THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e. g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Learning Universal Representations Across Tasks and Domains

Wei-Hong Li

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2022

Abstract

A longstanding goal in computer vision research is to produce broad and general-purpose systems that work well on a broad range of vision problems and are capable of learning concepts only from few labelled samples. In contrast, existing models are limited to work only in specific tasks or domains (datasets), *e.g*., a semantic segmentation model for indoor images (Silberman et al., 2012). In addition, they are data inefficient and require large labelled dataset for each task or domain. While there has been works proposed for domain/task-agnostic representations by either loss balancing strategies or architecture design, it remains a challenging problem on optimizing such universal representation network. This thesis focuses on addressing the challenges of learning universal representations that generalize well over multiple tasks (*e.g*. segmentation, depth estimation) or various visual domains (*e.g*. image object classification, image action classification). In addition, the thesis also shows that these representations can be learned from partial supervision and transferred and adopted to previously unseen tasks/domains in a data-efficient manner.

The first part of the dissertation focuses on learning *universal representations*, *i.e*. a single universal network for multi-task learning (*e.g*., learning a single network jointly for different dense prediction tasks like segmentation and depth estimation) and multi-domain learning (*e.g*. image classification for various vision datasets, each collected for a different problem like texture, flower or action classification). Learning such universal representations by jointly minimizing the sum of all task-specific losses is challenging because of the interference between tasks and it leads to unbalanced results (*i.e*. some tasks dominate or interfere other tasks and the universal network performs worse than task/domain-specific networks each of which is trained for a task/domain independently). Hence a new solution is proposed to regularize the optimization of the universal network by encouraging the universal network to produce the same features as the ones of task-specific networks. The experimental results demonstrate that the proposed method learns a single universal network that performs well for multiple tasks or various visual domains.

Despite the recent advances in multi-task learning of dense prediction problems, most methods rely on expensive labelled datasets. Relaxing this assumption gives rise to a new multi-task learning setting, called *multi-task partially-supervised learning* in this thesis, in which the goal is to jointly learn of multiple dense prediction tasks on partially annotated data (*i.e*. not all the task labels are available for each training image). In the thesis, a label efficient approach is proposed to successfully leverage

task relations to supervise its multi-task learning when data is partially annotated. In particular, the proposed method learns to map each task pair to a *joint pairwise task-space* which enables sharing information between them in a computationally efficient way through another network conditioned on task pairs, and avoids learning trivial cross-task relations by retaining high-level information about the input image.

The final part of the dissertation studies the problem of adapting a model to previously unseen tasks (from seen or unseen domains) with very few labelled training samples of the new tasks, *i.e*. cross-domain few-shot learning. Recent methods have focused on using various adaptation strategies for aligning their visual representations to new domains or selecting the relevant ones from multiple domain-specific feature extractors. In this dissertation, new methods are formulated to learn a single task-agnostic network from multiple domains during meta-training and attach light-weight task-specific parameters that are learned from limited training samples and adapt the task-agnostic network to accommodate the previously unseen tasks. Systematic analysis is performed to study various task adaptation strategies for few-shot learning. Extensive experimental evidence demonstrates that the proposed methods that learn a single set of task-agnostic representations and adapt the representations via residual adapters in matrix form attached to the task-agnostic model significantly benefits the cross-domain few-shot learning.

Acknowledgements

I feel grateful and very fortunate for studying and living in Edinburgh. I would like to thank everyone who has been with me and offered help to me during this unforgettable journey.

I would like to thank my supervisor Hakan Bilen for including me in the Visual Computing (VICO) Group at the University of Edinburgh, and for his enthusiasm and support throughout this thesis. Special thanks also to Timothy Hospedales and Iain Murray for their co-supervision. Thanks also to Amir Zamir and Laura Sevilla for finding time to evaluate my thesis and viva, and for all of the insightful comments. I am very hornored to have such distinguished researchers as my examiners.

I am grateful to my collaborators, Xialei Liu and Chuan-Sheng Foo for their contagious enthusiasm for universal representation learning and semi-supervised learning and dedicated hard work, and to friends in our group, office, flat, university and elsewhere who made the experience fun and memorable. I would like to thank several people here: Titas Anciukevicius, Lucas Deecke, Boyan Gao, Arushi Goel, Adrian Salazar Gomez, Shangmin Guo, Wenbin Hu, Vitor Ivanov, Zonglin Ji, Ruochun Jin, Taha Kocyigit, Changjiang Liu, Muyang Liu, Konda Reddy Mopuri, Octave Mariotti, Kunkun Pang, Simon Reinkemeier, Yinbing Tian, Robin Vogel, Yuan Wen, Yu Yang, Biao Zhang, Xueting Zhang, Ying Zhang, Bo Zhao, Yanpeng Zhao, Hao Zheng. Special thanks to Yuedong Chen and Jiabo Huang. We were looking for PhD opportunities together and supporting each other during the PhD study.

Lastly, I want to deeply thank my family for their unconditional love and support throughout my studies, especially when I was working remotely from home.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Wei-Hong Li*)

Table of Contents

Chapter 1

Introduction

Over the last decade, computer vision has advanced rapidly and achieved impressive results for many tasks like image classification for general object categories (Russakovsky et al., 2015), fungi categories (Brigit and Yin, 2018), or birds (Wah et al., 2011) so on, semantic segmentation (Long et al., 2015), depth estimation (Eigen et al., 2014), surface normal estimation (Wang et al., 2015) and so on. While the best performance in each task is achieved by designing and learning a single network per task in a dataset independently, these models are specialized to a dataset and heavily rely on large-scaled labelled training data.

An important goal in computer vision is to learn broad and general-purpose models that perform well on a wide range of vision problems and are capable of learning for new problems only from limited supervision. Datasets collected to achieve this goal contain either data from a single vision domain (*e.g*. indoor images (Silberman et al., 2012)) for multiple vision tasks (*e.g*. semantic segmentation (Long et al., 2015), depth estimation (Eigen et al., 2014)) or multiple sub-datasets (Rebuffi et al., 2017a) each sampled from a visual domain for a task (*e.g*. object classification (Russakovsky et al., 2015), action classification (Soomro et al., 2012)).

Learning universal representations, a single deep network, for jointly performing multiple vision tasks in a single domain (multi-task learning, MTL Fig. 1.1) (Caruana, 1997; Vandenhende et al., 2021) or over various visual domains (multi-domain learning, MDL Fig. 1.2) (Rebuffi et al., 2017a), is able to efficiently share features and computations across tasks and domains. This is computationally efficient and crucial in platforms with limited resources like mobile devices and autonomous vehicles and enables the network to learn more complete representations. This thesis focuses on learning universal representations that generalize well over multiple vision tasks and

various visual domains. A new method is formulated to learn these representations and show that they can be learned from partial supervision. As more complete universal representations are obtained, learning of new domains and tasks can be easier and performed efficiently from only a few samples by task adaptation.

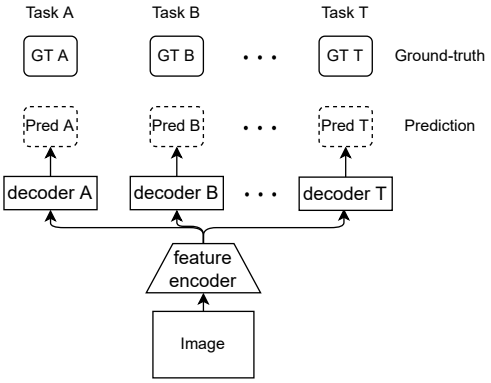


Figure 1.1: Multi-task Learning.

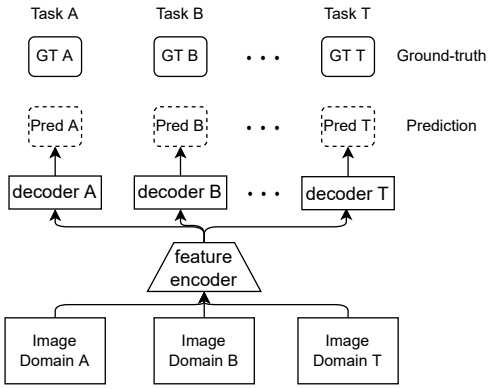


Figure 1.2: Multi-domain Learning.

Despite the high efficiency of compact universal representations, learning multiple problems simultaneously involves minimizing a weighted sum of multiple loss functions with different magnitudes and characteristics and thus results in unbalanced state of one loss dominating the optimization and poor results compared to learning a separate model for each problem (Kendall et al., 2018; Gong et al., 2019; Chen et al., 2018b). As highlighted in Sec. 2.2, existing methods propose to address this challenge by either designing better network architectures (*e.g*. Misra et al. (2016); Liu et al. (2019); Vandenhende et al. (2020a,b)) or optimization strategies by loss weighting (*e.g*. Poggi et al. (2020); Guo et al. (2018)) or gradient modification (*e.g*. Chen et al. (2018b); Yu et al. (2020); Liu et al. (2021a)).

Chapter 2 instead tackles the challenge from a different perspective and proposes a new learning formulation that distills knowledge of multiple task/domain-specific networks into a single deep neural network after aligning its representations with the task/domain-specific ones through small capacity adapters (Sec. 2.3), namely Universal Representation Learning (URL). Extensive experimental results in Sec. 2.5 demonstrate that universal representations learned by the proposed method achieve state-of-the-art performances in learning of multiple dense prediction problems in NYU-v2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) and multiple image classification problems from diverse domains in Visual Decathlon Dataset (Rebuffi et al., 2017a).

Most recent methods in learning universal representations require all training data to be labelled for all tasks, which is costly and impractical. In real-world scenarios,

curating fully annotated dataset for multiple dense prediction tasks often involves using multiple sensors to collect annotations for several tasks and requires very accurate synchronization between sensors. Hence synchronization errors result in partially annotated data and algorithms that can learn from such data are needed. To this end, Chapter 3 proposes a more realistic and general setting for multi-task dense prediction problems where not all the task labels are available for each image and called multi-task partially supervised learning (MTPSL). In particular, this chapter assumes that each image is at least labelled for one task and each task at least has few labelled images and aims to learn a multi-task learning model from them.

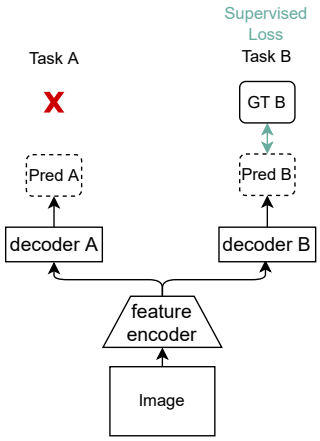


Figure 1.3: Supervised Learning.

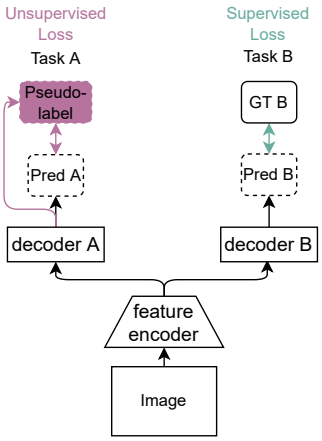


Figure 1.4: Semi-supervised Learning.

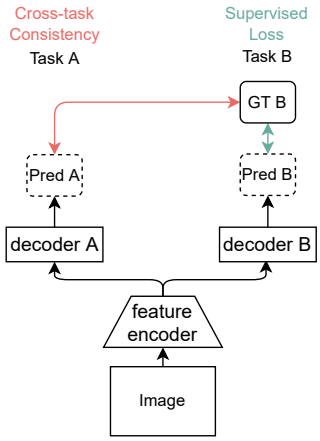


Figure 1.5: Cross-task Consistency Learning (Ours).

While one can train the MTL model only on available labels Fig. 1.3 or extend existing single-task semi-supervised learning methods to MTL Fig. 1.4, the former can not extract task-specific information from the images for the unlabelled tasks and the latter does not guarantee consistency across the related tasks. To this end, in Chapter 3, a multi-task training procedure that successfully leverages task relations Fig. 1.5 to provide supervision signal for unlabelled images is formulated (Sec. 3.3). In particular, Chapter 3 proposes to learn to map each task pair to a joint pairwise task-space which enables sharing information between them in a computationally efficient way through another network conditioned on task pairs, and avoids learning trivial cross-task relations by retaining high-level information about the input image. Sec. 3.4 demonstrates that the proposed method effectively exploits the images with unlabelled tasks and outperforms existing semi-supervised learning approaches and related methods on three standard benchmarks.

An advantage of universal representations is their ability to generalize to unseen domains from few samples only. Hence we look at the few-shot classification problem (Lake et al., 2011; Miller et al., 2000) that aims at learning a model that can be efficiently adapted to recognize unseen classes from few samples. A typical strategy for few-shot learning involves two steps: learning a task-agnostic network (universal representations) from a large training set (*i.e.* meta-training) and adapting this task-agnostic network to learn new classes from a given support set and the adapted model is then evaluated on a query set where the support set and query set are sampled form a testing set.

