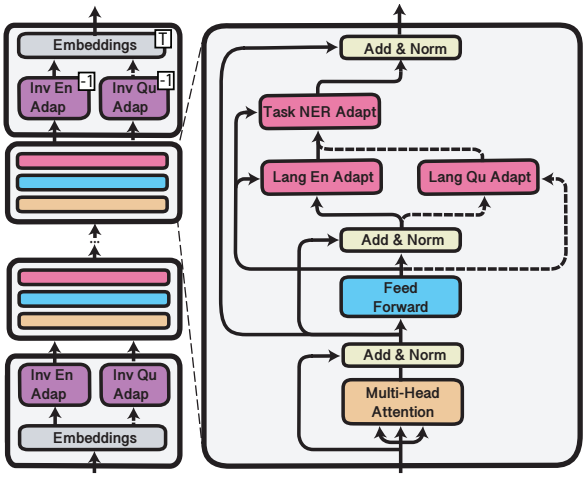
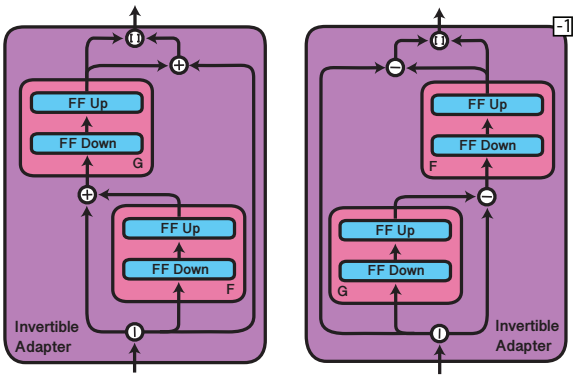


Figure 1: The MAD-X framework inside a Transformer model. Input embeddings are fed into the invertible adapter whose inverse is fed into the tied output embeddings. Language and task adapters are added to each Transformer layer. Language adapters and invertible adapters are trained via masked language modelling (MLM) while the pretrained multilingual model is kept frozen. Task-specific adapters are stacked on top of source language adapters when training on a downstream task such as NER (full lines). During zero-shot cross-lingual transfer, source language adapters are replaced with target language adapters (dashed lines).

[2] Pfeiffer et al. (2020a) perform an extensive hyperparameter search over adapter positions, activation functions, and residual connections within each Transformer layer. They arrive at an architecture variant that performs on par with that of Houlsby et al. (2019), while being more efficient.



(a) The invertible adapter (b) The inversed adapter

Figure 2: The invertible adapter (a) and its inverse (b). The input is split and transformed by projections F and G, which are coupled in an alternating fashion. | indicates the splitting of the input vector, and [ ] indicates the concatenation of two vectors. $+$ and $-$ indicate element-wise addition and subtraction, respectively.

simply replace the source language adapter with its target language component. For instance, as illustrated in Figure 1, we can simply replace a language-specific adapter trained for English with a language-specific adapter trained for Quechua at inference time. This, however, requires that the underlying multilingual model does not change during fine-tuning on the downstream task. In order to ensure this, we additionally introduce task adapters that capture task-specific knowledge.

4.2 Task Adapters

Task adapters TA_l at layer l have the same architecture as language adapters. They similarly consist of a down-projection $\mathbf{D} \in \mathbb{R}^{h \times d}$, a ReLU activation, followed by an up-projection. They are stacked on top of the language adapters and thus receive the output of the language adapter LA_l as input, together with the residual $\mathbf{r}_l$ of the Transformer's feed-forward layer[3]:

$$\mathsf{TA}_l(\mathbf{h}_l, \mathbf{r}_l) = \mathbf{U}_l(\mathsf{ReLU}(\mathbf{D}_l(\mathsf{LA}_l))) + \mathbf{r}_l \quad (2)$$

The output of the task adapter is then passed to another layer normalisation component. Task adapters are the only parameters that are updated when training on a downstream task (e.g., NER) and aim to capture knowledge that is task-specific but generalises across languages.

[3]Initial experiments showed that this residual connection performs better than one to the output of the language adapter.

4.3 Invertible Adapters

The majority of the "parameter budget" of pretrained multilingual models is spent on token embeddings of the shared multilingual vocabulary. Despite this, they underperform on low-resource languages (Artetxe et al., 2020; Conneau et al., 2020), and are bound to fare even worse for languages not covered by the model's training data.

In order to mitigate this mismatch between multilingual and target language vocabulary, we propose invertible adapters. They are stacked on top of the embedding layer while their respective inverses precede the output embedding layer (see Figure 1). As input and output embeddings are tied in multilingual pretrained models, invertibility allows us to leverage the same set of parameters for adapting both input and output representations. This is crucial as the output embeddings, which get discarded during task-specific fine-tuning might otherwise overfit to the pretraining task.

To ensure this invertibility, we employ Nonlinear Independent Component Estimation (NICE; Dinh et al., 2015). NICE enables the invertibility of arbitrary non-linear functions through a set of coupling operations (Dinh et al., 2015). For the invertible adapter, we split the input embedding vector $\mathbf{e}_i$ of the i-th token into two vectors of equal dimensionality $\mathbf{e}_{1,i}, \mathbf{e}_{2,i} \in \mathbb{R}^{h/2}$.[4] For two arbitrary non-linear function F and G, the forward pass through our invertible adapter $A_{inv}()$ is:

$$\begin{aligned} &\mathbf{o}_1 = F(\mathbf{e}_2) + \mathbf{e}_1; \;\; \mathbf{o}_2 = G(\mathbf{o}_1) + \mathbf{e}_2 \\ &\mathbf{o} = [\mathbf{o}_1, \mathbf{o}_2] \end{aligned} \quad (3)$$

where $\mathbf{o}$ is the output of the invertible adapter A_{inv} and $[\cdot, \cdot]$ indicates concatenation of two vectors.

Correspondingly, the inverted pass through the adapter, thus A^{-1}_{inv}, is computed as follows:

$$\begin{aligned} &\mathbf{e}_2 = \mathbf{o}_2 - G(\mathbf{o}_1); \;\; \mathbf{e}_1 = \mathbf{o}_1 - F(\mathbf{e}_2) \\ &\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2]. \end{aligned} \quad (4)$$

$\mathbf{e}$ is the output of $A^{-1}_{Inv}()$. For the non-linear transformations F and G, we use similar down- and up-projections as for the language and task adapters:

$$\begin{aligned} F(\mathbf{x}) &= \mathbf{U}_F(\mathsf{ReLU}(\mathbf{D}_F(\mathbf{x}))) \\ G(\mathbf{x}) &= \mathbf{U}_G(\mathsf{ReLU}(\mathbf{D}_G(\mathbf{x}))). \end{aligned} \quad (5)$$

where $\mathbf{D}_F, \mathbf{D}_G \in \mathbb{R}^{\frac{h}{4} \times \frac{h}{2}}$ and $\mathbf{U}_F, \mathbf{U}_G \in \mathbb{R}^{\frac{h}{2} \times \frac{h}{4}}$ and $\mathbf{x}$ is a placeholder for $\mathbf{e}_1, \mathbf{e}_2, \mathbf{o}_1$ and $\mathbf{o}_2$. We

[4]For brevity, we further leave out the dependency on i.

illustrate the complete architecture of the invertible adapter and its inverse in Figure 2.

The invertible adapter has a similar function to the language adapter, but aims to capture token-level language-specific transformations. As such, it is trained together with the language adapters using MLM on unlabelled data of a specific language. During task-specific training we use the fixed invertible adapter of the source language, and replace it with the target-language invertible during zero-shot transfer. Importantly, our invertible adapters are much more parameter-efficient compared to the approach of Artetxe et al. (2020), which learns separate token embeddings for every new language.

An Illustrative Example We provide a brief walk-through example from Figure 1. Assuming English (*En*) as the source language and Quechua (*Qu*) as the target language, we first pretrain invertible adapters A^{En}_{Inv} and A^{Qu}_{Inv}, and language adapters A^{En}_{Lang} and A^{Qu}_{Lang} with MLM for which the output of the last layer is passed through $A^{En\ -1}_{Inv}$. We then train a task adapter for the NER task A^{NER}_{Task} on the English NER training set. During training, embeddings are passed through A^{En}_{Inv}. At every layer of the model the data is first passed through the fixed A^{En}_{Lang} and then into the NER adapter A^{NER}_{Task}. For zero-shot inference, the English invertible and language adapters A^{En}_{Inv} and A^{En}_{Lang} are replaced with their Quechua counterparts A^{Qu}_{Inv} and A^{Qu}_{Lang} while the data is still passed through the NER task adapter A^{NER}_{Task}.

5 Experiments

Data We conduct experiments on three tasks: Named entity recognition (NER), question answering (QA), and causal commonsense reasoning (CCR). For NER, we use the WikiANN (Pan et al., 2017) dataset, which was partitioned into train, development, and test portions by Rahimi et al. (2019). For QA, we employ the XQuAD dataset (Artetxe et al., 2020), a cross-lingual version of SQuAD (Rajpurkar et al., 2016). For CCR, we rely on XCOPA (Ponti et al., 2020a), a cross-lingual version of COPA (Roemmele et al., 2011).

Languages The partitioned version of WikiANN covers 176 languages. In order to obtain a comprehensive comparison to state-of-the-art cross-lingual methods under different evaluation conditions, we select languages based on: **a)** variance in data availability (by selecting languages with a range of respective Wikipedia sizes); **b)** their presence in pretrained multilingual models; more precisely, whether data in the particular language was included in the pretraining data of both multilingual BERT and XLM-R or not; and **c)** typological diversity to ensure that different language types and families are covered. In total, we can discern four categories in our language set: **1)** high-resource languages and **2)** low-resource languages covered by the pretrained SOTA multilingual models (i.e., by mBERT and XLM-R); as well as **3)** low-resource languages and **4)** truly low-resource languages not covered by the multilingual models. We select four languages from different language families for each category. We highlight characteristics of the 16 languages from 11 language families in Table 1.

Language	ISO code	Language family	# of Wiki articles	Covered by SOTA?
English	en	Indo-European	6.0M	✓
Japanese	ja	Japonic	1.2M	✓
Chinese	zh	Sino-Tibetan	1.1M	✓
Arabic	ar	Afro-Asiatic	1.0M	✓
Javanese	jv	Austronesian	57k	✓
Swahili	sw	Niger-Congo	56k	✓
Icelandic	is	Indo-European	49k	✓
Burmese	my	Sino-Tibetan	45k	✓
Quechua	qu	Quechua	22k	
Min Dong	cdo	Sino-Tibetan	15k	
Ilokano	ilo	Austronesian	14k	
Mingrelian	xmf	Kartvelian	13k	
Meadow Mari	mhr	Uralic	10k	
Maori	mi	Austronesian	7k	
Turkmen	tk	Turkic	6k	
Guarani	gn	Tupian	4k	

Table 1: Languages in our NER evaluation.

We evaluate on all possible language pairs (i.e., on the Cartesian product), using each language as a source language with every other language (including itself) as a target language. This subsumes both the standard *zero-shot cross-lingual transfer setting* (Hu et al., 2020) as well as the standard *monolingual in-language* setting.

For CCR and QA, we evaluate on the 12 and 11 languages provided in XCOPA and XQuAD respectively, with English as source language. XCOPA contains a typologically diverse selection of languages including two languages (Haitian Creole and Quechua) that are unseen by our main model. XQuAD comprises slightly less typologically diverse languages that are mainly high-resource.

Model	en	ja	zh	ar	jv	sw	is	my	qu	cdo	ilo	xmf	mi	mhr	tk	gn	avg
XLM-R^{Base}	44.2	38.2	40.4	36.4	37.4	42.8	47.1	**26.3**	27.4	18.1	28.8	**35.0**	16.7	**31.7**	20.6	**31.2**	32.6
XLM-R^{Base} MLM-SRC	39.5	45.2	34.7	17.7	34.5	35.3	43.1	20.8	26.6	21.4	28.7	22.4	18.1	25.0	27.6	24.0	29.0
XLM-R^{Base} MLM-TRG	54.8	**47.4**	**54.7**	51.1	38.7	48.1	53.0	20.0	29.3	16.6	27.4	24.7	15.9	26.4	26.5	28.5	35.2
MAD-X^{Base} – LAD – INV	44.5	38.6	40.6	42.8	32.4	43.1	48.6	23.9	22.0	10.6	23.9	27.9	13.2	24.6	18.8	21.9	29.8
MAD-X^{Base} – INV	52.3	46.0	46.2	56.3	**41.6**	48.6	52.4	23.2	32.4	**27.2**	30.8	33.0	**23.5**	29.3	30.4	28.4	37.6
MAD-X^{Base}	**55.0**	46.7	47.3	**58.2**	39.2	**50.4**	**54.5**	24.9	**32.6**	24.2	**33.8**	34.3	16.8	**31.7**	**31.9**	30.4	**38.2**
mBERT	48.6	50.5	50.6	50.9	45.3	48.7	51.2	17.7	31.8	20.7	33.3	26.1	**20.9**	31.3	**34.8**	**30.9**	37.1
MAD-X^{mBERT}	**52.8**	**53.1**	**53.2**	**55.5**	**46.3**	**50.9**	**51.4**	**21.0**	**37.7**	**22.1**	**35.0**	**30.0**	18.6	**31.8**	33.0	25.1	**38.6**
XLM-R^{Large}	47.10	46.52	46.43	45.15	39.21	43.96	48.69	**26.18**	26.39	15.12	22.80	**33.67**	19.86	27.70	29.56	**33.78**	34.6
MAD-X^{Large}	**56.30**	**53.37**	**55.6**	**59.41**	**40.40**	**50.57**	**53.22**	24.55	**33.89**	**26.54**	**30.98**	33.37	**24.31**	**28.03**	**30.82**	26.38	**39.2**

Table 2: NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e. the columns are source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones (see also Table 1).

5.1 Baselines

The baseline models are based on different approaches to multilingual model adaptation for cross-lingual transfer, discussed previously in §3.

XLM-R The main model we compare against is XLM-R (Conneau et al., 2020), the current state-of-the-art model for cross-lingual transfer (Hu et al., 2020). It is a Transformer-based model pretrained for 100 languages on large cleaned Common Crawl corpora (Wenzek et al., 2020). For efficiency, we use the XLM-R Base configuration as the basis for most of our experiments. However, we note that the main idea behind the MAD-X framework is not tied to any particular pretrained model, and the framework can be easily adapted to other pretrained multilingual models as we show later in §6 (e.g., multilingual BERT). First, we compare against XLM-R in the standard setting where the entire model is fine-tuned on labelled data of the task in the source language.

XLM-R^{Base} MLM-SRC; XLM-R^{Base} MLM-TRG In §3, we have proposed target language adaptation as a simple method to adapt pretrained multilingual models for better cross-lingual generalisation on the downstream task. As a sanity check, we also compare against adapting to the source language data; we expect it to improve in-language performance but not to help with transfer. In particular, we fine-tune XLM-R with MLM on unlabelled source language (XLM-R^{Base} MLM-SRC) and target language data (XLM-R^{Base} MLM-TRG) prior to task-specific fine-tuning.

5.2 MAD-X: Experimental Setup

For the **MAD-X** framework, unless noted otherwise, we rely on the XLM-R Base architecture; we evaluate the full MAD-X, MAD-X without invertible adapters (–INV), and also MAD-X without language and invertible adapters (– LAD – INV). We use the Transformers library (Wolf et al., 2020) for all our experiments. For fine-tuning via MLM on unlabelled data, we train on the Wikipedia data of the corresponding language for 250,000 steps, with a batch size of 64 and a learning rate of $5e-5$ and $1e-4$ for XLM-R (also for the -SRC and -TRG variants) and adapters, respectively. We train models on NER data for 100 epochs with a batch size of 16 and 8 for high-resource and low-resource languages, respectively, and a learning rate of $5e-5$ and $1e-4$ for XLM-R and adapters, respectively. We choose the best checkpoint for evaluation based on validation performance. Following Pfeiffer et al. (2020a), we learn language adapters, invertible adapters, and task adapters with dimensionalities of 384, 192 (384 for both directions), and 48, respectively. XLM-R Base has a hidden layer size of 768, so these adapter sizes correspond to reductions of 2, 2, and 16.

For NER, we conduct five runs of fine-tuning on the WikiAnn training set of the source language—except for XLM-R^{Base} MLM-TRG for which we conduct one run for efficiency purposes for every source language–target language combination. For QA, we conduct three runs of fine-tuning on the English SQuAD training set, evaluate on all XQuAD target languages, and report mean F_1 and exact match (EM) scores. For CCR, we conduct three runs of fine-tuning on the respective English training set, evaluate on all XCOPA target languages, and report accuracy scores.

6 Results and Discussion

Named Entity Recognition As our main summary of results, we average the cross-lingual transfer results of each method for each source language

across all 16 target languages on the NER dataset. We show the aggregated results in Table 2. Moreover, in the appendix we report the detailed results for all methods across each single language pair, as well as a comparison of methods on the most common setting with English as source language.

In general, we observe that XLM-R performance is indeed lowest for unseen languages (the right half of the table after the vertical dashed line). XLM-R^{Base} MLM-SRC performs worse than XLM-R, which indicates that source-language fine-tuning is not useful for cross-lingual transfer in general.[5] On the other hand, XLM-R^{Base} MLM-TRG is a stronger transfer method than XLM-R on average, yielding gains in 9/16 target languages. However, its gains seem to vanish for low-resource languages. Further, there is another disadvantage, outlined in §3: XLM-R^{Base} MLM-TRG requires fine-tuning the full large pretrained model separately for each target language in consideration, which can be prohibitively expensive.

MAD-X without language and invertible adapters performs on par with XLM-R for almost all languages present in the pretraining data (left half of the table). This mirrors findings in the monolingual setting where task adapters have been observed to achieve performance similar to regular fine-tuning while being more parameter-efficient (Houlsby et al., 2019). However, looking at unseen languages, the performance of MAD-X that only uses task adapters deteriorates significantly compared to XLM-R. This shows that task adapters alone are not expressive enough to bridge the discrepancy when adapting to an unseen language.

Adding language adapters to MAD-X improves its performance across the board, and their usefulness is especially pronounced for low-resource languages. Language adapters help capture the characteristics of the target language and consequently provide boosts for unseen languages. Even for high-resource languages, the addition of language-specific parameters yields substantial improvements. Finally, invertible adapters provide further gains and generally outperform only using task and language adapters: for instance, we observe gains with MAD-X over MAD-X –INV on 13/16 target languages. Overall, the full MAD-X framework improves upon XLM-R by more than 5 F_1 points on average.

[5] However, there are some examples (e.g., JA, TK) where it does yield slight gains over the standard XLM-R transfer.

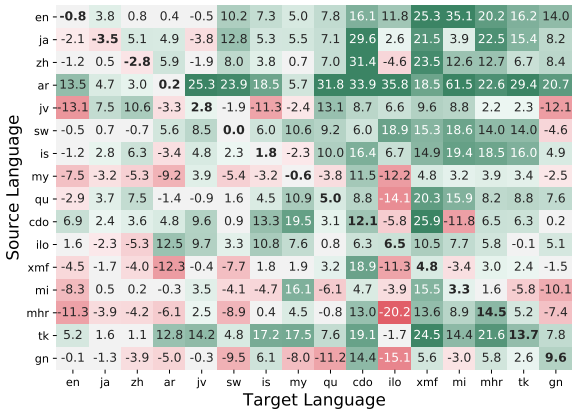


Figure 3: Relative F_1 improvement of MAD-X^{Base} over XLM-R^{Base} in cross-lingual NER transfer.

To demonstrate that our framework is model-agnostic, we also employ two other strong multilingual models, XLM-R^{Large} and mBERT as foundation for MAD-X and show the results in Table 2. MAD-X shows consistent improvements even over stronger base pretrained models.

For a more fine-grained impression of the performance of MAD-X in different languages, we show its relative performance against XLM-R in the standard setting in Figure 3. We observe the largest differences in performance when transferring from high-resource to low-resource and unseen languages (top-right quadrant of Figure 3), which is arguably the most natural setup for cross-lingual transfer. In particular, we observe strong gains when transferring from Arabic, whose script might not be well represented in XLM-R's vocabulary. We also detect strong performance in the in-language monolingual setting (diagonal) for the subset of low-resource languages. This indicates that MAD-X may help bridge the perceived weakness of multilingual versus monolingual models. Finally, MAD-X performs competitively even when the target language is high-resource.[6]

Causal Commonsense Reasoning We show results on transferring from English to each target language on XCOPA in Table 3. For brevity, we only show the results of the best fine-tuning set-

[6] In the appendix, we also plot relative performance of the full MAD-X method (with all three adapter types) versus XLM-R^{Base} MLM-TRG across all language pairs. The scores lead to similar conclusions as before: the largest benefits of MAD-X are observed for the set of low-resource target languages (i.e., the right half of the heatmap). The scores also again confirm that the proposed XLM-R^{Base} MLM-TRG transfer baseline is more competitive than the standard XLM-R transfer across a substantial number of language pairs.

Model	en	et	ht	id	it	qu	sw	ta	th	tr	vi	zh	avg
XLM-R^{Base}	66.8	58.0	51.4	65.0	60.2	51.2	52.0	58.4	62.0	56.6	65.6	**68.8**	59.7
XLM-R^{Base} MLM-TRG	66.8	59.4	50.0	**71.0**	61.6	46.0	**58.8**	60.0	**63.2**	**62.2**	**67.6**	67.4	61.2
MAD-X^{Base}	**68.3**	**61.3**	**53.7**	65.8	**63.0**	**52.5**	56.3	**61.9**	61.8	60.3	66.1	67.6	**61.5**

Table 3: Accuracy scores of all models on the XCOPA test sets when transferring from English. Models are first fine-tuned on SIQA and then on the COPA training set.

	en	ar	de	el	es	hi	ru	th	tr	vi	zh	avg
XLM-R^{Base}	83.6 / 72.1	66.8 / 49.1	**74.4 / 60.1**	73.0 / **55.7**	76.4 / **58.3**	68.2 / 51.7	**74.3 / 58.1**	66.5 / **56.7**	68.3 / 52.8	73.7 / 53.8	51.3 / 42.0	70.6 / 55.5
XLM-R^{Base} MLM-TRG	**84.7 / 72.6**	**67.0 / 49.2**	73.7 / 58.8	**73.2** / 55.7	**76.6 / 58.3**	**69.8 / 53.6**	**74.3** / 57.9	67.0 / 55.8	**68.6 / 53.0**	**75.5 / 54.9**	52.2 / **43.1**	**71.1 / 55.7**
MAD-X^{Base} – INV	83.3 / 72.1	64.0 / 47.1	72.0 / 55.8	71.0 / 52.9	74.6 / 55.5	67.3 / 51.0	72.1 / 55.1	64.1 / 51.8	66.2 / 49.6	73.0 / 53.6	50.9 / 40.6	67.0 / 53.2
MAD-X^{Base}	83.5 / **72.6**	65.5 / 48.2	72.9 / 56.0	72.9 / 54.6	75.9 / 56.9	68.2 / 51.3	73.1 / 56.7	**67.8** / 55.9	67.0 / 49.8	73.7 / 53.3	**52.7** / 42.8	70.3 / 54.4

Table 4: F_1 / EM scores on XQuAD with English as the source language for each target language.

ting from Ponti et al. (2020a)—fine-tuning first on SIQA (Sap et al., 2019) and on the English COPA training set—and report other possible settings in the appendix. Target language adaptation outperforms XLM-R^{Base} while MAD-X^{Base} achieves the best scores. It shows gains in particular for the two unseen languages, Haitian Creole (ht) and Quechua (qu). Performance on the other languages is also generally competitive or better.

Question Answering The results on XQuAD when transferring from English to each target language are provided in Table 4. The main finding is that MAD-X achieves similar performance to the XLM-R baseline. As before, invertible adapters generally improve performance and target language adaptation improves upon the baseline setting. We note that all languages included in XQuAD can be considered high-resource, with more than 100k Wikipedia articles each (cf. Wikipedia sizes of NER languages in Table 1). The corresponding setting can be found in the top-left quadrant in Figure 3 where relative differences are comparable.

These and XCOPA results demonstrate that, while MAD-X excels at transfer to unseen and low-resource languages, it achieves competitive performance even for high-resource languages and on more challenging tasks. These evaluations also hint at the modularity of the adapter-based MAD-X approach, which holds promise of quick adaptation to more tasks: we use exactly the same language-specific adapters in NER, CCR, and QA for languages such as English and Mandarin Chinese that appear in all three evaluation language samples.

7 Further Analysis

Impact of Invertible Adapters We also analyse the relative performance difference of MAD-X

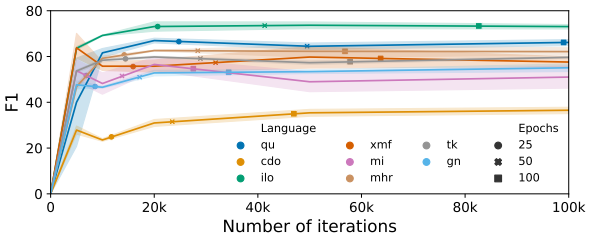


Figure 4: Cross-lingual NER performance of MAD-X transferring from English to the target languages with invertible and language adapters trained on target language data for different numbers of iterations. Shaded regions denote variance in F_1 scores across 5 runs.

Model		+ Params	% Model
MAD-X^{Base}		8.25M	3.05
MAD-X^{Base}	– INV	7.96M	2.94
MAD-X^{Base}	– LAD – INV	0.88M	0.32

Table 5: Number of parameters added to XLM-R Base, and as a fraction of its parameter budget (270M).

with and without invertible adapters for each source language–target language pair on the NER data set (see Section D in the appendix). Invertible adapters improve performance for many transfer pairs, and particularly when transferring to low-resource languages. Performance is only consistently lower with a single low-resource language as source (Maori), likely due to variation in the data.

Sample Efficiency The main adaptation bottleneck of MAD-X is training language adapters and invertible adapters. However, due to the modularity of MAD-X, once trained, these adapters have an advantage of being directly reusable (i.e., "plug-and-playable") across different tasks (see the discussion in §6). To estimate the sample efficiency of adapter training, we measure NER performance on

several low-resource target languages (when transferring from English as the source) conditioned on the number of training iterations. The results are given in Figure 4. They reveal that we can achieve strong performance for the low-resource languages already at 20k training iterations, and longer training offers modest increase in performance.

Moreover, in Table 5 we present the number of parameters added to the original XLM-R Base model per language for each MAD-X variant. The full MAD-X model for NER receives an additional set of 8.25M adapter parameters for every language, which makes up only 3.05% of the original model.

8 Conclusion

We have proposed MAD-X, a general modular framework for transfer across tasks and languages. It leverages a small number of additional parameters to mitigate the capacity issue which fundamentally hinders current multilingual models. MAD-X is model-agnostic and can be adapted to any current pre-trained multilingual model as foundation. We have shown that it is particularly useful for adapting to languages not covered by the multilingual model's training model, while also achieving competitive performance on high-resource languages.

In future work, we will apply MAD-X to other pre-trained models, and employ adapters that are particularly suited for languages with certain properties (e.g. with different scripts). We will also evaluate on additional tasks, and investigate leveraging pre-trained language adapters from related languages for improved transfer to truly low-resource languages with limited monolingual data.

Acknowledgments

Jonas Pfeiffer is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. The work of Ivan Vulić is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). We thank Laura Rimell for feedback on a draft.

We would like to thank Isabel Pfeiffer for the logo illustrations.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 4623–4637.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Multilingual Alignment of Contextual Word Representations. In *8th International Conference on Learning Representations, ICLR 2020, Virtual Conference, April 26 - May 1, 2020*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 8440–8451.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7057–7067.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kardas, Sylvain Gugger, and Jeremy Howard. 2019. Multifit: Efficient multi-lingual language model fine-tuning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5701–5706.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Conference of the Association*

for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 710–721.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 748–756.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 12-18 July 2020, Virtual Conference*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint*.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific BERT models. *arXiv preprint*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1946–1958.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020b. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), EMNLP 2020, Virtual Conference, 2020*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4996–5001.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020a. XCOPA: A Multilingual Dataset for Causal Commonsense Reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Virtual Conference*.

Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Marinela Parovic, Roi Reichart, and Anna Korhonen. 2020b. Parameter space factorization for zero-shot learning across tasks and languages. *Transactions of the Association for Computational Linguistics 2020*.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 151–164.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna

Gurevych. 2020. AdapterDrop: On the Efficiency of Adapters in Transformers. *arXiv preprint*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *arXiv preprint*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5986–5995.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language Adaptation for Truly Universal Dependency Parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Virtual Conference*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for Finnish. *arXiv preprint*.

Zirui Wang, Jiateng Xie, Ruochen Xu, Yiming Yang, Graham Neubig, and Jaime Carbonell. 2020. Cross-lingual Alignment vs Joint Training: A Comparative Study and A Simple Unified Framework. In *8th International Conference on Learning Representations, ICLR 2020, Virtual Conference, April 26 - May 1, 2020*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 4003–4012.

Georg Wiese, Dirk Weissenborn, and Mariana L. Neves. 2017. Neural Domain Adaptation for Biomedical Question Answering. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 281–289.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi andArt Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), EMNLP 2020, Virtual Conference, 2020*.

Shijie Wu, Alexis Conneau, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 6022–6034.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 833–844.

A Evaluation data

- Named Entity Recognition (NER). Data: WikiANN (Rahimi et al., 2019). Available online at:
  `www.amazon.com/clouddrive/share/d3KGCRCIYwhKJF0H3eWA26hjg2ZCRhjpEQtDL70FSBN`.
- Causal Commonsense Reasoning (CCR). Data: XCOPA (Ponti et al., 2020a). Available online at:
  `github.com/cambridgeltl/xcopa`
- Question Answering (QA). Data: XQuAD (Artetxe et al., 2020). Available online at:
  `github.com/deepmind/xquad`

B NER zero-shot results from English

We show the F1 scores when transferring from English to the other languages averaged over five runs in Table 6.

C NER results per language pair

We show the F1 scores on the NER dataset across all combinations of source and target language for all of our comparison methods in Figures 5 (XLM-R^{Base}), 6 (XLM-R^{Base} MLM-SRC), 7 (XLM-R^{Base} MLM-TRG), 8 (MAD-X^{Base} –

	en	ja	zh	ar	jv	sw	is	my	qu	cdo	ilo	xmf	mi	mhr	tk	gn	avg
mBERT	84.8	**26.7**	**38.5**	38.7	**57.8**	66.0	65.7	42.9	54.9	14.20	63.5	31.1	21.8	46.0	47.2	45.4	44.0
XLM-R	83.0	15.2	19.6	41.3	56.1	63.5	67.2	46.9	58.3	20.47	61.3	32.2	15.9	41.8	43.4	41.0	41.6
XLM-R^{Base} MLM-SRC	84.2	8.45	11.0	27.3	44.8	57.9	59.0	35.6	52.5	21.4	60.3	22.7	22.7	38.1	44.0	41.7	36.5
XLM-R^{Base} MLM-TRG	84.2	9.30	15.5	**44.5**	50.2	**77.7**	71.7	**55.5**	**68.7**	**47.6**	**84.7**	**60.3**	43.6	56.3	56.4	50.6	52.8
MAD-X – LAD – inv	82.0	15.6	20.3	41.0	54.4	66.4	67.8	48.8	57.8	16.9	59.9	36.9	14.3	44.3	41.9	42.9	41.9
MAD-X – INV	82.2	16.8	20.7	36.9	54.1	68.7	71.5	50.0	59.6	39.2	69.9	54.9	48.3	58.1	53.1	52.8	50.3
MAD-X	82.3	19.0	20.5	41.8	55.7	73.8	**74.5**	51.9	66.1	36.5	73.1	57.6	**51.0**	**62.1**	**59.7**	**55.1**	**53.2**

Table 6: NER F1 scores for zero-shot transfer from English.

LAD – INV), 9 (MAD-X^{Base} – INV), 10 (MAD-X^{Base}), 11 (mBERT), 12 (MAD-X^{mBERT}) , 13 (XLM-R^{Large}), and 14 (MAD-X^{mBERT}). Each score is averaged over five runs.

D Relative improvement of MAD-X over baselines in cross-lingual NER transfer

We show the heatmaps which depict relative F1 improvements of the full MAD-X^{Base} framework in the cross-lingual NER transfer task over: (a) the baseline model XLM-R^{Base} MLM-TRG (Figure 15) and (b) the MAD-X^{Base} variant without invertible adapters: MAD-X^{Base} –INV (Figure 16).

The heatmap which depicts relative F1 improvements of the full MAD-X^{mBERT} framework over mBERT can be found in Figure 17.

Finally, the heatmap which depicts relative F1 improvements of the full MAD-X^{Large} framework over XLM-R^{Large} can be found in Figure 18.

E XCOPA results for all settings

We show the results on XCOPA for all fine-tuning settings in Table 7.

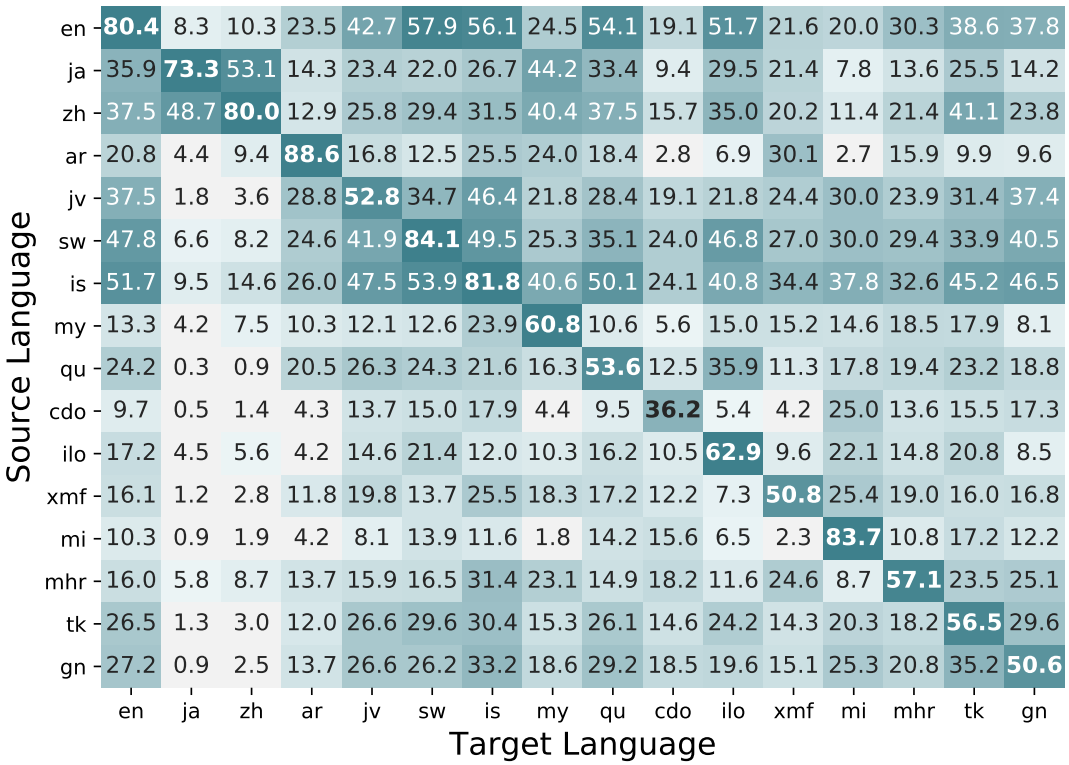


Figure 5: Mean F1 scores of XLM-R^{Base} in the standard setting (XLM-R^{Base}) for cross-lingual transfer on NER.