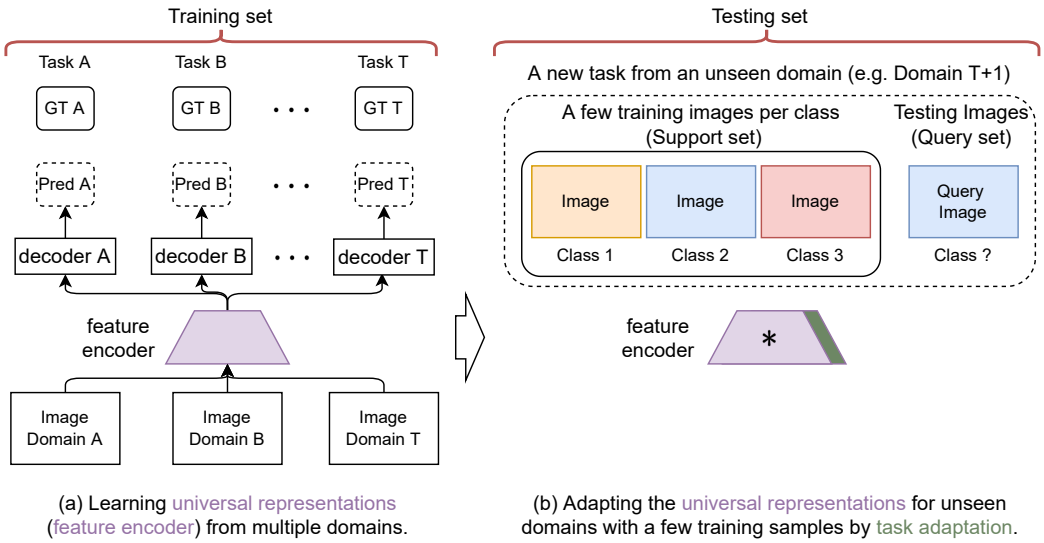


Figure 1.6: Cross-domain Few-shot Learning

Chapter 4 focuses on cross-domain few-shot learning, where test data is sampled from an unknown or previously unseen domain (Triantafillou et al., 2020). To learn universal representations (task-agnostic network), previous methods use various adaptation strategies for aligning their visual representations to new domains Requeima et al. (2019); Bateni et al. (2020b,a) or select the relevant ones from multiple domain-specific feature extractors (Dvornik et al., 2020; Liu et al., 2021b; Triantafillou et al., 2021; Liu et al., 2021c). Chapter 4 shows that the URL method proposed in Chapter 2 can be applied to this problem and learns a single task-agnostic (universal) network from multiple domains Fig. 1.6(a). Universal representations can still benefit from small modifications, *i.e.* task-specific adaptations, and Chapter 4 propose to attach light-weight adapters into the task-agnostic network and learn them from the support set Fig. 1.6(b). Systematic analysis is also performed to study various task adaptation strategies for cross-domain few-shot learning. Extensive experimental results in Sec. 4.4

shows that the URL method learns well generalized features for previously unseen tasks and requires less and fixed computational cost during inference time regardless of the number of domains. Adapting the representations via residual adapters in matrix form attached to the task-agnostic model significantly benefits learning new domains and tasks with few samples.

To conclude the thesis, Chapter 5 summarizes the broader impact of the proposed methods, discusses their limitations, and proposes future research directions for universal representations learning across tasks and domains.

1.1 Contributions

The main contributions of this thesis are as following:

- Universal Representation Learning (URL, Sec. 2.4): an optimization algorithm for learning compact universal representations over multiple vision tasks and various visual domains by distilling knowledge of multiple task/domain-specific networks into a single deep neural network after aligning its representations with the task/domain-specific ones through small capacity adapters. As learning universal representations over visually diverse domains is more challenging than learning multiple tasks of a single domain, a loss function based on CKA (Kornblith et al., 2019) similarity is proposed for aligning features. It is demonstrated that URL is able to learn universal representations for both multi-task learning and multi-domain learning and it achieves better or comparable results compared with task/domain-specific models while URL only requires one network for multiple tasks and domains. It outperforms vanilla MTL and MDL methods and related optimization strategies in several benchmarks, including multi-task dense prediction and multi-domain classification.

- Multi-task Partially Supervised Learning (MTPSL, Sec. 3.3.1): a more practical and general setting for multi-task learning, which aims at learning multi-task learning models on partially annotated data where not all the tasks labels are available for each image.

- Regularized Conditional Cross-task Joint Space Mapping (Sec. 3.3.2): a new algorithm for leveraging cross-task relations between related tasks to learn MTL model from partially annotated data. The proposed method is shown to efficiently

learn from partially annotated data and outperforms related baselines in three multi-task learning benchmarks for dense prediction problems.

- Task-specific adapters (TSA Sec. 4.3.2) for cross-domain few-shot learning. Adapting a model for previously unseen domains and tasks from very few samples is challenging due to limited ability of learning deep models from few samples and the large generalization domain gap between training and testing samples. TSA is proposed to attach light-weight task-specific adapters to a frozen task-agnostic model obtained during meta-training and learn the adapters from the support set during meta-testing. Systematic study shows that using residual adapters in matrix form with a pre-classifier alignment obtains the best performance and significantly boost the performance of existing methods in the leaderboard [1] of a recent challenging MetaDataset (Triantafillou et al., 2020).

1.2 Thesis Structure

The thesis is divided into five chapters. Chapter 1 contains the introduction, followed by Chapter 2, which proposes a new universal representation learning algorithm that generalizes over multiple vision tasks and over various visual domains. Chapter 3 focuses on multi-task partially supervised learning and presents a new method that efficiently leverages the cross-task relations between related tasks to learn the MTL model from partially annotated data. Chapter 4 focuses on cross-domain few-shot learning. Chapter 4 demonstrates that the URL method proposed in Chapter 2 learns well generalized universal features from multiple domains and proposes to efficiently adapt the universal features with task-specific adapters for previously unseen domains and tasks. Chapter 5 introduces ideas for future research and discusses the broader impact of the methods proposed in this work.

The technical content in this thesis (Chapters 2 to 4) is based on peer-reviewed papers and a journal submission which is currently under review. The publications associated with the individual chapters are as follows:

- Chapter 2 is based on "Universal Representations: A Unified Look at Multiple Task and Domain Learning", W.H. Li, X. Liu, H. Bilen, *arXiv preprint*

[1] `https://github.com/google-research/meta-dataset#leaderboard-in-progress` (accessed on July 13th, 2022)

arXiv:2204.02744 (2022) and "Knowledge distillation for multi-task learning", W.H. Li, H. Bilen, *European Conference on Computer Vision Workshop* (2020).

- Chapter 3 is based on "Learning Multiple Dense Prediction Tasks from Partially Annotated Data", W.H. Li, X. Liu, H. Bilen, *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022).

- Chapter 4 is based on "Universal representation learning from multiple domains for few-shot classification", W.H. Li, X. Liu, H. Bilen, *IEEE/CVF International Conference on Computer Vision* (2021) and "Cross-domain Few-shot Learning with Task-specific Adapters", W.H. Li, X. Liu, H. Bilen, *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022).

Chapter 2

Learning Universal Representations

Learning universal representations through a single network to jointly perform multiple tasks or generalize over visually diverse domains remains a challenging problem. Existing methods attempt to address this challenge by designing network architectures that better share parameters across tasks or domains and keep irrelevant ones task-specific or balancing optimization techniques by either losses weighting or gradients updating schemes. In this chapter, we focus on addressing the optimization challenges of learning universal representations in a new direction. This chapter begins with an introduction of the background of universal representations learning and the key idea of the proposed method for both multi-task dense prediction and multi-domain classification in Sec. 2.1, reviewing related literatures in Sec. 2.2, elaborating the proposed URL method by knowledge distillation in Sec. 2.3, evaluating and analyzing the proposed method in multi-task dense prediction and multi-domain classification problems in Sec. 2.5, concluding in Sec. 2.6.

2.1 Introduction

A major limitation of state-of-the-art image interpretation systems is their narrow scope. Different models are learned to recognize faces (Taigman et al., 2014; Parkhi et al., 2015; Schroff et al., 2015; Zhong et al., 2016), textures (Cimpoi et al., 2014), sketches (Eitz et al., 2012) and drawings (Lake et al., 2015), various fine-grained flower (Nilsback and Zisserman, 2008), bird (Wah et al., 2011), fungi categories (Brigit and Yin, 2018), detect (Ren et al., 2015; Liu et al., 2016) and segment (Dai et al., 2016) object categories, and perform various low and mid-level tasks such as depth (Eigen et al., 2014), surface normal estimation (Wang et al., 2015), so on.

In contrast, humans in the early years of their development develop powerful internal visual representations that are subject to small refinements in response to later visual experience (Atkinson, 2002; Maurer and Lewis, 2001; Lewis and Maurer, 2005). Once these visual representations are formed, they are *universal* and later employed in many diverse vision tasks from reading text, recognizing faces to interpreting visual art forms.

Presence of universal representations in computer vision (Bilen and Vedaldi, 2017), has important implications. First, it means that vision has limited complexity. A growing number of visual domains and tasks[1] can be modeled with a bounded number of representations. As a result, one can use a compact set of representations for learning multiple domains and tasks, and efficiently share features and computations across them, which is crucial in platforms with limited computational resources such as mobile devices and autonomous cars. Second, as we obtain more complete universal representations, learning of new domains and tasks can be easier and performed efficiently from only few samples by transfer learning.

In practice, learning universal representations requires addressing several challenges. First, modelling diverse visual data demands deep network architectures that can simultaneously learn representations while selectively sharing only the relevant representations across multiple tasks and domains.

To this end, previous multi-task works proposed controlling representation sharing across tasks through latent connections (Misra et al., 2016; Ruder et al., 2019), constructing branched deep neural networks based on task affinities (Vandenhende et al., 2020a), custom attention mechanisms (Liu et al., 2019), neural architecture search (Liang et al., 2018; Bruggemann et al., 2020; Guo et al., 2020), developing progressive communication across multiple tasks through recurrent networks (Bilen and Vedaldi, 2016; Zhang et al., 2018; Vandenhende et al., 2020b), multi-scale feature sharing (Xu et al., 2018a; Vandenhende et al., 2020b). Other previous works assume that features extracted from pretrained deep networks on ImageNet provide basis for universal representations and adapt them with a set of compact adapters to various domains (Rebuffi et al., 2017a, 2018; Rosenfeld and Tsotsos, 2018; Deecke et al., 2022). In cross-domain few-shot classification, where the goal is to generalize to unseen tasks and domains from few samples, features from multiple domain-specific networks are considered as universal

[1]While domain and and task definitions vary in previous work and are used interchangeably, in our experiments each domain denotes a data domain, dataset such as ImageNet, Omniglot and each task denotes either different prediction task such as semantic segmentation and depth estimation, and also same prediction task such as image classification, albeit, over different sets of categories. A subtle difference between two settings is that for a single image there can be multiple tasks defined, however, only a single domain.

features and transferred to previously unseen domains and tasks after a post selection step (Dvornik et al., 2020; Liu et al., 2021b).

The second challenge is to develop training algorithms to learn representations that achieve good performance not only in one of the tasks or domains but in all of them. This problem is especially visible when the training involves jointly minimizing a set of loss functions (*i.e.* one for each task) with significantly different difficulty levels, magnitudes, and characteristics. Thus a naive strategy of uniformly weighing multiple losses can lead to sub-optimal performances and searching for optimal weights in a continuous hyperparameter space can be prohibitively expensive. Previous works (Chen et al., 2018b; Sener and Koltun, 2018; Kendall et al., 2018; Guo et al., 2018; Liu et al., 2019) address the *unbalanced loss optimization* problem by weighing loss functions based on the task-dependent uncertainty of the model at training time (Kendall et al., 2018), proposing Pareto optimal solution (Sener and Koltun, 2018), eliminating conflicting gradient components between the tasks (Yu et al., 2020). Although these methods are shown to improve over the uniform weighing loss strategy in some benchmarks, they do not consistently outperform the baseline that simply weighs each loss function with a constant scalar (Vandenhende et al., 2021).