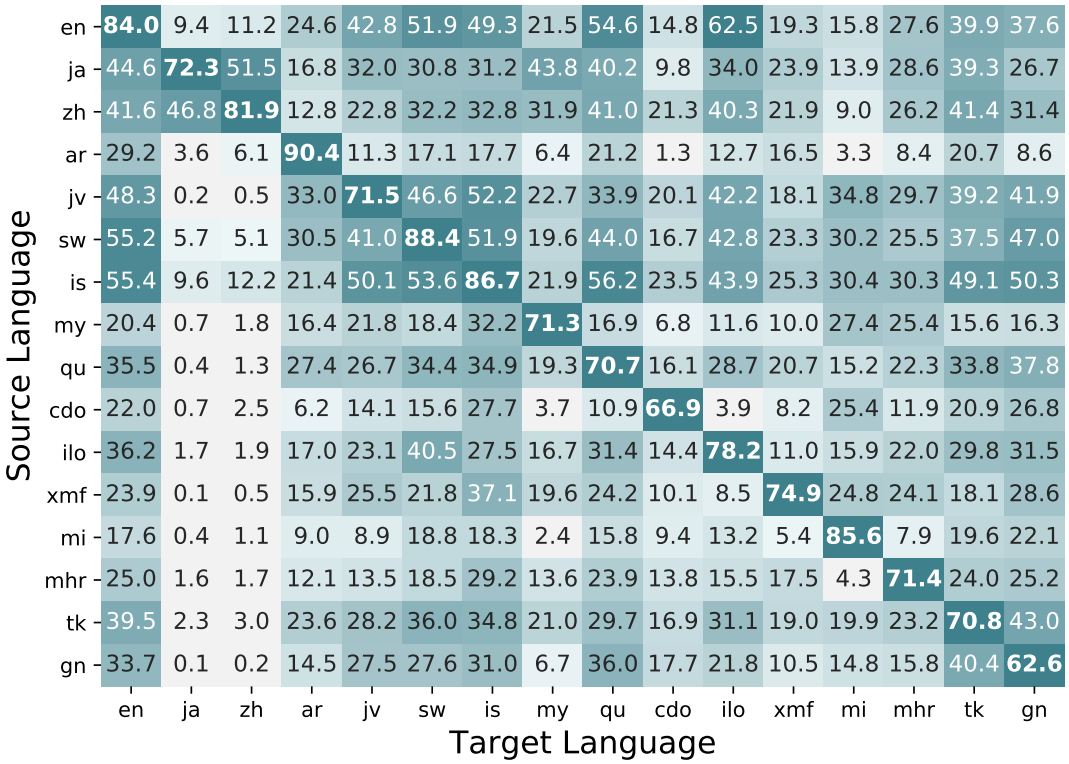


Figure 6: Mean F1 scores of XLM-R^{Base} with MLM fine-tuning on source language data (XLM-R^{Base} MLM-SRC) for cross-lingual transfer on NER.

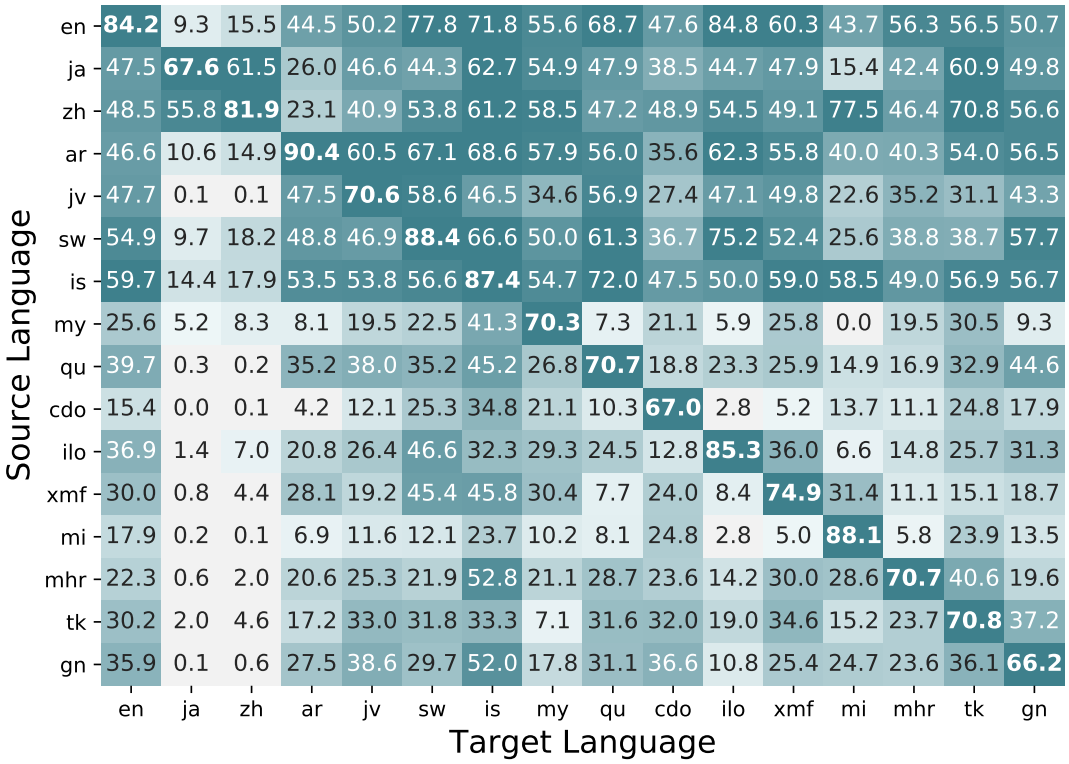


Figure 7: Mean F1 scores of XLM-R^{Base} with MLM fine-tuning on target language data (XLM-R^{Base} MLM-TRG) for cross-lingual transfer on NER.

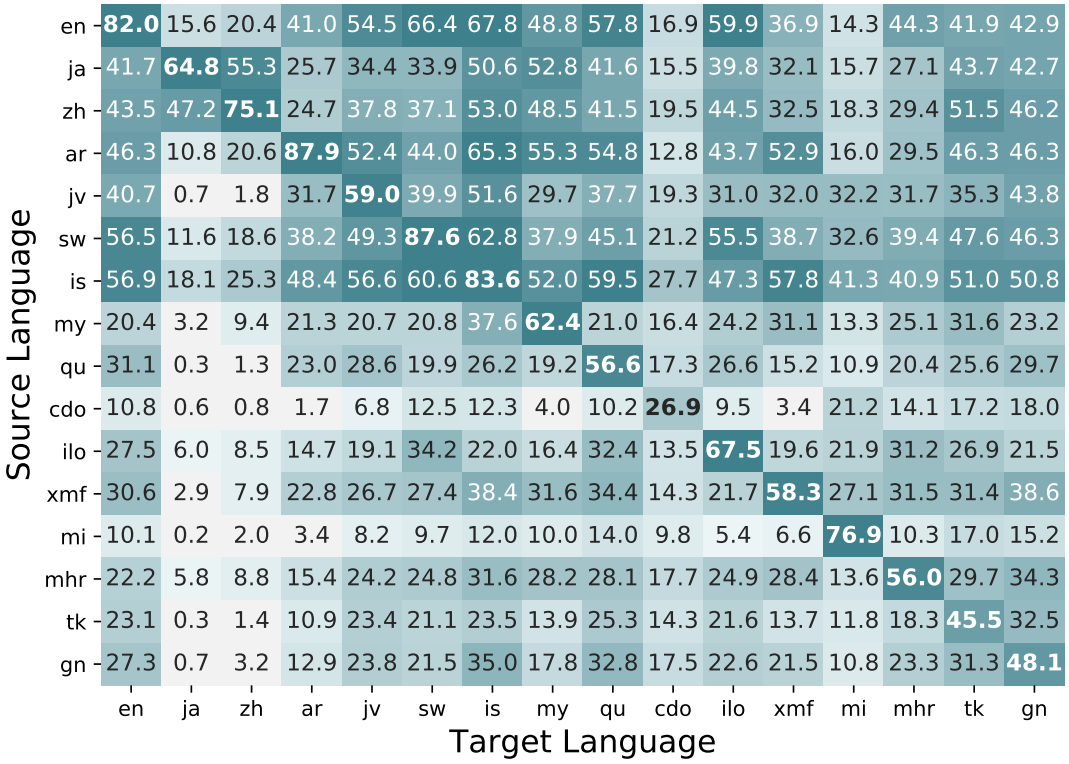


Figure 8: Mean F1 scores of our framework without language adapters and invertible adapters (MAD-X^{Base} – LAD – INV) for cross-lingual transfer on NER.

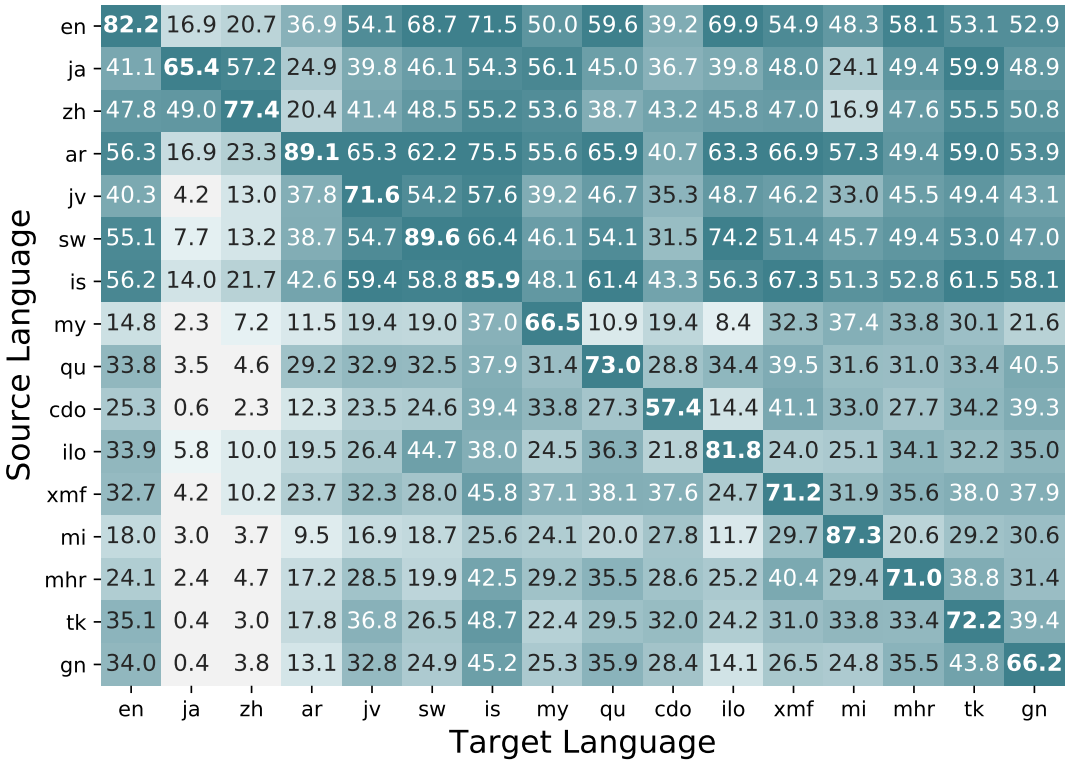


Figure 9: Mean F1 scores of our framework without invertible adapters (MAD-X^{Base} – INV) for cross-lingual transfer on NER.

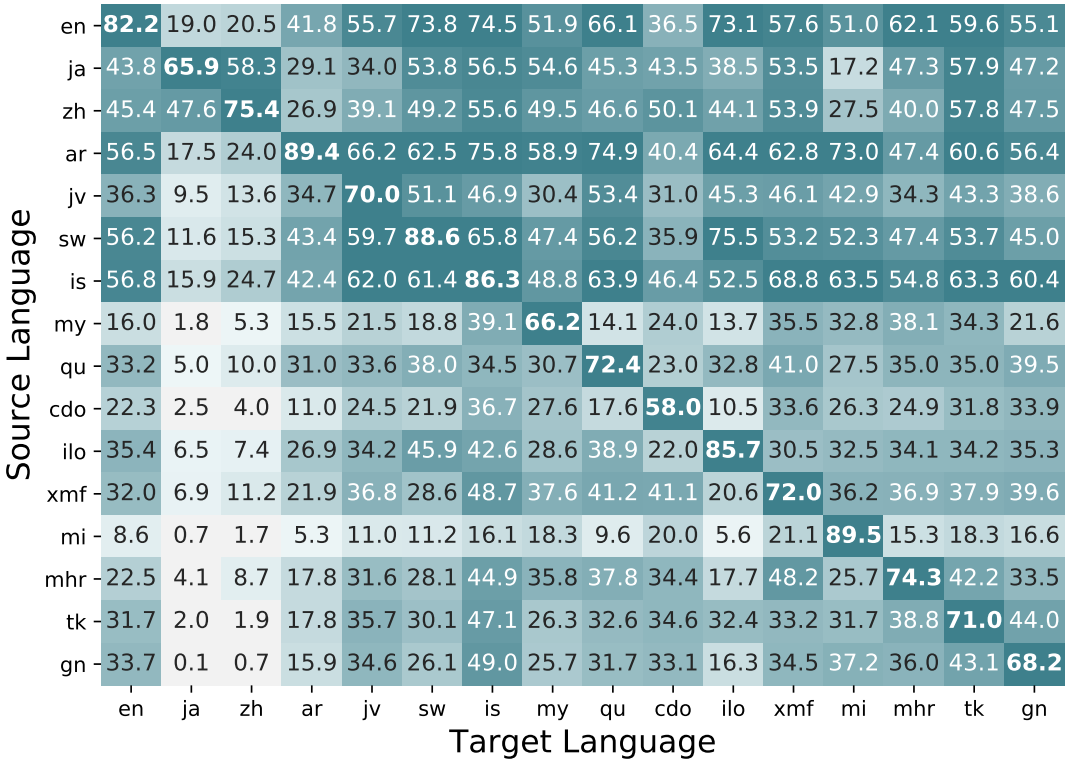


Figure 10: Mean F1 scores of our complete adapter-based framework (MAD-X^{Base}) for cross-lingual transfer on NER.

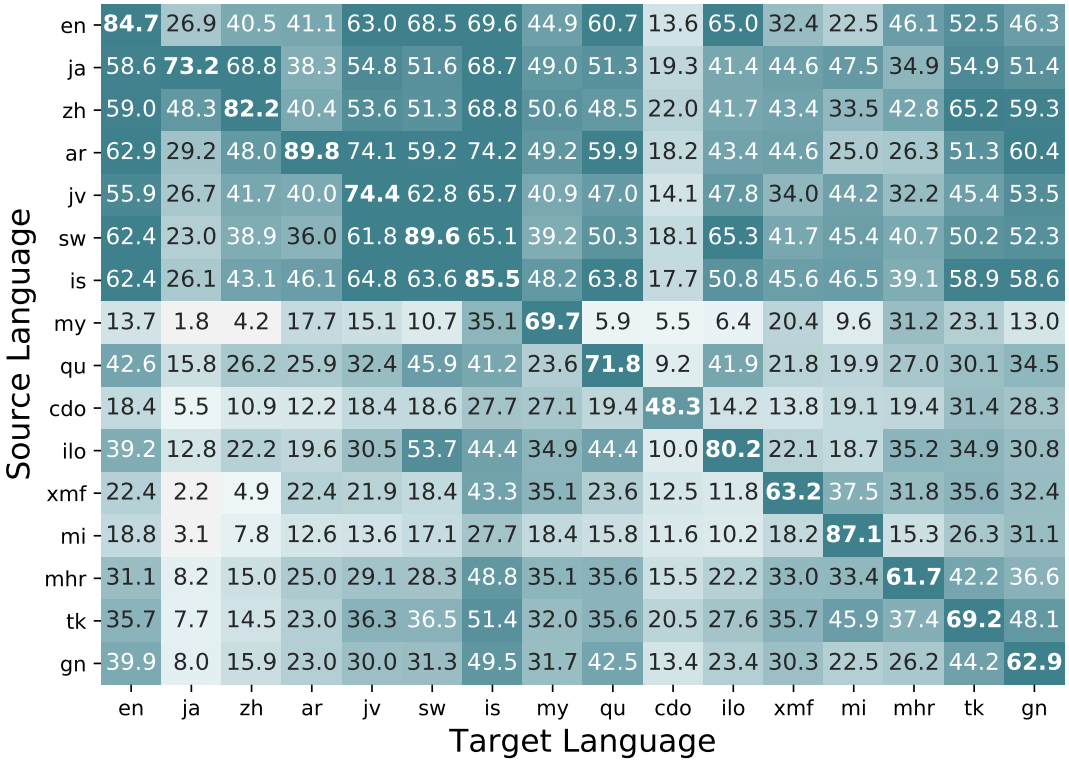


Figure 11: Mean F1 scores of mBERT for cross-lingual transfer on NER.

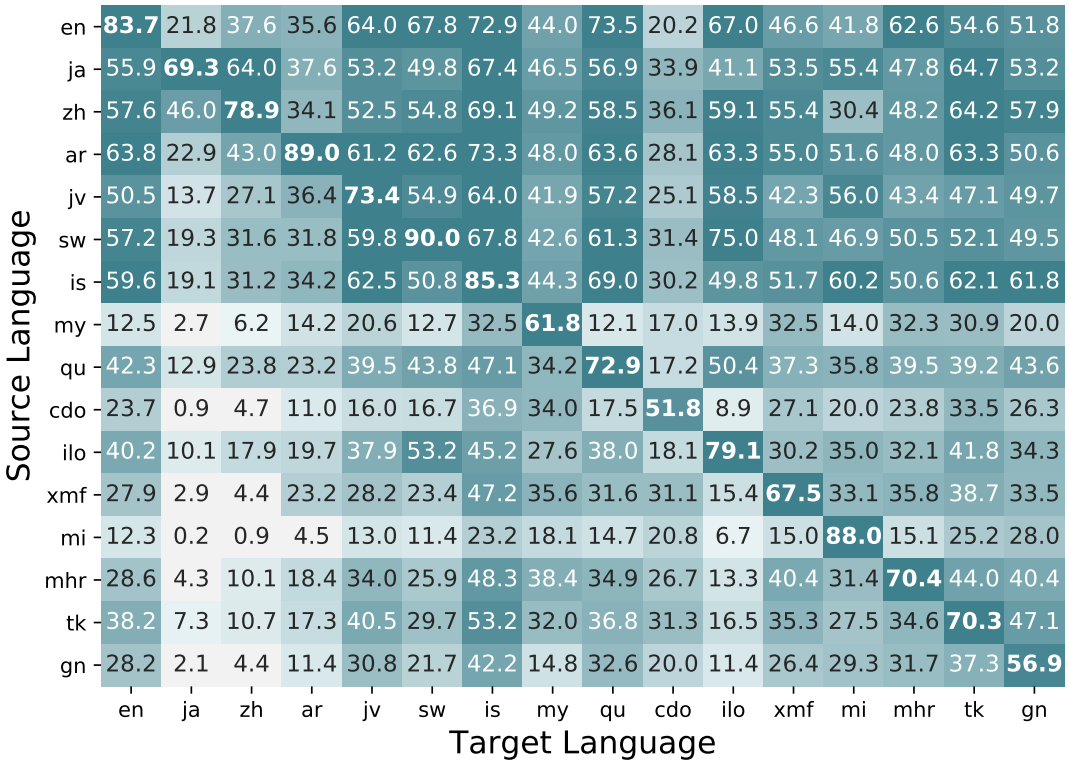


Figure 12: Mean F1 scores of our complete adapter-based framework (MAD-X^{mBERT}) for cross-lingual transfer on NER.

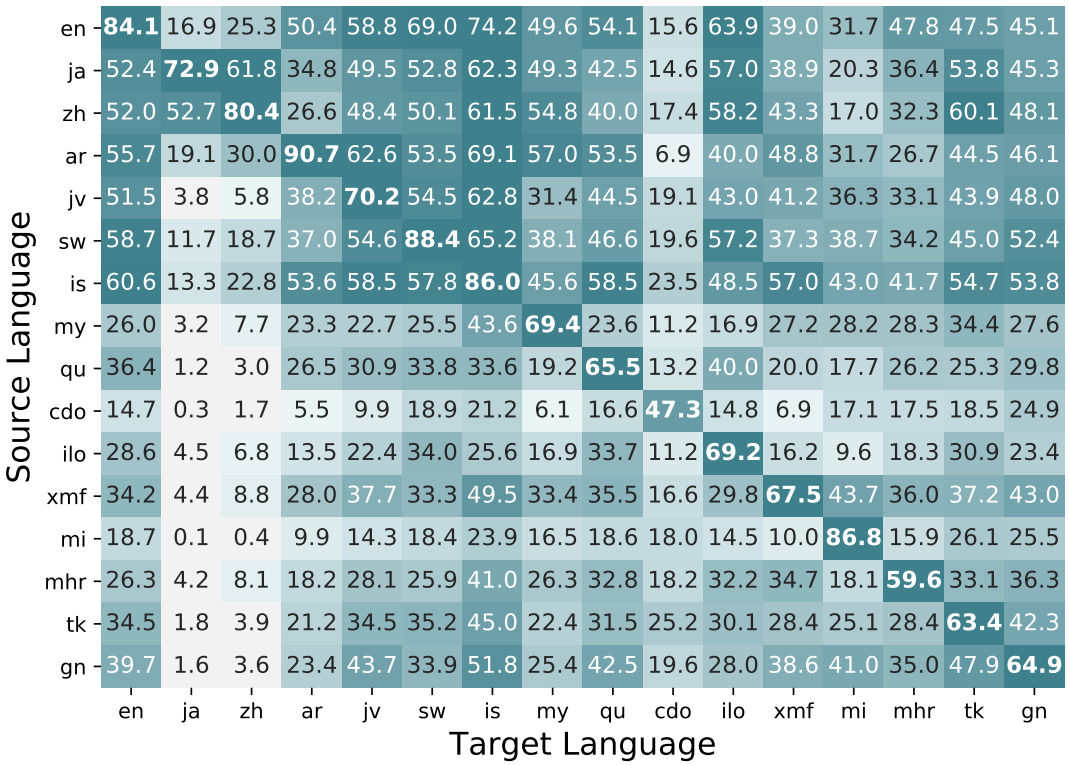


Figure 13: Mean F1 scores of XLM-R^{Large} for cross-lingual transfer on NER.

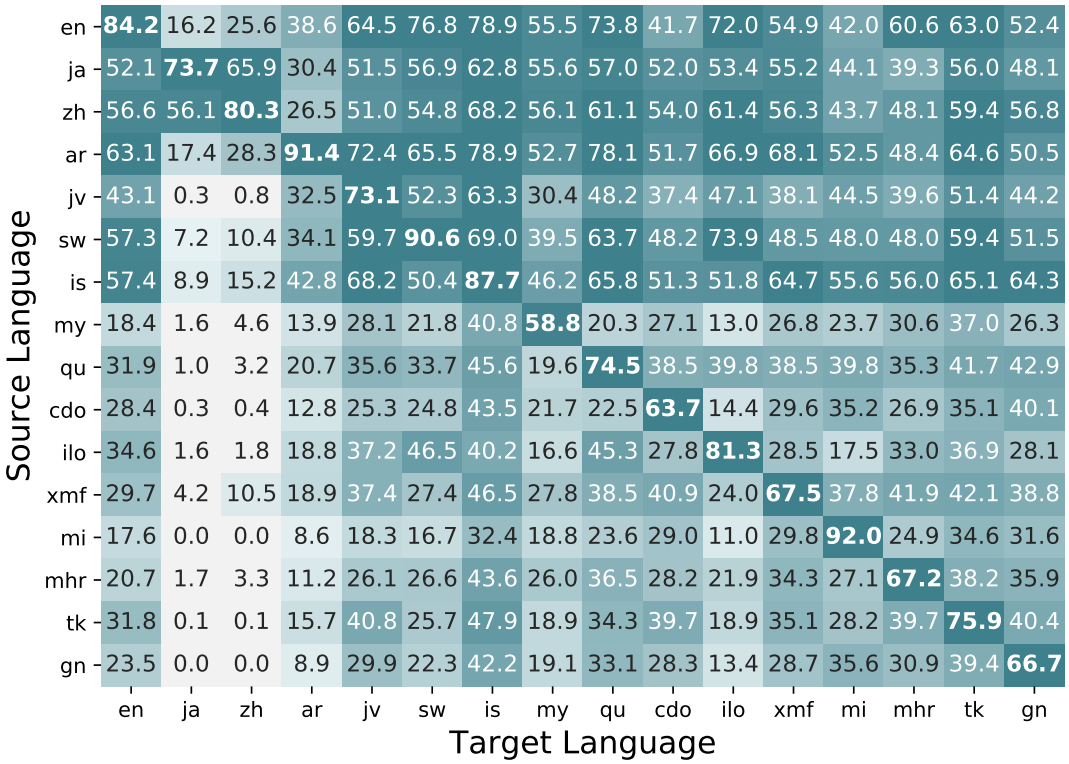


Figure 14: Mean F1 scores of our complete adapter-based framework (MAD-X^{Large}) for cross-lingual transfer on NER.

Model	en	et	ht	id	it	qu	sw	ta	th	tr	vi	zh	avg
XLM-R$^{Base}_{\rightarrow COPA}$	57.6	59.8	49.4	58.0	56.0	50.7	57.2	56.6	52.8	56.2	58.5	56.6	55.8
XLM-R^{Base} MLM-TRG$_{\rightarrow COPA}$	57.6	57.8	48.6	60.8	54.4	49.5	55.4	55.8	54.2	54.8	57.6	57.2	55.3
XLM-R$^{Base}_{\rightarrow SIQA}$	68.0	59.4	49.2	67.2	63.6	51.0	57.6	58.8	61.6	60.4	65.8	66.0	60.7
XLM-R$^{Base}_{\rightarrow SIQA \rightarrow COPA}$	66.8	58.0	51.4	65.0	60.2	51.2	52.0	58.4	62.0	56.6	65.6	**68.8**	59.7
XLM-R^{Base} MLM-TRG$_{\rightarrow SIQA \rightarrow COPA}$	66.8	59.4	50.0	**71.0**	61.6	46.0	**58.8**	60.0	**63.2**	**62.2**	**67.6**	67.4	61.2
MAD-X$^{Base}_{\rightarrow COPA}$	48.1	49.0	51.5	50.7	50.7	49.1	52.7	52.5	48.7	53.3	52.1	50.4	50.7
MAD-X$^{Base}_{\rightarrow SIQA}$	67.6	59.7	51.7	66.2	**64.4**	**54.0**	53.9	61.3	61.1	60.1	65.4	66.7	61.0
MAD-X$^{Base}_{\rightarrow SIQA \rightarrow COPA}$	**68.3**	**61.3**	**53.7**	65.8	63.0	52.5	56.3	**61.9**	61.8	60.3	66.1	67.6	**61.5**

Table 7: Accuracy scores of all models on the XCOPA test sets when transferring from English. Models are either only fine-tuned on the COPA training set ($\rightarrow COPA$), only fine-tuned on the SIQA training set ($\rightarrow SIQA$) or fine-tuned first on SIQA and then on COPA ($\rightarrow SIQA \rightarrow COPA$).

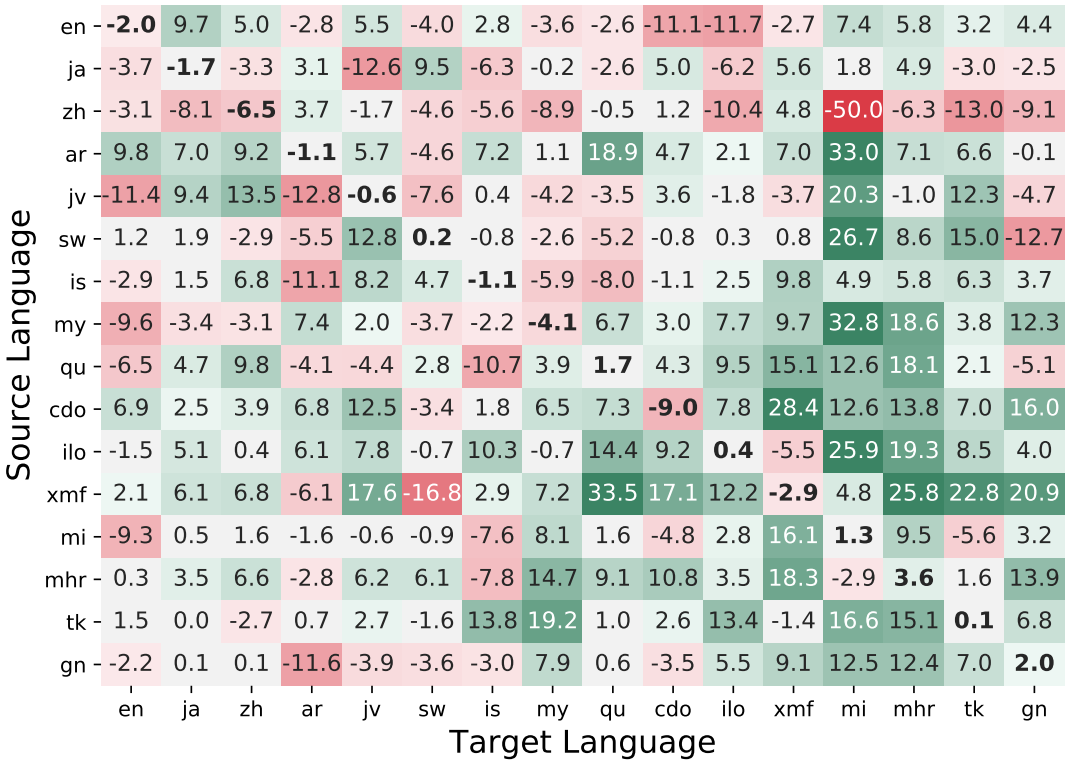


Figure 15: Relative F_1 improvement of MAD-X^{Base} over XLM-R^{Base} MLM-TRG in cross-lingual NER transfer.

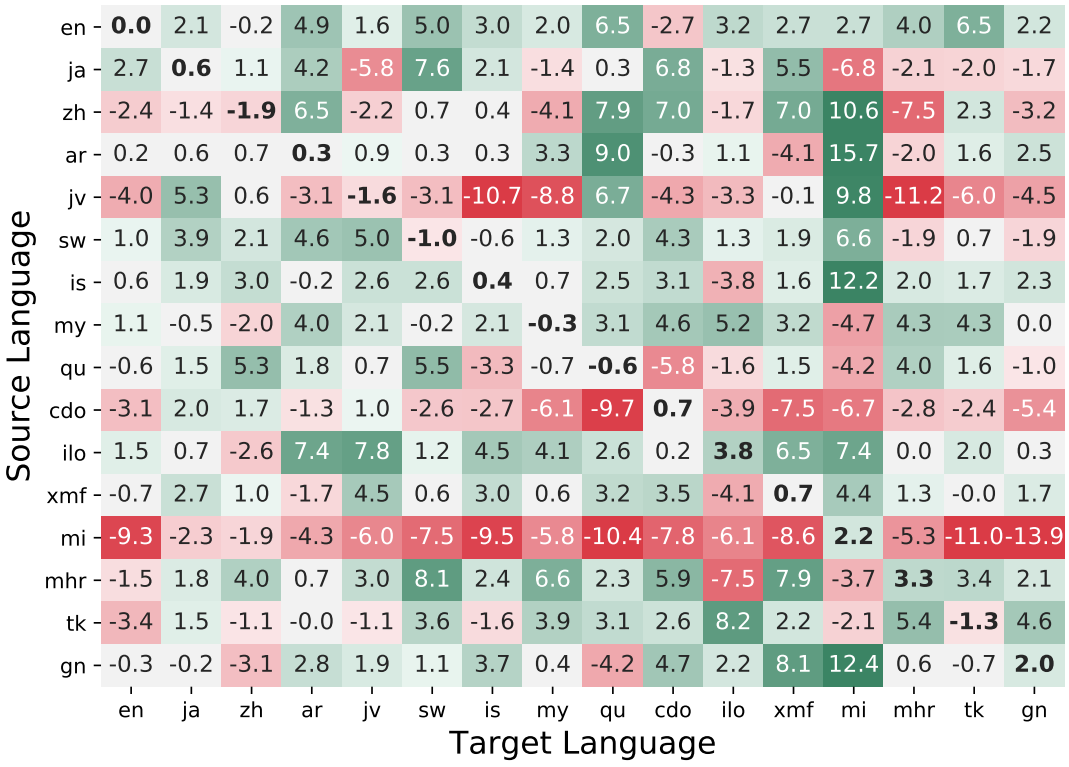


Figure 16: Relative F_1 improvement of MAD-X^{Base} over MAD-X^{Base} –INV in cross-lingual NER transfer.

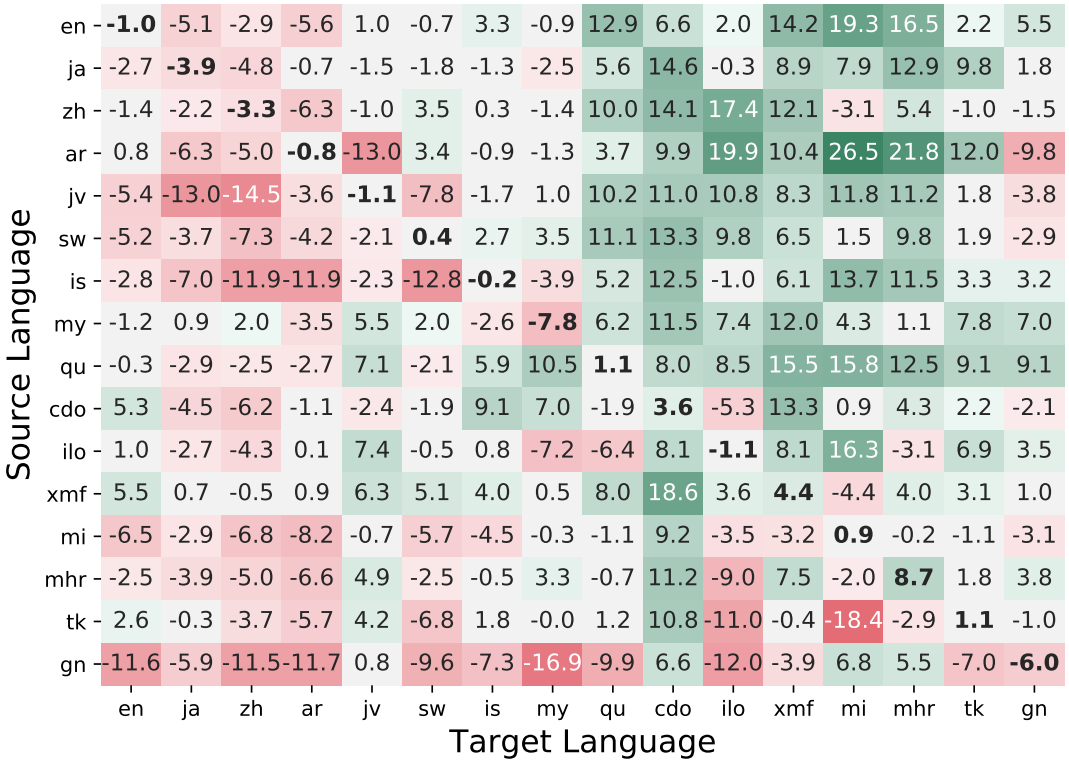


Figure 17: Relative F_1 improvement of MAD-X^{mBERT} over mBERT in cross-lingual NER transfer.

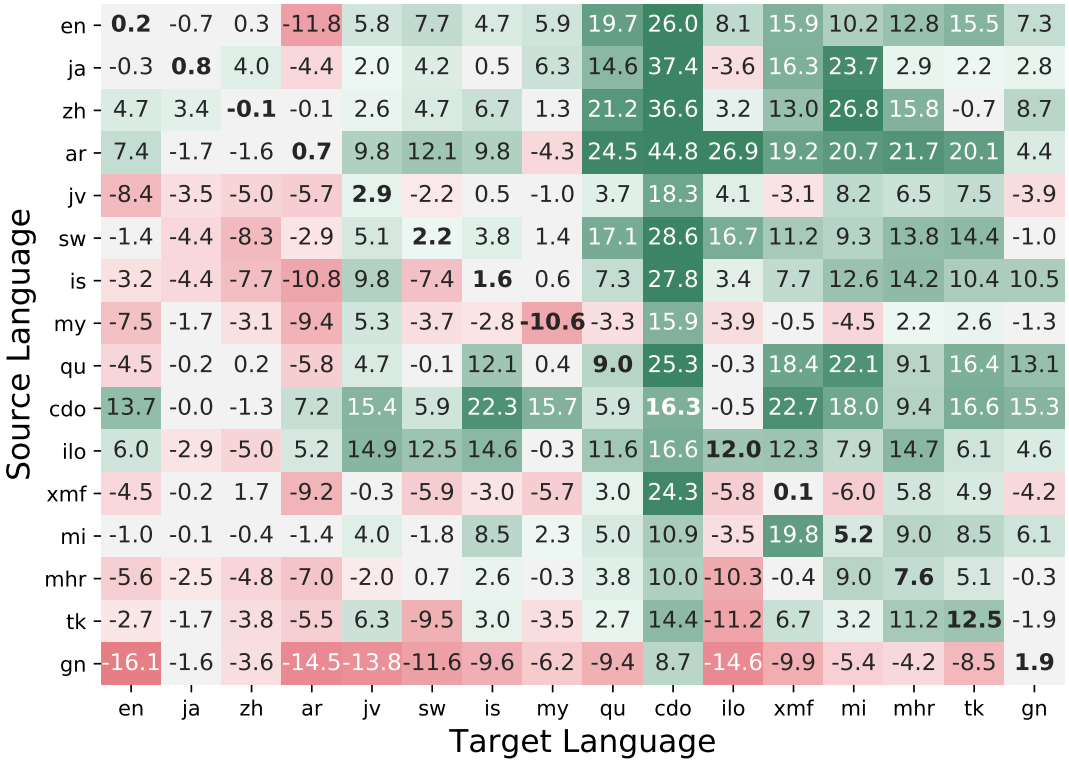


Figure 18: Relative F_1 improvement of MAD-X^{Large} over XLM-R^{Large} in cross-lingual NER transfer.