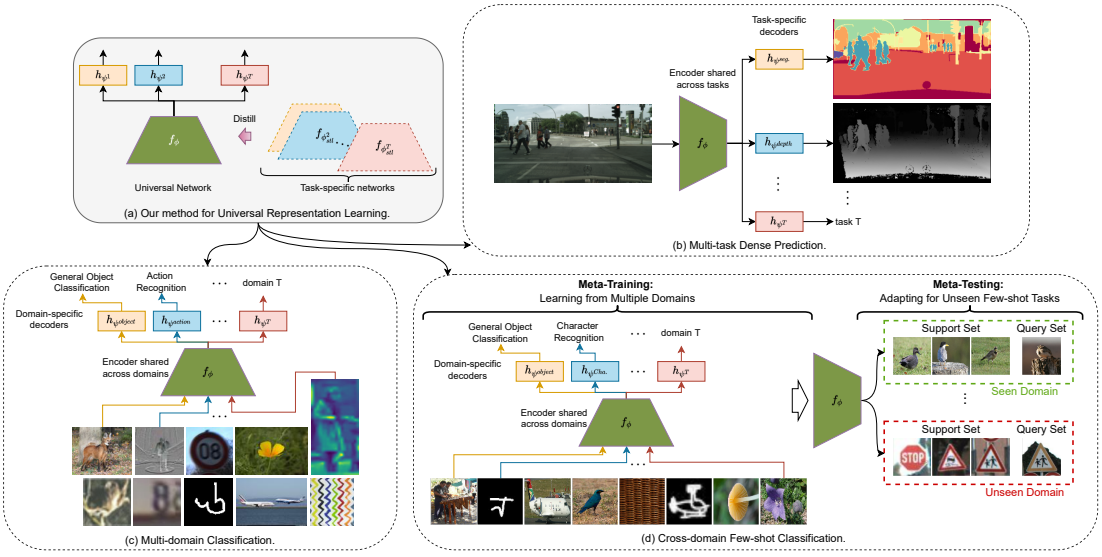


Figure 2.1: We propose a Universal Representation Learning framework in (a) that generalizes over multi-task dense prediction tasks (b), multi-domain many-shot learning (c), cross-domain few-shot learning (d) (validated and detailed in Chapter 4).

In this chapter, we focus on the second challenge. Inspired from knowledge distillation (Romero et al., 2015; Hinton et al., 2014), we approach the problem from a different perspective and propose a general methodology for universal representation learning

that can be applied to a diverse set of problems including multi-task and multi-domain learning in few- and many-shot settings. We propose a two-stage procedure for universal representation learning where we first train a set of task or domain-specific models and freeze their parameters, and then distill their knowledge to a universal representation network while simultaneously training it over multiple-tasks/domains. In contrast to the standard knowledge distillation, in our setting each "teacher" network is trained for either significantly different task (*e.g.* semantic segmentation, depth estimation) and/or domain (*e.g.* flowers, handwritten characters), and encodes significantly different representations. Hence naively distilling their representations into a single network would result in poor performance. To this end, we propose aligning the universal network with the individual ones via small task-specific adapters before the distillation, and using specific loss functions that are invariant to certain transformations between representations. Our method has multiple key advantages over the previous work. First, in contrast to relying solely on weighing the individual loss functions (Kendall et al., 2018) or modifying the direction of their gradients (Yu et al., 2020) that are limited to prevent one task dominating or interfering the rest, we propose more explicit control on the model parameters through knowledge distillation such that representations from all tasks/domains are included in the universal representations. Second, unlike task/domain-specific loss functions with different characteristics which are difficult to balance, the distillation loss function is the same for all tasks/domains and hence provides balanced optimization by design. Third, unlike Dvornik et al. (2020); Liu et al. (2021b) that employ multiple feature extractors, our model learns a single set of universal representations (a single feature extractor) over multiple domains which has a fixed computational cost regardless of the number of domains at inference. Finally, our method can be successfully incorporated to various state-of-the-art multi-task/domain customized network architectures (Vandenhende et al., 2020b; Rebuffi et al., 2018) and loss balancing strategies (Kendall et al., 2018; Liu et al., 2021c,a).

We illustrate our universal representation learning method and its applications to three standard vision problems in Fig. 2.1. The common step for all the applications is to first train a task- or domain-specific model and then distill their knowledge to a single universal network (see Fig. 2.1(a)). We show that the universal representations (depicted by the green network) can successfully be employed in jointly learning (i) multiple dense vision problems such as semantic segmentation, depth estimation (see Fig. 2.1(b)), (ii) multiple image classification problems from diverse datasets such as ImageNet (Deng et al., 2009), Omniglot (Lake et al., 2015), FGVC Aircraft (Maji et al.,

2013) (see Fig. 2.1(c)), (iii) learning to classify images from few training samples of unseen tasks and domains (detailed in Fig. 2.1(d) and Chapter 4). In all applications, the computations and representations are largely shared across tasks and domains through the universal network, while light-weight task or domain-specific heads are used to obtain the predictions by mapping the universal representations to the task output space.

2.2 Related Work

2.2.1 Multi-task Learning

Multi-task Learning (MTL) (Caruana, 1997) aims at learning a single model that can infer all desired task outputs given an input. Existing methods can be broadly divided into two groups. The first one (Misra et al., 2016; Ruder et al., 2019; Gao et al., 2019; Liu et al., 2019; Lu et al., 2017; Vandenhende et al., 2020a; Bruggemann et al., 2020; Guo et al., 2020; Bragman et al., 2019; Xu et al., 2018a; Zhang et al., 2019, 2018; Vandenhende et al., 2020b; Zhou et al., 2020; Bruggemann et al., 2021; Xu et al., 2022; Raychaudhuri et al., 2022; Ye and Xu, 2022; Wallingford et al., 2022) focuses on improving network architecture by better sharing information across tasks and learning task-specific representation by devising cross-task attention mechanism (Misra et al., 2016), task-specific attention modules (Liu et al., 2019), gating strategies (Bruggemann et al., 2020; Guo et al., 2020; Raychaudhuri et al., 2022), developing progressive communication across multiple tasks through recurrent networks (Bilen and Vedaldi, 2016; Zhang et al., 2018; Vandenhende et al., 2020b; Bruggemann et al., 2021; Xu et al., 2022), learning by adapting a pretrained network to another task by selecting which layers to tune (Wallingford et al., 2022), using Vision Transformer (Dosovitskiy et al., 2020) for better capture spatial position and tasks relations (Ye and Xu, 2022), so on.

The second group aims to address the unbalanced optimization that is caused by jointly optimizing multiple task loss functions with varying characteristics through either actively changing weight of each loss term (Kendall et al., 2018; Liu et al., 2019; Guo et al., 2018; Chen et al., 2018b; Lin et al., 2019; Sener and Koltun, 2018; Liu et al., 2021c, 2022b) and/or modifying the gradients of loss functions w.r.t. the shared network weights to alleviate the conflicts among tasks (Yu et al., 2020; Liu et al., 2021a; Chen et al., 2020c; Chennupati et al., 2019; Suteu and Guo, 2019). More specifically, Kendall et al. (2018) proposes to weigh multiple loss functions by considering the homoscedastic uncertainty of each task during training. Chen et al. (2018b) develops a

training strategy, namely GradNorm, that looks at the gradient's norm of each task and learns the weight to normalize each task's gradient so as to balance the losses for MTL. In MGDA (Sener and Koltun, 2018), Sener *et al*. formulate the MTL as a multiple objectives optimization problem and proposed an approximation Pareto optimization method using Frank-Wolfe algorithm to learn weights of losses. To design the weighting scheme, Guo et al. (2018) observe that the imbalances in task difficulty can lead to an unnecessary emphasis on easier tasks, thus neglecting and slowing progress on difficult tasks. Based on the observation, they introduce dynamic task prioritization for MTL, which allows the model to dynamically prioritize difficult tasks during training, where the difficulty is inversely proportional to performance. Liu et al. (2022b) formulate the loss weighting as a meta-learning problem where the loss weights are optimized such that the performance of the network learned with the loss weights obtains good performance on the validation set. Rather than weighing the losses, Yu et al. (2020) propose a form of gradient "surgery" that projects each task's gradient onto the normal plane of the gradient of any other task and modifies the gradients for each task so as to minimize negative conflict with other task gradients during the MTL optimization. Liu et al. (2021a) modifies the parameter update such that the update not only minimizes the average of task-specific losses but also decreases each task-specific loss.

Our method is complementary to the first line of work. In fact, we show in Sec. 2.5 that our method can be used to boost the state-of-the-art multi-task architectures in dense prediction problems. While our goal is aligned with the one of the second group, we propose a significantly different strategy based on knowledge distillation to solve the unbalanced loss optimization problem. To this end, we first train a task-specific model for each task in an offline stage and freeze their parameters. We then train the universal representation (multi-task) network for minimizing task-specific loss and also for producing the same features with the task-specific networks. As each task-specific network encodes different features, we introduce small task-specific adapters to project the universal features to the task-specific features and then minimize the discrepancy between task-specific and universal features. In contrast to prior works that either rely solely on weighing the individual loss functions (*e.g*. Uncertainty (Kendall et al., 2018)) or modifying their gradients for parameter updates (*e.g*. PCGrad (Yu et al., 2020)) that are limited to prevent one task dominating or interfering the rest, our method provides a more direct control on the model parameters through knowledge distillation such that representations from all tasks are included in the universal representations. Second, unlike task-specific loss functions with different characteristics which are difficult to

balance, the distillation loss function is the same for all tasks and hence provides a balanced optimization by design. In addition, in this chapter, we show that our method can also generalize to learning multiple diverse visual domains for standard image classification (Rebuffi et al., 2017a) and few-shot classification (Triantafillou et al., 2020).

2.2.2 Multi-domain learning (MDL)

A parallel line of research is learning representations jointly on multiple domains (Bilen and Vedaldi, 2017; Rosenfeld and Tsotsos, 2018; Rebuffi et al., 2018; Deecke et al., 2022). Unlike Ganin et al. (2016); Tzeng et al. (2017); Hoffman et al. (2018); Xu et al. (2018b); Peng et al. (2019); Sun et al. (2019b) that focus on domain adaptation, this line of work aims at learning a single set of universal representations over multiple tasks and visual domains. Bilen and Vedaldi (Bilen and Vedaldi, 2017) proposed to learn a compact multi-domain representation for standard image classification in multiple visual domains using domain-specific scaling parameters. This idea was later extended to the use of domain-specific adapters (Rebuffi et al., 2017a; Rosenfeld and Tsotsos, 2018; Rebuffi et al., 2018) and latent domains learning without access to domain annotations by learning gating functions to select domain-specific adapters for the given images (Deecke et al., 2022). In this chapter, we also learn a single set of universal (multi-domain) representations by sharing most of the computation across domains (*e.g.* the feature encoder is shared across all domains, followed by multiple domain-specific classifiers).

However, unlike Rebuffi et al. (2017a); Rosenfeld and Tsotsos (2018); Rebuffi et al. (2018); Deecke et al. (2022) that use representations pretrained only on ImageNet (Deng et al., 2009) as the universal ones, and then learn additional domain-specific representations resulting, our method is capable of learning a single set of universal representations from multiple domains which requires less number of parameters and it is also a significantly harder task due to the challenges in the multi-loss optimization. In addition, those works do not scale up to multi-task learning in a single domain, as they require running a network with the corresponding task-specific adaptors for each task separately, while ours requires only a single forward computation for all tasks.

2.2.3 Knowledge distillation

Our work is related to knowledge distillation (KD) methods (Hinton et al., 2014; Li and Bilen, 2020; Ma and Mei, 2019; Phuong and Lampert, 2019; Romero et al., 2015; Tian et al., 2020a) that distill the knowledge of an ensemble of large teacher models to a small student neural network at the classifier (Hinton et al., 2014) and intermediate layers (Romero et al., 2015). Born-Again Neural Networks (Furlanello et al., 2018) uses KD proposes to consecutively distill knowledge from an identical teacher network to a student network, which is further applied to few-shot learning by Tian et al. (2020b) and multi-task learning by Clark et al. (2019).

While our method can also be seen as a distillation method, our goal differs significantly. In contrast to the standard KD that aims to learn a single task/domain network from multiple teachers, our goal is to learn a multi-task/domain network. The difference is subtle. Multi-task/domain learning typically involves solving an unbalanced optimization, while aligning the predictions of the multi-task/domain (student) network with the task-specific (teacher) networks does not necessarily alleviate this issue, as this alignment problem leads to another unbalanced optimization problem due to varying dimensionality of task outputs and loss functions required for matching different tasks' predictions (Clark et al., 2019), *e.g.*, a kl-divergence loss for classification and l2-norm loss for regression. While alignment of intermediate representations are studied for KD by Romero et al. (2015), such an alignment is substantially harder when the representations vary significantly across different teacher networks. We demonstrate that mapping the student (or universal) representations to the teacher's representation space before the alignment is crucial. Finally, we also show that intermediate representation matching across very diverse domains can indeed be improved by using a loss function that is invariant to linear transformations, inspired from Centered Kernel Alignment (CKA) similarity (Kornblith et al., 2019).

2.3 Preliminaries

In this section, we review the problem setting for single-task and multi-task learning to provide the required background for universal representation learning. Let $\mathcal{D}$ be a training set consisting of N RGB training images and their respective labels. We consider two general label settings, multi-task learning (MTL) and multi-domain learning (MDL). In MTL, we assume that training images are sampled from a single distri-

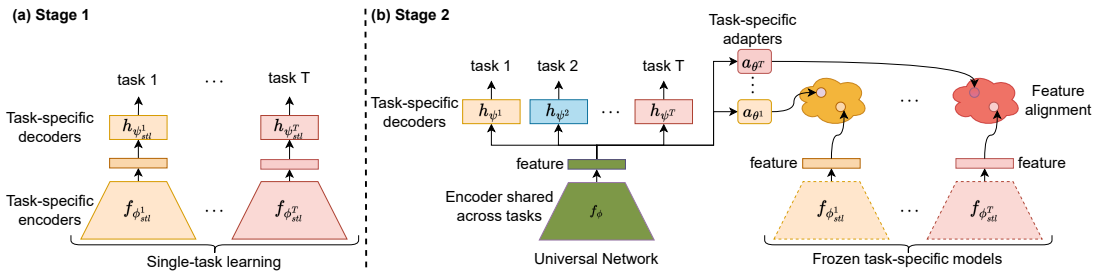


Figure 2.2: **Illustration of universal representation learning**. In the first stage (a), we learn a task-specific deep network for each task. In the second stage (b), our goal is to learn a multi-task network that shares the feature encoder across all tasks and build multiple task-specific decoders on top of the feature encoder such that it performs well on these tasks compared to task-specific models trained in (a). To achieve this, we train such a multi-task network by jointly minimizing task-specific losses and aligning the feature between the multi-task network and task-specific network. For the feature alignment, we introduce a set of task-specific adapters to transform the feature from multi-task network to task-specific space before the alignment with task-specific features.

bution, *i.e*. dataset, and each training image $\boldsymbol{x}$ is associated with labels for T tasks, $y = \{\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^T\}$. This is a common setting for dense prediction problems where multiple tasks such as semantic segmentation and depth estimation are performed on the same image. In MDL, the training set contains samples from T different domains, where each image is associated only with a single domain and its domain-specific task. The standard MDL benchmarks (Rebuffi et al., 2017a; Triantafillou et al., 2020) contain images from diverse datasets (*e.g*. ImageNet, Omniglot, VGG Flowers), each with a classification task over a mutually exclusive set of categories. Hence, an image $\boldsymbol{x}$ associated with domain t is labelled only with the domain-specific task $\boldsymbol{y}^t$. In both MTL and MDL settings, our goal is to learn a function $\hat{y}^t$ for each task t in MTL and for each domain t in MDL that accurately predicts the ground-truth label $\boldsymbol{y}^t$ of previously unseen images.

Note that while in MTL different tasks involve solving different problems such as semantic segmentation and depth estimation, in MDL they involve solving the same problem, *e.g*. image classification, however over a different set of categories. We do not focus on other scenarios where images from different domains are associated with the same task (*e.g*. digit recognition from hand-written notes and real street-numbers).

2.3.1 Single-task Learning

Single-task learning (STL) involves learning a task-specific function $\hat{y}^t_{stl}$ [2] independently for each task by optimizing a task-specific loss function $\ell^t(\hat{y}^t_{stl}, \boldsymbol{y}^t)$ (*e.g.* cross-entropy for classification), which measures the mismatch between the ground-truth label and prediction as following:

$$\min_{\phi^t_{stl}, \psi^t_{stl}} \frac{1}{N} \sum_{n=1}^{N} \ell^t(\hat{y}^t_{stl}(\boldsymbol{x}_n), \boldsymbol{y}^t_n), \quad \forall t \in \{1, \dots, T\} \tag{2.1}$$

where each $\hat{y}^t_s$ is composed of i) a feature encoder $f_{\phi^t_{stl}} : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{C\times H'\times W'}$ parameterized by ϕ^t_{stl} that takes in an $H \times W$ dimensional RGB image and outputs a $H' \times W'$ dimensional feature map with C channels, where $C > 3$, $H' < H$ and $W' < W$; ii) a decoder $h_{\psi^t_{stl}} : \mathbb{R}^{C\times H'\times W'} \rightarrow \mathbb{R}^{O^t\times H^t\times W^t}$ that decodes the extracted feature to predict the label for the task t, *i.e.* $\hat{y}^t_{stl}(\boldsymbol{x}) = h_{\psi^t_{stl}} \circ f_{\phi^t_{stl}}(\boldsymbol{x})$ where O^t, H^t, W^t are the dimensions of the output space for task t and ψ^t_{stl} are its parameters.

2.3.2 Multi-task Learning

A more efficient design is to share a significant portion of the computations and parameters across the tasks via a common feature encoder $f_\phi : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{C\times H'\times W'}$, *i.e.* convolutional neural network parameterized by ϕ that takes in an image $\boldsymbol{x}$ and produces a $H' \times W'$ dimensional C feature maps. The parameters ϕ are shared across all the tasks. In this setting, f_ϕ is followed by T task-specific decoders $h_{\psi^t} : \mathbb{R}^{C\times H'\times W'} \rightarrow \mathbb{R}^{O^t\times H^t\times W^t}$, each with its own task-specific weights ψ^t that decodes the extracted feature to predict the label for the task t, *i.e.* $\hat{y}^t(\boldsymbol{x}^t) = h_{\psi^t} \circ f_\phi(\boldsymbol{x}^t)$. A common way of learning $\hat{y}^t$ for all tasks is to jointly optimize the shared and task-specific parameters as following:

$$\min_{\phi, \{\psi^t\}|_{t=1}^T} \frac{1}{N} \sum_{n=1}^{N} \sum_{\boldsymbol{y}^t_n \in y_n} \lambda^t \ell^t(\hat{y}^t(\boldsymbol{x}_n), \boldsymbol{y}^t_n), \tag{2.2}$$

where λ^t is a scaling hyperparameter for task t to balance the loss functions among the tasks. However, obtaining good universal representations through solving Eq. (2.2) is a challenging problem, as it requires to leverage commonalities between the tasks while balancing their loss functions and minimizing interference (negative transfer (Chen et al., 2018b; Yu et al., 2020)) between them. Hence solving Eq. (2.2) often leads to lower results than the ones of task-specific models, where each task is independently learned.

[2] The subscript *stl* in $\hat{y}^t_{stl}$ indicates single-task learning

2.4 Universal Representation Learning

Motivated by these challenges, previous methods mainly focus on dynamically balancing the loss functions through loss weights (λ^t) or manipulating gradients of each task w.r.t. the shared encoder to alleviate the conflicts between them while optimizing Eq. (2.2). However, as reported in a study by Vandenhende et al. (2021), existing solutions fail to improve over carefully tuning these hyperparameters. We hypothesize that modifying hyperparameters and/or gradients provide only a limited control on the learned representations, and propose a different view on this problem, a two stage procedure inspired by the knowledge distillation methods (Romero et al., 2015; Hinton et al., 2014). Assuming that single-task learning often performs well when sufficient training data is available, we argue that single-task representations provide powerful representations and hence they provide good approximations to universal representations.

To this end, we first train task-specific deep networks $\{\hat{y}^t_{stl}\}_{t=1}^T$ as described in Sec. 2.3.1, where each network consists of a task-specific feature encoder $f_{\phi^t_{stl}}$ and decoder $h_{\psi^t_{stl}}$ with parameters ϕ^t_{stl} and ψ^t_{stl} respectively. In the second stage, we freeze their weights, the task-specific feature extractors $f_{\phi^t_{stl}}$ and decoders $h_{\psi^t_{stl}}$, and transfer their knowledge to learn a single set of universal representations by minimizing the distance between the task-specific and universal representations for given training samples (see Fig. 2.2).

As minimizing the distance between a single set of universal representations and multiple single-task representations would not yield a satisfactory solution, *i.e.* the average of the single-task representations, we instead first map the universal representations to each task-specific representation space through small task-specific adapters, and then compute the distance in this space. In addition, we consider minimizing the distance between the outputs of the universal and single-task networks as Hinton et al. (2014). With the introduction of these two distillation terms, Eq. (2.2) can be rewritten as:

$$\begin{aligned}\min_{\phi,\{\psi^t,\theta^t\}|_{t=1}^T} \frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{y}^t_n\in y_n}\Big(&\lambda^t\ell^t(\hat{y}^t(\boldsymbol{x}_n),\boldsymbol{y}^t_n)+\\ &\lambda^t_f\ell_f(a_{\theta^t}\circ f_\phi(\boldsymbol{x}^t_n), f_{\phi^t_{stl}}(\boldsymbol{x}^t_n)) + \lambda^t_p\ell^t_p(\hat{y}^t(\boldsymbol{x}^t_n),\hat{y}^t_{stl}(\boldsymbol{x}^t_n))\Big),\end{aligned} \tag{2.3}$$

where λ^t_f and λ^t_p are task-specific hyperparameters for distilling representations and predictions respectively. $a_{\theta^t}\colon \mathbb{R}^{C\times H'\times W'} \to \mathbb{R}^{C\times H'\times W'}$ is the adapter for task t which is parameterized by θ^t, ℓ_f and ℓ_p are distance functions in the task representation space t.

While a single distance function is used for distilling representations (*i.e.* ℓ_f), distilling predictions may require a task-specific distance function (*i.e.* ℓ_p^t). We provide these details in Sec. 2.5. The adapters are jointly trained along with the network parameters. Note that we discard the task-specific networks and adapters at test time, only use the universal network to predict the labels of unseen images. Hence, the inference time of our method is fixed and does not depend on the number of tasks/domains.

Next we describe how the universal representations are learned for different scenarios including multi-task dense prediction, multi-domain classification and cross-domain few-shot classification problems.

2.4.1 Multi-Task Dense Prediction

In multi-task dense prediction setting in a single domain, each image $\boldsymbol{x}$ is associated with labels for all tasks. The spatial dimensions of labels for each task is equal to the image size – hence it is called dense or pixelwise prediction – and is the same for all tasks. We consider semantic segmentation, monocular depth estimation and surface normal prediction in our experiments.

In this setting, we only minimize the difference between intermediate representations of the universal network and task-specific ones, and do *not* minimize the difference between the predictions of the universal and single-task networks as Clark et al. (2019); Hinton et al. (2014); Romero et al. (2015) and set λ_p^t to zero. In our preliminary experiments, we observed that matching the predictions leads to a significant drop in the final performance. As each task prediction has a different magnitude range and characteristic, we argue that jointly minimizing these distances along with other loss terms in Eq. (2.3) leads to a challenging unbalanced optimization.

Let $\boldsymbol{m}^t = a_{\theta^t} \circ f_\phi(\boldsymbol{x}) \in \mathbb{R}^{C\times H'\times W'}$ and $\boldsymbol{s}^t = f_{\phi_{stl}^t}(\boldsymbol{x}) \in \mathbb{R}^{C\times H'\times W'}$ denote representations obtained from the universal and single-task encoder for a given image $\boldsymbol{x}_n$ and task t, respectively. We normalize two feature maps with L2 Norm: $\tilde{\boldsymbol{m}}_{chw} = \boldsymbol{m}_{chw}/||\boldsymbol{m}_{\cdot hw}||_2$ and $\tilde{\boldsymbol{s}}_{chw} = \boldsymbol{s}_{chw}/||\boldsymbol{s}_{\cdot hw}||_2$, where $\boldsymbol{m}_{chw}$ indicates a hidden unit at c,h,w in $\boldsymbol{m}$. We then measure the distance between two normalized feature maps by using the Euclidean distance function for ℓ_f as following:

$$\ell_f(\boldsymbol{m},\boldsymbol{s}) = \sum_{w=1}^{W}\sum_{h=1}^{H}\sum_{c=1}^{C} \|\tilde{\boldsymbol{m}}_{chw} - \tilde{\boldsymbol{s}}_{chw}\|_2^2 . \tag{2.4}$$

We investigate different designs for a_θ (*e.g.* aligning features without adapters, with linear adapter or nonlinear) and ℓ_f (*e.g.* Attention Transfer Komodakis and Zagoruyko

(2017), Cosine Distance) and show that using linear adapters for a_θ with Euclidean distance function for ℓ_f obtains the best performance in Sec. 2.5.5.

2.4.2 Multi-Domain Classification

We also consider a multi-domain scenario as in Visual Decathlon (Rebuffi et al., 2017a), where the training set $\mathcal{D}$ contains T subdatasets, each sampled from a different domain. Each image is associated with only one domain and hence one task. The associated task is known in both train and test time as in in Visual Decathlon (Rebuffi et al., 2017a). Like the multi-dense prediction problem, our goal is to learn a single network with a shared feature encoder f_ϕ across the domains, thus the tasks. Unlike the multi-dense prediction problem, the output of the feature encoder is a vector (*i.e.* $H' = 1$ and $W' = 1$) and also the predictions (*i.e.* $H^t = 1$ and $W^t = 1$). In particular, the feature encoder f_ϕ is a convolutional neural network followed by an average global pooling layer as in ResNet (He et al., 2016).

Following the two stage procedure, we first independently train a set of domain-specific deep networks $\{\hat{y}^t_{stl}\}_{t=1}^T$ by Eq. (2.1) where each consists of a specific feature encoder $f_{\phi^t_{stl}}$ and classifier $h_{\psi^t_{stl}}$ with parameters ϕ^t_{stl} and ψ^t_{stl} respectively in the first stage. Unlike the previous setting that trains multiple task-specific networks on the same training set, this setting involves training each domain network on a different training set from a different domain. In the second stage, we then learn the universal network over training images of multiple domains using Eq. (2.3). Rather than setting λ_p in Eq. (2.3) to zero as in Sec. 2.4.1, here we use KL divergence loss for ℓ_p in Eq. (2.3) to align predictions of single-task and universal networks.