Chapter 9

UNKs Everywhere: Adapting Multilingual Language Models to New Scripts

UNKs Everywhere: Adapting Multilingual Language Models to New Scripts

Jonas Pfeiffer[1], **Ivan Vulić**[2], **Iryna Gurevych**[1], **Sebastian Ruder**[3]
[1]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[2]Language Technology Lab, University of Cambridge
[3]DeepMind
pfeiffer@ukp.tu-darmstadt.de

Abstract

Massively multilingual language models such as multilingual BERT offer state-of-the-art cross-lingual transfer performance on a range of NLP tasks. However, due to limited capacity and large differences in pretraining data sizes, there is a profound performance gap between resource-rich and resource-poor target languages. The ultimate challenge is dealing with under-resourced languages not covered at all by the models and written in scripts *unseen* during pretraining. In this work, we propose a series of novel data-efficient methods that enable quick and effective adaptation of pretrained multilingual models to such low-resource languages and unseen scripts. Relying on matrix factorization, our methods capitalize on the existing latent knowledge about multiple languages already available in the pretrained model's embedding matrix. Furthermore, we show that learning of the new dedicated embedding matrix in the target language can be improved by leveraging a small number of vocabulary items (i.e., the so-called *lexically overlapping* tokens) shared between mBERT's and target language vocabulary. Our adaptation techniques offer substantial performance gains for languages with unseen scripts. We also demonstrate that they can yield improvements for low-resource languages written in scripts covered by the pretrained model.

1 Introduction

Massively multilingual language models pretrained on large multilingual data, such as multilingual BERT (mBERT; Devlin et al., 2019) and XLM-R (Conneau et al., 2020a) are the current state-of-the-art vehicle for effective cross-lingual transfer (Hu et al., 2020). However, while they exhibit strong transfer performance between resource-rich and similar languages (Conneau et al., 2020a; Artetxe et al., 2020), these models struggle with transfer to low-resource languages (Wu and Dredze, 2020) and languages not represented at all in their pretraining corpora (Pfeiffer et al., 2020b; Müller et al., 2021; Ansell et al., 2021). The most extreme challenge is dealing with *unseen languages with unseen scripts* (i.e., the scripts are not represented in the pretraining data; see Figure 1), where the pretrained models are bound to fail entirely if they are used off-the-shelf without any *further model adaptation*.

Amharic	Tibetan	Divehi
የእግዚአብሔርን	[illegible]	[illegible]
የሚሰጡዋቸውን	[illegible]	[illegible]
የእግዚአብሔር	[illegible]	[illegible]
ቤተክርስቲያን	[illegible]	[illegible]
ኢትዮጵያውያን	[illegible]	[illegible]

Figure 1: Example tokens of unseen scripts.

Existing work focuses on the embedding layer and learns either a new embedding matrix for the target language (Artetxe et al., 2020) or adds new tokens to the pretrained vocabulary. While the former has only been applied to high-resource languages, the latter approaches have been limited to languages with seen scripts (Chau et al., 2020; Müller et al., 2021) and large pretraining corpora (Wang et al., 2020). Another line of work adapts the embedding layer as well as other layers of the model via adapters (Pfeiffer et al., 2020b; Üstün et al., 2020). Such methods, however, cannot be directly applied to languages with unseen scripts.

In this work, we first empirically verify that the original tokenizer and the original embedding layer of a pretrained multilingual model fail for languages with unseen script. This implies that dedicated in-language tokenizers and embeddings are a crucial requirement for any successful model adaptation. The key challenge is aligning new target language embeddings to the pretrained model's representations while leveraging knowledge encoded in the existing embedding matrix. We systematize existing approaches based on the pretrained information they utilize and identify *lexically overlapping tokens* that are present in both vocabularies as key carriers of such information (Søgaard et al.,

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 10186–10203
November 7–11, 2021.

2018).[1] We then present novel, effective, and data-efficient methods for adapting pretrained multilingual language models to resource-low languages written in different scripts. Beyond lexical overlap, our methods rely on factorized information from the embedding matrix and token groupings.

We evaluate our approaches in the named entity recognition (NER) task on the standard WikiAnn dataset (Rahimi et al., 2019) and Dependency Parsing (DP; Nivre et al., 2016). We use 4 diverse resource-rich languages as source languages, and transfer to 17 and 6 resource-poor target languages respectively, including 5 languages with unseen scripts (Amharic, Tibetan, Khmer, Divehi, Sinhala). We show that our adaptation techniques offer unmatched performance for languages with unseen scripts. They also yield improvements for low-resource and under-represented languages written in scripts covered by the pretrained model.

Contributions. **1)** We systematize and compare current model adaptation strategies for low-resource languages with seen and unseen scripts. **2)** We measure the impact of initialization when learning new embedding layers, and demonstrate that non-random initialization starting from a subset of seen lexical items (i.e., *lexically overlapping* vocabulary items) has a strong positive impact on task performance for resource-poor languages. **3)** We propose methods for learning low-dimensional embeddings, which reduce the number of trainable parameters and yield more efficient model adaptation. Our approach, based on matrix factorization and language clusters, extracts relevant information from the pretrained embedding matrix. **4)** We show that our methods outperform previous approaches with both resource-rich and resource-poor languages. They substantially reduce the gap between random and lexically-overlapping initialization, enabling better model adaption to unseen scripts.

The code for this work is released at github.com/Adapter-Hub/UNKs_everywhere.

2 Background: Multilingual Model Adaptation for Cross-lingual Transfer

Recent language models (Devlin et al., 2019; Conneau et al., 2020a), based on Transformer architectures (Vaswani et al., 2017) and pretrained on massive amounts of multilingual data, have recently surpassed (static) cross-lingual word embedding spaces (Ruder et al., 2019; Glavas et al., 2019) as the state-of-the-art paradigm for cross-lingual transfer in NLP (Pires et al., 2019; Wu and Dredze, 2019; Wu et al., 2020; Hu et al., 2020; K et al., 2020). However, recent studies have also indicated that even current state-of-the-art models such as XLM-R (Large) still do not yield reasonable transfer performance across a large number of target languages (Hu et al., 2020). The largest drops are reported for resource-poor target languages (Lauscher et al., 2020), and (even more dramatically) for languages not covered at all during pretraining (Pfeiffer et al., 2020b).

Standard Cross-Lingual Transfer Setup with a state-of-the-art pretrained multilingual model such as mBERT or XLM-R is **1)** fine-tuning it on labelled data of a downstream task in a source language and then **2)** applying it directly to perform inference in a target language (Hu et al., 2020). However, as the model must balance between many languages in its representation space, it is not suited to excel at a specific language at inference time without further adaptation (Pfeiffer et al., 2020b).

Adapters for Cross-lingual Transfer. Adapter-based approaches have been proposed as a remedy (Rebuffi et al., 2017, 2018; Houlsby et al., 2019; Stickland and Murray, 2019; Bapna and Firat, 2019; Pfeiffer et al., 2020a, 2021). In the cross-lingual setups, the idea is to increase the multilingual model capacity by storing language-specific knowledge of each language in dedicated parameters (Pfeiffer et al., 2020b; Vidoni et al., 2020). We start from MAD-X (Pfeiffer et al., 2020b), a state-of-the-art adapter-based framework for cross-lingual transfer. For completeness, we provide a brief overview of the framework in what follows.

MAD-X comprises three adapter types: language, task, and invertible adapters; this enables learning language and task-specific transformations in a modular and parameter-efficient way. As in prior work (Rebuffi et al., 2017; Houlsby et al., 2019), adapters are trained while keeping the parameters of the pretrained multilingual model fixed. *Language adapters* are trained via masked language modeling (MLM) on unlabelled target language data. *Task adapters* are trained via task-specific objectives on labelled task data in a source language while also keeping the language adapters fixed. Task and language adapters are stacked: this enables the adaptation of the pretrained multilin-

[1] Even languages with unseen scripts share some tokens, e.g. numbers, foreign named entities written in their original scripts, etc. with seen languages.

gual model to languages not covered in its pretraining data. MAD-X keeps the same task adapter while substituting the source language adapter with the target language adapter at inference.

In brief, the adapters A_l at layer l consist of a down-projection $\mathbf{D} \in \mathbb{R}^{h \times d}$ where h is the hidden size of the Transformer model and d is the dimension of the adapter, followed by a GeLU activation (Hendrycks and Gimpel, 2016) and an up-projection $\mathbf{U} \in \mathbb{R}^{d \times h}$ at every layer l:

$$\mathsf{A}_l(\mathbf{h}_l, \mathbf{r}_l) = \mathbf{U}_l(\mathsf{GeLU}(\mathbf{D}_l(\mathbf{h}_l))) + \mathbf{r}_l. \quad (1)$$

$\mathbf{h}_l$ and $\mathbf{r}_l$ are the Transformer hidden state and the residual at layer l, respectively. The residual connection $\mathbf{r}_l$ is the output of the Transformer's feed-forward layer whereas $\mathbf{h}_l$ is the output of the subsequent layer normalization. For further technical details, we refer the reader to Pfeiffer et al. (2020b).

Current model adaptation approaches (Chau et al., 2020; Wang et al., 2020) generally fine-tune *all* model parameters on target language data. Instead, we follow the more computationally efficient adapter-based paradigm where we keep model parameters fixed, and only train language adapters and target language embeddings. Crucially, while the current adapter-based methods offer extra capacity, they do not offer mechanisms to deal with extended vocabularies of many resource-poor target languages, and do not adapt their representation space towards the target language adequately. This problem is exacerbated when dealing with unseen languages and scripts.[2]

3 Cross-lingual Transfer of Lexical Information

The embedding matrix of large multilingual models makes up around 50% of their entire parameter budget (Chung et al., 2021). However, it is not clear how to leverage this large amount of information most effectively for languages that are not adequately represented in the shared multilingual vocabulary due to lack of pretraining data.

A key challenge in using the lexical information encoded in the embedding matrix is to overcome a mismatch in vocabulary between the pretrained model and the target language. To outline this issue, in Table 1 we show for the languages in our NER evaluation the proportion of tokens in each language that are effectively *unknown* (UNK) to mBERT: they occur in the vocabulary of a separately trained monolingual tokenizer (§4.3), but cannot even be composed by subword tokens from the original mBERT's vocabulary. Table 1 also provides the proportion of *lexically overlapping* tokens, i.e., tokens that are present both in mBERT's and monolingual in-language vocabularies. The zero-shot performance of mBERT generally deteriorates with less lexical overlap and more UNKs in a target language: see Figure 2. Pearson's ρ correlation scores between the lexical overlap and proportion of UNKs (see Table 1) and NER performance are 0.443 and -0.798, respectively.

Language	iso	Family	Script	% UNKs	% Lex Overl.
English*	en	Indo-Europ.	Latin	0%	66%
Chinese*	zh	Sino-Tibetan	Chinese	0%	79%
Japanese*	ja	Japonic	Japanese	0%	99%
Arabic*	ar	Afro-Asiatic	Arabic	1%	6%
Georgian	ka	Kartvelian	Georgian	2%	27%
Urdu	ur	Indo-Europ.	Arabic	5%	34%
Hindi	hi	Indo-Europ.	Devanagari	2%	33%
Min Dong	cdo	Sino-Tibetan	Chinese	6%	53%
Māori	mi	Austronesian	Latin	1%	45%
Ilokano	ilo	Austronesian	Latin	2%	48%
Guarani	gn	Tupian	Latin	3%	42%
Mingrelian	xmf	Kartvelian	Georgian	7%	22%
Sindhi	sd	Indo-Europ.	Arabic	30%	25%
Erzya	myv	Uralic	Cyrilic	1%	33%
Bhojpuri	bh	Indo-Europ.	Devanagari	1%	28%
Wolof	wo	Niger-Congo	Latin	1%	31%
Amharic	am	Afro-Asiatic	Ge'ez	86%	13%
Tibetan	bo	Sino-Tibetan	Tibetan	66%	20%
Khmer	km	Austroasiatic	Khmer	79%	19%
Divehi	dv	Indo-Europ.	Thaana	85%	14%
Sinhala	si	Indo-Europ.	Sinhala	75%	23%

Table 1: Languages in our NER evaluation, together with their language family and common script. For each monolingual vocabulary (§4.3), we compute the proportion of tokens that cannot be composed by the subword tokens from the original mBERT vocabulary (UNKs) as well as of *lexically overlapping* tokens that are present in both vocabularies. *: For the high-resource source languages, proportions are calculated with regard to the tokenizers used by Rust et al. (2021).

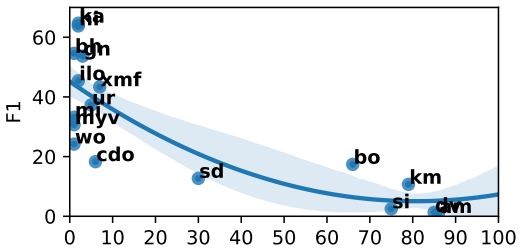


Figure 2: mBERT's zero-shot cross-lingual transfer performance with respect to the proportion of UNKs in mBERT's original vocabulary relative to the target language (see Table 1).

[2] An alternative approach based on transliteration (Müller et al., 2021) side-steps script adaptation but relies on language-specific heuristics, which are not available for most languages.

Method	Special tokens	Lexical overlap	Latent semantic concepts	Language clusters	New params	# of new params	Reference
EL-RAND	✓				$\mathbf{X}'$	7.68M	Artetxe et al. (2020)
EL-LEX	✓	✓			$\mathbf{X}'$	7.68M	Chau et al. (2020); Wang et al. (2020)
MF^C_*-RAND	✓		✓	✓	$\mathbf{F}', \mathbf{I}'$	1M + C·10k	Ours
MF^C_*-LEX	✓	✓	✓	✓	$\mathbf{F}', \mathbf{I}'$	1M + C·10k	Ours

Table 2: Overview of our methods and related approaches together with the pretrained knowledge they utilize. We calculate the number of new parameters per language with $V' = 10k$, $D = 768$, and $D' = 100$. We do not include up-projection matrices $\mathbf{G}$ as these are learned only once and make up a comparatively small number of parameters.

Recent approaches such as invertible adapters (Pfeiffer et al., 2020b) that adapt embeddings in the pretrained multilingual vocabulary may be able to deal with lesser degrees of lexical overlap. Still, they cannot deal with UNK tokens. In the following, we systematize existing approaches and present novel ways of adapting a pretrained model to the vocabulary of a target language that can handle this challenging setting, present most acutely when adapting to languages with unseen scripts. We summarize the approaches in Table 2 based on what types of pretrained information they utilize. All approaches rely on a new vocabulary V', learned on the target language data.

3.1 Target-Language Embedding Learning

A straightforward way to adapt a pretrained model to a new language is to learn new embeddings for the language. Given the new vocabulary V', we initialize new embeddings $\mathbf{X}' \in \mathbb{R}^{|V'| \times D}$ for all V' vocabulary items where D is the dimensionality of the existing embeddings $\mathbf{X} \in \mathbb{R}^{|V| \times D}$, and only initialize special tokens (e.g. [CLS], [SEP]) with their pretrained representations. We train the new embeddings of the $\mathbf{X}'$ with the pretraining task. This approach, termed EL-RAND, was proposed by Artetxe et al. (2020): they show that it allows learning aligned representations for a new language but only evaluate on high-resource languages. The shared special tokens allow the model to access a minimum amount of lexical information, which can be useful for transfer (Dufter and Schütze, 2020). Beyond this, this approach leverages knowledge from the existing embedding matrix only implicitly to the extent that the higher-level hidden representations are aligned to the lexical representations.

3.2 Initialization with Lexical Overlap

To leverage more lexical information, we can apply shared initialization not only to the special tokens but to all lexically overlapping tokens. Let us denote this vocabulary subset with V'_{lex}, and $V'_{rand} = V' \setminus V'_{lex}$. In particular, we initialize the embeddings of all lexically overlapping tokens $\mathbf{X}'_{lex}$ from V'_{lex} with their pretrained representations from the original matrix $\mathbf{X}$, while the tokens from V'_{rand} receive randomly initialized embeddings $\mathbf{X}'_{rand}$. We then fine-tune all target language embeddings $\mathbf{X}' = \mathbf{X}'_{lex} \cup \mathbf{X}'_{rand}$ on the target language data. Wang et al. (2020) cast this as extending V with new tokens. In contrast, we seek to disentangle the impact of vocabulary size and pretrained information. As one variant of this approach, Chau et al. (2020) only add the 99 most common tokens of a new language to V.

Initialization with lexical overlap, termed EL-LEX, allows us to selectively leverage the information from the pretrained model on a per-token level based on surface-level similarity. Intuitively, this should be most useful for languages that are lexically similar to those seen during pretraining and have a substantial proportion of lexically overlapping tokens. However, such lexical overlap is a lot rarer for languages that are written in different scripts. For such languages, relying on surface-level string similarity alone may not be enough.

3.3 Embedding Matrix Factorization

We therefore propose to identify latent semantic concepts in the pretrained model's embedding matrix that are general across languages and useful for transfer. Further, to allow modeling flexibility we propose to learn a grouping of similar tokens. We achieve this by factorizing the pretrained embedding matrix $\mathbf{X} \in \mathbb{R}^{|V| \times D}$ into lower-dimensional word embeddings $\mathbf{F} \in \mathbb{R}^{|V| \times D'}$ and C shared up-projections $\mathbf{G}^1, \ldots, \mathbf{G}^C \in \mathbb{R}^{D' \times D}$ that encode general cross-lingual information:

$$\mathbf{X} \approx \sum_{c \in C} \text{diag}(\mathbf{i}_c) \mathbf{F} \mathbf{G}^c \qquad (2)$$

D' is the dimensionality of the lower-dimensional embeddings. $\mathbf{I} = [\mathbf{i}_1, \ldots, \mathbf{i}_C] \in \mathbb{R}^{|V| \times C}$ is an

indicator matrix where $i_{v,c} = 1$ iff token v is associated with the c-th up-projection, else 0.[3] diag$(\cdot)$ creates a diagonal matrix from a vector, i.e. diag$(\mathbf{i}_c) \in \mathbb{R}^{|V|\times|V|}$.

MF^1-$*$. In the basic case where $C = 1$, Eq. (2) simplifies to $\mathbf{X} \approx \mathbf{FG}$. As $\mathbf{X}$ is unconstrained, we follow Ding et al. (2008) and interpret this as a semi-non-negative matrix factorization (Semi-NMF) problem. In Semi-NMF, $\mathbf{G}$ is restricted to be non-negative while no restrictions are placed on the signs of $\mathbf{F}$. The Semi-NMF is computed via an iterative updating algorithm that alternatively updates $\mathbf{F}$ and $\mathbf{G}$ where the Frobenius norm is minimized.[4]

$\mathbf{G}$ is shared across all tokens and thus encodes general properties of the original embedding matrix $\mathbf{X}$ whereas $\mathbf{F}$ stores token-specific information. $\mathbf{G}$ only needs to be pretrained once and can be used and fine-tuned for every new language. To this end, we simply learn new low-dimensional embeddings $\mathbf{F}' \in \mathbb{R}^{|V'|\times D'}$ with the pretraining task, which are up-projected with $\mathbf{G}$ and fed to the model.

MF$^C_{\text{KMEANS}}$-$*$. When $C > 1$, each token is associated with one of C up-projection matrices. Grouping tokens and using a separate up-projection matrix per group may help balance sharing information across typologically similar languages with learning a robust representation for each token (Chung et al., 2020). We propose two approaches to automatically learn such a clustering.

In our first, pipeline-based approach, we first cluster $\mathbf{X}$ into C clusters using KMeans. For each cluster, we then factorize the subset of embeddings $\mathbf{X}^c$ associated with the c-th cluster separately using Semi-NMF equivalently as for MF^1-$*$.

For a new language, we learn new low-dim embeddings $\mathbf{F}' \in \mathbb{R}^{|V'|\times D'}$ and a randomly initialized matrix $\mathbf{Z} \in \mathbb{R}^{|V'|\times C}$, which allows us to compute the cluster assignment matrix $\mathbf{I}' \in \mathbb{R}^{|V'|\times C}$. Specifically, for token v, we obtain its cluster assignment as $\arg\max$ of $\mathbf{z}'_{v,\cdot}$. As $\arg\max$ is not differentiable, we employ the Straight-Through Gumbel-Softmax estimator (Jang et al., 2017) defined as:

$$i'_{v,c} = \frac{\exp(\log(z_{v,c}) + g_c)/\tau)}{\sum_{j=1}^{C} \exp(\log(z_{v,j}) + g_j)/\tau)}, \quad (3)$$

where τ is a temperature parameter, and $\mathbf{g} \in \mathbb{R}^{|V|}$ corresponds to samples from the Gumbel distribution $g_j \sim -\log(-\log(u_j))$ with $u_j \sim \mathcal{U}(0,1)$ being the uniform distribution. $\mathbf{z}_{v,\cdot}$ can be seen as "logits" used for assigning the v-th token a cluster. As $\tau \to 0$, the softmax becomes an $\arg\max$ and the Gumbel-Softmax distribution approximates more closely the categorical distribution. $\mathbf{I}' \in \mathbb{R}^{|V'|\times C}$ represents the one-hot encoded, indicator function over possible clusters, with learnable parameters $\mathbf{Z}$. As before, $i'_{v,c} = 1$ iff new token v is associated with up-projection c, else 0.

MF$^C_{\text{NEURAL}}$-$*$. We can also learn the cluster assignment and up-projections jointly. Specifically, we parameterize $\mathbf{G}$ in Eq. (2) using a neural net where we learn the indicator matrix $\mathbf{I}$ equivalently to Eq. (3). The objective minimizes the L_2-norm between the original and predicted embeddings:

$$\mathcal{L} = ||\mathbf{X} - \sum_{c\in C} \text{diag}(\mathbf{i}_c)\mathbf{F}\mathbf{G}^c||_2 \quad (4)$$

For a new language, we proceed analogously.

MF$^*_*$-RAND and MF$^*_*$-LEX. Finally, we can combine different initialization strategies (see §3.1 and §3.2) with the embedding matrix factorization technique. We label the variant which relies on random initialization, see §3.1, as MF^1-RAND. The variant, which relies on lexically overlapping tokens from §3.2 can leverage both surface-level similarity as well as latent knowledge in the embedding matrix; we simply initialize the embeddings of overlapping lexical tokens (from V'_{lex}) in $\mathbf{F}'$ with their low-dim representations from $\mathbf{F}$. The remaining tokens (from V'_{rand}) are randomly initialized in $\mathbf{F}'$.

Factorizing the embedding matrix has the additional benefit of reducing the number of trainable parameters and correspondingly the amount of storage space required for each additional language. This is especially true when $D \gg D'$.[5]

4 Experiments

Data. For pretraining, we leverage the Wikipedia dumps of the target languages. We conduct experiments on named entity recognition (NER) and dependency parsing (DP). For NER, we use the WikiAnn (Pan et al., 2017) dataset, partitioned into train, dev, and test portions by Rahimi et al. (2019). For DP we use Universal Dependencies (UD; Nivre et al., 2016, 2020; Zeman et al., 2020).

[3] We use bold upper case letters ($\mathbf{X}$) for matrices, subscripts with bold letters ($\mathbf{x}_i$) for rows or columns and subscripts with standard weight letters for specific elements (x_i, $x_{i,j}$).

[4] For more details see Ding et al. (2008).

[5] For instance, when $|V| = |V|' = 10k$, $D = 768$, and $D' = 100$ as in our experiments we reduce the amount of storage space required by 82% per language.

Languages. WikiAnn offers a wide language coverage (176 languages) and, consequently, a number of language-related comparative analyses. In order to systematically compare against state-of-the-art cross-lingual methods under different evaluation conditions, we identify **1)** low-resource languages where the script has been covered by mBERT but the model has not been specifically trained on the language **2)** as well as low-resource languages with scripts not covered at all by pretraining. In each case, we select four languages, taking into account variance in data availability and typological diversity. We select four high-resource source languages (English, Chinese, Japanese, Arabic) in order to go beyond English-centric transfer. We evaluate the cross-lingual transfer performance of these 4 languages to the 17 diverse languages. For DP, we chose the subset that occurs in UD. We highlight the properties of all 21 languages in Table 1.

4.1 Baselines

mBERT (Standard Transfer Setup). We primarily focus on mBERT as it has been shown to work well for low-resource languages (Pfeiffer et al., 2020b). mBERT is trained on the 104 languages with the largest Wikipedias.[6] In the standard cross-lingual transfer setting (see §2), the full model is fine-tuned on the target task in the (high-resource) source language, and is evaluated on the test set of the target (low-resource) language.

MAD-X. We follow Pfeiffer et al. (2020b) and stack task adapters on top of pretrained language adapters (see §2). When training the model on source language task data, only the task adapter is trained while the original model weights and the source language adapter are frozen. At inference, the source language adapter is replaced with the target language adapter.

MAD-X 2.0. The adapter in the last transformer layer is not encapsulated between frozen transformer layers, and can thus be considered an extension of the prediction head. This places no constraints on the representation of the final adapter, possibly decreasing transfer performance when replacing the language adapters for zero-shot transfer. In this work, we thus propose to drop the adapter in the last transformer layer, and also evaluate this novel variant of the MAD-X framework.

[6] See Appendix Table 12 for the list of all 104 covered languages with corresponding scripts.

4.2 Methods

We experiment with the methods from Table 2 and discussed in §3, summarized here for clarity.

EL-* We randomly initialize embeddings for all tokens—except special tokens—in the new vocabulary (EL-RAND) or initialize embeddings of lexically overlapping tokens with their pretrained representations (EL-LEX).

MF^1-* We randomly initialize lower-dimensional embeddings (MF^1-RAND) or initialize lexically overlapping tokens with their corresponding lower-dimensional pretrained representation (MF^1-LEX) while using a single pretrained projection matrix.

MF$^{10}_*$-* We learn assignments to 10 clusters via k-means and up-projection matrices via Semi-NMF (MF$^{10}_{\text{KMEANS}}$-$*$). Alternatively, we learn cluster assignments with Gumbel-Softmax and up-projection matrices jointly (MF$^{10}_{\text{NEURAL}}$-$*$). For new tokens we use random (MF$^{10}_*$-RAND) or lexical overlap initialisation (MF$^{10}_*$-LEX).

4.3 Experimental Setup

Previous work generally fine-tunes the entire model on the target task (Chau et al., 2020; Wang et al., 2020). To extend the model to a new vocabulary, Artetxe et al. (2020) alternatingly freeze and fine-tune embeddings and transformer weights for pretraining, and target task fine-tuning, respectively. We find that this approach largely underperforms adapter-based transfer as proposed by Pfeiffer et al. (2020b), and we thus primarily focus on adapter-based training in this work.[7]

Adapter-Based Transfer. We largely follow the experimental setup of Pfeiffer et al. (2020b), unless noted otherwise. We obtain language adapters for the high-resource languages from AdapterHub.ml (Pfeiffer et al., 2020a) and train language adapters and embeddings for the low-resource languages jointly while keeping the rest of the model fixed. For zero-shot transfer, we replace the source language adapter with the target adapter, and also replace the entire embedding layer with the new embedding layer specialized to the target language. MAD-X 2.0 consistently outperforms MAD-X (see §5); we thus use this setup for all our methods.

Tokenizer. We learn a new WordPiece tokenizer for each target language with a vocabulary size of 10k using the HuggingFace tokenizer library.[8]

[7] We present results with such full model transfer in §A.1.

[8] https://github.com/huggingface/tokenizers

	Seen Languages				Unseen Languages but Covered Scripts										New Scripts						*Mac.*
	ka	ur	hi	**Avg**	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	**Avg**	am	bo	km	dv	si	**Avg**	***Avg***
mBERT	64.7	37.3	**63.8**	55.3	18.3	33.2	45.4	53.7	43.4	12.7	30.7	54.6	24.2	35.1	0.9	17.4	10.7	1.3	2.5	6.6	*30.3*
MAD-X	63.9	51.4	58.7	58.0	29.1	45.6	45.5	52.7	51.9	34.0	57.8	57.0	49.5	47.0	**10.8**	**24.8**	17.6	16.8	16.8	17.4	*40.2*
MAD-X 2.0	**65.6**	**55.3**	61.0	**60.6**	**30.7**	**50.6**	**64.0**	**56.3**	**52.6**	**37.1**	**63.0**	**59.8**	**55.5**	**52.2**	10.7	24.7	**18.1**	**22.2**	**18.7**	**18.9**	***43.9***
EL-RAND	**65.8**	47.8	**63.8**	**59.1**	38.0	7.3	**57.9**	48.5	59.4	44.2	35.2	55.5	5.1	39.0	**42.9**	**53.9**	59.5	32.7	51.5	48.1	*45.2*
MF^{1}-RAND	63.5	48.5	57.8	56.6	39.8	**19.3**	43.6	47.3	57.9	45.1	**60.4**	44.4	**45.0**	**44.8**	42.6	49.0	61.7	**43.4**	**52.1**	**49.8**	***48.3***
MF^{10}_{KMEANS}-RAND	64.3	**52.6**	60.4	**59.1**	20.7	11.7	51.2	50.8	58.2	**45.2**	49.0	**60.0**	42.5	43.2	37.3	37.5	**64.2**	18.5	47.2	40.9	*45.4*
MF^{10}_{NEURAL}-RAND	63.2	49.4	60.6	57.7	**40.7**	5.5	48.5	**52.0**	**60.4**	27.2	46.7	54.5	41.2	41.9	42.7	38.4	63.7	38.7	48.0	46.3	*46.0*
EL-LEX	**69.3**	57.9	**66.4**	**64.5**	46.8	32.2	**58.8**	54.1	57.0	44.6	**56.3**	59.5	50.1	51.0	**46.5**	51.5	61.0	**47.2**	**56.3**	**52.5**	*53.6*
MF^{1}-LEX	65.5	54.0	61.5	60.4	46.6	39.8	55.2	**54.8**	**57.2**	45.0	55.0	59.1	48.1	51.2	38.6	42.5	63.9	43.8	51.5	48.0	*51.2*
MF^{10}_{KMEANS}-LEX	66.3	**59.5**	61.1	62.3	**48.9**	47.5	46.4	53.6	56.7	46.2	58.4	**61.0**	**57.9**	**53.0**	45.5	**51.6**	64.2	44.9	46.2	50.5	***53.9***
MF^{10}_{NEURAL}-LEX	67.0	55.8	62.6	61.8	47.8	**50.8**	53.1	53.9	55.6	**46.4**	50.4	60.6	51.8	52.3	46.7	43.0	**65.2**	46.3	51.5	50.5	*53.4*

(a) Named Entity Recognition: Mean F_1 test results for UD averaged over 5 runs and averaged over the 4 source languages.

	Seen Languages			Unseen Languages but Covered Scripts				New Script	
	hi	ur	**Avg**	bh	myv	wo	**Avg**	am	***Macro Avg***
	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS
mBERT	**45.8 / 29.5**	34.9 / **20.5**	**40.3 / 25.0**	**33.9 / 18.6**	29.7 / 13.0	27.5 / 7.5	30.4 / 13.0	10.1 / 3.8	*30.3 / 15.5*
MAD-X 2.0	42.0 / 26.1	**35.0** / 20.1	38.5 / 23.1	31.0 / 16.2	**47.6 / 29.8**	**35.4 / 18.4**	**38.0 / 21.4**	**14.0 / 7.7**	***34.2 / 19.7***
EL-RAND	41.9 / 25.9	32.9 / 18.4	37.4 / 22.1	29.9 / 13.8	45.4 / 23.9	24.3 / 7.2	33.2 / 15.0	30.1 / 11.3	*34.1 / 16.7*
MF^{1}-RAND	41.7 / 26.2	33.7 / 19.4	37.7 / 22.8	28.7 / 13.7	47.6 / 27.2	**32.7 / 15.5**	36.3 / 18.8	**34.3 / 13.3**	**36.4 / 19.2**
MF^{10}_{KMEANS}-RAND	40.7 / 25.1	30.4 / 17.7	35.6 / 21.4	**31.9 / 16.3**	47.8 / 27.7	29.6 / 13.7	**36.4 / 19.2**	22.6 / 10.1	*33.8 / 18.4*
MF^{10}_{NEURAL}-RAND	**43.1 / 27.0**	**34.7 / 20.0**	**38.9 / 23.5**	30.8 / 14.7	**48.6 / 28.7**	21.7 / 9.4	33.7 / 17.6	32.4 / 12.4	*35.2 / 18.7*
EL-LEX	41.9 / 26.1	34.0 / 19.6	37.9 / 22.9	30.6 / 16.2	48.8 / 27.7	37.9 / 19.2	39.1 / 21.0	34.0 / 9.9	*36.6 / 19.8*
MF^{1}-LEX	42.1 / 26.7	34.0 / 20.1	38.0 / 23.4	**31.2 / 16.4**	**49.0 / 28.9**	37.6 / 19.5	**39.3 / 21.6**	**34.8 / 12.9**	*36.8 / **20.8***
MF^{10}_{KMEANS}-LEX	42.2 / 27.0	34.7 / 20.3	38.5 / 23.6	31.1 / **16.4**	47.7 / 28.6	33.8 / 17.3	37.5 / 20.8	32.2 / 13.2	*37.0 / 20.5*
MF^{10}_{NEURAL}-LEX	**43.0 / 27.4**	**36.1 / 21.2**	**39.6 / 24.3**	30.9 / 16.2	46.8 / 27.6	**38.7 / 20.4**	38.8 / 21.4	31.4 / 12.1	***37.8 / 20.8***

(b) Dependency Parsing: Mean UAS and LAS test results for UD averaged over 5 runs and averaged over the 4 source languages.

Table 3: Mean test results for (a) NER and (b) DP. The top group (first two/three rows) includes models, which leverage the original tokenizer which is not specialized for the target language. The second group (last eight rows) include models with new tokenizers. We separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical init (∗-LEXINIT) by the dashed line. Bold numbers indicate best-scoring models of the respective group, underlined numbers the best performance overall. Source languages are *en*, *ar*, *zh*, and *ja*.

Semi-NMF. We factorize the pretrained embedding matrix of mBERT using Semi-NMF (Ding et al., 2008) leveraging the default implementation provided by Bauckhage et al. (2011).[9] We train for 3,000 update steps and leverage the corresponding matrices **F** and **G** as initialization for the new vocabulary. We choose the reduced embedding dimensionality $D' = 100$. **F** is only used when initializing the (lower-dimensional) embedding matrix with lexically overlapping representations.

Masked Language Modeling. For MLM pretraining we leverage the entire Wikipedia corpus of the respective language. We train for 200 epochs or ∼100k update steps, depending on the corpus size. The batch size is 64; the learning rate is $1e-4$.

Task Fine-tuning. Our preliminary experiments suggested that fine-tuning the model for a smaller number of epochs leads to better transfer performance in low-resource languages in general. We thus fine-tune all the models for 10 epochs, evaluating on the source language dev set after every epoch. We then take the best-performing model according to the dev F_1 score, and use it in zero-shot transfer. We train all the models with a batch size of 16 on high resource languages. For NER we use learning rates $2e-5$ and $1e-4$ for full fine-tuning and adapter-based training, respectively. For DP, we use a transformer-based variant (Glavas and Vulic, 2021) of the standard deep biaffine attention dependency parser (Dozat and Manning, 2017) and train with learning rates $2e-5$ and $5e-4$ for full fine-tuning and adapter-based training respectively.