Though we use domain-specific adapters to map the universal features to the domain-specific space, learning a single set of representations over substantially diverse domains still remains challenging, requires to model complex non-linear relations between and hence a more elaborate distance function (ℓ_f) than the Euclidean one. To this end, we propose to adopt the Centered Kernel Alignment (CKA) (Kornblith et al., 2019) similarity index with the Radial Basis Function (RBF) kernel that is originally proposed as an analysis tool to measure similarities between neural network representations and shown to be invariant to various transformations, capable of capturing meaningful non-linear similarities between representations of higher dimension than the number of data points. Differently from the original goal, we use CKA as a loss function to minimize the distance between universal and domain-specific representations rather

than an analysis tool.

Next we briefly describe CKA. Given a set of images $\{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_B^t\}$, let $\mathbf{M} = [a_{\theta^t} \circ f_\phi(\boldsymbol{x}_1^t), \ldots, a_{\theta^t} \circ f_\phi(\boldsymbol{x}_B^t)]^\top \in \mathrm{R}^{B \times C}$ and $\mathbf{S} = [f_{\phi_{stl}^t}(\boldsymbol{x}_1^t), \ldots, f_{\phi_{stl}^t}(\boldsymbol{x}_B^t)]^\top \in \mathrm{R}^{B \times C}$ denote the features that are computed by the multi-domain network adapted by a_{θ^t} and domain-specific networks respectively. We first compute the RBF kernel matrices $\mathbf{P}$ and $\mathbf{T}$ of $\mathbf{M}$ and $\mathbf{S}$ respectively and then use two kernel matrices $\mathbf{P}$ and $\mathbf{T}$ to measure CKA similarity between $\mathbf{M}$ and $\mathbf{S}$:

$$\mathrm{CKA}(\mathbf{M}, \mathbf{S}) = \mathrm{tr}(\mathbf{PHTH}) / \sqrt{\mathrm{tr}(\mathbf{PHPH})\mathrm{tr}(\mathbf{THTH})}, \tag{2.5}$$

where $\mathrm{tr}(\cdot)$ and $\mathbf{H}$ denote the trace of a matrix and centering matrix $\mathbf{H}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ respectively. The loss $\ell_f(\mathbf{M}, \mathbf{Y})$ can be derived as $\ell_f(\mathbf{M}, \mathbf{S}) = 1 - \mathrm{CKA}(\mathbf{M}, \mathbf{S})$ as *dissimilarity* between the multi-domain and domain-specific features. As the original CKA similarity requires the computation of the kernel matrices over the whole datasets, which is not scalable to large datasets, we follow Nguyen et al. (2021) and compute them over each minibatch in our training. We refer to Kornblith et al. (2019); Nguyen et al. (2021) for more details.

Discussion. In contrast to the previous setting where the universal representations are learned over multiple domains from sufficiently large data, cross-domain few-shot learning requires obtaining the domain-specific knowledge from only few samples in an unseen domain, which is extremely challenging. Hence, our hypothesis is that transferring universal representations should yield more effective learning of new domains, as the base assumption is that there is a bounded number of representations for vision problems.

2.5 Experiments

In this section, we analyze and evaluate our method in three problems, i) learning multiple dense prediction tasks on two popular benchmarks, NYU-v2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) in Sec. 2.5.1, ii) learning multiple diverse visual domains on the Visual Domain Decathlon (Rebuffi et al., 2017a) in Sec. 2.5.4 and iii) few-shot classification on MetaDataset (Triantafillou et al., 2020) in Sec. 4.4. Finally, we conduct extensive analysis over various design choices in Sec. 2.5.5. [3]

[3]The code and models that are used in our experiments will be available at `https://github.com/VICO-UoE/UniversalRepresentations`.

2.5.1 Learning multiple dense prediction tasks

Here, we evaluate our method on learning universal representations for performing multiple dense prediction tasks on two standard multi-task learning benchmarks NYU-v2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) as in the prior works (Liu et al., 2019, 2021a).

Datasets and experimental setting. We follow the training and evaluation settings in the prior works (Liu et al., 2019, 2021a) for both single-task and multi-task learning in both datasets. More specifically, *NYU-V2* (Silberman et al., 2012) contains RGB-D indoor scene images, where we evaluate performances on 3 tasks, including 13-class semantic segmentation, depth estimation, and surface normals estimation. We use the true depth data recorded by the Microsoft Kinect and surface normals provided by Eigen and Fergus (2015) for depth and surface normal estimation as in the prior work Liu et al. (2019). All images are resized to 288×384 resolution as in MTAN (Liu et al., 2019). We follow the default setting (Silberman et al., 2012; Liu et al., 2019) where 795 and 654 images are used for training and testing, respectively. *Cityscapes* (Cordts et al., 2016) consists of street-view images, which are labeled for two tasks: 7-class semantic segmentation[4] and depth estimation. We resize the images to 128×256 to speed up the training as Liu et al. (2019).

In both NYU-v2 and Cityscapes, we follow the training and evaluation protocol in the prior work (Liu et al., 2019). We apply our method and all the baseline methods to two common multi-task architectures, encoder and decoder based ones. The encoder-based methods only share information in the encoder before decoding each task with an independent task-sepcific decoder while the decoder-based approaches also exchange information during the decoding stage (Vandenhende et al., 2021). For the encoder-based, we use the SegNet (Badrinarayanan et al., 2017) as the backbone. Following Liu et al. (2019), we use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation in Cityscapes, and cosine similarity loss for surface normal estimation in NYU-v2. We use the exactly same hyper-parameters including learning rate, optimizer and also the same evaluation metrics, mean intersection over union (mIoU), absolute error (aErr) and mean error (mErr) in the predicted angles to evaluate the semantic segmentation, depth estimation and surface normals estimation task, respectively in the prior work (Liu et al., 2019).

[4]The original version of Cityscapes provides labels 7&19-class semantic segmentation. We follow the 7-class semantic segmentation evaluation protocol as in the prior work (Liu et al., 2019) to be able to compare to the related works.

For the decoder-based, we build our method on PAD-Net (Xu et al., 2018a) and MTI-Net (Vandenhende et al., 2020b) that use multi-scale feature extractor (encoder) based on the HRNet-18 (Sun et al., 2019a) initialized with ImageNet pretrained weight as the feature encoder. We use the same loss functions, evaluation metrics, and training and evaluation protocol as done for SegNet backbone. For our method, we use the uniform loss weights (*i.e.* $\lambda^t = 1$ for all tasks) for task-specific losses, unless stated otherwise. As we do *not* minimize the difference between predictions of the universal and single-task networks, we set λ_p^t in Eq. (2.3) to zero. We then first split the train set as train and validation set to search $\lambda_f^t \in \{1,2\}$ by cross-validation and train our network on the whole training set. We set λ_f^t to 1 for semantic segmentation and depth and 2 for surface normal estimation. Please refer to Appendix A.1.1 for more details.

Multi-task performance. In addition to the abovementioned evaluation metric for each task, following prior works (Vandenhende et al., 2021; Li et al., 2022), we also report the multi-task performance $\triangle$MTL whih measures the average per-task drop in performance w.r.t. the single-task baseline:

$$\triangle \text{MTL} = \frac{1}{T}\sum_{t=1}^{T}(-1)^{\ell_t}(P^t - P_{stl}^t)/P_{stl}^t, \tag{2.6}$$

where $\ell_t = 1$ if a lower value of P^t means better performance for metric of task t, and 0 otherwise. P^t and P_{stl}^t are performance (*e.g.* mIoU for semantic Segmentation) of the universal (multi-task) network and single-task network, respectively.

2.5.2 Encoder-based Architecture

Compared methods. Encoder-based architectures, including the vanilla MTL using *SegNet* (Badrinarayanan et al., 2017) that shares the whole feature encoder across all tasks and consists of task-specific decoders, and *MTAN* (Liu et al., 2019) which extends the vanilla MTL baseline by sharing the SegNet across tasks and using task-specific attention modules in each layer to extract task-specific features. We compare our method to the single-task learning (STL) baseline, *i.e.* train individual network per task, the vanilla multi-task learning network with uniform loss weights (MTL), and balanced optimization strategies, including Uncertainty (Kendall et al., 2018), GradNorm (Chen et al., 2018b), MGDA (Sener and Koltun, 2018), DWA (Liu et al., 2019), PCGrad (Yu et al., 2020), GradDrop (Chen et al., 2020c), IMTL (Liu et al., 2021c) and CAGrad (Liu et al., 2021a). We also consider the BAM (Clark et al., 2019) which is originally designed for natural language processing and adapt this

method for visual dense prediction tasks by aligning dense predictions. Importantly, this model performs knowledge distillation on predictions when learning the multi-task network and hence comparing this method sheds light onto the importance of matching intermediate representations in the task-specific spaces. Here we use KL-divergence loss, l1-norm loss and cosine similarity loss as knowledge distillation loss on predictions for semantic segmentation, depth estimation and surface normal estimation, respectively. We reproduce all methods in the same settings for fair comparison and the results of the compared methods are similar or better than the ones reported in the corresponding papers.

Results on NYU-v2. Table 2.1 depicts the results of our method and other compared approaches in NYU-v2. We see that the vanilla MTL (Uniform) using SegNet achieves better performance in depth estimation, however, its performance drops in surface normal estimation in comparison with STL. This indicates that joint optimization of multiple tasks with uniform loss weights leads to unbalanced results, and overall worse performance than STL models in ΔMTL metric. While only few balanced optimization algorithms help to improve the MTL performance, IMTL-H and CAGrad obtains the best when applied to SegNet and MTAN. IMTL-H improves by balancing the pace at which tasks are learned via looking at the projection onto individual tasks of the average gradient w.r.t. the shared parameters while CAGrad achieves improvement by modifying the parameter update such that the update not only minimizes the average of task-specific losses but also decreases each task-specific loss. While BAM optimizes the multi-task learning network by knowledge distillation, it performs worse than the Uniform baseline as it aligns the predictions which requires using different loss functions (*e.g.* cross-entropy for segmentation) and requires solving another unbalanced optimization problem. This shows that simply distilling the predictions of the multiple single task network leads to poor performance. Finally, our method outperforms these methods with either SegNet or MTAN backbones significantly, +7.84% MTL performance improvement over the IMTL-H, the best baseline. The results suggest that distilling features from multiple single-task networks provides a more effective learning of shared representations.

Results on Cityscapes. We also evaluate all methods in Cityscapes and report the results in Tab. 2.2. Similar to the results in NYU-v2, BAM obtains worse results compared with the STL methods (*e.g.*-0.58% MTL performance when using the vanilla MTL method with SegNet). Among the loss balancing methods, Uncertainty and IMTL-

Arch	Method	Seg. (mIoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
SegNet	STL	40.54	0.6276	**24.28**	+0.00
	Uniform	40.22	0.5196	29.09	-1.13
	Uncertainty (Kendall et al., 2018)	37.35	0.5014	26.74	+0.72
	GradNorm (Chen et al., 2018b)	40.15	0.5824	27.70	-2.60
	MGDA (Sener and Koltun, 2018)	39.77	0.5669	29.05	-3.95
	DWA (Liu et al., 2019)	40.27	0.5247	28.71	-0.82
	PCGrad (Yu et al., 2020)	39.55	0.5236	28.54	-1.13
	GradDrop (Chen et al., 2020c)	39.25	0.5226	29.41	-2.52
	IMTL-H (Liu et al., 2021c)	40.62	0.5224	26.14	+3.11
	CAGrad (Liu et al., 2021a)	37.99	0.5196	25.77	+1.61
	BAM (Clark et al., 2019)	37.72	0.5571	28.58	-4.47
	Ours	**45.52**	**0.4912**	24.57	**+10.95**
MTAN	STL	38.69	0.5701	24.85	+0.00
	Uniform	40.60	0.5238	26.87	+1.65
	Uncertainty (Kendall et al., 2018)	39.59	0.5382	26.21	+0.83
	GradNorm (Chen et al., 2018b)	38.97	0.5269	26.22	+0.94
	MGDA (Sener and Koltun, 2018)	40.08	0.5410	26.84	+0.23
	DWA (Liu et al., 2019)	40.34	0.5418	27.65	-0.68
	PCGrad (Yu et al., 2020)	39.42	0.5290	27.17	-0.07
	GradDrop (Chen et al., 2020c)	39.19	0.5552	27.35	-2.05
	IMTL-H (Liu et al., 2021c)	41.12	0.5200	25.73	+3.86
	CAGrad (Liu et al., 2021a)	42.17	0.5227	25.42	+5.01
	BAM (Clark et al., 2019)	40.49	0.5412	26.84	+0.58
	Ours	**43.91**	**0.5019**	**24.58**	**+8.85**

Table 2.1: Test performance on NYU-v2. We evaluate single-task learning (STL) method and multi-task learning methods (MTL) on NYU-v2. Mean intersection over union (mIoU) for semantic segmentation, absolute error (aErr) for depth estimation, mean error (mErr) for surface normal estimation and multi-task performance (△MTL) are reported.