5 Results and Discussion

The main results are summarised in Table 3a for NER, and in Table 3b for DP. First, our novel MAD-X 2.0 considerably outperforms the MAD-X version of Pfeiffer et al. (2020b). However, while both MAD-X versions improve over mBERT for unseen scripts, the performance remains quite low on average. The corresponding mBERT tokenizer is not able to adequately represent unseen scripts: many tokens are substituted by UNKs (see Table 1), culminating in the observed low performance.

For our approaches that learn new embedding matrices, we observe that for languages seen during pretraining, but potentially underrepresented by the

[9] https://github.com/cthurau/pymf

	ka	ur	hi	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	am	bo	km	dv	si
Numbers	11%	8%	6%	26%	17%	8%	8%	15%	9%	11%	11%	7%	13%	9%	3%	12%	9%
Lat Char	4%	3%	3%	2%	2%	2%	2%	5%	4%	3%	4%	3%	8%	5%	5%	7%	4%
Lat (S)W	10%	9%	7%	54%	61%	64%	65%	10%	10%	9%	26%	71%	19%	30%	31%	17%	30%
Oth Char	55%	36%	48%	18%	19%	26%	25%	60%	39%	26%	24%	18%	60%	52%	61%	63%	57%
Oth (S)W	20%	43%	36%	0%	1%	0%	0%	10%	38%	51%	35%	1%	0%	3%	0%	1%	0%

Table 4: Grouping of tokens that lexically overlap between the original mBERT tokenizer and the tokenizer of the target language. *Numbers* includes all tokens which include at least one number; *Lat Char* indicates all Latin tokens of length 1; *Lat (S)W* includes all (sub)words that include a Latin character but are of length > 1. Consequently, *Oth Char* and *Oth (S)W* consists of characters and (sub)words respectively, which do not include Latin characters.

cdo	mi	ilo	gn	bo	km	dv	si
Northumberland	Massachusetts	Melastomataceae	establecimiento	University	languages	government	International
Massachusetts	Encyclopedia	munisipalidad	vicepresidente	Therefore	language	Chinese	Bangladesh
International	Pennsylvania	Internasional	Internacional	suffering	formula	govern	wikipedia
Pennsylvania	Jacksonville	internasional	internacional	existence	disease	system	Australia
Philadelphia	Turkmenistan	International	Independencia	practice	control	ation	Zimbabwe

Table 5: Longest lexically overlapping (sub)words.

model (e.g. Georgian (ka), Urdu (ur), and Hindi (hi)), the proposed methods outperform MAD-X 2.0 for all tasks. This is in line with contemporary work (Rust et al., 2021), which emphasizes the importance of tokenizer quality for the downstream task. Consequently, for unseen languages with under-represented scripts, the performance gains are even larger, e.g., we see large improvements for Min Dong (cdo), Mingrelian (xmf), and Sindhi (sd). For unseen languages with the Latin script, our methods perform competitively (e.g. Maori (mi), Ilokano (ilo), Guarani (gn), and Wolof (wo)): this empirically confirms that the Latin script is adequately represented in the original vocabulary. The largest gains are achieved for languages with unseen scripts (e.g. Amharic (am), Tibetan (bo), Khmer (km), Divehi (dv), Sinhala (si)), as these languages are primarily represented as UNK tokens by the original mBERT tokenizer.

We observe improvements for most languages with lexical overlap initialization. This adds further context to prior studies which found that a shared vocabulary is not necessary for learning multilingual representations (Conneau et al., 2020b; Artetxe et al., 2020): while it is possible to generalize to new languages without lexical overlap, leveraging the overlap still offers additional gains.

The methods based on matrix factorization (MF^*_*-$*$) improve performance over full-sized embedding methods (EL^*_*-$*$), especially in the setting without lexical overlap initialization ($*$-RAND). This indicates that by factorizing the information encoded in the original embedding matrix we are able to extract relevant information for unseen languages. Combining matrix factorization with lexical overlap initialization (MF^*_*-LEX), zero-shot performance improves further for unseen languages with covered scripts. This suggests that the two methods complement each other. For 6/9 of these languages, we find that encoding the embeddings in multiple up-projections (MF^{10}_*-$*$) achieves the peak score. This in turn verifies that grouping similar tokens improves the robustness of token representations (Chung et al., 2020). For unseen languages with covered scripts, this model variant also outperforms MAD-X 2.0 on average.

For languages with unseen scripts we find that MF has smaller impact. While the encoded information supports languages similar to those seen by the model in pretraining, languages with unseen scripts are too distant to benefit from this latent multilingual knowledge. Surprisingly, lexical overlap is helpful for languages with unseen scripts.

Overall, we observe that both $\text{MF}^{10}_{\text{KMEANS}}$-$*$ and $\text{MF}^{10}_{\text{NEURAL}}$-$*$ perform well for most languages, where the KMEANS variant performs better for NER and the NEURAL variant performs better for UD.

6 Further Analysis

6.1 Lexically Overlapping (Sub)Words

We perform a quantitative analysis of lexically overlapping tokens, i.e. that occur both in mBERT's and monolingual in-language vocabularies; see Table 4. For languages with scripts not covered by the mBERT tokenizer, most lexically-overlapping tokens are single characters of a non-Latin script.

	Seen Languages			Unseen Languages but Covered Scripts									New Scripts				
	ka	ur	hi	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	am	bo	km	dv	si
MF^{10}_{KMEANS}-LEX 100	66.3	59.5	61.1	48.9	47.5	46.4	53.6	56.7	46.2	58.4	61.0	57.9	45.5	51.6	64.2	44.9	46.2
MF^{10}_{KMEANS}-LEX 300	65.1	46.8	56.8	26.6	5.5	53.7	10.0	52.2	43.5	37.4	53.2	3.7	21.5	7.4	63.9	38.8	51.2

Table 6: Mean F_1 test results averaged over 5 runs and the 4 high-resource source languages English, Chinese, Japanese, and Arabic. The first row presents results with 100-, the second row 300-dimensional embeddings.

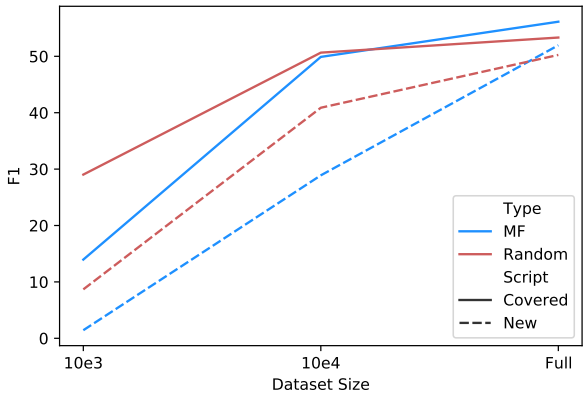


Figure 3: Sample efficiency. For "MF" we leverage the MF^{10}_{KMEANS}-LEX setting.

We further present the top 5 longest lexically overlapping (sub) words for four languages with scripts covered during pretraining (Min Dong, Maori, Ilokano, and Guarani) and four languages with unseen scripts (Tibetan, Khmer, Divehi, Sinhala) in Table 5. We observe that frequent lexically overlapping tokens are named entities in the Latin script, indicating that NER may not be the best evaluation task to objectively assess generalization performance to such languages. If the same named entities also occur in the training data of higher-resource languages, the models will be more successful at identifying them in the unseen language, which belies a lack of deeper understanding of the low-resource language. This might also explain why greater performance gains were achieved for NER than for DP.

6.2 Sample Efficiency

We have further analyzed the sample efficiency of our approaches. We find that EL-LEX is slightly more sample-efficient than MF^{10}_{KMEANS}-LEX, where the latter outperforms the former with more data available. In Figure 3 we plot the zero-shot transfer performance where the adapters and embeddings were pretrained on different amounts of data. We further find that lower-dimensional embeddings tend to outperform higher-dimensional embeddings for the majority of languages. In Table 6 we compare 300 with 100 dimensional embeddings for the MF^{10}_{KMEANS}-LEX approach.

6.3 Script Clusters

We analyzed the KMeans clusters based on the tokens that consist of characters of a certain script in Figure 4 of the Appendix. We find distinct script-based groups; For instance, 5 clusters consist primarily of Latin-script tokens[10], two clusters predominantly consist of Chinese, and a few Korean tokens. Interestingly, 2 clusters consisted of Cyrilic, and Arabic scripts as well as scripts used predominantly in India, varying slightly in their distribution. Lastly, one cluster included tokens of all except the Latin script.

7 Conclusion

We have systematically evaluated strategies for model adaptation to unseen languages with seen and unseen scripts. We have assessed the importance of the information stored within the original embedding matrix by means of leveraging lexically overlapping tokens, and extracting latent semantic concepts. For the latter, we have proposed a new method of encoding the embedding matrix into lower- dimensional embeddings and up-projections. We have demonstrated that our methods outperform previous approaches on NER and dependency parsing for both resource-rich and resource-poor scenarios, reducing the gap between random and lexical overlap initialisation, and enabling more effective model adaptation to unseen scripts.

Acknowledgments

Jonas Pfeiffer is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. The work of Ivan Vulić is supported by the ERC Grant LEXICAL (no. 648909) and the ERC PoC Grant MultiConvAI (no. 957356).

We thank Laura Rimell, Nils Reimers, Michael Bugert and the anonymous reviewers for insightful feedback and suggestions on a draft of this paper.

[10]The majority of tokens of mBERT use the Latin script.

References

Judit Ács. 2019. Exploring BERT's Vocabulary. *Blog Post*.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021. MAD-G: Multilingual Adapter Generation for Efficient Cross-Lingual Transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, Online.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 4623–4637.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548.

Christian Bauckhage, Kristian Kersting, and Christian Thurau. 2011. Factorizing gigantic matrices. *ECML-PKDD 2011: Tutorial*.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual bert, a small treebank, and a small corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 1324–1334.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking Embedding Coupling in Pre-trained Language Models. In *9th International Conference on Learning Representations, ICLR 2021, Online, 2021, Conference Track Proceedings*.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4536–4546.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 8440–8451.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chris HQ Ding, Tao Li, and Michael I Jordan. 2008. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for bert's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4423–4437.

Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 710–721.

Goran Glavas and Ivan Vulic. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? an empirical investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3090–3104. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International*

Conference on Machine Learning, ICML 2020, 12-18 July 2020, Virtual Conference.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual BERT: an empirical study. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online.

Benjamin Müller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. When being unseen from mbert is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 448–462. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1946–1958.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 487–503. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 46–54. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7654–7673. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4996–5001.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 151–164.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In *Proceedings of the 59th Annual Meeting of the Association*

for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 3118–3135. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ruder, and Ivan Vulic. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 778–788.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5986–5995.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language Adaptation for Truly Universal Dependency Parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2302–2315. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Marko Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. In *arXiv preprint*.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2649–2656.

Shijie Wu, Alexis Conneau, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 6022–6034.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 833–844.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Angelina Aquino, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Ethan Chi, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Hildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phuong Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Yuan Li, Nikola

Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Arzucan Özgür, Balkız Öztürk Başaran, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Aya Wakasa, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Hanzhi Zhu, and Anna Zhuravleva. 2020. Universal dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Appendices

A.1 Full Model Transfer

For comparability with previous work, which generally fine-tunes the entire model (Chau et al., 2020; Wang et al., 2020), we follow Artetxe et al. (2020) and learn a new embedding matrix for the target language while freezing the pretrained transformer weights. For training on the target task, we fine-tune the transformer weights of the pretrained model while keeping the original embedding layer frozen. For zero-shot transfer, we replace the existing with the new embedding matrix trained on the target language.

A.2 Results: Named Entity Recognition

We present non-aggregated NER transfer performance when transferring from English, Chinese, Japanese, and Arabic in Tables 7, 8, 9, and 10 respectively. *12Ad* indicates whether (✓) or not (✗) an adapter is placed in the 12th transformer layer. We additionally present the results for full model fine-tuning (FMT-∗).

A.3 Results: Dependency Parsing

We present non-aggregated DP transfer performance when transferring from English, Chinese, Japanese, and Arabic in Tables 11a, 11b, 11c, and 11d respectively.

A.4 Script Clusters

We present the groups of scripts within the 10 KMeans clusters in Figure 4. We follow Ács (2019) in grouping the scripts into languages.

A.5 Language information

Table 12 lists all 104 languages and corresponding scripts which mBERT was pretrained on.

A.6 Hardware Setup

All experiments were conducted on a single NVIDIA V100 GPU with 32 Gb of VRAM.

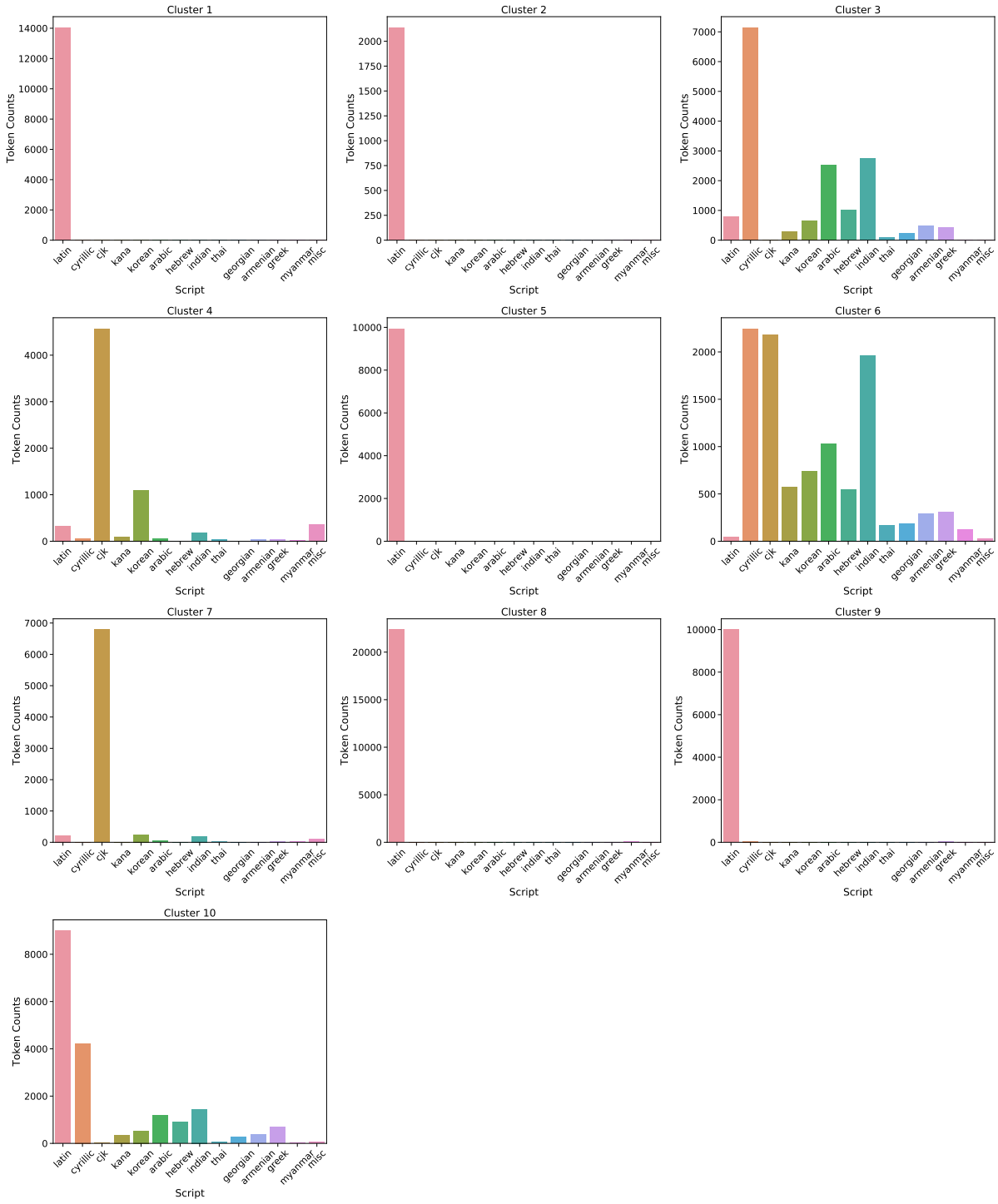


Figure 4: Number of tokens of each script grouped into the 10 KMeans clusters. We follow (Ács, 2019) in splitting the utf-8 tokens into scripts.

	12Ad	Seen Languages				Unseen Languages but Covered Scripts										New Scripts					
		ka	ur	hi	**Avg**	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	**Avg**	am	bo	km	dv	si	**Avg**
mBERT		64.3	34.3	65.3	54.6	17.3	22.2	62.7	46.7	34.4	12.1	47.2	48.8	26.7	35.4	0.0	11.4	4.9	0.3	0.3	3.4
MAD-X	✓	64.8	44.5	60.7	56.7	25.1	38.4	61.2	52.7	46.0	25.3	47.7	52.1	47.1	43.9	14.5	20.4	14.5	13.9	16.7	16.0
MAD-X 2.0	✗	67.4	54.4	64.3	62.1	24.5	51.8	79.2	54.4	49.0	30.9	58.1	55.6	56.8	51.1	8.0	16.6	18.4	23.0	22.0	17.6
FMT-RANDINIT		65.1	48.9	63.3	59.1	26.9	12.6	67.1	20.9	54.6	37.5	8.0	47.1	10.9	31.7	28.6	29.3	51.6	27.0	50.0	37.3
EL-RAND	✓	66.6	35.0	62.9	54.8	33.5	11.0	58.8	49.0	47.8	37.5	20.0	53.6	4.1	35.0	36.8	40.2	57.6	35.5	48.4	43.7
EL-RAND	✗	70.1	39.2	67.5	58.9	34.0	16.4	66.6	53.0	57.9	39.7	35.1	56.4	4.9	40.4	46.4	50.8	61.4	38.9	52.6	50.0
MF^{1}-RAND	✓	62.1	46.2	61.2	56.5	29.5	28.1	51.4	41.6	47.3	40.8	38.5	33.8	29.0	37.8	39.6	39.3	54.7	35.4	46.6	43.1
MF^{1}-RAND	✗	67.2	51.5	63.2	60.6	33.3	33.7	58.0	48.6	56.1	44.3	55.2	43.2	41.8	46.0	43.3	42.1	63.4	41.6	54.7	49.0
MF^{10}_{KMEANS}-RAND	✓	63.2	52.0	61.4	58.9	28.4	7.2	52.8	45.6	51.3	15.3	30.0	47.8	32.8	34.6	37.4	34.1	55.1	33.7	44.4	40.9
MF^{10}_{KMEANS}-RAND	✗	67.3	55.5	63.9	62.3	35.1	6.3	65.0	57.3	59.1	20.4	44.8	55.2	42.6	42.9	46.8	41.8	60.1	37.5	51.0	47.4
MF^{10}_{NEURAL}-RAND	✓	63.9	51.7	62.6	59.4	14.0	4.1	57.0	49.4	48.2	35.6	23.6	52.4	34.8	35.5	31.6	32.0	58.3	17.2	35.6	34.9
MF^{10}_{NEURAL}-RAND	✗	68.4	56.3	62.8	62.5	16.0	4.7	65.9	56.4	57.6	42.2	39.1	58.5	44.8	42.8	37.3	31.0	61.7	18.9	42.2	38.2
FMT-LEXINIT		67.6	51.8	63.1	60.9	42.2	31.3	72.4	47.6	56.7	35.6	53.1	53.4	45.6	48.7	34.9	25.9	49.9	43.9	55.4	42.0
EL-LEX	✓	70.1	51.4	65.2	62.2	42.5	37.9	56.8	50.8	52.5	41.1	38.8	56.1	54.2	47.9	40.0	41.9	60.8	43.3	52.6	47.7
EL-LEX	✗	73.1	56.0	69.2	66.1	42.9	34.3	66.1	54.9	56.9	44.0	50.8	59.4	61.4	52.3	46.5	46.6	60.0	48.3	60.3	52.3
MF^{1}-LEX	✓	66.4	41.1	62.0	56.5	41.3	37.2	57.1	46.5	47.0	35.7	29.5	50.6	36.5	42.4	36.4	40.0	59.1	35.6	44.1	43.0
MF^{1}-LEX	✗	71.0	51.7	65.6	62.8	43.3	55.6	64.9	57.5	53.2	42.1	47.5	57.4	45.9	51.9	43.1	41.8	63.1	44.0	53.6	49.1
MF^{10}_{KMEANS}-LEX	✓	68.0	56.4	63.7	62.7	38.1	45.5	60.2	47.8	46.5	39.8	33.5	52.2	50.1	46.0	42.2	36.8	61.6	39.1	50.5	46.0
MF^{10}_{KMEANS}-LEX	✗	71.0	58.0	65.8	64.9	44.7	65.7	67.1	55.5	55.2	43.9	41.0	57.3	59.2	54.4	49.0	36.3	65.3	45.5	55.3	50.3
MF^{10}_{NEURAL}-LEX	✓	65.7	53.6	60.2	59.8	43.7	51.1	52.7	51.0	46.7	41.1	46.6	55.9	48.8	48.6	43.8	37.3	58.9	39.1	41.1	44.0
MF^{10}_{NEURAL}-LEX	✗	70.6	60.5	63.9	65.0	45.3	65.0	61.9	53.6	53.0	43.2	56.9	60.2	60.2	55.5	53.5	46.1	61.1	44.5	43.2	49.7

Table 7: Mean F_1 NER test results averaged over 5 runs transferring from high resource language **English** to the low resource languages. *12Ad* indicates whether (✓) or not (✗) an adapter is placed in the 12th transformer layer. The top group (first three rows) includes models which leverage the original tokenizer which is not specialized for the target language. The second group (last 18 rows) include models with new tokenizers. Here we separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical initialization (∗-LEXINIT) by the dashed line. We additionally present the results for full model fine-tuning (FMT-∗).

	12Ad	Seen Languages				Unseen Languages but Covered Scripts										New Scripts					
		ka	ur	hi	**Avg**	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	**Avg**	am	bo	km	dv	si	**Avg**
mBERT		66.0	33.3	60.2	53.2	21.2	42.3	42.6	59.5	49.1	14.2	29.4	60.1	25.1	38.2	0.2	13.1	21.8	1.4	2.0	7.7
MAD-X	✓	62.9	50.8	54.1	55.9	32.2	42.5	40.6	56.4	57.3	44.4	68.6	63.1	55.1	51.1	14.6	24.4	24.1	19.8	18.9	20.4
MAD-X 2.0	✗	63.9	52.4	55.7	57.3	37.0	42.1	64.7	57.9	54.9	42.8	72.0	63.8	60.1	55.1	13.5	29.0	23.0	21.5	20.2	21.4
FMT-RANDINIT		64.6	37.2	56.5	52.8	36.7	4.1	48.1	15.7	57.8	45.8	7.0	48.9	15.8	31.1	26.4	28.8	53.0	31.1	41.6	36.2
EL-RAND	✓	62.8	43.7	57.2	54.6	41.4	6.8	51.7	45.5	57.4	48.1	38.7	58.4	13.4	40.2	41.8	49.8	51.6	30.1	54.3	45.5
EL-RAND	✗	63.0	48.3	59.3	56.9	44.8	4.4	54.3	47.5	60.1	47.0	46.5	58.6	8.7	41.3	43.5	51.8	53.6	29.7	51.3	46.0
MF^{1}-RAND	✓	59.3	38.1	50.3	49.2	38.2	10.0	34.7	44.4	54.1	46.5	59.8	43.0	47.8	42.1	41.6	49.1	59.7	42.3	53.6	49.2
MF^{1}-RAND	✗	59.5	42.8	52.9	51.7	44.2	10.2	35.9	44.3	54.2	44.8	64.2	43.4	47.4	43.2	43.9	48.6	55.9	45.3	52.4	49.2
MF^{10}_{KMEANS}-RAND	✓	60.0	40.7	55.5	52.1	38.6	1.0	43.2	44.4	64.3	28.6	55.0	61.0	50.9	43.0	39.3	23.5	63.7	34.6	45.0	41.2
MF^{10}_{KMEANS}-RAND	✗	59.6	43.9	56.8	53.4	48.1	3.1	47.6	50.3	63.1	33.0	60.6	59.7	51.9	46.4	44.9	25.3	62.1	37.5	45.7	43.1
MF^{10}_{NEURAL}-RAND	✓	62.5	44.1	54.1	53.5	21.3	2.2	44.7	46.1	58.4	50.8	39.7	63.9	47.6	41.6	37.5	38.7	65.4	18.3	50.8	42.1
MF^{10}_{NEURAL}-RAND	✗	62.1	50.0	56.3	56.1	20.7	4.1	49.7	51.4	57.9	52.5	48.2	65.6	48.3	44.3	38.3	42.9	64.3	18.1	52.5	43.2
FMT-LEXINIT		67.1	36.9	60.2	54.7	59.6	15.6	49.3	55.4	57.3	45.5	52.3	64.6	50.4	50.0	29.3	31.3	58.2	45.2	48.6	42.5
EL-LEX	✓	68.9	49.7	61.7	60.1	51.4	26.6	54.3	53.5	54.6	48.4	59.7	62.2	59.3	52.2	43.9	47.1	60.2	48.3	55.7	51.0
EL-LEX	✗	68.0	53.1	64.0	61.7	54.7	24.4	54.4	55.3	52.4	48.4	60.7	59.6	58.1	52.0	48.0	53.5	60.7	50.3	54.8	53.5
MF^{1}-LEX	✓	61.8	42.8	56.4	53.7	51.2	16.3	47.2	49.7	52.6	47.2	56.2	60.0	53.7	48.2	41.5	39.4	61.4	41.7	44.8	45.8
MF^{1}-LEX	✗	61.9	47.8	57.7	55.8	48.2	18.8	51.5	55.8	54.9	46.7	61.3	60.4	52.6	50.0	42.9	41.9	58.3	44.8	49.6	47.5
MF^{10}_{KMEANS}-LEX	✓	62.9	47.2	58.3	56.1	49.2	31.8	53.8	53.2	51.1	50.1	56.0	61.6	58.5	51.7	42.7	43.6	62.4	46.9	50.0	49.1
MF^{10}_{KMEANS}-LEX	✗	63.3	50.9	59.8	58.0	51.8	29.6	52.5	55.3	51.5	50.7	59.9	65.1	56.0	52.5	44.9	44.4	61.0	49.9	48.0	49.6
MF^{10}_{NEURAL}-LEX	✓	62.6	55.1	55.6	57.8	52.9	32.3	40.0	54.4	59.4	53.4	61.1	60.8	67.4	53.5	44.3	50.1	65.7	40.9	53.4	50.9
MF^{10}_{NEURAL}-LEX	✗	63.7	57.1	58.4	59.7	54.1	34.2	42.0	54.6	59.0	49.6	63.4	62.9	66.9	54.1	41.5	53.8	62.3	45.4	49.6	50.5

Table 8: Mean F_1 NER test results averaged over 5 runs transferring from high resource language **Chinese** to the low resource languages. *12Ad* indicates whether (✓) or not (✗) an adapter is placed in the 12th transformer layer. The top group (first three rows) includes models which leverage the original tokenizer which is not specialized for the target language. The second group (last 18 rows) include models with new tokenizers. Here we separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical initialization (∗-LEXINIT) by the dashed line. We additionally present the results for full model fine-tuning (FMT-∗).

	12Ad	Seen Languages				Unseen Languages but Covered Scripts										New Scripts					
		ka	ur	hi	**Avg**	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	**Avg**	am	bo	km	dv	si	**Avg**
mBERT		64.2	35.5	59.8	53.2	17.7	42.6	39.5	50.2	46.5	14.2	23.9	56.0	21.6	34.7	3.4	23.2	6.2	1.4	7.6	8.3
MAD-X	✓	62.7	49.8	55.9	56.1	29.7	43.8	28.3	48.0	53.2	34.2	58.0	54.7	52.9	44.7	6.0	31.4	22.4	16.7	17.2	18.7
MAD-X 2.0	✗	64.7	52.5	57.8	58.3	31.8	46.6	43.5	54.6	51.7	40.0	60.6	59.9	50.8	48.8	7.2	31.2	22.7	24.4	22.1	21.5
FMT-RANDINIT		64.2	36.4	60.2	53.6	33.3	3.8	49.7	15.5	56.1	47.2	11.7	54.6	12.7	31.6	27.5	33.0	58.5	30.4	52.5	40.4
EL-RAND	✓	63.0	37.1	58.2	52.8	35.8	3.9	47.3	43.7	50.1	41.6	22.5	54.7	5.7	33.9	34.9	50.6	52.7	32.3	44.6	43.0
EL-RAND	✗	65.6	43.1	60.7	56.5	37.4	2.6	52.6	47.4	58.1	46.7	31.2	56.8	4.3	37.5	40.3	53.5	60.7	32.3	52.8	47.9
MF^{1}-RAND	✓	60.8	43.1	52.7	52.2	36.7	9.1	32.4	42.0	55.2	42.1	57.1	43.1	46.7	40.5	37.2	48.3	56.8	37.9	50.6	46.2
MF^{1}-RAND	✗	63.5	44.6	55.7	54.6	43.3	12.2	28.9	44.9	57.7	48.5	62.4	46.0	55.9	44.4	41.4	46.8	57.6	40.1	52.8	47.7
MF^{10}_{KMEANS}-RAND	✓	58.8	36.5	56.5	50.6	34.5	0.9	24.0	45.7	57.8	20.6	27.3	52.4	42.2	33.9	33.5	39.9	61.9	34.7	40.7	42.1
MF^{10}_{KMEANS}-RAND	✗	64.4	39.0	58.4	53.9	40.3	1.3	31.3	48.8	57.0	26.5	46.1	55.5	43.8	39.0	36.2	40.7	63.6	37.0	47.2	44.9
MF^{10}_{NEURAL}-RAND	✓	59.4	37.4	55.4	50.7	22.6	2.6	29.3	44.8	52.1	45.5	28.3	58.5	41.3	36.1	31.0	42.4	63.9	17.0	45.5	40.0
MF^{10}_{NEURAL}-RAND	✗	62.8	44.4	56.7	54.6	19.5	3.7	32.6	46.0	57.3	44.5	45.4	62.3	42.9	39.4	33.8	40.1	63.7	18.2	44.5	40.0
FMT-LEXINIT		66.1	40.4	62.0	56.2	52.4	13.7	45.5	52.8	56.7	43.0	61.2	64.0	50.2	48.9	31.5	30.0	60.7	40.2	53.9	43.3
EL-LEX	✓	60.3	51.7	61.8	57.9	42.3	20.1	55.3	47.5	50.8	42.7	60.9	59.8	55.2	48.3	39.8	44.8	60.2	39.9	52.7	47.5
EL-LEX	✗	68.8	52.5	62.9	61.4	46.6	22.6	51.3	50.4	55.0	44.7	60.1	63.4	56.0	50.0	45.0	47.3	61.7	45.4	57.7	51.4
MF^{1}-LEX	✓	62.7	47.4	58.6	56.2	47.0	22.1	38.0	49.6	51.6	42.0	57.1	59.2	55.2	46.9	34.9	41.4	63.3	41.1	43.0	44.8
MF^{1}-LEX	✗	64.6	52.1	59.4	58.7	48.8	26.7	43.0	51.0	55.1	44.6	58.3	63.1	57.3	49.8	32.2	42.0	63.8	43.1	50.4	46.3
MF^{10}_{KMEANS}-LEX	✓	62.2	46.9	57.4	55.5	43.2	33.6	31.8	46.1	53.5	42.3	35.8	60.2	55.8	44.7	37.8	41.9	60.9	44.7	50.5	47.2
MF^{10}_{KMEANS}-LEX	✗	67.0	51.5	59.4	59.3	51.2	45.8	35.6	49.9	55.0	45.8	50.5	64.3	56.7	50.5	42.6	38.7	63.6	44.3	51.2	48.1
MF^{10}_{NEURAL}-LEX	✓	62.1	49.2	56.4	55.9	45.1	20.5	29.2	47.1	51.3	45.9	48.2	62.3	62.2	45.8	33.6	50.4	67.1	40.0	45.9	47.4
MF^{10}_{NEURAL}-LEX	✗	66.2	52.4	58.5	59.0	48.4	36.0	29.6	51.5	54.9	46.8	56.3	63.6	65.9	50.3	38.1	54.7	64.6	43.6	46.8	49.6

Table 9: Mean F_1 NER test results averaged over 5 runs transferring from high resource language **Japanese** to the low resource languages. *12Ad* indicates whether (✓) or not (✗) an adapter is placed in the 12th transformer layer. The top group (first three rows) includes models which leverage the original tokenizer which is not specialized for the target language. The second group (last 18 rows) include models with new tokenizers. Here we separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical initialization (∗-LEXINIT) by the dashed line. We additionally present the results for full model fine-tuning (FMT-∗).

	12Ad	Seen Languages				Unseen Languages but Covered Scripts										New Scripts					
		ka	ur	hi	**Avg**	cdo	mi	ilo	gn	xmf	sd	myv	bh	wo	**Avg**	am	bo	km	dv	si	**Avg**
mBERT		64.4	46.2	70.0	60.2	17.0	25.8	36.7	58.4	43.6	10.3	22.3	53.2	23.6	32.3	0.0	21.9	9.8	1.9	0.3	6.8
MAD-X	✓	65.3	60.6	64.1	63.4	29.5	57.8	52.0	53.5	51.2	32.3	57.0	57.9	43.2	48.3	7.9	22.9	9.6	16.8	14.4	14.3
MAD-X 2.0	✗	66.5	61.9	66.1	64.9	29.5	61.8	68.7	58.4	54.8	34.8	61.3	59.8	54.5	53.7	14.1	22.0	8.5	20.0	10.4	15.0
FMT-RANDINIT		64.3	44.0	66.9	58.4	37.2	2.0	59.6	20.0	64.9	41.1	9.1	55.8	4.9	32.7	30.2	38.3	56.7	31.7	54.7	42.3
EL-RAND	✓	61.8	59.1	64.9	61.9	31.8	6.7	52.7	43.3	55.4	38.0	29.4	48.6	3.0	34.3	36.9	59.8	61.4	27.7	44.3	46.0
EL-RAND	✗	64.4	60.7	67.8	64.3	35.9	5.9	58.1	45.9	61.5	43.6	27.8	50.4	2.8	36.9	41.4	59.4	62.2	30.1	49.2	48.5
MF^{1}-RAND	✓	61.8	52.9	57.8	57.5	36.4	18.5	46.8	49.3	59.0	39.0	57.5	40.3	34.2	42.3	36.4	58.2	66.7	45.2	47.0	50.7
MF^{1}-RAND	✗	63.8	55.1	59.5	59.5	38.4	21.4	51.8	51.6	63.5	42.8	59.8	44.9	35.0	45.5	42.0	58.4	69.8	46.7	48.6	53.1
MF^{10}_{KMEANS}-RAND	✓	60.4	55.5	61.5	59.1	32.3	10.5	43.5	47.6	60.0	25.5	29.7	44.2	21.0	34.9	36.6	46.5	68.3	40.1	46.3	47.5
MF^{10}_{KMEANS}-RAND	✗	61.3	59.4	63.5	61.4	39.3	11.4	49.9	51.4	62.4	28.8	35.3	47.8	26.6	39.2	43.1	45.7	69.1	42.7	48.2	49.8
MF^{10}_{NEURAL}-RAND	✓	62.9	56.4	63.2	60.8	24.5	65.0	49.0	46.1	58.1	40.7	56.9	49.0	29.5	46.6	35.0	35.0	65.5	15.5	43.2	38.8
MF^{10}_{NEURAL}-RAND	✗	64.1	59.6	65.8	63.2	26.8	34.2	56.6	49.2	59.8	41.5	63.4	53.7	33.8	46.6	39.6	36.1	67.1	18.9	49.6	42.3
FMT-LEXINIT		64.3	47.6	67.5	59.8	47.0	28.5	68.9	53.4	66.0	43.3	49.1	57.5	43.0	50.7	34.6	32.1	62.6	45.9	50.4	45.2
EL-LEX	✓	62.0	68.0	67.1	65.7	40.7	47.5	52.1	54.3	58.4	35.8	53.8	53.5	24.8	46.8	46.1	59.2	61.5	42.1	52.4	52.3
EL-LEX	✗	67.5	70.0	69.5	69.0	42.9	47.7	63.4	55.7	63.6	41.4	53.6	55.4	25.1	49.9	46.5	58.6	61.8	44.8	52.3	52.8
MF^{1}-LEX	✓	63.3	63.4	61.7	62.8	41.0	58.9	53.6	50.1	64.7	40.3	54.8	53.9	31.3	49.8	33.5	38.9	67.8	42.4	47.3	46.0
MF^{1}-LEX	✗	64.7	64.3	63.4	64.1	46.3	58.1	61.3	54.6	65.7	46.4	53.1	55.5	36.5	53.1	36.1	44.4	70.2	43.1	52.2	49.2
MF^{10}_{KMEANS}-LEX	✓	64.8	61.9	64.2	63.7	35.9	52.0	56.0	51.7	57.5	41.5	50.3	51.1	30.3	47.4	46.2	52.6	69.2	44.1	49.2	52.3
MF^{10}_{KMEANS}-LEX	✗	66.5	62.9	65.4	64.9	43.3	62.0	57.5	55.0	60.7	45.2	50.1	55.9	35.1	51.6	50.5	52.4	70.9	45.5	51.6	54.2
MF^{10}_{NEURAL}-LEX	✓	63.4	66.5	63.4	64.4	44.7	38.6	40.7	52.0	58.7	42.5	52.9	54.6	34.8	46.6	46.4	53.4	64.3	44.4	42.5	50.2
MF^{10}_{NEURAL}-LEX	✗	64.7	68.3	63.7	65.5	47.9	54.9	52.0	54.6	59.8	45.3	57.0	57.3	38.7	51.9	48.9	51.9	68.6	46.2	45.3	52.2

Table 10: Mean F_1 NER test results averaged over 5 runs transferring from high resource language **Arabic** to the low resource languages. *12Ad* indicates whether (✓) or not (✗) an adapter is placed in the 12th transformer layer. The top group (first three rows) includes models which leverage the original tokenizer which is not specialized for the target language. The second group (last 18 rows) include models with new tokenizers. Here we separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical initialization (∗-LEXINIT) by the dashed line. We additionally present the results for full model fine-tuning (FMT-∗).