H obtain the best MTL performance in both backbones (*i.e*. SegNet and MTAN) but their performance is lower than the STL models in both tasks. This shows the difficulty of optimizing the MTL network in a balanced way in this problem. Our method obtains significant gains on both tasks than all the compared methods and also achieves better results than the STL results. The results again demonstrate that our method is able to optimize the MTL model in a more balanced way and to achieve better overall results. In addition, our method has much less parameters (one network) than the STL models (two networks) in Cityscapes.

Arch	Method	Seg. (mIoU) ↑	Depth (aErr) ↓	△MTL ↑
SegNet	STL	74.19	0.0122	+0.00
	Uniform	73.82	0.0126	-0.74
	Uncertainty (Kendall et al., 2018)	72.74	0.0123	-0.29
	GradNorm (Chen et al., 2018b)	73.67	0.0130	-2.75
	MGDA (Sener and Koltun, 2018)	73.86	0.0130	-2.43
	DWA (Liu et al., 2019)	73.51	0.0126	-1.13
	PCGrad (Yu et al., 2020)	73.55	0.0126	-1.16
	GradDrop (Chen et al., 2020c)	73.09	0.0125	-0.96
	IMTL-H (Liu et al., 2021c)	73.26	0.0124	-0.56
	CAGrad (Liu et al., 2021a)	74.50	0.0136	-4.34
	BAM (Clark et al., 2019)	74.02	0.0123	-0.58
	Ours	**75.53**	**0.0119**	**+2.21**
MTAN	STL	75.92	0.0119	+0.00
	Uniform	75.31	0.0119	-0.56
	Uncertainty (Kendall et al., 2018)	74.95	0.0121	-1.73
	GradNorm (Chen et al., 2018b)	74.88	0.0123	-2.70
	MGDA (Sener and Koltun, 2018)	75.84	0.0129	-4.65
	DWA (Liu et al., 2019)	75.39	0.0121	-1.42
	PCGrad (Yu et al., 2020)	75.62	0.0122	-1.73
	GradDrop (Chen et al., 2020c)	75.69	0.0123	-2.15
	IMTL-H (Liu et al., 2021c)	75.33	0.0120	-1.20
	CAGrad (Liu et al., 2021a)	75.45	0.0124	-2.69
	BAM (Clark et al., 2019)	75.74	0.0122	-1.44
	Ours	**76.42**	**0.0117**	**+0.81**

Table 2.2: Testing results on Cityscapes. We evaluate single-task learning (STL) method and multi-task learning methods (MTL) on Cityscapes. Mean intersection over union (mIoU) for semantic segmentation, absolute error (aErr) for depth estimation and multi-task performance (△MTL) are reported.

Incorporating loss balancing to ours. While our method achieves consistent improvements over all the target tasks, solving Eq. (2.3) also involves minimizing a weighted sum of multiple loss terms. Hence here we investigate whether our method can also benefit from dynamically setting weights of the individual loss terms in NYU-v2. In particular, we use SegNet as backbone, and we dynamically update the weights of task-specific losses (*i.e.* λ^t) with keeping the weight of distillation loss fixed (λ_f). We evaluate our method with each of three best performing loss balancing methods, *i.e.* Uncertainty, IMTL-H and CAGrad and report the results in Tab. 2.3. We see that our method is complementary to these loss balancing methods and it significantly improves

the performance of loss balancing methods (*e.g.* Ours (Uncertainty) obtains about +12 improvement in MTL performance over the Uncertainty). Also, we can see that by applying our method to loss balancing methods obtains better performance than using our method with the Uniform MTL baseline.

Arch	Method	Seg. (mIoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
SegNet	STL	40.54	0.6276	24.28	+0.00
	Uniform	40.22	0.5196	29.09	-1.13
	Ours (Uniform)	**45.52**	**0.4912**	**24.57**	**+10.95**
	Uncertainty (Kendall et al., 2018)	37.35	0.5014	26.74	+0.72
	Ours (Uncertainty)	**46.03**	**0.4780**	**24.04**	**+12.80**
	IMTL-H (Liu et al., 2021c)	40.62	0.5224	26.14	+3.11
	Ours (IMTL-H)	**46.33**	**0.5000**	**23.65**	**+12.41**
	CAGrad (Liu et al., 2021a)	37.99	0.5196	25.77	+1.61
	Ours (CAGrad)	**45.58**	**0.5059**	**23.68**	**+11.44**

Table 2.3: Testing results on NYU-v2. We evaluate single-task learning (STL) method and multi-task learning methods (MTL) on NYU-v2. Mean intersection over union (mIoU) for semantic segmentation, absolute error (aErr) for depth estimation, mean error (mErr) for surface normal estimation and multi-task performance (△MTL) are reported.

2.5.3 Decoder-based Architectures

We also apply our method to the decoder-based methods, PAD-Net (Xu et al., 2018a) and MTI-Net (Vandenhende et al., 2020b) which are particularly designed for MTL by exchanging information during the decoding stage and achieve state-of-the-art performances in MTL (Vandenhende et al., 2021). Apart from these results, we also include results of the vanilla MTL method using the same backbone (HRNet-18 (Sun et al., 2019a)) of PAD-Net and MTI-Net as baseline and we report all results in NYU-v2 and Cityscapes in Tab. 2.4 and Tab. 2.5.

First we see that decoder based methods achieve better performance than the encoder based ones, as they use more powerful customized architectures and initialized with pre-trained ImageNet weights. Similar to encoder-based methods, the vanilla MTL method obtains better performance in Segmentation than STL while it performs worse in depth and surface normal estimation than STL models in NYU-v2 due to the unbalanced optimization. In Cityscapes, it performs worse in both tasks than STL baselines. Our method when applied to the vanilla MTL method using HRNet-18 backbone

improves the performance over the vanilla MTL method and achieves a balanced MTL performance (*i.e*. better or comparable results than STL) in both datasets.

We see in Tab. 2.4 and Tab. 2.5 that the decoder-based methods (PAD-Net and MTI-Net) obtain better performance than the vanilla MTL method by first employing a multi-task network to make initial task predictions, and then leveraging features from these initial predictions to improve each task output (MTI-Net obtains +1.72 MTL performance in NYU-v2). Here, PAD-Net improves over the vanilla MTL by aggregating information from the initial task predictions of other tasks by spatial attention for estimating the final task output while MTI-Net extends the PAD-Net to a multi-scale procedure by making initial task predictions and distilling information at each individual scale (of feature). However, they still suffer from the unbalanced optimization problem (*e.g*. MTI-Net obtains worse performance in surface normal estimation in NYU-v2 and semantic segmentation in Cityscapes, respectively). Building our method on these decoder-based methods helps to boost their performance (Ours (MTI-Net) vs MTI-Net: +4.11% vs +1.72%). These results indicate that our method can be used with various architectures and enable more balanced performance over multiple tasks, and boost their overall performance.

Method	Seg. (mIoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
STL	53.07	0.3608	20.90	+0.00
MTL	53.39	0.3626	23.35	-3.87
Ours (MTL)	**54.03**	**0.3565**	**21.43**	**+0.15**
PAD-Net (Xu et al., 2018a)	54.07	0.3583	22.60	-1.85
Ours (PAD-Net)	**54.75**	**0.3537**	**21.80**	**+0.28**
MTI-Net (Vandenhende et al., 2020b)	54.59	0.3353	21.90	+1.72
Ours (MTI-Net)	**56.48**	**0.3317**	**21.36**	**+4.11**

Table 2.4: Testing results on NYU-v2. We evaluate single-task learning (STL) method and decoder-based multi-task learning methods (MTL) with HRNet on NYU-v2. Mean intersection over union (mIoU) for semantic segmentation, absolute error (aErr) for depth estimation, mean error (mErr) for surface normal estimation and multi-task performance (△MTL) are reported.

Method	Seg. (mIoU) ↑	Depth ↓	△MTL ↑
STL	76.92	0.0111	+0.00
MTL	76.67	0.0115	-1.86
Ours (MTL)	**77.20**	**0.0111**	**+0.06**
PAD-Net (Xu et al., 2018a)	77.83	0.0109	+1.47
Ours (PAD-Net)	**77.84**	**0.0107**	**+2.48**
MTI-Net (Vandenhende et al., 2020b)	76.61	0.0111	-0.17
Ours (MTI-Net)	**77.10**	**0.0109**	**+0.77**

Table 2.5: Testing results on Cityscapes. We evaluate single-task learning (STL) method and decoder-based multi-task learning methods (MTL) with HRNet on Cityscapes. Mean intersection over union (mIoU) for semantic segmentation, absolute error (aErr) for depth estimation and multi-task performance (△MTL) are reported.

Model #images	#params	ImNet 1.3m	Airc. 7k	C100 50k	DPed 30k	DTD 4k	GTSR 40k	Flwr 2k	OGlt 26k	SVHN 70k	UCF 9k	avg ↑	S ↑	△MDL ↑
Feature (ImNet) (Rebuffi et al., 2017a)	1	59.67	23.31	63.11	80.33	45.37	68.16	73.69	58.79	43.54	26.80	54.28	544	- -
Scratch (Rebuffi et al., 2017a)	10	59.87	57.10	75.73	91.20	37.77	96.55	56.30	88.74	96.63	43.27	70.23	1625	- -
Finetune (Rebuffi et al., 2017a)	10	59.87	60.34	82.12	92.82	55.53	97.53	81.41	87.69	96.55	51.20	76.51	2500	+0.00
Serial RA (Rebuffi et al., 2017a)	2	59.67	56.68	81.20	93.88	50.85	97.05	66.24	89.62	96.13	47.45	73.88	2118	-3.95
DAN (Rosenfeld and Tsotsos, 2018)	2.17	57.74	64.12	80.07	91.30	56.54	98.46	86.05	89.67	96.77	49.38	77.01	2851	+0.60
Piggyback (Mallya et al., 2018)	1.28	57.69	65.29	79.87	96.99	57.45	97.27	79.09	87.63	**97.24**	47.48	76.60	2838	+0.00
MDL	1	59.30	67.21	79.60	97.08	57.66	97.28	83.88	87.41	96.39	45.76	77.16	2866	+0.75
Parallel RA (Rebuffi et al., 2018)	2	60.32	64.21	81.91	94.73	58.83	**99.38**	84.68	89.21	96.54	50.94	78.07	3412	+2.20
Parallel RA SVD (Rebuffi et al., 2018)	1.5	60.32	66.04	81.86	94.23	57.82	99.24	85.74	89.25	96.62	**52.50**	78.36	3398	+2.70
Ours	1	61.45	74.35	80.45	97.55	59.57	98.24	86.73	89.72	96.95	49.01	79.40	3626	+4.19
Ours (RA)	2	**61.97**	**77.17**	**82.31**	**97.58**	**60.69**	98.75	**87.28**	**90.40**	97.14	51.89	**80.52**	**4005**	**+5.96**

Table 2.6: Universal Representation Learning on Visual Decathlon. Accuracy on the test sets of individual dataset, average accuracy of 10 datasets (avg), evaluation score (S), multi-domain learning performance (△MDL) and the number of parameters (#params) w.r.t. a single task network are reported.

2.5.4 Multi-domain Learning

Here we evaluate our method on learning universal representations for multiple image classification tasks over multiple diverse domains in Visual Decathlon Benchmark (Rebuffi et al., 2017a).

Dataset. The *Visual Decathlon Benchmark* (Rebuffi et al., 2017a) consists of 10 different well-known datasets: including ILSVRC_2012 (ImNet) (Russakovsky et al., 2015), FGVC-Aircraft (Airc.) (Maji et al., 2013), CIFAR-100 (C100) (Krizhevsky et al., 2009), Daimler Mono Pedestrian Classification Benchmark (DPed) (Munder and Gavrila, 2006), Describable Texture Dataset (DTD) (Cimpoi et al., 2014), German Traffic Sign Recognition (GTSR) (Houben et al., 2013), Flowers102 (Flwr) (Nilsback

and Zisserman, 2008), Omniglot (OGlt) (Lake et al., 2015), Street View House Numbers (SVHN) (Netzer et al., 2011), UCF101 (UCF) (Soomro et al., 2012). In this benchmark (Rebuffi et al., 2017a), images are resized to a common resolution of roughly 72 pixels to accelerate training and evaluation by the organizers.

Implementation details. We follow Rebuffi et al. (2017a, 2018), use the official train/val/test splits, evaluation protocol, also use the ResNet-26 (He et al., 2016) as the backbone for domain-specific network and universal network. In our universal network, the backbone (*i.e*. ResNet-26) is shared across all domains and followed by domain-specific linear classifiers. We use the same data augmentation (random crop, flipping) and SGD as optimizer, and train domain-specific networks and our universal network for 120 epochs as in the prior works (Rebuffi et al., 2017a, 2018). Here we set the loss weights to 1 (*i.e*. $\lambda^t = 1$ for all tasks) and perform cross-validation to search loss weights $(\lambda_f^t, \lambda_p^t)$ in $\{0.1, 1, 10\}$ for knowledge distillations on features and predictions, and set λ_f and λ_p to 10 for ImageNet, 0.1 for DPed, and 1 for other datasets. Please refer to Appendix A.1.2 for more details.