	Seen Languages			Unseen Languages but Covered Scripts				New Script
	hi	ur	**Avg**	bh	myv	wo	**Avg**	am
	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS
mBERT	48.0 / 34.4	36.4 / 23.9	42.2 / 29.2	33.0 / 18.3	33.6 / 14.3	33.0 / 9.3	33.2 / 14.0	15.5 / 12.6
MAD-X 2.0	44.6 / 31.2	37.6 / 23.9	41.1 / 27.5	29.5 / 16.1	60.5 / 43.2	50.5 / 33.8	46.8 / 31.1	11.7 / 9.6
EL-RAND	45.9 / 32.5	38.2 / 23.9	42.1 / 28.2	29.9 / 13.6	57.8 / 35.0	33.2 / 11.1	40.3 / 19.9	26.4 / 13.0
MF^{1}-RAND	45.3 / 33.3	37.8 / 25.2	41.6 / 29.3	26.6 / 13.3	62.7 / 41.5	42.2 / 27.2	43.8 / 27.3	32.7 / 15.5
MF^{10}_{KMEANS}-RAND	46.3 / 33.1	37.8 / 25.6	42.1 / 29.3	31.4 / 16.0	62.9 / 42.8	24.9 / 13.7	39.7 / 24.1	29.7 / 14.9
MF^{10}_{NEURAL}-RAND	42.6 / 30.5	33.7 / 22.6	38.1 / 26.5	33.2 / 17.8	61.6 / 41.5	38.3 / 22.2	44.4 / 27.2	28.4 / 12.7
EL-LEX	47.1 / 33.1	38.1 / 24.9	42.6 / 29.0	31.3 / 17.2	63.3 / 40.9	51.3 / 34.0	48.6 / 30.7	32.4 / 9.1
MF^{1}-LEX	45.8 / 33.5	38.1 / 26.2	42.0 / 29.8	33.2 / 17.6	62.6 / 42.4	50.0 / 35.2	48.6 / 31.8	35.1 / 16.1
MF^{10}_{KMEANS}-LEX	46.0 / 33.4	38.8 / 26.5	42.4 / 30.0	33.1 / 17.7	64.2 / 44.5	52.6 / 35.9	50.0 / 32.7	27.7 / 14.1
MF^{10}_{NEURAL}-LEX	45.1 / 32.9	36.8 / 25.1	41.0 / 29.0	32.1 / 17.0	62.3 / 42.9	45.2 / 29.6	46.5 / 29.8	29.6 / 15.3

(a) Zero-shot DP scores. Source language: **English**.

	Seen Languages			Unseen Languages but Covered Scripts				New Script
	hi	ur	**Avg**	bh	myv	wo	**Avg**	am
	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS
mBERT	53.1 / 28.7	38.7 / 19.4	45.9 / 24.1	37.8 / 16.8	33.0 / 16.1	28.2 / 9.2	33.0 / 14.0	7.7 / 0.6
MAD-X 2.0	48.2 / 25.0	39.2 / 19.1	43.7 / 22.0	33.4 / 14.8	52.5 / 32.4	37.6 / 20.1	41.2 / 22.5	12.6 / 9.8
EL-RAND	46.2 / 22.7	34.1 / 15.7	40.1 / 19.2	30.0 / 10.0	49.6 / 25.8	22.7 / 6.9	34.1 / 14.2	27.3 / 11.0
MF^{1}-RAND	48.0 / 25.5	36.0 / 17.7	42.0 / 21.6	29.9 / 12.2	54.5 / 32.0	34.6 / 17.3	39.7 / 20.5	33.8 / 13.7
MF^{10}_{KMEANS}-RAND	46.5 / 25.0	35.7 / 17.8	41.1 / 21.4	29.5 / 11.4	54.9 / 32.6	22.8 / 11.4	35.7 / 18.5	34.3 / 13.3
MF^{10}_{NEURAL}-RAND	44.3 / 22.9	31.7 / 15.3	38.0 / 19.1	32.2 / 13.6	53.6 / 31.7	29.9 / 15.6	38.6 / 20.3	32.9 / 11.7
EL-LEX	47.1 / 24.6	36.9 / 17.7	42.0 / 21.1	31.6 / 14.0	53.5 / 29.6	40.3 / 20.9	41.8 / 21.5	34.4 / 10.8
MF^{1}-LEX	47.4 / 26.1	36.8 / 18.3	42.1 / 22.2	31.3 / 13.9	54.3 / 32.7	39.5 / 21.2	41.7 / 22.6	37.6 / 14.8
MF^{10}_{KMEANS}-LEX	47.6 / 25.8	38.4 / 18.9	43.0 / 22.4	33.5 / 14.3	54.5 / 32.7	40.5 / 22.5	42.8 / 23.2	28.4 / 11.2
MF^{10}_{NEURAL}-LEX	45.0 / 24.8	36.7 / 18.4	40.9 / 21.6	32.2 / 14.6	53.0 / 31.3	35.0 / 19.1	40.1 / 21.7	34.2 / 15.4

(b) Zero-shot DP scores. Source language: **Chinese**.

	Seen Languages			Unseen Languages but Covered Scripts				New Script
	hi	ur	**Avg**	bh	myv	wo	**Avg**	am
	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS
mBERT	57.0 / 40.7	43.7 / 29.8	50.3 / 35.2	45.9 / 30.6	27.7 / 9.3	18.9 / 2.3	30.9 / 14.0	10.0 / 0.9
MAD-X 2.0	55.5 / 38.3	45.4 / 30.4	50.5 / 34.3	42.6 / 27.2	37.9 / 19.7	22.2 / 6.1	34.2 / 17.7	12.3 / 9.7
EL-RAND	55.7 / 38.4	44.5 / 28.1	50.1 / 33.2	42.4 / 24.9	31.8 / 9.5	16.3 / 1.9	30.2 / 12.1	36.9 / 14.3
MF^{1}-RAND	53.1 / 36.7	42.7 / 27.4	47.9 / 32.1	38.5 / 23.9	36.6 / 14.6	21.1 / 4.3	32.0 / 14.3	36.6 / 15.2
MF^{10}_{KMEANS}-RAND	54.6 / 38.0	44.6 / 28.7	49.6 / 33.3	41.3 / 25.9	36.6 / 15.5	14.1 / 2.3	30.7 / 14.6	33.1 / 14.0
MF^{10}_{NEURAL}-RAND	52.2 / 35.8	40.9 / 26.7	46.6 / 31.3	42.9 / 27.5	35.5 / 15.0	18.8 / 4.3	32.4 / 15.6	29.0 / 10.2
EL-LEX	52.9 / 36.3	45.1 / 29.1	49.0 / 32.7	40.6 / 27.1	34.9 / 13.3	22.6 / 5.3	32.7 / 15.2	37.6 / 11.6
MF^{1}-LEX	53.1 / 36.8	44.2 / 28.6	48.7 / 32.7	41.5 / 27.0	36.1 / 14.8	23.9 / 5.6	33.8 / 15.8	31.2 / 12.5
MF^{10}_{KMEANS}-LEX	53.4 / 37.8	45.7 / 30.4	49.6 / 34.1	41.9 / 27.0	36.9 / 16.2	23.6 / 5.6	34.1 / 16.3	40.9 / 16.5
MF^{10}_{NEURAL}-LEX	53.5 / 37.9	44.5 / 29.2	49.0 / 33.5	41.3 / 27.0	35.6 / 16.7	20.4 / 4.8	32.4 / 16.1	32.6 / 14.5

(c) Zero-shot DP scores. Source language: **Japanese**.

	Seen Languages			Unseen Languages but Covered Scripts				New Script
	hi	ur	**Avg**	bh	myv	wo	**Avg**	am
	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS	UAS / LAS
mBERT	25.1 / 14.2	20.9 / 8.9	23.0 / 11.6	18.9 / 8.7	24.5 / 12.3	29.9 / 9.1	24.4 / 10.0	7.3 / 1.1
MAD-X 2.0	19.8 / 9.8	17.9 / 7.0	18.8 / 8.4	18.8 / 6.7	39.7 / 23.7	31.2 / 13.6	29.9 / 14.6	19.4 / 1.5
EL-RAND	19.7 / 10.0	15.0 / 6.0	17.4 / 8.0	17.5 / 6.6	42.2 / 25.5	25.2 / 8.9	28.3 / 13.7	30.0 / 6.7
MF^{1}-RAND	20.3 / 9.4	18.4 / 7.3	19.4 / 8.4	19.7 / 5.3	36.7 / 20.7	33.1 / 13.3	29.8 / 13.1	34.0 / 8.9
MF^{10}_{KMEANS}-RAND	25.1 / 11.9	20.9 / 7.9	23.0 / 9.9	21.0 / 5.5	40.0 / 23.8	25.1 / 10.0	28.7 / 13.1	32.7 / 7.6
MF^{10}_{NEURAL}-RAND	23.8 / 11.4	15.3 / 6.3	19.5 / 8.9	19.1 / 6.1	40.3 / 22.6	31.2 / 12.7	30.2 / 13.8	0.0 / 5.8
EL-LEX	20.5 / 10.6	15.8 / 6.8	18.1 / 8.7	19.0 / 6.7	43.3 / 26.9	37.2 / 16.6	33.2 / 16.8	31.4 / 8.3
MF^{1}-LEX	22.0 / 10.5	16.9 / 7.1	19.5 / 8.8	18.9 / 7.3	42.8 / 25.6	36.9 / 16.1	32.9 / 16.3	35.4 / 8.0
MF^{10}_{KMEANS}-LEX	25.2 / 12.8	21.7 / 8.8	23.4 / 10.8	15.1 / 5.6	31.6 / 17.0	38.0 / 17.5	28.2 / 13.4	28.7 / 6.6
MF^{10}_{NEURAL}-LEX	25.3 / 12.5	21.0 / 8.3	23.2 / 10.4	18.9 / 6.8	39.9 / 23.5	34.5 / 15.9	31.1 / 15.4	32.4 / 7.7

(d) Zero-shot DP scores. Source language: **Arabic**.

Table 11: Mean UAS and LAS test results for UD averaged over 5 runs transferring from the high-resource source languages (a) **English**, (b) **Chinese**, (c) **Japanses**, and (d) **Arabic**. The top group (first two rows) includes models which leverage the original tokenizer which is not specialized for the target language. The second group (last four rows) include models with new tokenizers. Here we separate models with randomly initialized embeddings (∗-RANDINIT) from models with lexical initialization (∗-LEXINIT) by the dashed line.

Language	Language code	Script
Afrikaans	af	Latin
Albanian	sq	Latin
Arabic	ar	Arabic
Aragonese	an	Latin
Armenian	hy	Armenian
Asturian	ast	Latin
Azerbaijani	az	Latin
Bashkir	ba	Cyrillic
Basque	eu	Latin
Bavarian	bar	Latin
Belarusian	be	Cyrillic
Bengali	bn	Bengali
Bishnupriya-manipuri	bpy	Bengali
Bosnian	bs	Latin
Breton	br	Latin
Bulgarian	bg	Cyrillic
Burmese	my	Myanmar
Catalan	ca	Latin
Cebuano	ceb	Latin
Chechen	ce	Cyrillic
Chinese-simplified	zh-Hans	Chinese
Chinese-traditional	zh-Hant	Chinese
Chuvash	cv	Cyrillic
Croatian	hr	Latin
Czech	cs	Latin
Danish	da	Latin
Dutch	nl	Latin
English	en	Latin
Estonian	et	Latin
Finnish	fi	Latin
French	fr	Latin
Galician	gl	Latin
Georgian	ka	Georgian
German	de	Latin
Greek	el	Greek
Gujarati	gu	Gujarati
Haitian	ht	Latin
Hebrew	he	Hebrew
Hindi	hi	Devanagari
Hungarian	hu	Latin
Icelandic	is	Latin
Ido	io	Latin
Indonesian	id	Latin
Irish	ga	Latin
Italian	it	Latin
Japanese	ja	Japanese
Javanese	jv	Latin
Kannada	kn	Kannada
Kazakh	kk	Cyrillic
Kirghiz	ky	Cyrillic
Korean	ko	Korean
Latin	la	Latin

Language	Language code	Script
Latvian	lv	Latin
Lithuanian	lt	Latin
Lombard	lmo	Latin
Low-saxon	nds	Latin
Luxembourgish	lb	Latin
Macedonian	mk	Cyrillic
Malagasy	mg	Latin
Malay	ms	Latin
Malayalam	ml	Malayalam
Marathi	mr	Devanagari
Minangkabau	min	Latin
Mongolian	mn	Cyrillic
Nepali	ne	Devanagari
Newar	new	Devanagari
Norwegian-bokmal	nb	Latin
Norwegian-nynorsk	nn	Latin
Occitan	oc	Latin
Persian	fa	Arabic
Piedmontese	pms	Latin
Polish	pl	Latin
Portuguese	pt	Latin
Punjabi	pa	Gurmukhi
Romanian	ro	Latin
Russian	ru	Cyrillic
Scots	sco	Latin
Serbian	sr	Cyrillic
Serbo-croatian	hbs	Latin
Sicilian	scn	Latin
Slovak	sk	Latin
Slovenian	sl	Latin
South-azerbaijani	azb	Arabic
Spanish	es	Latin
Sundanese	su	Latin
Swahili	sw	Latin
Swedish	sv	Latin
Tagalog	tl	Latin
Tajik	tg	Cyrillic
Tamil	ta	Tamil
Tatar	tt	Cyrillic
Telugu	te	Telugu
Thai	th	Thai
Turkish	tr	Latin
Ukrainian	uk	Cyrillic
Urdu	ur	Arabic
Uzbek	uz	Latin
Vietnamese	vi	Latin
Volapuk	vo	Latin
Waray-waray	war	Latin
Welsh	cy	Latin
West	fy	Latin
Western-punjabi	lah	Arabic
Yoruba	yo	Latin

Table 12: List of languages used in the pretraining. Taken from Chung et al. (2020)

Chapter 10

xGQA: Cross-Lingual Visual Question Answering

xGQA: Cross-Lingual Visual Question Answering

Jonas Pfeiffer[1], **Gregor Geigle**[1], **Aishwarya Kamath**[2], **Jan-Martin O. Steitz**[3]
Stefan Roth[3], **Ivan Vulić**[4], **Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[2]Center for Data Science, New York University
[3]Visual Inference Lab, Technical University of Darmstadt
[4]Language Technology Lab, University of Cambridge

Abstract

Recent advances in multimodal *vision and language* modeling have predominantly focused on the English language, mostly due to the lack of multilingual multimodal datasets to steer modeling efforts. In this work, we address this gap and provide xGQA, a new multilingual evaluation benchmark for the visual question answering task. We extend the established English GQA dataset (Hudson and Manning, 2019) to 7 typologically diverse languages, enabling us to detect and explore crucial challenges in cross-lingual visual question answering. We further propose new adapter-based approaches to adapt multimodal transformer-based models to become multilingual, and—vice versa—multilingual models to become multimodal. Our proposed methods outperform current state-of-the-art multilingual multimodal models (e.g., M^{3}P) in zero-shot cross-lingual settings, but the accuracy remains low across the board; a performance drop of around 38 accuracy points in target languages showcases the difficulty of zero-shot cross-lingual transfer for this task. Our results suggest that simple cross-lingual transfer of multimodal models yields latent multilingual multimodal misalignment, calling for more sophisticated methods for vision and multilingual language modeling.[1]

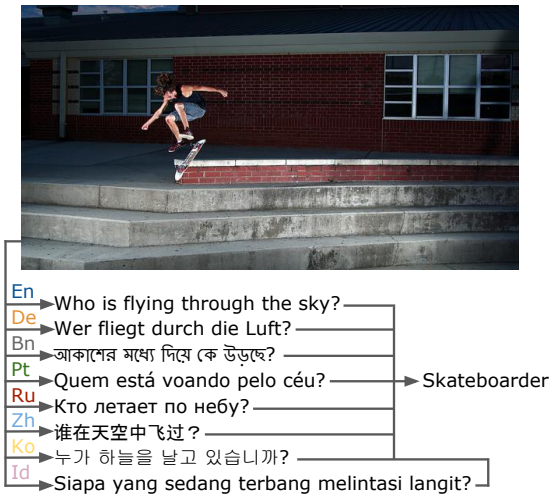


Figure 1: Example taken from the xGQA dataset with the same question uttered in 8 languages.

1 Introduction

Transformer-based architectures (Vaswani et al., 2017) have become ubiquitous in NLP (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020, *inter alia*) and in computer vision (CV) (Carion et al., 2020; Dosovitskiy et al., 2021), offering unmatched task performance. Having a shared architecture for multiple modalities opened up possibilities for effective fusion of information, yielding impressive performance gains across various multimodal tasks such as image captioning, phrase grounding, visual question answering, referring expression comprehension and image-text retrieval (Lu et al., 2019; Tan and Bansal, 2019; Li et al., 2020b; Zhang et al., 2021; Ni et al., 2021; Kamath et al., 2021; Miech et al., 2021; Frank et al., 2021; Bugliarello et al., 2021; Radford et al., 2021; Jia et al., 2021; Eichenberg et al., 2021; Singh et al., 2021; Fu et al., 2021; Yang et al., 2021; Yuan et al., 2021; Wang et al., 2021a; Li et al., 2021; Geigle et al., 2022, *inter alia*). Yet, progress in this area has been limited mostly to the English language, as the main multimodal datasets consist only of English text. Due to the scarcity of multilingual evaluation benchmarks, there has been limited development of models that tackle this joint problem.

Aiming to address this gap, in this paper we propose **xGQA**, a multilingual evaluation benchmark for the visual question answering task, extending the monolingual English-only GQA dataset (Hudson and Manning, 2019). For xGQA we manually translate and adapt the balanced GQA test-dev set into 7 new languages from 7 language families, covering 5 distinct scripts; see Figure 1 and Ta-

[1]The xGQA dataset is available online at: https://github.com/Adapter-Hub/xGQA.

Findings of the Association for Computational Linguistics: ACL 2022, pages 2497 - 2511

ble 1 later. In addition, we provide new fixed data splits to guide cross-lingual few-shot learning experiments, where only a small number of examples in the target language are utilized.

As pretraining is (i) notoriously computationally expensive for high-resource languages and (ii) only limited amounts of multilingual multimodal resources are available, we also propose computationally efficient adapter-based (Houlsby et al., 2019) approaches as additional baselines for constructing multilingual multimodal models. In a nutshell, we extend multimodal models pretrained only on English text (Zhang et al., 2021) to become multilingual and—vice versa—multilingual models (Devlin et al., 2019) to become multimodal. To this end, we follow the approaches of Artetxe et al. (2020) and Pfeiffer et al. (2020b, 2021) and extend monolingual and multilingual models to new languages and scripts via learning new tokenizers and corresponding word-embedding matrices, as well as adapters for the target languages. To transfer the respective multilingual multimodal adapter-based models to the target task, we propose a novel *modality-specific split architecture*, which uses modality dependent adapter weights (see Figure 2 for an illustration of the architecture).

Our results clearly indicate that the proposed adapter-based architecture outperforms the recent state-of-the-art pretrained multilingual multimodal M^3P model (Ni et al., 2021) in zero-shot cross-lingual settings. However, the overall performance of zero-shot transfer remains low across the board, with an average drop of around 38 accuracy points across target languages. Using a small number of target language examples in a few-shot setup considerably improves performance for all approaches, but cross-lingual transfer performance still lags substantially behind source language performance. This demonstrates the inherent difficulty of the task, even though the corresponding questions are arguably simple as they are template based and only contain 8.5 words on average (see Figure 1).

Contributions. 1) We propose the first evaluation benchmark for cross-lingual visual question answering, covering 7 diverse target languages; **2)** we propose novel adapter-based approaches for the creation of multilingual multimodal models; **3)** we systematically benchmark state-of-the-art and new multilingual multimodal models in zero-shot and few-shot learning setups, demonstrating the difficulty of the proposed task and serving as strong reference points for future work; **4)** we provide a thorough analysis of the different approaches, highlighting the aspects and question types that lead to the most common model failures, again motivating future work in this domain.

2 Background and Related Work

Multilingual Language Models. Pretrained multilingual transformer-based LMs such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) adopt the same pretraining regime as their respective monolingual counterparts: BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). They are pretrained via self-supervised masked language modelling objective (MLM) on concatenated text corpora of more than 100 languages, where text is tokenized using WordPiece, SentencePiece or BytePair encodings. These multilingual models have been shown to work surprisingly well for cross-lingual tasks, despite the fact that they do not rely on direct cross-lingual supervision (e.g., parallel data, translation dictionaries; Pires et al., 2019; Wu and Dredze, 2019; Artetxe et al., 2020; Hu et al., 2020; K et al., 2020; Rust et al., 2021).

Vision and Language Models. Most transformer-based multimodal models (Lu et al., 2019; Tan and Bansal, 2019; Chen et al., 2020; Li et al., 2020a; Gan et al., 2020; Li et al., 2020b; Bugliarello et al., 2021; Ni et al., 2021, *inter alia*) jointly encode text tokens and image region features by preprocessing images using object detection models—such as Faster R-CNN (Ren et al., 2015)—to extract features for regions of interest (RoI) (Anderson et al., 2018). The image region features are passed through an affine layer, which learns to project the region features to the joint embedding space of the multimodal transformer. The bounding box coordinates of the RoI act as positional embeddings for the visual features. As such, they undergo an affine transformation to the embedding space and are combined with their respective image region representation. The position-aware image region embeddings get passed into the transformer. The multi-head attention then attends over all text and image inputs at every layer, learning a joint representation of both modalities. On the other hand, Kamath et al. (2021) avoid using object detectors as a black-box for pre-extracting these region features and instead make it a central part of the multimodal transformer architecture. Training the object detector end-to-end with the multimodal transformer

adds flexibility and better representation capacity.

Similar to MLM, multimodal transformer-based models are trained with self-supervised objectives such as masked feature regression, masked object detection, masked attribute detection, and contrastive losses such as cross-modality matching (Tan and Bansal, 2019). Typically, image captioning datasets are used for pretraining such as COCO (Lin et al., 2014), Flickr30k (Plummer et al., 2015), Conceptual Captions (CC) (Sharma et al., 2018), and SBU (Ordonez et al., 2011). Similar to unimodal language models, the [CLS] token is used as a contextual representation for classification tasks.

Multilingual multimodal models have also been proposed recently: M^{3}P (Ni et al., 2021) is trained on the Wikipedias of 50 different languages and the English multimodal CC dataset. In order to align tokens of languages other than English with image representations, M^{3}P utilizes a code-switching mechanism, where words of the English CC examples are randomly replaced with words from corresponding bilingual dictionaries. In UC^2, Zhou et al. (2021) augment English multimodal datasets with other languages via machine translation and propose masked region-to-token modeling and visual translation language modeling.[2]

Adapters (Rebuffi et al., 2017; Houlsby et al., 2019) have been introduced as a more efficient fine-tuning strategy for transfer learning in NLP and CV. Instead of fine-tuning all the weights of a pretrained model on the target task, small feed-forward layers are introduced at each layer of the pretrained model. During task fine-tuning, only the adapter weights are updated, while the pretrained parameters remain fixed/frozen. Adapters have been shown to be very training efficient (Rücklé et al., 2021), and among an increasing amount of applications they can be utilized to transfer between domains (Rücklé et al., 2020) and tasks (Poth et al., 2021), and in machine translation (Bapna and Firat, 2019; Philip et al., 2020; Le et al., 2021) and cross-lingual transfer (Pfeiffer et al., 2020b, 2021; Üstün et al., 2020; Ansell et al., 2021, *inter alia*) scenarios.

Datasets. Pretraining and fine-tuning data for multilingual multimodal models is typically based on (multimodal information from) Wikipedia (**WikiCaps**, **WIT**, Schamoni et al., 2018; Srinivasan et al., 2021), or on available downstream task data. **Multi30k** (Elliott et al., 2016) is a multilingual image captioning dataset for retrieval-type questions, covering English, German, French, and Czech; **GEM** (Su et al., 2021) covers image and video retrieval tasks across 20 and 30 different languages, respectively; **HowTo100M** (Huang et al., 2021) is a multilingual and multimodal pretraining dataset for image and video retrieval; **MultiSubs** (Wang et al., 2021b) focuses on fill-in-the-blank tasks and lexical translation, covering English, Spanish, German, Portuguese, and French. Gao et al. (2015); Shimizu et al. (2018) propose bilingual visual question answering datasets for English, and Chinese and Japanese respectively. In contemporary work Liu et al. (2021) propose **MaRVL**, a binary multilingual question answering dataset similar to NLVR2 (Suhr et al., 2019), spanning 5 typologically diverse languages (Chinese, Tamil, Swahili, Indonesian, and Turkish).

Previous datasets predominantly focus on (arguably simpler) retrieval-type tasks, only cover a small set of similar languages (e.g., Multi30k, MultiSubs), or only cover binary questions. In contrast, we propose the first multilingual visual question answering dataset, which covers a typologically more diverse set of languages.

Most recently, **IGLUE** (Bugliarello et al., 2022)—a multilingual multimodal benchmark that integrates xGQA—was proposed: IGLUE brings together visual question answering, cross-modal retrieval, grounded reasoning, and grounded entailment tasks across 20 diverse languages.

3 xGQA

The original English GQA dataset (Hudson and Manning, 2019) was constructed by leveraging Visual Genome scene graphs (Krishna et al., 2017). An English question engine that utilizes *content* (i.e. information about objects, attributes, and relations provided) and *structure* (a linguistic grammar that couples hundreds of structural patterns and detailed lexical semantic resources) was used to generate over 22 million diverse questions, which are visually grounded in the image scene graphs. As the questions are automatically generated using templates, they do not necessarily reflect the wide spectrum of natural language, making any assumptions on the performance in the wild difficult.

Each question is associated with additional metadata such as **structural types**: (1) *verify* for yes/no questions (e.g. "Do you see any cats?"), (2) *query* for all open questions (e.g. "Who is wearing

[2]The model weights of UC^2 were not released by the time of experimentation.

Language	iso	Family	Script	Speakers
English	en	IE:Germanic	Latin	400M
German	de	IE:Germanic	Latin	95M
Portuguese	pt	IE:Romance	Latin	250M
Russian	ru	IE:Slavic	Cyrillic	150M
Indonesian	id	Austronesian	Latin	43M
Bengali	bn	IE:Iranian	Bengali	230M
Korean	ko	Koreanic	Korean	77M
Chinese	zh	Sino-Tibetan	Chinese	1.2B

Table 1: Languages covered by xGQA. IE stands for Indo-European.

jeans?"), (3) *choose* for questions that present two alternatives to choose from (e.g. "Is it red or blue?"), (4) *logical* which involve logical inference (e.g. "Is the field soft and snowy"), and (5) *compare* for comparison questions between two or more objects (e.g. "Are all the animals zebras?"). For further details regarding the metadata, we refer the reader to Hudson and Manning (2019).

Dataset Design. The principal objective when devising xGQA was to create a genuinely typologically diverse multimodal and multilingual evaluation benchmark for visual question answering. We utilize the balanced[3] test-dev set of GQA, which consists of 12,578 questions about 398 images.[4] Due to the defined structural patterns, the formulation of the questions is simple, with an average length of 8.5 words.[5] The resulting xGQA dataset covers translations in 7 languages, each representing a distinct language family, and contains examples written in 5 different scripts (see Table 1).

Few-Shot Data Splits. In order to conduct cross-lingual few-shot learning experiments, we provide new data splits of different sizes. We split on images and add all questions associated with the image to the respective set. The development and test sets consist of 50 and 300 images, respectively. The training splits consist of 1, 5, 10, 20, 25, and 48 images, see Table 2. We ensure that the distribution of structural types within each set is maintained.

Set	Test	Dev	Train					
#Img	300	50	1	5	10	20	25	48
#Ques	9666	1422	27	155	317	594	704	1490

Table 2: Few-shot dataset sizes. The GQA test-dev set is split into new development, test sets, and training splits of different sizes. We maintain the distribution of structural types in each split.

xGQA is the first truly typologically diverse multilingual multimodal benchmark, unlocking new experimentation and analysis opportunities in cross-lingual zero-shot and few-shot scenarios. While the questions in xGQA are intuitive and easy for humans to solve, we later show that current state-of-the-art models still have difficulty with transfer.

4 Baselines

To analyze the performance and current gaps on xGQA, we first evaluate the recently proposed M^{3}P model, which has been pretrained on multilingual and multimodal data. However, pretraining is computationally expensive and only limited amounts of multilingual multimodal resources are available. Therefore, we further propose new and more efficient approaches that (1) extend state-of-the-art multilingual language models to the multimodal domain and (2) provide multilingual capabilities to state-of-the-art multimodal models.

Unless noted otherwise, we follow the predominant fine-tuning strategy for GQA; a prediction head is placed on top of the output of a pretrained transformer. All possible 1853 answers of the GQA task are mapped to a class label. The question associated with an image together with the position-aware region features are passed as input to the transformer, supervised using a cross-entropy loss.[6]

4.1 Multimodal → Multilingual

OSCAR+^{Emb}. To extend a monolingual transformer LM to a multilingual domain, Artetxe et al. (2020) fine-tune a new word-embedding layer in the target language. Inspired by this idea, we now describe how we extend the current state-of-the-art monolingual multimodal transformer model OSCAR+ (Zhang et al., 2021) to learn new embeddings for the target languages.

In the *language-extension* phase, we replace the embedding matrix of OSCAR+ with a randomly

[3]To reduce biases in the conditional answer distribution Hudson and Manning (2019) utilize the structural metadata to downsample and create balanced datasets that are more robust against shortcuts and guesses.

[4]We chose to translate the test-dev set of GQA, as the labels for test-std are not released.

[5]For this reason, we chose to hire university students that are currently conducting their (Computer Science or Computational Linguistics) studies in English and are all fluent English speakers to translate the question into their native language. They were paid above the minimum hourly wage of the country of their respective university. After all questions have been translated, another, independent native speaker then verified the translations based on random spot checks.

[6]For instance, we use this strategy to fine-tune all parameters of M^{3}P on the GQA training data.

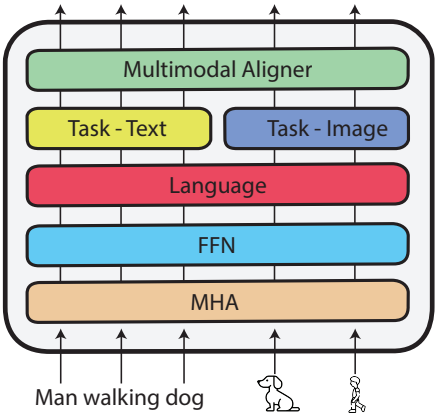


Figure 2: Architecture of an adapter-based multilingual multimodal model. Text and image inputs share the weights of the multi-head attention (MHA) and feed-forward (FFN) layers, as well as the *language* and *multimodal align* adapters. Each modality is passed through a modality specific *task* adapter, the outputs of which are concatenated.

initialized embedding matrix.[7] The transformer weights are frozen while only the newly introduced embeddings are fine-tuned on unlabeled text data of the target language with the MLM objective.

In the *target-task* phase, the original OSCAR+ model is fine-tuned on the English training data of GQA, where the transformer layers are fine-tuned, but the embedding layer is frozen. During inference, the embedding layer is replaced with the target language's embedding layer.

OSCAR+^{Ada}. We extend this by adding adapters.

In the *language-extension* phase we follow Pfeiffer et al. (2021) in order to extend the model to the target languages. Similar to OSCAR+^{Emb}, we train a new embedding layer. We further add *language* adapters at every transformer layer. Given that OSCAR+ is trained on English text, we follow Pfeiffer et al. (2020b) when training English *language* adapter modules, without replacing the embedding matrix. The transformer weights are frozen while only the *newly* introduced embeddings and *language* adapter weights are fine-tuned on unlabeled text data of the language.

For the *target-task* phase, we propose a novel modality-split architecture (see Figure 2) inspired by the cross-lingual transfer method of Pfeiffer et al. (2020b). At each transformer layer, text and image representations are passed through the pretrained multi-head attention (MHA) and feed-forward (FFN) layers. Both image and text representations are also passed through the pre-trained *language* adapters. Each modality is then passed through modality-specific *text* and *image task* adapters and next through a shared *multimodal alignment* adapter.[8] We follow Pfeiffer et al. (2020b), freezing transformer, embedding and *language* adapter weights during training, thus fine-tuning only the *task* and *multimodal aligner* adapter weights, together with the prediction head. At inference time, the embedding layer and the *language* adapters are replaced with the target language weights.

4.2 Multilingual → Multimodal

mBERT^{Ada}. For experiments where we extend a multilingual model to become multimodal, we utilize mBERT (Devlin et al., 2019).

Given that mBERT is able to represent many different languages, it is not necessary to learn new embedding layers for the target languages in the *language-extension* phase. Instead, we utilize the mBERT-compatible *language* adapters available on AdapterHub.ml (Pfeiffer et al., 2020a).[9]

For the *target-task* phase, we follow OSCAR+ for the image representation layer, where image features are combined with their respective positional information and passed through an affine transformation layer. We experiment with the same adapter architecture from Figure 2, as described for OSCAR+^{Ada}. We again freeze transformer, embedding and *language* adapter weights during training. However, in contrast to OSCAR+*, we randomly initialize and fine-tune the affine image transformation layer. We also fine-tune the *task*, *multimodal aligner* adapter weights, and prediction head, all on the GQA task. At inference time, the embedding layer and the *language* adapters are replaced with the corresponding target language weights.

5 Experimental Setup

5.1 Language-Extension Phase

For OSCAR+^{Emb} and OSCAR+^{Ada}, we follow the general setups proposed by Pfeiffer et al. (2020b,

[7] Following Pfeiffer et al. (2021), we copy the embeddings of lexically overlapping tokens (if such tokens exist) from the original embedding space to the new embedding space, as it typically works better than fully random initialization.

[8] We have compared multiple different architectures as illustrated in Figure 6 in the Appendix, finding this setup to perform best. We present results of the alternative architectures also in the Appendix.

[9] While all xGQA languages already have readily available language adapters on AdapterHub, any hypothetical extension of experiments to languages without such adapters would involve training their dedicated language adapters, e.g., following the procedure of Pfeiffer et al. (2020b).

2021). We train a new word-piece tokenizer for each target language with a vocabulary size of 30k. We fine-tune the randomly initialized embedding layer, and (for OSCAR+^{Ada}) adapter layers for 100k update steps with a batch size of 64 and a learning rate of $1e{-}4$. For mBERT^{Ada}, we utilize the language adapters from AdapterHub.ml.

5.2 Fine-tuning on GQA

We follow the standard setup proposed by Li et al. (2020b), passing the representation of the [CLS] token through a prediction head. We fine-tune the respective models using a cross-entropy loss with labels being all possible answers in the GQA dataset. Following prior work (Li et al., 2020b), we use a batch size of 192 and train for 5 epochs on the unbalanced GQA training portion.

M^{3}P. We fine-tune all weights of the pretrained model with a learning rate of $3e{-}5$.

OSCAR+^{Emb}, OSCAR+^{Ada}, and mBERT^{Ada}. We use the pretrained weights and image region features provided by Zhang et al. (2021). However, we do not pass the object attribute labels as inputs to the model. The object attribute labels are in English and utilizing them in cross-lingual scenarios is non-trivial.[10] We leave this for future work.

For the OSCAR+^{Emb} setting, we fine-tune the transformer weights and the prediction head and freeze the embedding layer, using a learning rate of $3e{-}5$. For the OSCAR+^{Ada} and mBERT^{Ada} settings, we add adapter layers as described in §4.1 and illustrated in Figure 2. We freeze all pretrained weights–including embeddings, transformer layers, and language adapters–and only fine-tune the newly introduced adapters and the prediction head. For mBERT^{Ada}, we also add and train the affine image transformation layer. We fine-tune the adapter-based models with a learning rate of $1e{-}4$.

5.3 Zero-Shot Cross-Lingual Transfer

For zero-shot cross-lingual evaluation, we utilize the model fine-tuned on the GQA training data and evaluate on the multilingual xGQA test data. The model checkpoint that performed best on the English GQA validation data is selected for transfer.

M^{3}P. As the model is pre-trained to cover, among others, xGQA languages, no additional steps are required for cross-lingual transfer.

OSCAR+^{Emb}. We replace the English embedding layer with the target-language embedding layer.

OSCAR+^{Ada}. We replace the English embedding and language adapter layers with the embedding and adapters layers of the target language.

mBERT^{Ada}. We replace the language adapter layers with the adapters layers of the target language.

5.4 Few-Shot Cross-Lingual Transfer

For few-shot cross-lingual scenarios we follow Lauscher et al. (2020) and start from the same fine-tuned model as for zero-shot transfer (see §5.3). We then fine-tune the same parts of the model as when training on the English training data as in §5.2, but on the small portions of multimodal data available in the target language. We train on the different data splits, consisting of 1, 5, 10, 15, 20, 25, and 48 images (see Table 2). We experiment with training for a different number of epochs (5, 10) using different learning rates ($1e{-}5$ and $5e{-}5$ for M^{3}P and OSCAR+^{Emb}, and $5e{-}5$ and $1e{-}4$ for OSCAR+^{Ada} and mBERT^{Ada}). We find that training for longer and with a larger learning rate performed best for all settings.

6 Results and Discussion

The main results are presented in Table 3 (zero-shot experiments) and in Table 4 (few-shot).

6.1 Zero-Shot Cross-Lingual Transfer

One of our core findings is that multimodal zero-shot cross-lingual transfer is extremely difficult; we witness an average drop in accuracy of more than 38 points on the target languages of the xGQA dataset compared to English GQA scores (e.g., compare the results with M^{3}P).

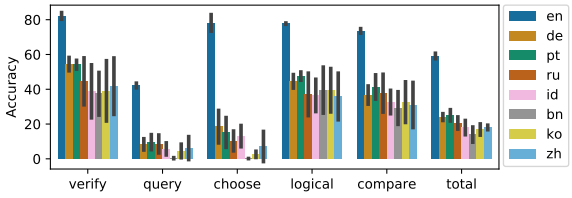
While, as expected, OSCAR+ achieves the best accuracy on the English test set, the massively multilingual models—M^{3}P and mBERT—perform considerably better in cross-lingual transfer.[11] This

[10] The replaced tokenizer and embedding representations of the target language potentially do not adequately represent English terms, resulting in a misalignment between the question (in the target language) and the object attributes (in English).

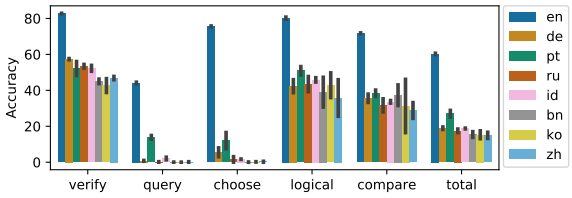
[11] The superior accuracy of OSCAR+ on the English test set is expected as the model was pretrained on large English multimodal data. We find that fine-tuning all transformer weights (OSCAR+^{Emb}) achieves slightly better results than only training adapter weights (OSCAR+^{Ada}). Our slightly lower scores compared to results by Zhang et al. (2021) can be explained by us (1) not fine-tuning the embedding layer, and (2) not utilizing the attribute labels. Further, previous works that focus only on English add the official *validation* set to the *training* set, use the official *test-dev* set as their dev set, and report their test scores of the official GQA test benchmark *test-std* for which labels are not available. Our scores follow the training splits, where we use the official *test-dev* set as the final test set, as described before in §3.

model	en	de	pt	ru	id	bn	ko	zh	mean
M3P	58.43 ±1.4	23.93 ±3.2	24.37 ±4.0	20.37 ±3.4	**22.57 ±6.1**	**15.83 ±3.6**	16.90 ±3.8	18.60 ±1.0	20.37
OSCAR+^{Emb}	**62.23 ±0.3**	17.35 ±1.0	19.25 ±0.4	10.52 ±4.0	18.26 ±0.4	14.93 ±2.0	17.10 ±1.8	16.41 ±3.2	16.26
OSCAR+^{Ada}	60.30 ±0.4	18.91 ±0.8	27.02 ±2.3	17.50 ±1.2	18.77 ±0.3	15.42 ±2.0	15.28 ±2.7	14.96 ±2.1	18.27
mBERT^{Ada}	56.25 ±0.5	**29.76 ±2.3**	**30.37 ±1.8**	**24.42 ±1.1**	19.15 ±2.8	15.12 ±1.9	**19.09 ±0.9**	**24.86 ±1.8**	**23.25**

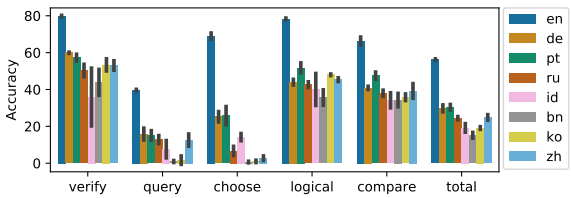
Table 3: Zero-shot transfer results when transferring from English GQA. Average accuracy and standard deviation are reported. Best results are highlighted in **bold**; *mean* scores are not averaged over the source language (English).



(a) M3P

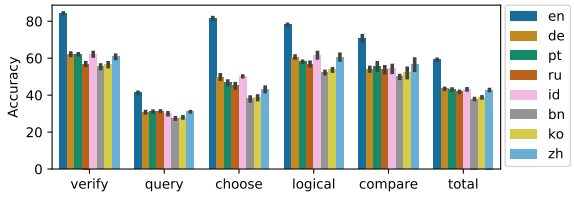


(b) OSCAR+^{Ada}

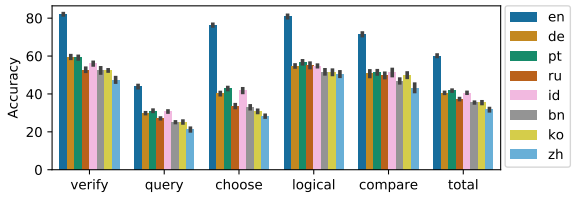


(c) mBERT^{Ada}

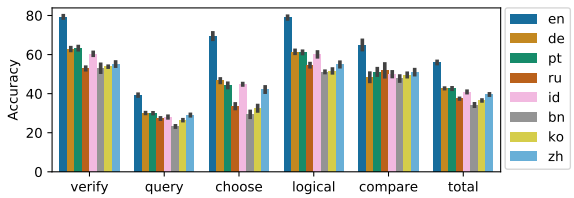
Figure 3: Zero-shot accuracy across different languages and structural question types from xGQA.



(a) M3P



(b) OSCAR+^{Ada}



(c) mBERT^{Ada}

Figure 4: Few-shot accuracy (with 48 images) across different languages and question types from xGQA.

indicates, that joint multilingual pretraining is important and a simple multilingual adapter-based or embedding-based extension of monolingual models achieves inferior cross-lingual performance.

While the pretraining method M^{3}P achieves better accuracy on the English test set, the adapter-based multimodal extension of mBERT outperforms M^{3}P in cross-lingual transfer. We hypothesize that, when fine-tuning all transformer weights on monolingual multimodal data, the cross-lingual alignment breaks within M^{3}P. However, this does not happen in adapter-based settings, as the multilingual weights are frozen and thus remain intact.

Analysis of Structural Question Types. Figure 3 depicts our analysis of the structural question types in zero-shot experiments. We observe large drops in accuracy especially for *query* and *choose* type questions. *Query* type questions are free-form and thus semantically the most difficult to answer, even in the source language (English). This explains the overall low accuracy across all approaches in zero-shot settings for this question type.

This is in stark contrast with the *choose*-type questions, which the models perform very well on in the source language. However, we report a substantial accuracy drop in zero-shot cross-lingual transfer. This decrease is most likely due to the nature of the question formulation and the modelling implementation. *Choose*-type questions are formulated such that the answer to the question is a word or phrase which appears in the question, i.e. "Is it red or blue?". The label classes, and consequently the prediction head, are constructed as a set of all answers appearing in the dataset. This means that the model learns a distributed repre-

Lang	Model	# Training Images						
		0	1	5	10	20	25	48
de	M3P	24.78	31.49	**39.31**	**41.05**	**42.22**	**42.54**	**43.16**
	OSCAR+^{Emb}	17.49	17.84	29.09	34.48	37.35	38.45	41.08
	OSCAR+^{Ada}	17.84	21.40	31.26	35.84	37.92	38.46	40.58
	mBERT^{Ada}	**32.41**	**33.87**	37.44	39.15	40.65	41.63	42.71
pt	M3P	26.73	32.98	37.23	**39.07**	**40.92**	**41.05**	43.06
	OSCAR+^{Emb}	19.36	22.55	32.42	36.37	39.01	40.15	**43.27**
	OSCAR+^{Ada}	24.58	29.61	34.73	37.46	38.82	39.70	41.75
	mBERT^{Ada}	**31.45**	**33.27**	**37.31**	38.88	40.51	41.03	42.62
ru	M3P	24.29	**32.32**	**36.71**	**38.53**	**39.94**	**40.13**	**41.85**
	OSCAR+^{Emb}	7.98	17.32	23.72	28.21	32.15	32.87	36.84
	OSCAR+^{Ada}	16.38	19.74	27.42	30.17	33.22	34.21	37.28
	mBERT^{Ada}	**25.51**	26.47	31.69	32.47	34.93	35.53	37.42
id	M3P	18.74	31.37	**37.24**	**38.65**	**41.07**	**42.00**	**43.12**
	OSCAR+^{Emb}	17.89	21.09	29.76	33.59	36.69	37.31	40.51
	OSCAR+^{Ada}	18.52	23.94	31.45	34.60	37.26	37.97	40.60
	mBERT^{Ada}	**19.77**	**31.99**	34.49	36.26	39.15	39.81	40.88
bn	M3P	**17.59**	17.33	**26.94**	**31.09**	**34.58**	**35.27**	**37.96**
	OSCAR+^{Emb}	13.35	**17.40**	21.67	26.61	31.94	32.78	36.97
	OSCAR+^{Ada}	13.96	15.60	22.35	27.20	31.25	31.81	35.45
	mBERT^{Ada}	13.38	11.33	23.10	26.55	31.60	32.26	34.18
ko	M3P	19.70	**22.94**	**32.28**	**35.50**	**37.72**	**37.84**	**38.61**
	OSCAR+^{Emb}	15.11	16.43	19.99	24.78	29.48	30.43	35.59
	OSCAR+^{Ada}	12.25	15.48	20.73	25.97	31.37	32.20	35.41
	mBERT^{Ada}	**19.92**	17.71	27.83	31.27	34.44	35.03	36.51
zh	M3P	19.66	**27.76**	**36.15**	**38.21**	**40.48**	**40.53**	**42.55**
	OSCAR+^{Emb}	12.66	14.77	19.17	22.13	27.97	29.08	33.24
	OSCAR+^{Ada}	13.20	15.12	19.67	22.74	26.81	28.19	31.69
	mBERT^{Ada}	**26.16**	23.47	32.93	35.82	38.22	37.89	39.57

Table 4: Average accuracy of few-shot results, utilizing different amounts of training data. The *0* column presents the best zero-shot results. These models are used as initialization for the subsequent few-shot experiments. **Bold** numbers indicate the best scores.

sentation of each answer in its final layer. Consequently, in cross-lingual transfer, the model is required to automatically align the question's options "red" or "blue" (translated in their respective language), with their English latent representation of the model's prediction head. The very low results in this category indicate that this cross-lingual word alignment breaks in zero-shot scenarios.

Overall, zero-shot transfer with our proposed multimodal adapter-based extension of mBERT (mBERT^{Ada}) achieves the best accuracy, with almost 3 points increase over M^{3}P and almost 5 points increase over OSCAR+. However, the overall accuracy of all approaches remains low in comparison to the results in English. This indicates that zero-shot multimodal cross-lingual transfer is extremely difficult, most likely due to the misalignment issue between visual and cross-lingual internal representations. To investigate this conjecture further, we run similar tests in few-shot setups, which should potentially mitigate the misalignment issue observed in zero-shot setups.

6.2 Few-Shot Cross-Lingual Transfer

The main results of few-shot experiments are provided in Table 4, while the plot illustrating the impact of different amounts of training data is shown in Figure 5. One crucial finding is that, as expected, utilizing an increasing amount of data instances in the target language consistently improves accuracy for all methods. This culminates in an improvement of up to 20 accuracy points when specializing the model with only 48 images in the target language. This indicates that a small number of target-language examples supports the models in partially repairing its internal cross-lingual multimodal alignment. Interestingly, we find that with as little as 5 images, and their corresponding questions, M^{3}P begins to outperform mBERT^{Ada}—the best performing zero-shot model.

We again analyze the impact of few-shot learning on accuracy across different structural question types, with the results depicted in Figure 4. The overall accuracy increases across all types compared to zero-shot scenarios (cf., Figure 3). However, the most pronounced gains are reported for *query* and *chose*-type questions, on which the model performed the worst in zero-shot setups. This implies the improved alignment between latent multimodal and multilingual representations, achieved via fine-tuning the model on a small amount of examples in the target language.

6.3 Language Transfer

We witness cross-lingual transfer capability patterns similar to those shown by previous work, where our models perform best on typologically close languages (Pires et al., 2019; Lauscher et al., 2020). Our models transfer best to German (de) and Portuguese (pt), both being part of the Indo-European (IE) language family and also sharing the same script (Latin) with the source language English (en). We see a small drop in accuracy for Russian (ru), Indonesian (id), and Chinese (zh) and a larger drop in accuracy for Bengali (bn) and Korean (ko). All of these languages are typologically different to the source language and in most cases do not share the same script. These differences highlight the importance of language diversity in cross-lingual transfer. Our benchmark thus enables experimentation and evaluation of multilingual multimodal models on a representative set of truly typologically diverse languages.

7 Contemporary Work

With the recent rise in interest in multilingual vision and language learning, contemporary work has

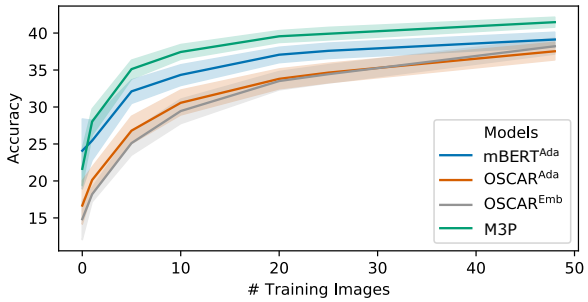


Figure 5: Few-shot accuracy with different training dataset sizes of the different approaches. Scores are averaged over all languages.

already further analyzed and extended the proposed xGQA dataset. We provide a brief description and pointers to this work in what follows.

Further Analysis. Liu et al. (2022) provide an extensive analysis of multilingual and multimodal models trained on cross-lingual visual question answering, and propose several approaches to mitigate the multilingual misalignment problem discussed in §6.1. Their results suggest that standard approaches taken from text-only cross-lingual transfer scenarios (Pires et al., 2019; Hu et al., 2020) do not leverage the full multilingual capability of the pretrained models. Interestingly, they find that a deeper prediction head does not have any measurable impact on the model's performance in the source language, while at the same time it considerably improves zero-shot transfer results across all target languages.

Translated Test Data. Bugliarello et al. (2022) propose the first benchmark for transfer learning across modalities, tasks, and languages, covering visual question answering, cross-modal retrieval, grounded reasoning, and grounded entailment tasks across 20 diverse languages. They extend the xGQA dataset by providing machine translated test-set questions and evaluate state-of-the-art monolingual multimodal models in a translate-test setup. In this setting, they achieve slightly better results. However, the performance remains to fall behind source language performance. The translate-test data can be found at iglue-benchmark.github.io.

8 Conclusion

We have proposed xGQA, a first cross-lingual evaluation benchmark for the visual question answering task. xGQA extends the English GQA dataset with development and test data in 7 more typologically diverse languages, covering 5 different scripts. As additional baselines, we have further proposed new adapter-based methods to extend unimodal multilingual models to become multimodal and—vice-versa—monolingual multimodal models to become multilingual. Our results have indicated that 1) efficient adapter-based methods slightly outperform the pretrained multilingual multimodal model M^3P in zero-shot scenarios, but 2) the overall zero-shot cross-lingual transfer yields harsh accuracy drops compared to the English performance for all models in comparison. Further, accuracy can be partially recovered via few-shot learning, where small amounts of training data are available in the target language. However, the large gaps remain, suggesting the inherent complexity of the cross-lingual task despite it being extremely intuitive and easy to solve by (bilingual) humans.

We hope that our dataset and error analysis will motivate future work on this task and, more broadly, in the exciting emerging domain of multilingual multimodal representation learning.

Acknowledgments

The Ubiquitous Knowledge Processing Lab acknowledges the financial support of the German Federal Ministry of Education and Research (BMBF) under the promotional reference 13N15897 (MISRIK), and the LOEWE initiative (Hesse, Germany) within the emergenCITY center. Jan-Martin O. Steitz is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. The work of Ivan Vulić is supported by a Huawei research donation and the ERC PoC Grant MultiConvAI: Enabling Multilingual Conversational AI (no. 957356). Stefan Roth is additionally supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 866008).

We thank Leonardo F. R. Ribeiro, Ji-Ung Lee, and Chen Liu for insightful feedback and suggestions on a draft of this paper.

References

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. 2021. Multimodal Pretraining Unmasked: A Meta-Analysis and a Unified Framework of Vision-and-Language BERTs. *Transactions of the Association for Computational Linguistics*, 9:978–994.

Emanuele Bugliarello, Fangyu Liu, Jonas Pfeiffer, Siva Reddy, Desmond Elliott, Edoardo Maria Ponti, and Ivan Vulić. 2022. IGLUE: A benchmark for transfer learning across modalities, tasks, and languages. *arXiv preprint*.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. UNITER: universal image-text representation learning. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 104–120. Springer.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Constantin Eichenberg, Sidney Black, Samuel Weinbach, Letitia Parcalabescu, and Anette Frank. 2021. MAGMA - multimodal augmentation of generative models through adapter-based finetuning. *arXiv preprint*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany. Association for Computational Linguistics.

Stella Frank, Emanuele Bugliarello, and Desmond Elliott. 2021. Vision-and-language or vision-for-language? on cross-modal influence in multimodal transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9847–9857. Association for Computational Linguistics.

Tsu-Jui Fu, Linjie Li, Zhe Gan, Kevin Lin, William Yang Wang, Lijuan Wang, and Zicheng Liu. 2021. VIOLET : End-to-end video-language transformers with masked visual-token modeling. *arXiv preprint*.

Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. 2020. Large-scale adversarial training for vision-and-language representation learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question answering. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2296–2304, Cambridge, MA, USA. MIT Press.

Gregor Geigle, Jonas Pfeiffer, Nils Reimers, Ivan Vulic, and Iryna Gurevych. 2022. Retrieve fast, rerank smart: Cooperative and joint approaches for improved cross-modal retrieval. *Transactions of the Association for Computational Linguistics*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Po-Yao Huang, Mandela Patrick, Junjie Hu, Graham Neubig, Florian Metze, and Alexander Hauptmann. 2021. Multilingual multimodal pre-training for zero-shot cross-lingual transfer of vision-language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2443–2459, Online. Association for Computational Linguistics.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6700–6709. Computer Vision Foundation / IEEE.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916. PMLR.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual BERT: an empirical study. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia. OpenReview.net.

Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. 2021. MDETR - modulated detection for end-to-end multimodal understanding. In *2021 IEEE International Conference on Computer Vision, ICCV 2021, Online, October 10-17, 2021*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Hang Le, Juan Miguel Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2021. Lightweight adapter tuning for multilingual speech translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 817–824. Association for Computational Linguistics.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020a. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11336–11344. AAAI Press.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. 2021. Grounded language-image pre-training. *arXiv preprint*.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020b. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 121–137. Springer.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer.

Chen Liu, Jonas Pfeiffer, Anna Korhonen, Ivan Vulić, and Iryna Gurevych. 2022. Delving deeper into

cross-lingual visual question answering. *arXiv preprint*.

Fangyu Liu, Emanuele Bugliarello, Edoardo Maria Ponti, Siva Reddy, Nigel Collier, and Desmond Elliott. 2021. Visually grounded reasoning across languages and cultures. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Online, November , 2021*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13–23.

Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2021. Thinking fast and slow: Efficient text-to-visual retrieval with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9826–9836. Computer Vision Foundation / IEEE.

Minheng Ni, Haoyang Huang, Lin Su, Edward Cui, Taroon Bharti, Lijuan Wang, Dongdong Zhang, and Nan Duan. 2021. M3P: learning universal representations via multitask multilingual multimodal pretraining. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3977–3986. Computer Vision Foundation / IEEE.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1143–1151.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10186–10203. Association for Computational Linguistics.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2641–2649.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to Pre-Train on? Efficient Intermediate Task Selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Online, November , 2021*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*

28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 91–99.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7930–7946. Association for Computational Linguistics.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020. MultiCQA: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2486, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, ACL 2021, Online, August 1-6, 2021*. Association for Computational Linguistics.

Shigehiko Schamoni, Julian Hitschler, and Stefan Riezler. 2018. A dataset and reranking method for multimodal MT of user-generated image captions. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas, AMTA 2018, Boston, MA, USA, March 17-21, 2018 - Volume 1: Research Papers*, pages 140–153. Association for Machine Translation in the Americas.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia. Association for Computational Linguistics.

Nobuyuki Shimizu, Na Rong, and Takashi Miyazaki. 2018. Visual question answering dataset for bilingual image understanding: A study of cross-lingual transfer using attention maps. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1918–1928. Association for Computational Linguistics.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2021. FLAVA: A foundational language and vision alignment model. *arXiv preprint*.

Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. 2021. WIT: wikipedia-based image text dataset for multimodal multilingual machine learning. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2443–2449. ACM.

Lin Su, Nan Duan, Edward Cui, Lei Ji, Chenfei Wu, Huaishao Luo, Yongfei Liu, Ming Zhong, Taroon Bharti, and Arun Sacheti. 2021. GEM: A general evaluation benchmark for multimodal tasks. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, pages 2594–2603. Association for Computational Linguistics.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5099–5110. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jianfeng Wang, Xiaowei Hu, Zhe Gan, Zhengyuan Yang, Xiyang Dai, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2021a. UFO: A unified transformer for vision-language representation learning. *arXiv preprint*.

Josiah Wang, Pranava Madhyastha, Josiel Figueiredo, Chiraag Lala, and Lucia Specia. 2021b. Multisubs: A large-scale multimodal and multilingual dataset. *arXiv preprint*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Faisal Ahmed, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2021. Crossing the format boundary of text and boxes: Towards unified vision-language modeling. *arXiv preprint*.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. 2021. Florence: A new foundation model for computer vision. *arXiv preprint*.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. VinVL: Revisiting Visual Representations in Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5579–5588.

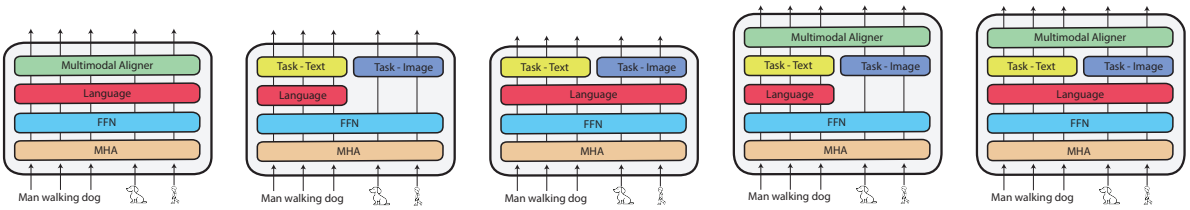
Mingyang Zhou, Luowei Zhou, Shuohang Wang, Yu Cheng, Linjie Li, Zhou Yu, and Jingjing Liu. 2021. UC2: universal cross-lingual cross-modal vision-and-language pre-training. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4155–4165. Computer Vision Foundation / IEEE.

A Appendix

We experiment with different multimodal adapter architectures as illustrated in Figure 6. In initial experiments we find that splitting the modalities (settings 2-5) outperforms a joint adapter (setting 1). However, a joint "alignment" architectures (settings 4-5) outperform settings where we only use modality-specific adapters (settings 2-3). We more thoroughly investigate settings 4-5 and report scores in Table 5. Interestingly, we find that when only using the language adapter for the textual inputs, cross-lingual accuracy drops for both OSCAR+ and mBERT; The difference is more pronounced for OSCAR+. We speculate that this is due to a latent misalignment of the representation spaces, partly due to the residual connection. Due to the better performance of setting 5 on average, we have reported scores of this architecture in the main paper (as illustrated in Figure 2).

model	Setting	en	de	pt	ru	id	bn	ko	zh	mean
OSCAR+^{Ada}	4	60.21	18.60	25.48	8.22	17.79	10.47	9.97	12.54	14.72
OSCAR+^{Ada}	5	**60.30**	**18.91**	**27.02**	**17.50**	**18.77**	**15.42**	**15.28**	**14.96**	**18.27**
mBERT^{Ada}	4	**57.83**	27.86	28.88	22.87	**20.86**	14.74	18.30	24.39	22.56
mBERT^{Ada}	5	56.25	**29.76**	**30.37**	**24.42**	19.15	**15.12**	**19.09**	**24.86**	**23.25**

Table 5: Zero-shot transfer results on xGQA for the different adapter architecture settings (as illustrated in Figure 6) when transferring from English GQA. Average accuracy is reported. Best results for each language and model type are highlighted in **bold**; *mean* scores are not averaged over the source language (English).



(a) Setting 1 (b) Setting 2 (c) Setting 3 (d) Setting 4 (e) Setting 5

Figure 6: The different multimodal multilingual adapter architectures we experimented with. The best performing architecture was setting 5, for which we present results in the main paper.

Chapter 11

Lifting the Curse of Multilinguality by Pre-training Modular Transformers

Lifting the Curse of Multilinguality by Pre-training Modular Transformers

Jonas Pfeiffer[*1,2,3], **Naman Goyal**[3], **Xi Victoria Lin**[3], **Xian Li**[3], **James Cross**[3], **Sebastian Riedel**[3], **Mikel Artetxe**[3]
[1]New York University, [2]TU Darmstadt,
[3]Meta AI

Abstract

Multilingual pre-trained models are known to suffer from *the curse of multilinguality*, which causes per-language performance to drop as they cover more languages. We address this issue by introducing language-specific modules, which allows us to grow the total capacity of the model, while keeping the total number of trainable parameters per language constant. In contrast with prior work that learns language-specific components post-hoc, we pre-train the modules of our **Cross**-lingual **Mod**ular (**X-MOD**) models from the start. Our experiments on natural language inference, named entity recognition and question answering show that our approach not only mitigates the negative interference between languages, but also enables positive transfer, resulting in improved monolingual and cross-lingual performance. Furthermore, our approach enables adding languages post-hoc with no measurable drop in performance, no longer limiting the model usage to the set of pre-trained languages.

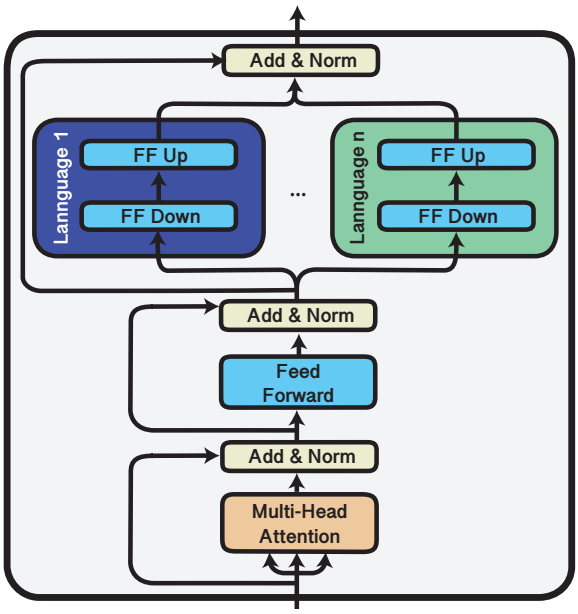


Figure 1: A transformer layer of our proposed modular architecture. The dark blue and green components illustrate the modular layers, which are language specific. The Multi-Head Attention and Feed-Forward components are shared by all languages.

1 Introduction

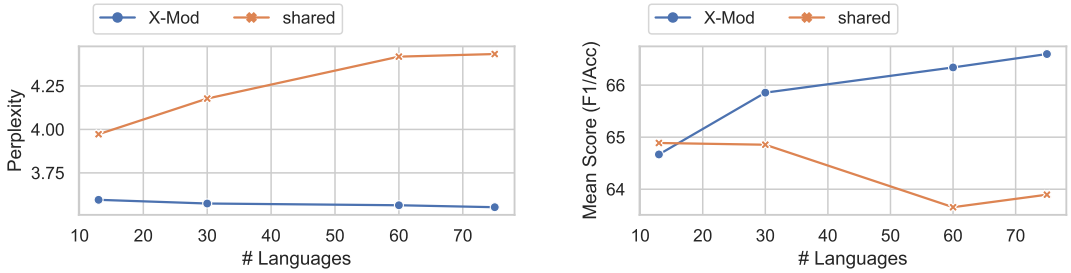
Recent work on multilingual NLP has focused on pre-training transformer-based models (Vaswani et al., 2017) on concatenated corpora of a large number of languages (Devlin et al., 2019; Conneau et al., 2020). These multilingual models have been shown to work surprisingly well in cross-lingual settings, despite the fact that they do not rely on direct cross-lingual supervision (e.g., parallel data or translation dictionaries; Pires et al., 2019; Wu and Dredze, 2019; Artetxe et al., 2020; Hu et al., 2020; K et al., 2020; Rust et al., 2021).

However, recent work has uncovered fundamental limitations of multilingual transformers. Conneau et al. (2020) observe that pre-training a model with a fixed capacity on an increasing amount of languages only improves its cross-lingual performance up to a certain point, after which performance drops can be measured—a phenomenon known as *the curse of multilinguality* (Figure 2). As such, prior work had to find a trade-off between supporting more languages and obtaining better performance on a smaller set of languages.

In this work, we address this problem by introducing language-specific, modular components during pre-training (Figure 1). Our **Cross**-lingual, **Mod**ular (**X-MOD**) language model shares the majority of the transformer parameters between all pre-training languages, while providing each language with individual capacity to learn idiosyncratic information without increasing the total number of trainable parameters per language. While previous adapter-based approaches (Figure 3a) extend pre-trained multilingual language models (LMs) with modular components *after* pre-training, we add modular components *during* pre-training, thereby

[*] Work done while interning at Meta AI.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3479 - 3495



(a) Mean Perplexity.

(b) Mean Performance on XNLI and NER.

Figure 2: Average (a) perplexity and (b) transfer performance on XNLI and NER across pre-trained languages when training on an increasing number of languages. Each model has seen the **same amount of examples** in each language. Lower perplexity and higher downstream score indicate better performance. Refer to Figure 4 for per-task performance, and Appendix A for per-language performance.

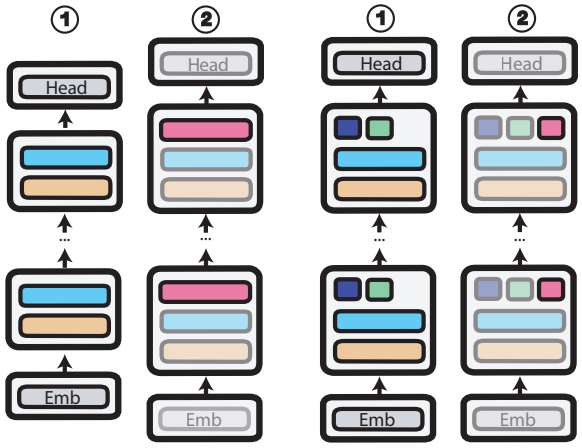
preparing the model to be extended to new languages post-hoc. Our experiments on natural language inference (NLI), named entity recognition (NER), and question answering (QA) demonstrate that our modular architecture not only is effective at mitigating interference between languages, but also achieves positive transfer, resulting in improved monolingual and cross-lingual performance. In addition, we show that X-MOD can be extended to unseen languages, with no measurable drop in performance, by learning its corresponding modules and leaving the shared parameters frozen. All in all, we propose a multilingual architecture that can scale to a large number of languages without any loss in performance, and can be further extended to new languages after pre-training.[1]

2 Background and related work

We provide a background on multilingual and modular language modelling, as well as approaches that extend LMs to new languages.

2.1 Multilingual transformers

Recent LMs (Devlin et al., 2019; Conneau et al., 2020), based on transformer architectures (Vaswani et al., 2017) and pre-trained on massive amounts of multilingual data, have surpassed (static) cross-lingual word embedding spaces (Ruder et al., 2019; Glavas et al., 2019) for cross-lingual transfer in NLP (Pires et al., 2019; Wu and Dredze, 2019; Wu et al., 2020; Hu et al., 2020; K et al., 2020). Transformer-based models are **1)** pre-trained on textual corpora using Masked Language Modelling (MLM). They are then **2)** fine-tuned on labelled data of a downstream task in a *source* language and **3)** directly applied to perform inference in a *target* language (Hu et al., 2020).



(a) Adapter-based

(b) X-MOD

Figure 3: Our proposed architecture in comparison to adapter-based approaches. (a) Previous approaches ① utilize non-modular pre-trained transformer models and ② extend them with modular adapter components. (b) We ① pre-train the transformer with modular units from the get-go, *preparing* the model to be ② extended with additional modular units later on. Yellow and light blue components indicate standard Multi-Head Attention and Feed-Forward layers. The remaining (non-gray) components are bottleneck (modular) units. Grayed-out components are frozen.

2.2 Modular language models

Modular approaches have a long standing history in NLP, preceding pre-trained models (Andreas et al., 2016). They have recently re-gained interest for transformer-based models, where mix-

[1]Code and pre-trained models are available at: https://github.com/pytorch/fairseq/tree/main/examples/xmod.

ture of experts (MoE; Shazeer et al., 2017) approaches have enabled training trillion parameters models in a distributed fashion (Fedus et al., 2021). More recently modular MoE approaches have been shown to improve domain-specific pre-training of LMs (Gururangan et al., 2021). In a similar trend, 'expert' modules have been added to (non-modular) pre-trained LMs post-hoc, predominantly referred to as adapters (Rebuffi et al., 2017, 2018; Houlsby et al., 2019). Next to being extremely parameter (Houlsby et al., 2019; Mahabadi et al., 2021a; He et al., 2022) and training efficient (Pfeiffer et al., 2020a; Rücklé et al., 2021), these modular approaches allow models to be extended to new data settings (Chen et al., 2019; Rücklé et al., 2020), where newly learned knowledge can be combined (Stickland and Murray, 2019; Wang et al., 2021a; Pfeiffer et al., 2021a; Lauscher et al., 2020a; Mahabadi et al., 2021b; Poth et al., 2021), or stacked for combinatory cross-lingual (Pfeiffer et al., 2020b, 2021b; Üstün et al., 2020; Vidoni et al., 2020; Ansell et al., 2021b,a; Wang et al., 2021b) as well as NMT scenarios (Bapna and Firat, 2019; Philip et al., 2020; Chronopoulou et al., 2020; Le et al., 2021; Üstün et al., 2021; Stickland et al., 2021; Garcia et al., 2021).