Results. In Visual Decathlon, we compare our method to *Feature i.e*. a feature extractor on ImageNet and learn classifiers on top of the feature extractor for other domains, and single domain learning models that are learned from *Scratch* or *Finetune* from the ImageNet pretrained feature extractor. We also compared our method with existing approaches, including Serial Residual Adapters (RA) (Rebuffi et al., 2017a), Parallel RA and Parallel RA SVD (Rebuffi et al., 2018), DAN (Rosenfeld and Tsotsos, 2018) and Piggyback (Mallya et al., 2018). Results are from the corresponding papers.

We report the results on the test split on each domains by the official online evaluation (Rebuffi et al., 2018) in Tab. 2.6, including testing accuracy in individual datasets, average accuracy over 10 datasets (avg), decathlon evaluation score (S) (Rebuffi et al., 2018), number of parameters (#params) w.r.t. one single task network. We also consider the multi-domain performance (*i.e*. $\triangle$MDL) as described in Eq. (2.6). First, while using ImageNet features requires only $1\times$ parameters, they do not generalize well to other datasets when a large domain gap is present (*e.g*. SVHN). In contrast, single-task learning models obtained by either learning from scratch or finetuning achieves significantly better performance (*e.g*. Finetune obtains 76.51 average accuracy and 2500 score) with the expense of 10 times more parameters.

We use Finetune as the baseline as in the prior works (Rebuffi et al., 2017a, 2018) and compute $\triangle$MDL metric for existing methods and ours. We can see that Serial RA which learns a set of domain-specific residual adapters for each task with a ImageNet

pretrained feature extractor greatly reduce the number of parameter to $2\times$ while it obtains slightly worse performance than Finetune (*e.g*. 73.88 vs 76.51 average accuracy for Serial RA and Finetune, respectively). The performance is further improved by DAN which constrains newly learned filters to be linear combinations of existing ones when adapting a pretrained model for other domains, *i.e*. DAN obtains 77.01 average accuracy, +0.60 MDL performance and only requires 2.17 parameters). Piggyback learns binary domain-specific masks to select effective filters to adapt a pretrained model for each domain (it obtains 76.60 average accuracy, +0.00 MDL performance and further reduces the number of parameters to 1.28). Connecting the RAs in parallel to the backbone (Parallel RAs) boosts the performance of the serial configuration while keeping the same computation cost (78.07 in average accuracy and +2.20 in MDL performance). The authors show that the performance can be further improved by decomposing residual adapters to low rank adapters through (78.36 average accuracy, +2.70 MDL performance and 1.5 parameters).

Finally, we show that our method (Ours) successfully learns a single feature extractor shared across all domains only with $1\times$ parameters, the same number of parameters with a single domain network. Our model obtains better results than the Finetune baseline and existing methods in most domains (Ours obtains 79.40 average accuracy and +4.19 MDL performance). This clearly shows that learning representations from all domains jointly produces more general features than ImageNet representations. However, this is challenging due to the optimization issues. The difference between our model and vanilla MDL model shows that simply optimizing over multiple domain-specific loss functions is not sufficient to obtain good representations and representation distillation is crucial. We also show that RAs can be incorporated into our universal network. Jointly learning a shared ResNet-26 backbone with residual adapters (*i.e*. Ours (RA)) boosts the performance, *e.g*. Ours (RA) obtains the best or second best performance in most datasets (*e.g*. Airc., DTD, etc), best average accuracy (80.52), best MDL performance (+5.96) while only requires 2 in parameters cost and best score (4005).

2.5.5 Further Analysis

In this section, we provide an extensive analysis over various adapter types, loss functions for knowledge distillation for multi-task learning and multi-domain learning.

Effect of adapters. As explained in Sec. 2.4, we employ adapters to align the universal representations with each task-specific representation (see a_{θ^t} in Eq. (2.3)). Here,

we evaluate our method without any adapters (by directly matching universal representations with task-specific ones), also with two different adapter parameterizations including *linear* adapters (*i.e*. each adapter is constructed by a linear 1×1 convolutional layer, this is the default setting in Sec. 2.5.1, Sec. 2.5.4), *nonlinear* adapters (*i.e*. each adapter consists of two linear convolutional layers and a ReLU layer between them). We report their results on the NYU-v2 dataset in Tab. 2.7. From the results, we can see that, though directly aligning features without the adapters improves performance on all tasks over the vanilla MTL baseline (Uniform), it still performs significantly worse than using either linear or non-linear adapters. This verifies that the adapters help align features between the multi-task network with features of different single-task networks. We also observe that, using linear and nonlinear adapters obtains comparable results and using linear adapters is sufficient. We hypothesize that there is a tradeoff between the complexity of adapters and informativeness of aligned features. For instance, using deep multi-layer adapters would overfit the data and align the pairs very accurately, hence leading to inferior representation transfer. Thus we argue that the linear adapters provide a good complexity/performance tradeoff.

Arch	Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
SegNet	STL	SL	40.54	0.6276	24.28	+0.00
	MTL	Uniform	40.22	0.5196	29.09	-1.13
		Ours (w/o adapter)	41.64	0.5086	25.88	+5.03
		Ours (nonlinear)	44.84	**0.4881**	24.59	+10.52
		Ours (linear)	**45.52**	0.4912	**24.57**	**+10.95**

Table 2.7: Testing results on NYU-v2. 'linear' means we use a linear convolutional layer for adapters and 'nonlinear' means we use non-linear adapters (*i.e*. each adapter consists of two linear convolutional layers and a ReLU layer between them). 'w/o adapter' means aligning features without any adapters

Loss functions for knowledge distillation. Here we evaluate various loss functions for distilling intermediate representations (*i.e*. $\ell_f(\cdot)$ in Eq. (2.3)) including standard ones such as L2, cosine distance, and also Attention Transfer (AT) (Komodakis and Zagoruyko, 2017) that align the spatial attention maps computed by averaging the feature maps along the channel dimension and CKA. Here, we use linear adapters for aligning features between multi-task and single-task networks before measuring their discrepancy with these loss functions.

Arch	Type	Loss	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
SegNet	STL	SL	40.54	0.6276	24.28	+0.00
	MTL	-/-	40.22	0.5196	29.09	-1.13
		AT	39.93	0.5060	28.81	-0.26
		Cosine	44.22	0.4969	25.45	+8.37
		L2	**45.52**	**0.4912**	**24.57**	**+10.95**

Table 2.8: Testing results on NYU-v2. 'MTL' and 'STL' means multi-task learning and single-task learning, respectively. 'AT', 'KL', 'Cosine' and 'L2' means using Attention Transfer, KL-divergence, Cosine and L2 loss functions respectively.

We first evaluate these loss functions for multiple dense prediction problems in NYU-v2 and report the results in Tab. 2.8 where our default loss function is L2. Here, we apply knowledge distillation with different loss functions to the vanilla MTL method with SegNet (Badrinarayanan et al., 2017) as backbone (Note that CKA loss function is not included as it requires too large memory cost to operate on feature maps). From the results, we can see that AT obtains the worst performance among these loss functions as it aligns the averaged features where some information is lost, but it still outperforms the vanilla MTL with uniform loss weights. While Cosine loss function performs better than AT, using L2 loss function obtains the best results.

As the official online evaluation (Rebuffi et al., 2018) of the Visual Decathlon Benchmark only allow 10 submissions for the evaluation and the validation set is not representative of the distribution of the test set for some datasets [5]. To this end, we evaluate various adapter and loss function designs in learning universal representations from multiple domains for cross-domain few-shot learning in Sec. 4.4.5 in MetaDataset (Triantafillou et al., 2020). As shown in Sec. 4.4.5, we again show that directly aligning features without adapters is worse than using linear adapters. We also show that the proposed CKA loss function for aligning features performs better than L2 and COSINE loss functions and adding additional KL loss function on predictions further boosts the performance (See Sec. 4.4.5 for detailed results).

[5]The validation set is not representative of the distribution of the test set for some datasetsas mentioned in this issue `https://github.com/srebuffi/residual_adapters/issues/3`.

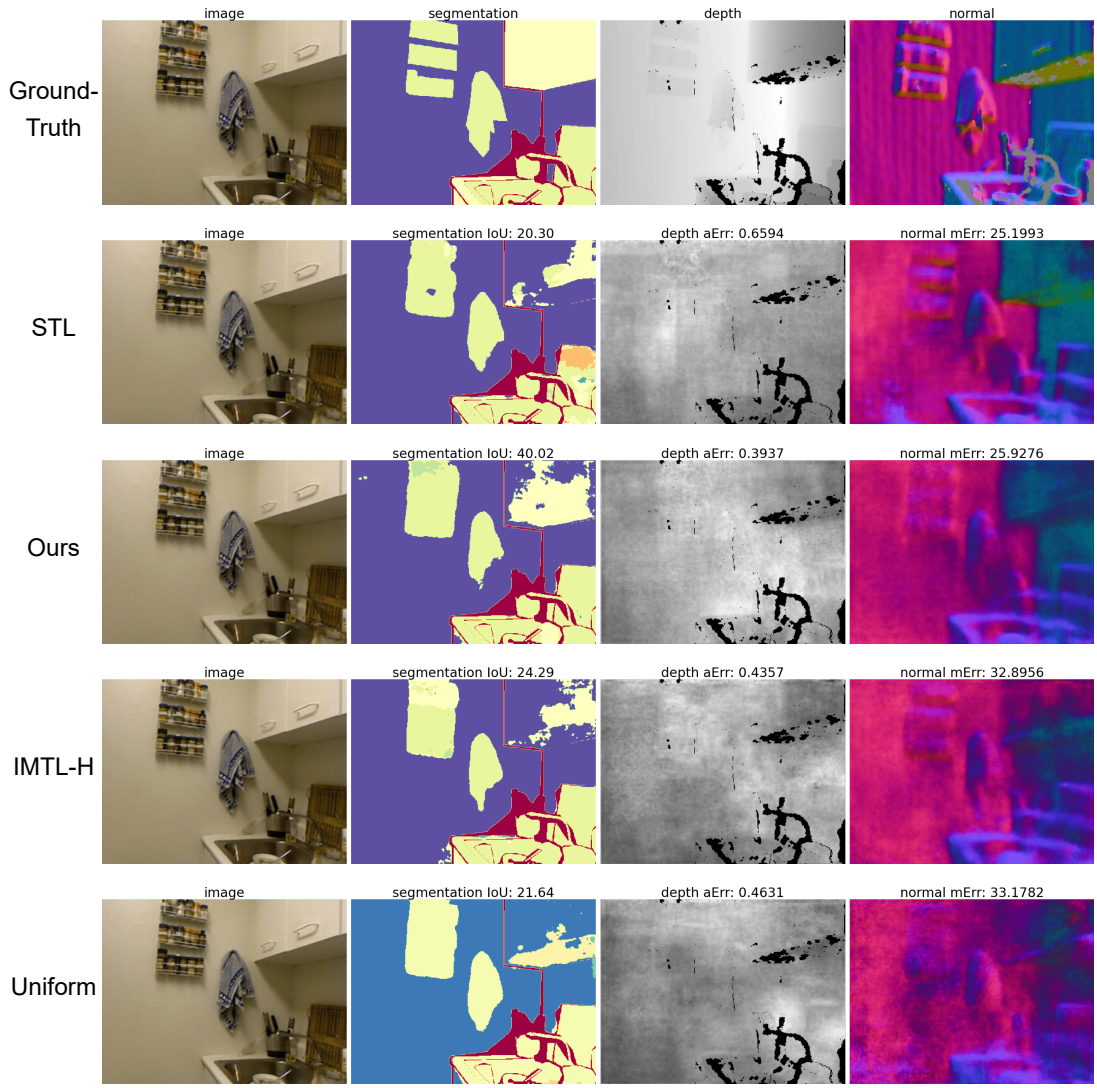


Figure 2.3: **Qualitative results on NYU-v2**. The fist column shows the RGB image, the second column plots the ground-truth or predictions with the IoU (↑) score of all methods for semantic segmentation, the third column presents the ground-truth or predictions with the absolute error (↓), and we show the prediction of surface normal with mean error (↓) in the last column.

2.5.6 Qualitative results

Here, we analyze our method and qualitatively compare our method to STL, MTL with Uniform loss weights and the best compared method, *i.e.* IMTL-H (Liu et al., 2021c) for multi dense prediction problem on NYU-v2 with SegNet backbone (see Fig. 2.3, see Appendix A.2 for more examples). We can see that, Uniform baseline obtains improvement on segmentation and depth estimation over STL, while it performs worse in surface normal estimation. Though dynamically balancing the loss values with

IMTL-H improves the overall performance, it still performs worse in surface normal estimation. Finally by distilling representations from single-task learning model to the universal network, our method can produce better or comparable results to STL, *i.e*. our method produces similar outputs for surface normal as STL and more accurate predictions for segmentation and depth estimation as our method enables a balanced optimization of universal network and a task can be benefited from another one. This indicates the effectiveness of our method on learning shared representations for multiple dense predictions.