2.3 Weaknesses, improvements, and extensions of language models

Next to the *curse of multilinguality*, recent works have shown substantially reduced cross-lingual and monolingual abilities of models for low-resource languages with smaller pre-training data (Wu and Dredze, 2020; Hu et al., 2020; Lauscher et al., 2020b; Artetxe et al., 2020; Pfeiffer et al., 2020b, 2021b; Chau et al., 2020b; Ponti et al., 2020).

K et al. (2020); Artetxe et al. (2020) show that a shared vocabulary is not necessary for cross-lingual transfer. Chung et al. (2021) demonstrate that decoupling the input embeddings from the prediction head improves the performance on a number of downstream tasks. Dufter and Schütze (2020) show that the number of parameters and training duration is interlinked with the model's multilingual capability. Chung et al. (2020); Rust et al. (2021) show that the tokenizer plays an important role in the per-language downstream task performance, which Clark et al. (2022); Xue et al. (2022); Tay et al. (2021) take to the extreme by proposing tokenizer-free approaches.

To extend a monolingual LM to other languages, Artetxe et al. (2020) train a new embedding layer with a corresponding target-language tokenizer, while freezing the pre-trained transformer weights. Tran (2020) extend a monolingual model to new languages using bilingual corpora. Wang et al. (2020); Chau et al. (2020a) extend the vocabulary of multilingual models with a small number of target-language tokens, to improve the performance in the target language. Muller et al. (2021) propose a transliteration based approach, Vernikos and Popescu-Belis (2021) propose subword mappings, and Pfeiffer et al. (2020b, 2021b); Vidoni et al. (2020); Ansell et al. (2021b) propose adapter-based approaches to extend multilingual models to unseen languages.

While these approaches achieve considerable performance gains over unseen languages, they are outperformed by standard full fine-tuning methods for seen languages. One can further argue that, as the pre-trained models have already been cursed by multilinguality, the adapter-based approaches build upon sub-optimal parameter initializations.[2] In our work, we consequently aim to **1)** modularize the model from the start to prepare the model to be **2)** extendable to new languages post-hoc.

3 Proposed approach

We propose X-MOD, a modular multilingual architecture that combines shared and language-specific parameters. In contrast to prior work, we pre-train modular models from the get-go. Our models can be extended to new languages after pre-training, and used for cross-lingual transfer learning in downstream tasks.

Architecture. As illustrated in Figure 1, we extend the transformer-based architecture from mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) by incorporating language-specific modules—bottleneck feed-forward layers—at every transformer layer. We learn a separate module for each language, whereas the attention and feed-forward components are shared. While the total number of parameters of the model grows linearly with the number of languages, the training and inference cost does not increase (as measured in FLOPs), as only the module in the relevant language is used for each input. Inspired by the adapter[3] architecture of Pfeiffer et al. (2021a) we

[2] We investigate this claim further in §6.2.

[3] The term 'adapter' refers to newly introduced layers within a pre-trained (frozen) model. These layers *adapt* the

place our 'modules' after the LayerNorm of the feed-forward transformer block, and the residual connection is placed after the LayerNorm;[4] the LayerNorm before and after the modular component is shared.[5]

Pre-training procedure. Similar to Conneau et al. (2020), we pre-train our model on MLM on combined monolingual corpora in multiple languages. Examples of each language are passed through the shared embedding matrix as well as the multi-head attention and feed-forward components at each layer. As each layer contains a language-specific modular component, the examples are routed through the respective designated modular bottleneck layer. Given that each example only requires access to a single module, modules can be efficiently stored on only a subset of GPUs in distributed training.

Extending to new languages. The modular design of our model allows us to extend it to new languages after pre-training. To that end, we learn new embeddings and adapter modules for the target language through MLM, while the rest of the components are frozen.[6] Consequently, we are able to extend the model to a new language by learning a small number of new parameters, without affecting performance in the set of pre-trained languages. Following Pfeiffer et al. (2021b), we learn a new subword vocabulary for the added languages, and initialize the embeddings of lexically overlapping tokens from the original embedding matrix.

Fine-tuning on downstream tasks. To transfer the models to cross-lingual downstream tasks, we fine-tune the shared weights only on the source language data, while keeping the modular components and the embedding layer frozen. We follow the standard fine-tuning procedure of adding a prediction head on top of the CLS token. We then replace the source language modules (as well as embedding layer for *added* languages) with the target language parameters, passing the text of the target language through the model.[7]

representations of the pre-trained mode; we train these modular components together with the transformer weights, and therefore refer to them as modules.

[4]We find that the residual connection proposed by Pfeiffer et al. (2021a) results in training instabilities when trained together with the transformer weights.

[5]Preliminary results showed that sharing the LayerNorm results in better cross-lingual transfer performance.

[6]Following Artetxe et al. (2020) we train positional embeddings.

[7]We initially also experimented with stacking adapters on top of the language modules similar to Pfeiffer et al. (2020b, 2021b). While this approach is considerably more parameter efficient, we find that fine-tuning all shared weights slightly outperformed the adapter-based approach.

4 Experimental design

We detail the baseline and models (§4.1), and their training (§4.2) and evaluation settings (§4.3).

4.1 Model variants

We pre-train separate models for all combinations along the following axes:

X-MOD vs. SHARED. To evaluate the effectiveness of our X-MOD model, we aim to compare ourselves to a conventional non-modular architecture. However, simply removing the modular component would be unfair, as the number of FLOPs and trainable parameters per language would not be the same—both in terms of pre-training, as well as fine-tuning. Consequently, for our baseline model—where all parameters should be *fully* shared between all languages—we include a single bottleneck layer right after the Feed-Forward component. Effectively, this is the same architecture as our X-MOD model, just with a single module that is shared by all languages. We refer to this as the SHARED model throughout this paper.[8] To extend the SHARED model to unseen languages, we follow Artetxe et al. (2020) and only learn a new embedding layer, freezing the transformer parameters. To fine-tune the SHARED model on a downstream task, we freeze the embedding layer, as well as the (single) module, thereby fine-tuning an equal amount of parameters on the downstream task as the X-MOD model.[9]

13 vs. 30 vs. 60 vs. 75 languages. So as to understand how each approach is affected by the curse of multilinguality, we pre-train the X-MOD and SHARED models on 4 increasing sets of languages. We start with an *initial* set of 13 typologically diverse languages that we evaluate on, and add additional languages for larger sets of 30, 60, and 75 languages. In addition, we keep a set of 7 held-out languages that we extend the pre-trained models to. Table 1 lists the specific languages in each

[8]Extending the **total** number of shared parameters would be unfair, as X-MOD and SHARED would not have the same FLOPs nor the same number of trainable parameters when fine-tuning.

[9]Adapter-based approach such as MAD-X (Pfeiffer et al., 2020b) would be an alternative. However, this would require training on languages twice—once during pre-training, and once when adding adapters—which is not directly comparable to X-MOD. Nonetheless, we report results in §6.2.

Pre-trained languages	13-LANGS	***en***, ***ar***, **fr**, ***hi***, ko, ***ru***, ***th***, ***vi***, ta, id, fi, **sw**, ka
	30-LANGS	13-LANGS + cs, eu, hr, hu, hy, it, lt, ml, mn, ms, pl, ro, si, sk, sq, sv, tl
	60-LANGS	30-LANGS + af, am, be, bn, ca, cy, da, eo, et, fa, ga, gl, gu, ha, is, ku, la, lv, mk, ne, nl, no, ps, pt, sa, sd, sl, so, sr, te
	75-LANGS	60-LANGS + as, br, bs, fy, gd, jv, kn, mg, mr, om, or, pa, su, xh, yi,
Added languages		**bg**, ***de***, ***el***, ***es***, ***tr***, **ur**, ***zh***,

Table 1: **Selection of languages.** We pre-train different models on 4 sets of languages, and further extend them to a set of held-out languages post-hoc. We evaluate on XNLI (languages in **bold**), NER (underlined languages) and XQuAD/MLQA (languages in *italic*). For more details about the language selection, see Appendix C.

group. The selection and split of *initial* as well as *added* languages is motivated by typological and geographical diversity, as well as the availability of downstream task evaluation data.

Controlling for total vs. per-language updates. Conneau et al. (2020) investigated the effect of adding more languages during pre-training, while training on an equal number of update steps. However, increasing the number of languages while keeping the number of updates constant results in the model seeing less data in each individual language. As such, it remains unclear if the curse of multilinguality happens because of negative interference, or simply because the number of updates for each specific language is smaller. So as to understand this, we compare (1) training on an equal number of *update steps* and (2) training on an equal number of *seen examples* per language. We start with the set of 13 languages (Table 1) and train the respective models for 125k update steps. When adding more languages, we compare (1) training models on each set of languages for 125k update steps, and (2) increasing the number of update steps such that the models are trained on the same number of examples in each of the initial 13 languages. For the latter, this amounts to training for 195k, 265k and 269k update steps, respectively.

4.2 Training details

Data and hyperparameters. We sample languages with $\alpha = 0.7$ and train our models with a batch size of 2048 across 64 V100 GPUs on the CC100 dataset (Conneau et al., 2020) using fairseq (Ott et al., 2019). All our models extend the *base* transformer architecture, with 12 layers and 768 dimensions. Modules are implemented with a bottleneck size of 384. The shared transformer weights account for 270M parameters, whereas each individual module accounts for 7M parameters. We train our models with a linear learning rate decay peaking at $7e{-}4$ during pre-training and $1e{-}4$ when adding languages.

Vocabulary. As we aim to identify the impact of *modularity* on the curse of multilinguality, we control for consistent tokenization across the different axes. We therefore tokenize using the XLM-R vocabulary for all our pre-training experiments.[10] However, for languages added post-hoc, we learn a *new* SentencePiece tokenizer for each of the target language,[11] as the languages potentially use scripts unseen by the original tokenizer.

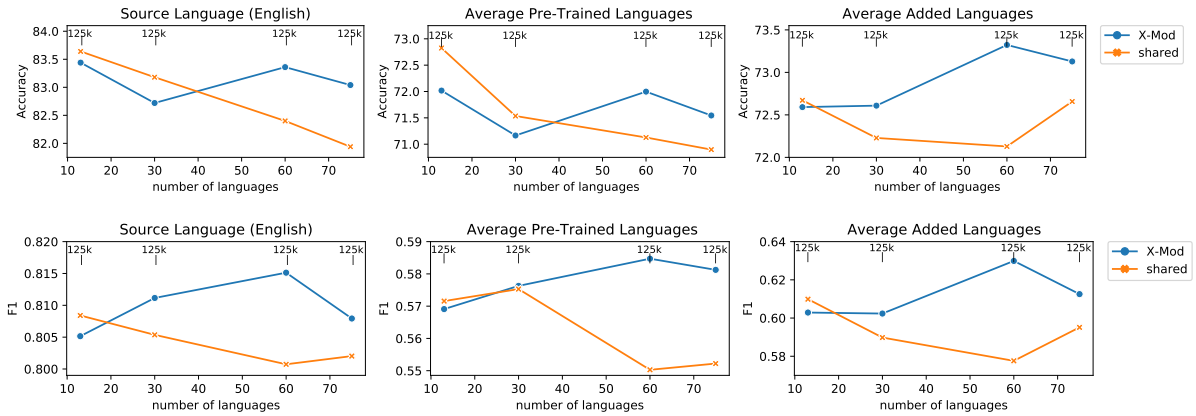
4.3 Evaluation

We conduct experiments on NLI, NER, and QA. In all cases, we fine-tune the model on English and measure the zero-shot transfer performance in other languages. For NLI we train on MultiNLI (Williams et al., 2018) and evaluate on XNLI (Conneau et al., 2018). For QA, we train on SQuAD (Rajpurkar et al., 2016) and evaluate on XQuAD (Artetxe et al., 2020) and MLQA (Lewis et al., 2020). For NER, we use WikiANN (Pan et al., 2017; Rahimi et al., 2019). We experiment with learning rates $1e{-}4$, $3e{-}4$, and $5e{-}4$ and train for 3 or 5 epochs for QA and 5 or 10 epochs for NER and NLI. For NER and NLI we take the hyperparameter setting performing best on the development sets, averaged across the pre-trained languages (Table 1). For SQuAD we take the best performing checkpoint evaluated on the English development set, and report the cross-lingual test set results.[12] All results are averaged across 5 random seed runs.

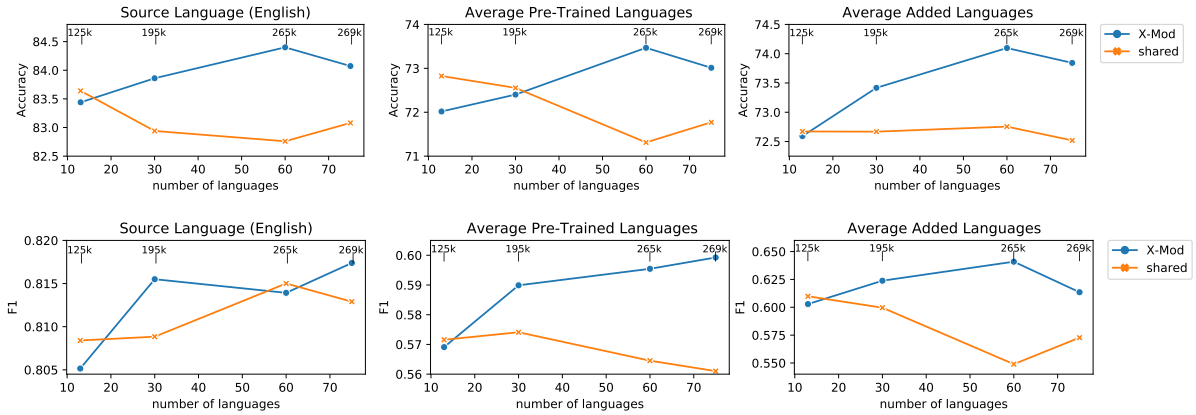
[10] Rust et al. (2021) have previously demonstrated the impact of the multilingual tokenizer on the downstream task performance: languages underrepresented in the sub-word vocabulary exhibit considerable performance drops when compared to vocabularies dedicated to the respective language.

[11] We train the new tokenizers for a vocabulary size of 30k.

[12] In contrast to NER and NLI, the cross-lingual evaluation benchmarks of SQuAD do not provide a development set for each target language on the basis of which the best checkpoint can be selected. Consequently, we select the checkpoint based



(a) All models are trained for 125k update steps. Models trained on **more languages** have seen **less examples** in each language.



(b) Models trained on more languages are trained longer. All models have seen the **same amount of examples** in each language.

Figure 4: Test set results on XNLI (top) and NER (bottom) for models trained on different numbers of languages. *Source Language (English)* only includes scores of the source language. *Average Pre-Trained Languages* includes all evaluation languages that the model was pre-trained on. *Average Added Languages* includes all languages that were added to the model after pre-training. Scores are averaged across all languages and random seeds.

5 Results and discussion

We present results for pre-trained languages in §5.1 and added languages in §5.2.

5.1 Pre-trained languages

In Figure 4 we plot downstream task results of models pre-trained on different amounts of languages. Table 2 reports the individual language performance for the models trained on 60 languages.

The Curse of Multilinguality. Conneau et al. (2020) showed that multilingual LMs trained on *increasing* amounts of languages, while *maintaining* the number of update steps, exhibit drops in downstream task XNLI performance. We reproduce these results, both in terms of language modelling perplexity (Figure 2a),[13] as well as downstream task performance on XNLI *and* NER (Figure 4a). We further find that the curse of multilinguality does not *only* happen *because* the total number of update steps per language decreases, but *also* when all SHARED models are trained on the *same* number of examples per language (Figure 4b). This confirms that fully shared architectures suffer from negative interference.

Lifting the Curse. While for the SHARED model we witness negative interference between languages in terms of perplexity, the X-MOD model is able to *maintain* performance, and even improves for a subset of languages. We observe similar patterns in the downstream task performance: In both our experimental setups—(1) we control for the number of update steps (Figure 4a); (2) we control for the number of per-language seen examples (Figure 4b)—our X-MOD model—in contrast to the SHARED model—is able to maintain, or

on the best performance on the English development set.

[13]For per-language perplexity see Appendix A.

		en	ar	fr	hi	ko	ru	th	vi	ta	id	fi	sw	ka	avg
NER	X-MOD	81.4	**78.9**	**77.2**	**70.1**	**53.0**	**59.1**	2.8	**66.2**	51.1	50.5	**78.6**	**73.4**	**67.3**	**62.8**
	SHARED	**81.5**	74.1	74.7	64.4	46.0	58.3	**4.0**	63.7	**52.5**	**51.5**	74.4	57.2	61.5	58.8
XNLI	X-MOD	**84.4**	**71.2**	**77.6**	**68.3**	-	**74.1**	**71.7**	**73.4**	-	-	-	**66.9**	-	**73.5**
	SHARED	82.8	69.2	75.6	66.6	-	73.2	68.5	72.5	-	-	-	62.1	-	72.5
XQuAD	X-MOD	**85.1**	**68.1**	-	**67.5**	-	**75.0**	**66.3**	**74.9**	-	-	-	-	-	**72.8**
	SHARED	83.8	64.6	-	65.8	-	72.7	63.0	72.6	-	-	-	-	-	70.4
MLQA	X-MOD	**80.1**	**58.6**	-	**60.7**	-	-	-	**67.5**	-	-	-	-	-	**66.7**
	SHARED	79.6	53.6	-	58.7	-	-	-	64.9	-	-	-	-	-	64.2

Table 2: Pre-trained language results for the modular and shared model variants, pre-trained on the set of 60 languages for 265k update steps. For NER and MLQA we report F_1, for XNLI *accuracy* scores. Scores are averaged across all 5 random seeds of the best hyperparameter setting, evaluated on the development set.

		bg	de	el	es	tr	ur	zh	avg
NER	X-MOD	**77.6**	**75.1**	**75.2**	**71.9**	**72.6**	**54.7**	**21.6**	**64.1**
	SHARED	74.9	66.3	69.6	49.1	64.8	50.4	9.2	54.9
XNLI	X-MOD	**77.4**	**75.4**	**76.2**	**78.5**	**72.4**	**64.9**	**73.8**	**74.1**
	SHARED	76.3	74.1	74.9	77.3	71.0	64.3	71.4	72.8
MLQA	X-MOD	-	**63.8**	-	**68.6**	-	-	**61.7**	**64.8**
	SHARED	-	58.9	-	66.7	-	-	56.5	60.7

Table 3: Results for added languages, for models pre-trained on the set of 60 languages for 265k update steps. We report F_1 and *accuracy* scores which are averaged across all 5 random seeds of the best hyperparameter setting on the development set.

even outperform model variants trained on less languages. These results demonstrate that the added per-language capacity is sufficient for the model to adequately represent all languages.

Surprisingly, X-MOD not only maintains performance, but actually slightly improves while we increase the number of languages we pre-train on. This is even the case for settings where the model sees *less* examples in the target language. This suggests that increasing the language diversity can have a positive impact on the model's cross-lingual representation capability.

X-MOD vs SHARED. Overall, the X-MOD model pre-trained on 60 languages achieves the best cross-lingual performance.[14] Our results on XNLI, NER, MLQA, and XQuAD in Table 2 demonstrate consistent performance gains over the SHARED model for every task and across (almost) all high- as well as low-resource languages.

[14]We find that the X-MOD model trained on 75 languages is less stable than the versions trained on less languages. We think that this can be attributed to the 15 added languages being extremely low resource—we only train for an additional 4k update steps—resulting in the respective randomly initialized modules being updated very infrequently. This variance could potentially be mitigated by training for longer.

5.2 Extending to unseen languages

We further evaluate the cross-lingual performance of languages added in the second step; (1) on the architectural side—comparing the SHARED with the X-MOD modelling variant—and (2) by comparing the performance when *pre-training* on the language, vs. when *adding* the language post-hoc.

Modular vs Shared. We evaluate if the additional per-language capacity improves the extendability of the X-MOD model. On the right in Figure 4a we plot the results for added languages on XNLI (top) and NER (bottom). Similarly, we plot the results for the models where we control for the number of seen examples per target language in Figure 4b. We find that the X-MOD model consistently outperforms the SHARED model, with a peak performance when pre-training on 60 languages, demonstrating that the language specific capacity is beneficial for adding new languages post-hoc. We report results for the 60 language versions in Table 3, demonstrating the consistent advantage of the X-MOD over the SHARED model.

Pre-training vs Adding Languages. To evaluate if there is a measurable difference on downstream performance for languages that we *pre-train* on vs. those we *add post-hoc*, we train 2 models on *different* initial sets of languages, adding the respectively missing ones in the second step. So as to understand if the typological similarity of languages has impact on the downstream task performance, we split the *initial* and *added* languages (Table 1) of our previous experiments into two parts. The *first* split consists of languages where the model was pre-trained on at least one language of the same language family (e.g. English vs. German). The *second* split consists of languages that are part of a **unique** language family, i.e. the model was **not**

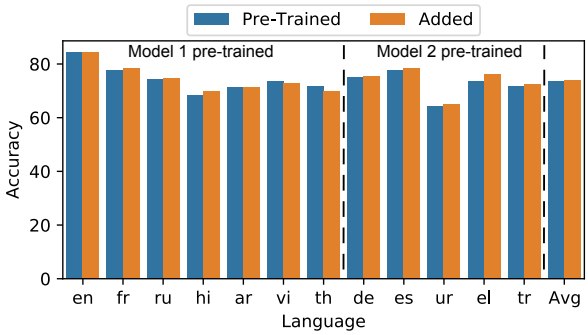


Figure 5: XNLI test set accuracy of X-MOD models pre-trained on different languages in comparison to those added post-hoc (Table 4).

Language	iso	Family	Script	Model 1	Model 2
English	en	IE: Germanic	Latin	pre-train	add
German	de	IE: Germanic	Latin	add	pre-train
French	fr	IE: Romance	Latin	pre-train	add
Spanish	es	IE: Romance	Latin	add	pre-train
Russian	ru	IE: Slavic	Cyrillic	pre-train	add
Ukranian	uk	IE: Slavic	Cyrillic	add	pre-train
Hindi	hi	IE: Iranian	Devanagari	pre-train	add
Urdu	ur	IE: Iranian	Arabic	add	pre-train
Arabic	ar	Afro-Asiatic	Arabic	pre-train	add
Hebrew	he	Afro-Asiatic	Hebrew	add	pre-train
Vietnamese	vi	Austro-Asiatic	Latin	pre-train	add
Thai	th	Kra-Dai	Thai	pre-train	add
Korean	ko	Koreanic	Korean	pre-train	add
Japanese	ja	Japonic	Japanese	add	pre-train
Greek	el	IE: Hellenic	Greek	add	pre-train
Turkish	tr	Turkic	Latin	add	pre-train

Table 4: Selection of 2 sets of languages that we either pre-train on, or add post-hoc. The last 6 languages in the list are part of language families which are *unique* in the total list of languages we pre-train on (Table 1), i.e. none of our models was pre-trained on a language of the same family.

pre-trained on a language of the same family (Table 4). Consequently, we pre-train two models on two sets of languages, adding the respective other set post-hoc.[15]

Our XNLI results (Figure 5) demonstrate that the per-language performance is on par when pre-training vs. when adding the language post-hoc.[16] We also find that the family does not have a measurable effect on the performance of the language. Our results therefore suggest that it is sufficient to train X-MOD on only a subset of languages for which sufficient pre-training data exists. Essentially, X-MOD has the potential to cover all languages of the world, as the model has the capability to be adapted to new languages post-hoc.

[15] In previous experiments, the modular model trained on 60 languages achieved the best performance. Therefore, the models in these experiments are also trained on 60 languages. Both models are trained on the same additional languages, i.e. the 60-LANGS of Table 1, where only the 13-LANGS differ.

[16] The models have seen an equal amount of examples in the respective languages in each case.

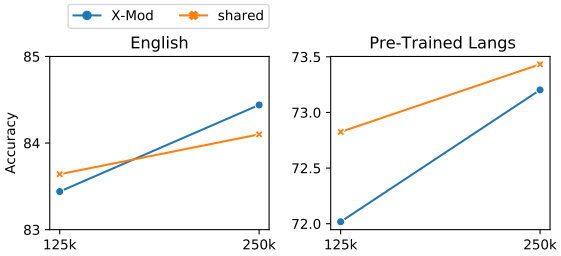


Figure 6: Results on XNLI when when pre-training on 13 languages for 125k and 250k update steps.

6 Further analysis

We further analyze the impact of the number of update steps on X-MOD (§6.1) and compare our method to adapter-based approaches (§6.2).

6.1 The importance of update steps

In Figure 4 we have witnessed a slight edge of the SHARED model over the X-MOD model, when training on only 13 languages and only training for 125k update steps. Dufter and Schütze (2020) found that it requires a large number of update steps for a model pre-trained on multiple languages to become multilingual; with the added per-language capacity we hypothesize that update steps also play an important role for modular models. We compare the downstream task performance of models pre-trained on 13 languages, when training for 125k with 250k update steps in Figure 6. When training for longer we find that the X-MOD model begins to outperforms the SHARED model in the source language, while almost closing the gap in the cross-lingual setting. This supports the hypothesis that the X-MOD model requires more update steps when training only on a small number of languages, in order for modularity to "kick-in".

6.2 X-MOD vs. Adapters

As illustrated in Figure 3, from an architecture perspective X-MOD is similar to previously proposed multilingual Adapter-based methods (MAD-X; Pfeiffer et al., 2020b). MAD-X utilizes a pre-trained massively multilingual transformer-based model and fine-tunes newly introduced adapter weights on languages the model has seen during pre-training, and ones the model has not been

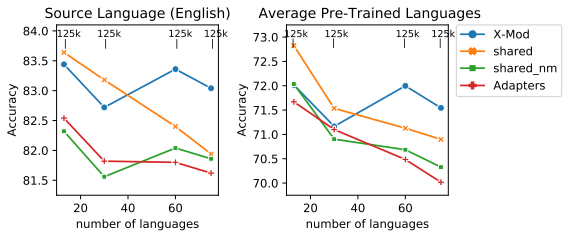


Figure 7: Comparison on XNLI of X-MOD and *shared* models with an Adapter baseline, all models are pre-trained for 125k update steps.

trained on. For a fair comparison in terms of *seen examples* and *number of update steps* we train a transformer model without module components (*shared_nm*) for 100k update steps on the respective languages (Table 1). We subsequently train adapters on each of the target languages for another 25k update steps.[17] We report results in comparison to X-MOD in Figure 7, here results for *shared_nm* are for a model that was trained for 125k update steps to instantiate a fair comparison.

Our results demonstrate that the additional capacity of adapters added *after* pre-training is not able to mitigate the curse of multilinguality which has already had a catastrophic impact on the shared transformer weights; the performance of the adapters strongly correlates with the performance of the corresponding fully shared model *shared_nm*. Consequently, adding language-specific capacity *during* pre-training is important, as the curse of multilinguality cannot be lifted post-hoc.

7 Conclusions

In this paper, we have evaluated the effectiveness of modular multilingual language modelling across multiple axes. We have demonstrated that by providing additional per-language capacity, while maintaining the total number of trainable parameters per language, we are not only able to mitigate negative interference between languages, but additionally achieve positive transfer. Our results suggest that it is sufficient to train our proposed X-MOD model only on a subset of languages for which sufficient amounts of textual data is available. Unseen languages can be added post-hoc, with no measurable drop in performance on XNLI. By *pre-training* the model in a modular fashion, we thus mitigate negative interference of idiosyncratic information, while simultaneously preparing the model to be extendable to unseen languages.

While in this work we have simulated language adding scenarios with a held out set of languages, in future work we aim to evaluate the performance on truly low-resource languages such as MasakhaNER (Adelani et al., 2021) and AmericasNLI (Ebrahimi et al., 2021). We further aim to evaluate the cross-lingual transfer performance from typologically more diverse source languages, besides English.

[17] We follow Pfeiffer et al. (2020b) and train adapter weights with a learning rate of 0.0001. While they have found that cross-lingual transfer performance of adapters converges at ∼20k update-steps, we would like to stress that our experimental setup is only **one** of multiple different valid versions. A more thorough investigation to find the optimal number of update steps for pre-training and subsequent adapter training is necessary, which was out of scope for this work.

Acknowledgments

We thank Samuel Broscheit for insightful feedback and suggestions on a draft of this paper, as well as the ARR reviewers and meta-reviewers for their valuable comments.

References

David Ifeoluwa Adelani, Jade Z. Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen Hassan Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba O. Alabi, Seid Muhie Yimam, Tajuddeen Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin P. Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane Mboup, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima Diop, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named Entity Recognition for African Languages. In *Transactions of the Association for Computational Linguistics 2021*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, San Diego California, USA, June 12-17, 2016, pages 1545–1554. The Association for Computational Linguistics.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulic. 2021a. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint*.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021b. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548. Association for Computational Linguistics.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020a. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020b. Parsing with multilingual bert, a small treebank, and a small corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 1324–1334.

Vincent S. Chen, Sen Wu, Alexander J. Ratner, Jen Weng, and Christopher Ré. 2019. Slice-based learning: A programming model for residual learning in critical data slices. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9392–9402.

Alexandra Chronopoulou, Dario Stojanovski, and Alexander Fraser. 2020. Reusing a Pretrained Language Model on Languages with Limited Corpora for Unsupervised NMT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2703–2711, Online. Association for Computational Linguistics.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4536–4546. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. CANINE: pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 8440–8451.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for BERT's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4423–4437, Online. Association for Computational Linguistics.

Abteen Ebrahimi, Manuel Mager, Arturo Oncevay, Vishrav Chaudhary, Luis Chiruzzo, Angela Fan, John Ortega, Ricardo Ramos, Annette Rios, Ivan Vladimir, Gustavo A. Giménez-Lugo, Elisabeth Mager, Graham Neubig, Alexis Palmer, Rolando A. Coto Solano, Ngoc Thang Vu, and Katharina Kann. 2021. AmericasNLI: Evaluating Zero-shot Natural Language Understanding of Pretrained Multilingual Models in Truly Low-resource Languages. *arXiv preprint*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint*.

Xavier Garcia, Noah Constant, Ankur Parikh, and Orhan Firat. 2021. Towards continual learning for multilingual machine translation via vocabulary substitution. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1184–1192, Online. Association for Computational Linguistics.

Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 710–721.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2021. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *10th International Conference on Learning Representations, ICLR 2022, Virtual Conference, April 25 - 29, 2022*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 12-18 July 2020, Virtual Conference*.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual BERT: an empirical study. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020a. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020b. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online.

Hang Le, Juan Miguel Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2021. Lightweight adapter tuning for multilingual speech translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 817–824. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 565–576. Association for Computational Linguistics.

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of*

the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1946–1958.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), EMNLP 2020, Virtual Conference, 2020*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Online, November , 2021*.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4996–5001.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online. Association for Computational Linguistics.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to pre-train on? efficient intermediate task selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10585–10605. Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 151–164.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7930–7946. Association for Computational Linguistics.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020. Multicqa: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2471–2486. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Asa Cooper Stickland, Alexandre Berard, and Vassilina Nikoulina. 2021. Multilingual domain adaptation for NMT: decoupling language and domain information with adapters. In *Proceedings of the Sixth Conference on Machine Translation, WMT@EMNLP 2021, Online Event, November 10-11, 2021*, pages 578–598. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Prakash Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint*.

Ke M. Tran. 2020. From english to foreign languages: Transferring pre-trained language models. *arXiv preprint*.

Ahmet Üstün, Alexandre Berard, Laurent Besacier, and Matthias Gallé. 2021. Multilingual unsupervised neural machine translation with denoising adapters. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6650–6662. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Giorgos Vernikos and Andrei Popescu-Belis. 2021. Subword mapping and anchoring across languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2633–2647, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marko Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. In *arXiv preprint*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1405–1418. Association for Computational Linguistics.

Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021b. Efficient test time adapter ensembling for low-resource language varieties. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 730–737, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Shijie Wu, Alexis Conneau, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 6022–6034.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages

833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics 2022*.

A Additional results

We report MLQA and XQuAD results on pre-trained languages in Tables 5 and 6, respectively, and MLQA results on added languages in Table 7. Table 8 report NER results on more languages.

Figures 9, 10 and 11 report per-language results as we increase the amount of languages on language modeling perplexity, XNLI and NER, respectively.

B Intermediate checkpoints

Our results in §6.1 suggest that, when the number of languages is small, X-MOD becomes more competitive with SHARED as the number of training steps increases. So as to understand if this behavior also holds for models covering more languages, we evaluate intermediate checkpoints for the 60-LANG model on XNLI. As shown in Figure 8, we find that the X-MOD model continuously outperforms the SHARED model. This suggests that the SHARED model immediately suffers from negative interference between languages, while the added, language-specific components of the X-MOD model are able to mitigate the curse of multilinguality, resulting in considerable performance gains at all evaluated checkpoints.

C Language selection

We provide more details about our selection of languages in Table 9.

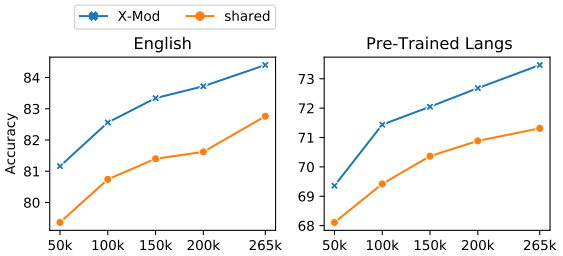


Figure 8: Results on XNLI using intermediate checkpoints of the models trained on 60 languages.

	en F1 / EM	ar F1 / EM	hi F1 / EM	vi F1 / EM	*avg* F1 / EM
X-MOD	**80.1 / 66.9**	**58.6 / 38.9**	**60.7 / 42.4**	**67.5 / 46.1**	**66.7 / 48.6**
SHARED	79.6 / 66.5	53.6 / 33.9	58.7 / 40.4	64.9 / 43.8	64.2 / 46.2

Table 5: Average F_1 and Exact Match results for **pre-trained languages**, on the test set of **MLQA** for the X-MOD and SHARED model variants, pre-trained on the set of 60 languages for 265k update steps. **Bold** numbers indicate better performance for the respective language.

	en F1 / EM	ar F1 / EM	hi F1 / EM	ru F1 / EM	th F1 / EM	vi F1 / EM	*avg* F1 / EM
X-MOD	**85.1 / 73.4**	**68.1 / 52.4**	**67.5 / 50.3**	**75.0 / 57.8**	**66.3 / 52.6**	**74.9 / 54.6**	**72.8 / 56.9**
SHARED	83.8 / 72.1	64.6 / 48.5	65.8 / 48.3	72.7 / 54.5	63.0 / 48.0	72.6 / 52.1	70.4 / 53.9

Table 6: Average F_1 and Exact Match results for **pre-trained languages**, on the test set of **XQuAD** for the X-MOD and SHARED model variants, pre-trained on the set of 60 languages for 265k update steps. **Bold** numbers indicate better performance for the respective language.

	de F1 / EM	es F1 / EM	zh F1 / EM	*avg* F1 / EM
X-MOD	**63.8 / 48.9**	**68.8 / 50.3**	**61.7 / 36.4**	**64.8 / 45.2**
SHARED	58.9 / 44.1	66.7 / 48.3	56.5 / 32.2	60.7 / 41.5

Table 7: Average F_1 and Exact Match results for **added languages**, on the test set of **MLQA** for the X-MOD and SHARED model variants, pre-trained on the set of 60 languages for 265k update steps. **Bold** numbers indicate better performance for the respective language.

	en	af	ar	bn	et	eu	fa	fi	fr	hi	hu	id	it	ka	ko	ru	sw	ta	th	vi	avg
X-MOD	81.4	**78.9**	43.5	**63.2**	**76.2**	**62.2**	**44.3**	**78.6**	**77.2**	**70.1**	**78.3**	50.5	**78.7**	**67.3**	**53.0**	**59.1**	**73.4**	51.1	2.8	**66.2**	**62.8**
SHARED	**81.5**	74.1	**44.2**	62.4	70.7	58.1	40.3	74.4	74.7	64.4	74.2	**51.5**	75.5	61.5	46.0	58.3	57.2	**52.5**	**4.0**	63.7	59.5

Table 8: Average F_1 results for **pre-trained languages**, on the test set of **NER** for the X-MOD and SHARED model variants, pre-trained on the set of 60 languages. **Bold** numbers indicate better performance for the respective language.

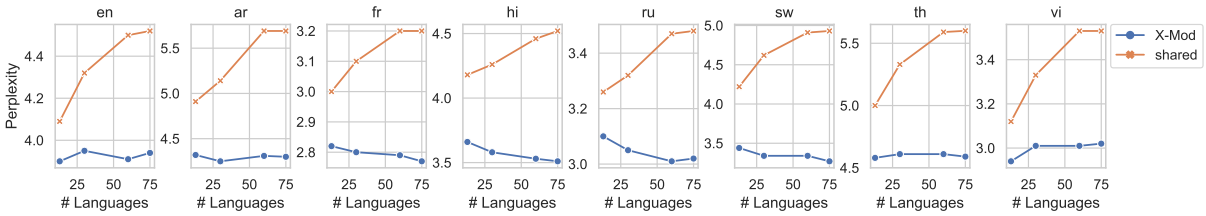
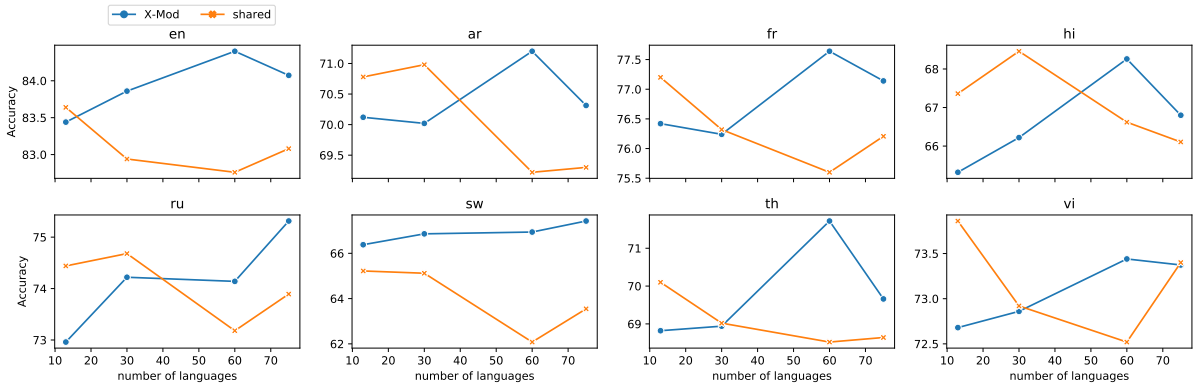
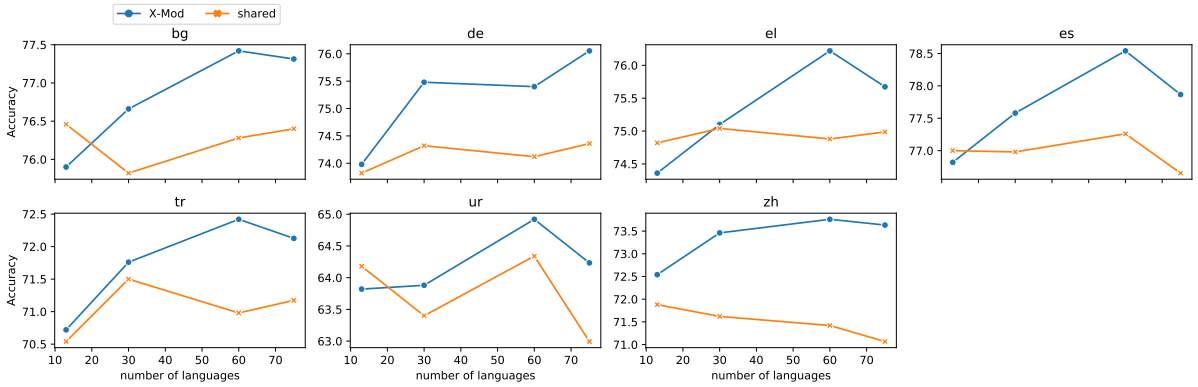


Figure 9: Perplexity when training on more languages. Each model has seen the **same amount of examples** in each language. Lower perplexity indicates better performance.

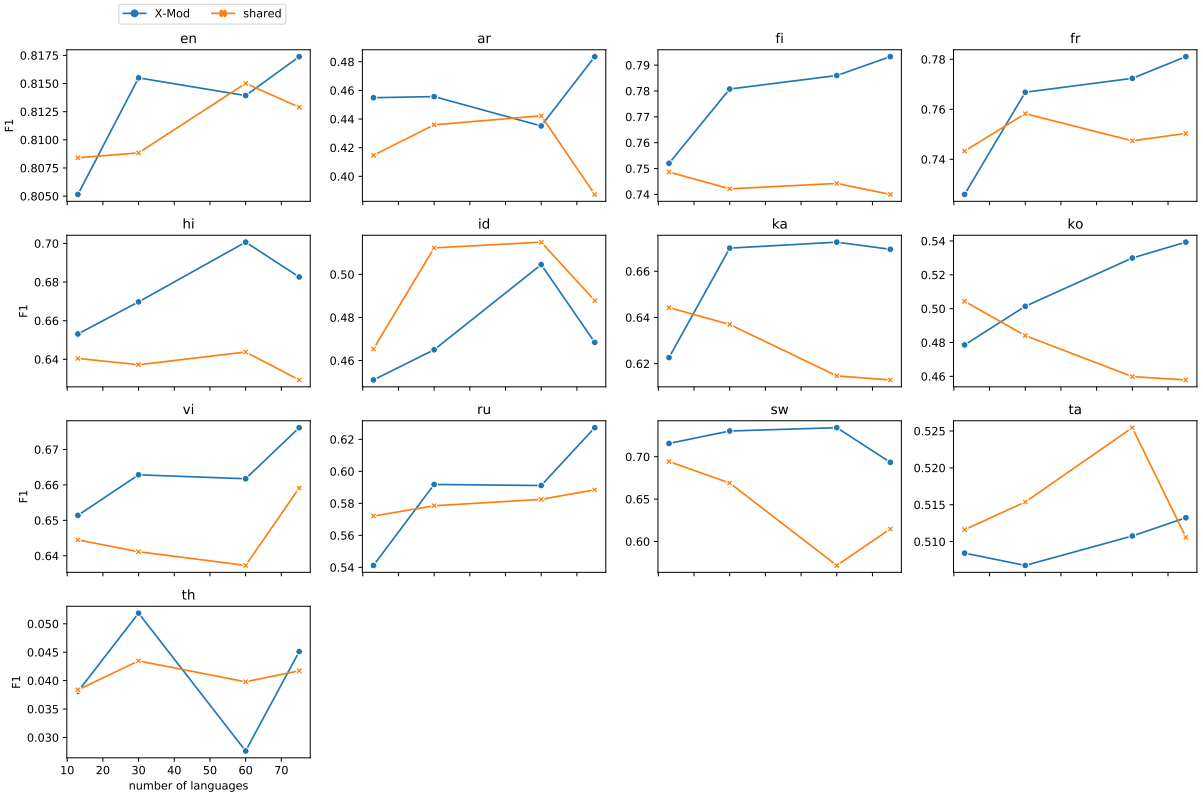


(a) Pre-Trained Languages

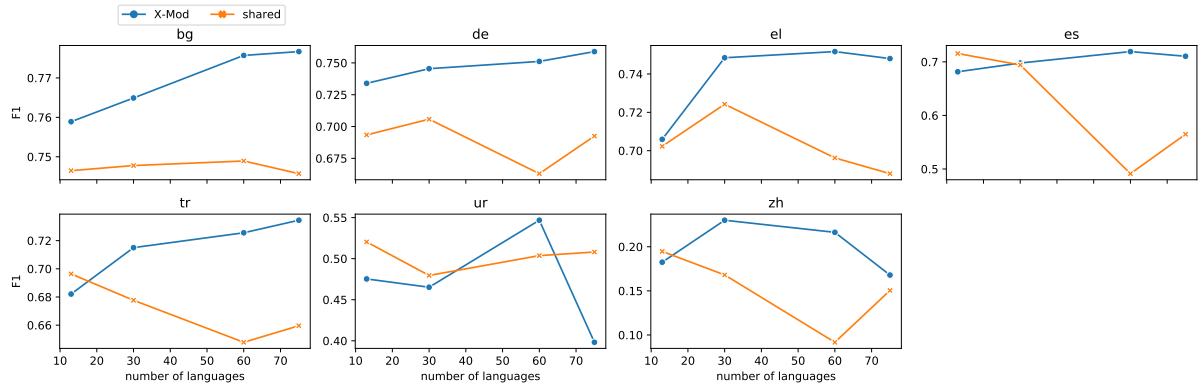


(b) Added Languages

Figure 10: Testset results on **XNLI** of pre-trained (top) and added (bottom) languages trained on different numbers of languages. Models trained on more languages are trained for longer → all models have seen the **same amount of examples** in each individual language. Scores are averaged across all random seeds.



(a) Pre-Trained Languages



(b) Added Languages

Figure 11: Testset results on **NER** of pre-trained (top) and added (bottom) languages trained on different numbers of languages. Models trained on more languages are trained for longer → all models have seen the **same amount of examples** in each individual language. Scores are averaged across all random seeds.

Language	iso	Family	Script	13	30	60	75
Afrikaans	af	IE:Germanic	Latin			✓	✓
Albanian	sq	IE:Albanian	Latin		✓	✓	✓
Amharic	am	Afro-Asiatic	Amharic		✓	✓	
Arabic	ar	Afro-Asiatic	Arabic	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Armenian	hy	IE:Armenian	Armenian		✓	✓	✓
Assamese	as	IE:Iranian	Assamese				✓
Basque	eu	Isolate	Latin		✓	✓	✓
Belarusian	be	IE:Slavic	Cyrillic			✓	✓
Bengali	bn	IE:Iranian	Bengali			✓	✓
Bosnian	bs	IE:Slavic	Latin				✓
Breton	br	IE:Celtic	Latin				✓
Bulgarian	bg	IE:Slavic	Cyrillic	+	+	+	+
Catalan	ca	IE:Romance	Latin			✓	✓
Chinese	zh	Sino-Tibetan	Chinese	+	+	+	+
Croatian	hr	IE:Slavic	Latin		✓	✓	✓
Czech	cs	IE:Slavic	Latin		✓	✓	✓
Danish	da	IE:Germanic	Latin			✓	✓
Dutch	nl	IE:Germanic	Latin			✓	✓
English	en	IE:Germanic	Latin	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Estonian	et	Uralic	Latin			✓	✓
Esperanto	eo	Constructed	Latin			✓	✓
Finnish	fi	Uralic	Latin	✓	✓	✓	✓
French	fr	IE:Romance	Latin	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Frisian	fy	IE:Germanic	Latin				✓
Galician	gl	IE:Romance	Latin			✓	✓
Georgian	ka	Kartvelian	Georgian	✓	✓	✓	✓
German	de	IE:Germanic	Latin	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Greek	el	IE:Hellenic	Greek	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Gujarati	gu	IE:Iranian	Gujarati			✓	✓
Hausa	ha	Afro-Asiatic	Latin			✓	✓
Hebrew	he	Afro-Asiatic	Hebrew	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Hindi	hi	IE:Iranian	Devanagari	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Hungarian	hu	Uralic	Latin		✓	✓	✓
Icelandic	is	IE:Germanic	Latin			✓	✓
Indonesian	id	Austronesian	Latin	✓	✓	✓	✓
Irish	ga	IE:Celtic	Latin			✓	✓
Italian	it	IE:Romance	Latin		✓	✓	✓
Japanese	ja	Japonic	Japanese	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Javanese	jv	Austronesian	Latin				✓
Kannada	kn	Dravidian	Kannada				✓
Korean	ko	Koreanic	Korean	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Kurdish	ku	IE:Iranian	Latin			✓	✓
Latin	la	IE:Romance	Latin			✓	✓

Language	iso	Family	Script	13	30	60	75
Latvian	lv	IE:Slavic	Latin			✓	✓
Lithuanian	lt	IE:Slavic	Latin		✓	✓	✓
Macedonian	mk	IE:Slavic	Cyrillic			✓	✓
Malagasy	mg	Austronesian	Latin				✓
Malay	ms	Austronesian	Latin		✓	✓	✓
Malayalam	ml	Dravidian	Malayalam		✓	✓	✓
Marathi	mr	IE:Iranian	Devanagari				✓
Mongolian	mn	Mongolian	Cyrillic		✓	✓	✓
Nepali	ne	IE:Iranian	Devanagari			✓	✓
Norwegian	no	IE:Germanic	Latin			✓	✓
Oriya	or	IE:Iranian	Odia				✓
Oromo	om	Afro-Asiatic	Ge'ez				✓
Pashto	ps	IE:Iranian	Arabic			✓	✓
Persian	fa	IE:Iranian	Arabic			✓	✓
Polish	pl	IE:Slavic	Latin		✓	✓	✓
Portuguese	pt	IE:Romance	Latin			✓	✓
Punjabi	pa	IE:Iranian	Gurmukhi				✓
Romanian	ro	IE:Romance	Latin		✓	✓	✓
Russian	ru	IE:Slavic	Cyrillic	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Sanskrit	sa	IE:Iranian	Devanagari			✓	✓
Scottish Gaelic	gd	IE:Germanic	Latin				✓
Serbian	sr	IE:Slavic	Cyrillic			✓	✓
Sindhi	sd	IE:Iranian	Arabic			✓	✓
Sinhala	si	IE:Iranian	Sinhala		✓	✓	✓
Slovak	sk	IE:Slavic	Latin		✓	✓	✓
Slovenian	sl	IE:Slavic	Latin			✓	✓
Somali	so	Afro-Asiatic	Latin			✓	✓
Spanish	es	IE:Romance	Latin	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Sundanese	su	Austronesian	Latin				✓
Swahili	sw	Niger-Congo	Latin	✓	✓	✓	✓
Swedish	sv	IE:Germanic	Latin		✓	✓	✓
Tagalog	tl	Austronesian	Latin		✓	✓	✓
Tamil	ta	Dravidian	Tamil	✓	✓	✓	✓
Telugu	te	Dravidian	Telugu			✓	✓
Thai	th	Kra-Dai	Thai	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Turkish	tr	Turkic	Latin	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Ukrainian	uk	IE:Slavic	Cyrillic	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Urdu	ur	IE:Iranian	Arabic	+,(✓)	+,(✓)	+,(✓)	+,(✓)
Vietnamese	vi	Austroasiatic	Latin	✓,(+)	✓,(+)	✓,(+)	✓,(+)
Welsh	cy	IE:Celtic	Latin			✓	✓
Xhosa	xh	Niger-Congo	Latin				✓
Yiddish	yi	IE:Germanic	Hebrew				✓

Table 9: List of languages we pre-train ✓on or add + in the different sets (13, 30, 60, 75). (·) indicates the respectively different pre-training/added languages of models 1 and 2 as described in §5.2 and Table 4. IE stands for Indo-European.

Part III

Epilogue

Chapter 12

Conclusion and Future Work

12.1 Conclusion

While transfer learning has been the dominating paradigm in natural language processing in recent years, with the ever increasing size of pre-trained models and their necessity to perform well in out-of-distribution scenarios, parameter-efficient and modular strategies have since emerged as promising research directions. In this thesis, we have disentangled modularity in natural language processing across four axes: *computation functions, routing functions, aggregation functions*, and the *training setting*. Along those axes we have made multiple contributions.

We have proposed a novel framework (AdapterHub; Chapter 6) which has continuously be extended to incorporate different *computation functions*, such as LoRA (Hu et al., 2022), bottleneck adapters (Houlsby et al., 2019), parallel adapters (He et al., 2022a), Prefix-Tuning (Li and Liang, 2021), inter alia. The framework allows for simple fixed *routing*, based on an input-id to route through designate parts of the model. Different architectures can be seamlessly combined, allowing for seamless *aggregation* such as sequential composition of modular components. The AdapterHub consequently provides a easy-to-use framework for the *training setting* of post-hoc adaptation.

We have proposed a novel *computation function* as well *aggregation function* in Chapter 7. The computation function simplifies previously proposed methods, while the aggregation function is the first which performs post-hoc composition using an attention mechanism.

In Chapters 8 - 10 we propose novel fixed *routing* as well as sequential *aggregation* mechanisms to disentangle language, task, and modality specific information into designated modular components. We demonstrate that by *training* modality, language, and task-specific components while freezing shared weights, we perform better in out-of-distribution scenarios, such as cross-lingual and cross-modal scenarios.

Finally, in Chapter 11 we investigated the impact of modularizing transformer based models during multilingual *pre-training*. While the majority of parameters are shared, we perform fixed *routing* to language-specific modular components. We demonstrate that modular pre-training mitigates catastrophic interference between languages, while preparing the model to be extended to more languages *post-hoc*.

12.2 Future Work

While in recent times modularity has been a hot-topic in research, there remain many interesting open research questions along the axes of modularity introduced in this thesis. We provide an overview of some of these directions for future work in what follows.

In Chapter 2 we introduced different computation functions, where methods have predominant focused either on parameter composition or on function composition. Naturally, a combination of these two orthogonal concepts can be imagined. If sparse subnetworks are trained within parameter-efficient function composition methods, such as bottleneck-adapters, efficient *aggregation* of these modular components can be envisioned. The combination of which would potentially allow for more flexibility and computational efficiency during training as significantly less parameters need to be investigated.

In Chapter 3 we have discussed different routing methods. Here, the dominating strategy to disentangle knowledge into modular parts of the model has been to utilize fixed routing strategies. However, this limits the usability of the proposed methods as they cannot be used on data which lacks the necessary meta data; when training on heterogeneous data, necessary labels, such as domain data, often do not exist. While learned routing methods do not require this meta data to perform routing a-priori, these methods suffer from training difficulties (as discussed in § 3.2). This opens up potential research directions aiming at improving the modularity capabilities of models during pre-training using learned routing mechanisms. Finally, there does not exist a benchmark or even a metric which compares routing mechanisms from a modularity perspective, which the research community desperately needs if this research thread receives more interest in upcoming years.

In Chapter 4 we have introduced different methods which combine the knowledge from multiple modular components. Here, one line of research has focused on performing arithmetic operations on sparse subnetworks. These arithmetic operations are based on an inductive bias placed upon the method by the user, i.e. the notion of subtracting and adding knowledge is strongly biased, especially because these methods are applied across all parameters and not only those which hold the information worth "subtracting". Future research can potentially focus on learning to compose the information within subnetworks directly, in order to improve the aggregation for out of distribution generalization.

Finally, in Chapter 5 we have discussed modular training strategies. Here a recent research thread around model merging has emerged (Wortsman et al., 2022, ;

inter alia). Simultaneously, integrating modular components for multi-task learning has been of interest to researchers for many years. Both methods are a means to cover many tasks within a single model, ideally achieving cross-task improvements, while mitigating performance drops in the individual tasks. These research threads have the potential of being *merged*, i.e. modular multi-task learning models can be envisioned which have specific components designed to be merged later. This potentially allows for an architecture which can be computationally efficiently trained while covering many modalities.

Bibliography

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Chanho Ahn, Eunwoo Kim, and Songhwai Oh. 2019. Deep elastic networks with model selection for multi-task learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6529–6538.

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. 2022. Git rebasin: Merging models modulo permutation symmetries. *CoRR*, abs/2209.04836.

Ferran Alet, Tomas Lozano-Perez, and Leslie P. Kaelbling. 2018. Modular meta-learning. In *Proceedings of The 2nd Conference on Robot Learning*, pages 856–868.

Tariq Alhindi, Jonas Pfeiffer, and Smaranda Muresan. 2019. Fine-tuned neural models for propaganda detection at the sentence and fragment levels. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 98–102, Hong Kong, China. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. 2022. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021a. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavas, Ivan Vulic, and Anna Korhonen. 2021b. MAD-G: multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Re-*

public, 16-20 November, 2021, pages 4762–4781. Association for Computational Linguistics.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Carliss Young Baldwin, Kim B Clark, Kim B Clark, et al. 2000. *Design rules: The power of modularity*. MIT press.

Dana H Ballard. 1986. Cortical connections and parallel processing: Structure and function. *Behavioral and brain sciences*, 9(1):67–90.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548. Association for Computational Linguistics.

Tim Baumgärtner, Kexin Wang, Rachneet Sachdeva, Gregor Geigle, Max Eichler, Clifton Poth, Hannah Sterz, Haritz Puerto, Leonardo F. R. Ribeiro, Jonas Pfeiffer, Nils Reimers, Gözde Şahin, and Iryna Gurevych. 2022. UKP-SQUARE: An online platform for question answering research. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–22, Dublin, Ireland. Association for Computational Linguistics.

Christos Baziotis, Mikel Artetxe, James Cross, and Shruti Bhosale. 2022. Multilingual machine translation with hyper-adapters. *CoRR*, abs/2205.10835.

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. 2022. AdapterHub playground: Simple and flexible few-shot learning with adapters. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–75, Dublin, Ireland. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2016. Conditional Computation in Neural Networks for faster models. In *Proceedings of ICLR 2016 - Workshop track*.

Hakan Bilen and Andrea Vedaldi. 2017. Universal representations: The missing link between faces, text, planktons, and cat breeds. *CoRR*, abs/1701.07275.

Grady Booch, Robert A Maksimchuk, Michael W Engle, Bobbi J Young, Jim Connallen, and Kelli A Houston. 2008. Object-oriented analysis and design with applications. *ACM SIGSOFT software engineering notes*, 33(5):29–29.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901.

Emanuele Bugliarello, Fangyu Liu, Jonas Pfeiffer, Siva Reddy, Desmond Elliott, Edoardo Maria Ponti, and Ivan Vulic. 2022. IGLUE: A benchmark for transfer learning across modalities, tasks, and languages. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2370–2392. PMLR.

Lucas Caccia, Edoardo Ponti, Lucas Liu, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2022. Multi-head adapter routing for data-efficient fine-tuning.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. 2020. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *Proceedings of NeurIPS 2020*.

Ignacio Cases, Clemens Rosenbaum, Matthew Riemer, Atticus Geiger, Tim Klinger, Alex Tamkin, Olivia Li, Sandhini Agarwal, Joshua D. Greene, Dan Jurafsky, Christopher Potts, and Lauri Karttunen. 2019. Recursive routing networks: Learning to compose modules for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3631–3648. Association for Computational Linguistics.

Michael B Chang, Abhishek Gupta, Sergey Levine, and Thomas L Griffiths. 2018. Automatically composing representation transformations as a means for generalization. *arXiv preprint arXiv:1807.04640*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846.

Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. 2019. On Self Modulation for Generative Adversarial Networks. In *Proceedings of ICLR 2019*.

Alexandra Chronopoulou, Matthew Peters, and Jesse Dodge. 2022. Efficient Hierarchical Domain Adaptation for Pretrained Language Models. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1336–1351.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Pa-

ganini, Jordan Hoffmann, Bogdan Damoc, Blake A. Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J. Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc'Aurelio Ranzato, Jack W. Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. 2022. Unified scaling laws for routed language models. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4057–4086. PMLR.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. 2017. Modulating early visual processing by language. *Advances in Neural Information Processing Systems*, 30.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. 2018. Essentially no barriers in neural network energy landscape. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

Dheeru Dua, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan. 2022. Tricks for training sparse translation models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3340–3345, Seattle, United States. Association for Computational Linguistics.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2014. Learning factored representations in a deep mixture of experts. In *2nd International Conference on*

Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *J. Mach. Learn. Res.*, 22:107:1–107:48.

William Fedus, Jeff Dean, and Barret Zoph. 2022. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Jerry A Fodor. 1983. *The modularity of mind*. MIT press.

Negar Foroutan, Mohammadreza Banaei, Rémi Lebret, Antoine Bosselut, and Karl Aberer. 2022. Discovering language-neutral sub-networks in multilingual language models. In *Proceedings of EMNLP 2022*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of ICLR 2019*.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR.

C. Daniel Freeman and Joan Bruna. 2017. Topology and geometry of half-rectified network optimization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling

of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8803–8812, Red Hook, NY, USA. Curran Associates Inc.

Gregor Geigle, Chen Liu, Jonas Pfeiffer, and Iryna Gurevych. 2022a. One does not fit all! on the complementarity of vision encoders for vision and language tasks. *arXiv preprint*.

Gregor Geigle, Jonas Pfeiffer, Nils Reimers, Ivan Vulić, and Iryna Gurevych. 2022b. Retrieve fast, rerank smart: Cooperative and joint approaches for improved cross-modal retrieval. *Transactions of the Association for Computational Linguistics*, 10:503–521.

Gregor Geigle, Jonas Stadtmüller, Wei Zhao, Jonas Pfeiffer, and Steffen Eger. 2021. TUDa at WMT21: Sentence-level direct assessment with adapters. In *Proceedings of the Sixth Conference on Machine Translation*, pages 911–919, Online. Association for Computational Linguistics.

Andrea Gesmundo and Jeff Dean. 2022a. An evolutionary approach to dynamic introduction of tasks in large-scale multitask learning systems. *CoRR*, abs/2205.12755.

Andrea Gesmundo and Jeff Dean. 2022b. munet: Evolving pretrained deep neural networks into scalable auto-tuning multitask systems. *CoRR*, abs/2205.10937.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. 2021. Recurrent independent mechanisms. In *International Conference on Learning Representations*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *CoRR*, abs/2204.07689.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah Smith, and Luke Zettlemoyer. 2022. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5557–5576. Association for Computational Linguistics.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meet-*

ing of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4921–4933, Online. Association for Computational Linguistics.

John B. Hampshire and Alex Waibel. 1992. The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(07):751–769.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. 2016. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Towards a unified view of parameter-efficient transfer learning. In *10th International Conference on Learning Representations, ICLR 2022, Virtual Conference, April 25 - 29, 2022*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer.

Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. 2022b. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, pages 8678–8690. PMLR.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of ICLR 2022*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *preprint*.

Robert A Jacobs, Michael I Jordan, and Andrew G Barto. 1991a. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive science*, 15(2):219–250.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991b. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Michael I. Jordan and Robert A. Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.*, 6(2):181–214.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 72–83, Florence, Italy. Association for Computational Linguistics.

Simran Khanuja and Melvin Johnson. 2021. MergeDistill: Merging Pre-trained Language Models using Distillation. In *Findings of ACL 2021*, pages 2874–2887.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

Louis Kirsch, Julius Kunze, and David Barber. 2018. Modular networks: Learning to decompose neural computation. *Advances in Neural Information Processing Systems*, 31.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3577–3599.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Quoc Le, Tamás Sarlós, Alex Smola, et al. 2013. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, volume 85, page 8.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of EMNLP 2021*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the Intrinsic Dimension of Objective Landscapes. In *Proceedings of ICLR 2018*.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *CoRR*, abs/2208.03306.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning language specific sub-network for multilingual machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 293–305, Online. Association for Computational Linguistics.

Chen Liu, Jonas Pfeiffer, Anna Korhonen, Ivan Vulić, and Iryna Gurevych. 2022a. Delving deeper into cross-lingual visual question answering. *Under Review at ARR March*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022b. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *CoRR*, abs/2205.05638.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022c. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association*

for Computational Linguistics (Volume 2: Short Papers), pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1930–1939. ACM.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient multi-task fine-tuning for Transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 565–576.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82.

Arun Mallya and Svetlana Lazebnik. 2018. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

Franco Manessi, Alessandro Rozza, Simone Bianco, Paolo Napoletano, and Raimondo Schettini. 2018. Automated pruning for deep neural network compression. In *2018 24th International conference on pattern recognition (ICPR)*, pages 657–664. IEEE.

Michael Matena and Colin Raffel. 2022. Merging Models with Fisher-Weighted Averaging. In *Proceedings of NeurIPS 2022*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Rahul S. Mehta. 2019. Sparse Transfer Learning via Winning Lottery Tickets. In *Workshop on Learning Transferable Skills (NeurIPS 2019)*.

Elliot Meyerson and Risto Miikkulainen. 2018. Beyond Shared Hierarchies: Deep Multitask Learning through Soft Layer Ordering. In *Proceedings of ICLR 2018*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning Convolutional Neural Networks. In *Proceedings of ICLR 2017*.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2022. Models with Conditional Computation Learn Suboptimal Solutions. *arXiv preprint*.

Vaishnavh Nagarajan and J. Zico Kolter. 2019. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11611–11622.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Johannes Von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. 2020. Continual learning with hypernetworks. In *Proceedings of ICLR 2020*.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. 2018. Learning independent causal mechanisms. In *International Conference on Machine Learning*, pages 4036–4044.

Marinela Parovic, Goran Glavas, Ivan Vulic, and Anna Korhonen. 2022. BAD-X: bilingual adapters improve zero-shot cross-lingual transfer. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1791–1799. Association for Computational Linguistics.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. FiLM: Visual reasoning with a general conditioning layer. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3942–3951.

Jonas Pfeiffer, Gregor Geigle, Aishwarya Kamath, Jan-Martin Steitz, Stefan Roth, Ivan Vulić, and Iryna Gurevych. 2022a. xGQA: Cross-lingual visual question answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2497–2511, Dublin, Ireland. Association for Computational Linguistics.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022b. Lifting the curse of multilinguality by pre-training modu-

lar transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Jonas Pfeiffer, Naman Goyal, Xi Victoria Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022c. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3479–3495. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Iryna Gurevych, and Sebastian Ruder. 2019a. What do Deep Networks Like to Read? *arXiv preprint*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021b. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 487–503.

Jonas Pfeiffer, Christian M. Meyer, Claudia Schulz, Jan Kiesewetter, Jan Zottmann, Michael Sailer, Elisabeth Bauer, Frank Fischer, Martin R. Fischer, and Iryna Gurevych. 2019b. FAMULUS: Interactive annotation and feedback generation for teaching diagnostic reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 73–78, Hong Kong, China. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Edwin Simpson, and Iryna Gurevych. 2019c. Low Resource Multi-Task Sequence Tagging - Revisiting Dynamic Conditional Random Fields. *arXiv preprint*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021c. UNKs

everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2021d. Unks everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10186–10203. Association for Computational Linguistics.

Jonathan Pilault, Amine El hattami, and Christopher Pal. 2021. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. In *International Conference on Learning Representations*.

Edoardo Ponti. 2021. *Inductive Bias and Modular Design for Sample-Efficient Neural Language Learning*. Ph.D. thesis, University of Cambridge.

Edoardo M. Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2022. Combining modular skills in multitask learning.

Edoardo M. Ponti, Ivan Vulić, Ryan Cotterell, Marinela Parovic, Roi Reichart, and Anna Korhonen. 2021. Parameter space factorization for zero-shot learning across tasks and languages. *Transactions of the Association for Computational Linguistics*, 9:410–428.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to pre-train on? Efficient intermediate task selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10585–10605, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*.

Janarthanan Rajendran, P Prasanna, Balaraman Ravindran, and Mitesh M Khapra. 2017. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. In *International Conference on Learning Representations*.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127. Computer Vision Foundation / IEEE Computer Society.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021. Smelting gold and silver for improved multilingual AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 742–750, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17555–17566.

Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. 2019. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7930–7946. Association for Computational Linguistics.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020. MultiCQA: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2486, Online. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual

language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *arXiv*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2339–2352. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Edwin Simpson, Jonas Pfeiffer, and Iryna Gurevych. 2020. Low Resource Sequence Tagging with Weak Labels. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8862–8869.

Karolina Stanczak, Edoardo Ponti, Lucas Torroba Hennigen, Ryan Cotterell, and Isabelle Augenstein. 2022. Same neurons, different languages: Probing morphosyntax in multilingual pre-trained models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1589–1598, Seattle, United States. Association for Computational Linguistics.

Jan-Martin O. Steitz, Jonas Pfeiffer, Iryna Gurevych, and Stefan Roth. 2021. Txt: Crossmodal end-to-end learning with transformers. In *Pattern Recognition - 43rd DAGM German Conference, DAGM GCPR 2021, Bonn, Germany, September 28 - October 1, 2021, Proceedings*, volume 13024 of *Lecture Notes in Computer Science*, pages 405–420. Springer.

Asa Cooper Stickland, Alexandre Berard, and Vassilina Nikoulina. 2021. Multilin-

gual domain adaptation for NMT: decoupling language and domain information with adapters. In *Proceedings of the Sixth Conference on Machine Translation, WMT@EMNLP 2021, Online Event, November 10-11, 2021*, pages 578–598. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. 2019. Many task learning with task routing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1375–1384.

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. Learning sparse sharing architectures for multiple tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8936–8943.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2020b. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740.

Richard Sutton. 1986. Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 823–832.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language Adaptation for Truly Universal Dependency Parsing. In *Proceedings of EMNLP 2020*.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, Gertjan van Noord, and Sebastian Ruder. 2022. Hyper-X: A Unified Hypernetwork for Multi-Task Multilingual Transfer. In *Proceedings of EMNLP 2022*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS 2017*.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. In *Proceedings of ACL 2022*.

Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *CoRR*, abs/2205.12410.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2021. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*.

RJ Williams. 1988. Toward a theory of reinforcement-learning connectionist systems. *Technical Report NU-CCS-88-3, Northeastern University*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. 2020. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184.

Teng Xi, Yifan Sun, Deli Yu, Bi Li, Nan Peng, Gang Zhang, Xinyu Zhang, Zhigang Wang, Jinwen Chen, Jian Wang, Lufei Liu, Haocheng Feng, Junyu Han, Jingtuo Liu, Errui Ding, and Jingdong Wang. 2022. UFO: unified feature optimization. *CoRR*, abs/2207.10341.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. 2021. M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082*.

Zonghan Yang and Yang Liu. 2022. On robust prefix-tuning for text classification. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Zhao You, Shulin Feng, Dan Su, and Dong Yu. 2022. Speechmoe2: Mixture-of-experts model with improved routing. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7217–7221. IEEE.

Fan Zhang, Duyu Tang, Yong Dai, Cong Zhou, Shuangzhi Wu, and Shuming Shi. 2022a. SkillNet-NLU: A Sparsely Activated Model for General-Purpose Natural Language Understanding. *arXiv*.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2022b. Emergent modularity in pre-trained transformers.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew M. Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. *CoRR*, abs/2202.09368.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Jianfeng Gao, and Tuo Zhao. 2022. Taming sparsely activated transformer with stochastic experts. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Appendix A

Data Handling

In accordance with DFG's "Principles for the Handling of Research Data",[1] we ensured the long-term preservation of research data and/or experimental software that has been developed as part of this dissertation. We made this data openly accessible when possible. The following software has been made available for the scientific community (see the repositories for licensing details):

- Chapter 6: https://github.com/Adapter-Hub
- Chapter 7: https://github.com/Adapter-Hub/adapter-transformers
- Chapter 8: https://github.com/Adapter-Hub/adapter-transformers
- Chapter 9: https://github.com/Adapter-Hub/adapter-transformers
- Chapter 10: https://github.com/Adapter-Hub/xGQA
- Chapter 11: https://github.com/facebookresearch/fairseq

All publications related to this thesis are publicly available on the ACL Anthology (aclweb.org/anthology/):

- Chapter 6: https://aclanthology.org/2020.emnlp-demos.7/
- Chapter 7: https://aclanthology.org/2021.eacl-main.39/
- Chapter 8: https://aclanthology.org/2020.emnlp-main.617/
- Chapter 9: https://aclanthology.org/2021.emnlp-main.800/
- Chapter 10: https://aclanthology.org/2022.findings-acl.196/
- Chapter 11: https://aclanthology.org/2022.naacl-main.255/

[1] https://www.dfg.de/download/pdf/foerderung/grundlagen_dfg_foerderung/forschungsdaten/leitlinien_forschungsdaten.pdf

Moreover, all research results of the aforementioned publications are documented in the present thesis, which is archived by the Universitäts- und Landesbibliothek Darmstadt.