2.6 Conclusion and Limitations

We showed that learning general features from multiple tasks and domains is an important step for better generalization in various computer vision problems including multiple dense prediction, multi-domain image classification and we also validate the same capability for cross-domain few-shot learning in Chapter 4. By distilling representations from multiple task-specific or domain-specific networks, we can successfully learn a single set of universal representations after aligning them via small task/domain-specific adapters. These representations are compact and generalize better to unseen samples, tasks and domains in multiple benchmarks.

A limitation of our method is that it requires multiple models to learn universal representations at train time which can be computationally expensive in the presence of many tasks. In future, we would like to alleviate this problem by hierarchically grouping similar tasks and learning a single model on them, and then distilling their knowledge into universal representations. We would also like to extend our method to problems that involve multiple tasks and multiple domains at the same time such as semantic segmentation and depth estimation from different cities (or domains).

Another limitation of learning universal representations for multiple dense prediction tasks is that standard methods are data inefficient. In particular, they require all the task labels are available to learn the multi-task learning model. However, in a real-world scenario, it is costly and difficult to collect such datasets and it is more practical to design algorithms to learn the MTL model in a more data efficient way, *e.g*. learning MTL model from partial supervision. In the next chapter, we formulate this problem in a more practical setting called multi-task partially supervised learning and present an efficient algorithm that leverages cross-task relations to learn the MTL model from partially annotated data.

Chapter 3

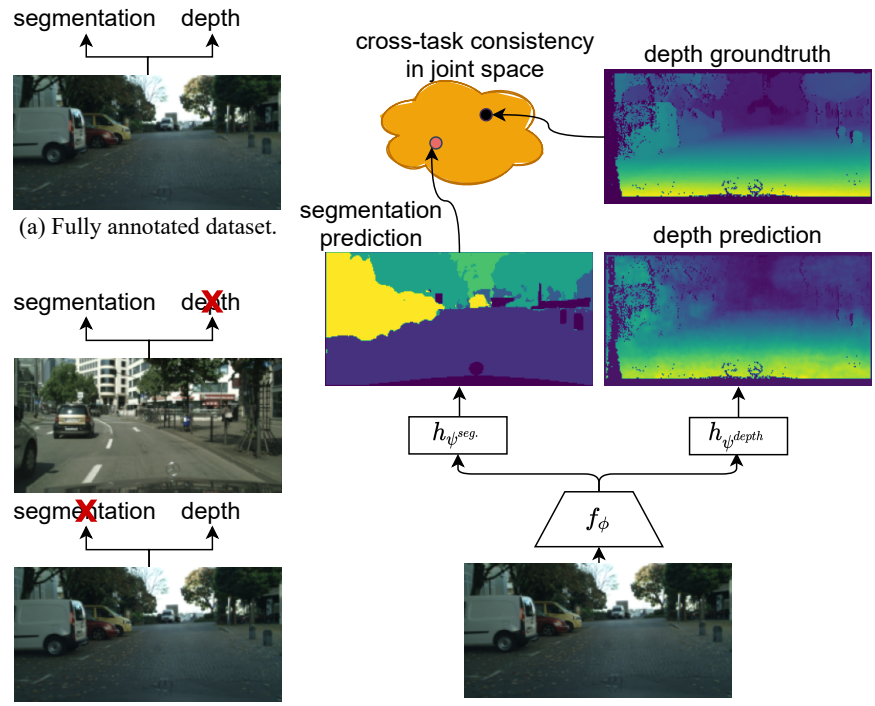
Multi-task Learning from Partially Annotated Data

In this chapter, we look at the dense prediction problems, where a dense prediction task aims to produce pixel-level predictions (*e.g.* semantic and instance segmentation, depth estimation). The recent MTL dense prediction methods rely on fully annotated training data where all training images are labelled for all tasks for training the multi-task learning model from them. To address this, we propose a new and more practical setting called multi-task partially supervised learning and presents an architecture-agnostic method to learn the MTL model from partially annotated data by leveraging cross-task relations.

This chapter begins with an introduction of background in MTL dense prediction from partially annotated data and the main idea of the proposed method in Sec. 3.1. Sec. 3.2 summarizes related literature of multi-task dense prediction and cross-task relations. The multi-task partially supervised learning setting and a novel method for learning MTL from partially annotated data are introduced in Sec. 3.3, followed by experiments in Sec. 3.4 and a conclusion in Sec. 3.5.

3.1 Introduction

Recent MTL dense prediction methods broadly focus on designing MTL architectures (Misra et al., 2016; Ruder et al., 2019; Gao et al., 2019; Liu et al., 2019; Lu et al., 2017; Vandenhende et al., 2020a; Bruggemann et al., 2020; Guo et al., 2020; Bragman et al., 2019; Xu et al., 2018a; Zhang et al., 2019, 2018; Vandenhende et al., 2020b; Zhou et al., 2020) that enable effective sharing of information across tasks and improving



(a) Fully annotated dataset. (b) Partially annotated dataset. (c) Cross-task consistency learning.

Figure 3.1: **Multi-task partially supervised learning.** We look at the problem of learning multiple tasks from partially annotated data (b) where not all the task labels are available for each image, which generalizes over the standard supervised learning (a) where all task labels are available. We propose a MTL method that employs a shared feature extractor (f_ϕ) with task-specific heads (h_ψ) and exploits label correlations between each task pair by mapping them into a *joint pairwise task-space* and penalizing inconsistencies between the provided ground-truth labels and predictions (c).

the MTL optimization (Gong et al., 2019; Kendall et al., 2018; Chen et al., 2018b; Liu et al., 2019; Guo et al., 2018; Sener and Koltun, 2018; Lin et al., 2019; Yu et al., 2020; Chen et al., 2020c; Li and Bilen, 2020) to balance the influence of each task-specific loss function and to prevent interference between the tasks. We refer to Vandenhende et al. (2021) for a more comprehensive review. One common and strong assumption in these works is that each training image has to be labelled for all the tasks. There are two main practical limitations to this assumption. First, curating multi-task image datasets (*e.g.* KITTI (Geiger et al., 2012) and CityScapes (Cordts et al., 2016)) typically involves using multiple sensors to produce ground-truth labels for several tasks and obtaining all the labels for each image requires very accurate synchronization between the sensors, which is a challenging research problem by itself (Voges and Wagner, 2018). Second, imagine a scenario where i) one would like to add a new task to an existing image

dataset which is already annotated for another task and ii) obtaining the ground-truth for the new task requires using a different sensor (*e.g*. depth camera) to the one which is used to capture the original data. In this case, labelling the previously recorded images for the new task will not be possible for many visual scenes (*e.g*. uncontrolled outdoor environments). Such real-world scenarios lead to obtaining partially annotated data and thus ask for algorithms that can learn from such data.

In this chapter, we look at a more realistic and general case of the MTL dense prediction problem where not all the task labels are available for each image and we call this setting *multi-task partially supervised learning*. In particular, we assume that each image is at least labelled for one task and there are at least few images labelled for each task and we would like to learn a multi-task model on them. A naive way of learning from such partial supervision is to train a multi-task model only on the available labels (*i.e*. by setting the weight of the corresponding loss function to 0 for the missing task labels). Though, in this setting, the MTL model is trained on all the images thanks to the parameter sharing across the tasks, it cannot extract the task-specific information from the images for the unlabelled tasks. To this end, one can extend existing single-task semi-supervised learning methods to MTL by penalizing the inconsistent predictions of images over multiple perturbations for the unlabelled tasks (*e.g*. consistency-based methods (Chen et al., 2020d; Liu et al., 2008; Terzopoulos et al., 2019; Latif et al., 2019; Imran et al., 2020)). While this strategy ensures consistent predictions over various perturbations, it does not guarantee consistency across the related tasks.

An orthogonal information that has recently been used in MTL is cross-task relation (Zamir et al., 2020; Lu et al., 2021; Saha et al., 2021) which aims at producing consistent predictions across task pairs. Unfortunately existing methods are not directly applicable for learning from partial supervision, as they require either each training image to be labelled with all the task labels (Zamir et al., 2020; Saha et al., 2021) or cross-task relations that can be analytically derived (Lu et al., 2021). In our setting, there are fewer training images available with ground-truth labels of each task pair and thus it is harder to learn the relationship. In addition, we focus on the general setting where one task label cannot be accurately obtained from another (*e.g*. from semantic segmentation to depth) and hence learning exact mappings between two task labels is not possible.

Motivated by these challenges, we propose a MTL approach that shares a feature extractor between tasks and also learns to relate each task pair in a learned *joint pairwise task-space* (illustrated in Fig. 3.1(c)), which encodes only the shared information between them and does not require the ill-posed problem of recovering labels of one

task from another one. There are two challenges to this goal. First, a naive learning of the joint pairwise task-spaces can lead to trivial mappings that take all predictions to the same point such that each task produces artificially consistent encodings with each other. To this end, we regulate learning of each mapping by penalizing its output to retain high-level information about the input image. Second, the computational cost of modelling each task pair relation can get exponentially expensive with the number of tasks. To address this challenge, we use a single encoder network to learn all the pairwise-task mappings, however, dynamically estimate its weights by conditioning them on the target task pair.

The main contributions of our method are as follows. We propose a new and practical setting for multi-task dense prediction problems and a novel MTL model that penalizes cross-task consistencies between pairs of tasks in joint pairwise task-spaces, each encoding the commonalities between pairs, in a computationally efficient manner. We show that our method can be incorporated to several architectures and significantly outperforms the related baselines in three standard multi-task benchmarks.

3.2 Related Work

3.2.1 Multi-task Semi-supervised Learning

Multi-task Learning (MTL) (Caruana, 1997; Vandenhende et al., 2021; Ruder, 2017; Zhang and Yang, 2017) aims at learning a single model that can infer all desired task outputs given an input. Standard multi-task learning works mainly focus on supervised setting, where each sample in the dataset is annotated for all desired tasks as highlighted in Sec. 2.2.

However, learning multi-task models on fully annotated data would require large-scale labeled data and it is costly to collect sufficient labeled data. Thus few works propose to learn multi-task learning model using semi-supervised learning strategy (Liu et al., 2008; Zhang and Yeung, 2009; Wang et al., 2009; Chen et al., 2020d; Liu et al., 2008; Terzopoulos et al., 2019; Latif et al., 2019; Imran et al., 2020) and they assume that the dataset consists of limited labeled annotated with all tasks labels and a large amount of unlabeled data. Liu *et al*. (Liu et al., 2008) extend single-task semi-supervised learning to multi-task learning by learning a classifier per task jointly under the constraint of a soft-sharing prior imposed over the parameters of the classifiers. In prior works (Chen et al., 2020d; Liu et al., 2008; Terzopoulos et al., 2019; Latif et al.,

2019; Imran et al., 2020), the authors employ a regularization term on the unlabeled samples of each tasks that encourages the model to produce 'consistent' predictions when its inputs are perturbed.

3.2.2 Cross-task Relations

A rich body of work (Liu et al., 2010; Bilen and Vedaldi, 2016; Zamir et al., 2016, 2018; Lu et al., 2021; Zamir et al., 2020; Saha et al., 2021; Zhou et al., 2017; Casser et al., 2019; Wang et al., 2021; Hoyer et al., 2021; Sun et al., 2021) study the relations between tasks in MTL. Most related to ours, Saha et al. (2021) explore the relations between segmentation and depth and propose a better fusion strategy to fuse two task predictions for domain adaptation. Zamir *et al*. (Zamir et al., 2020) study the cross-task consistency learned from groundtruth of all tasks for robust learning, *i.e*. the predictions made for multiple tasks from the same image are not independent, and therefore, are expected to be 'consistent'. Similar to Zamir et al. (2020), Lu *et al*. (Lu et al., 2021) propose to leverage the cross-task consistency between predictions of different tasks on unlabeled data in a mediator dataset when jointly learning multiple models for distributed. To regularize the cross-task consistency, Lu *et al*. (Lu et al., 2021) design multiple consistency losses according to the consistency between adjacent frames in videos, relations between depth and surface normal, etc. In this chapter, we also exploit the cross-task consistency in MTL, however, from partially annotated data where the mapping from one task label to another cannot be analytically derived or exactly learned. To this end, unlike Zamir et al. (2020); Lu et al. (2021), we learn a joint task-space for each task pair rather than measuring consistency in one's task space. Finally, our method learns cross-task in a more computationally efficient way than the prior works (Zamir et al., 2020; Lu et al., 2021) by sharing parameters across different mappings and conditioning its output on the related task-pair.

3.3 Method

3.3.1 Problem setting

Let $\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$ and $\boldsymbol{y}^t \in \mathbb{R}^{O^t\times H\times W}$ denote an $H \times W$ dimensional RGB image and its dense label for task t respectively, where O^t is the number of output channels for task t. Our goal is to learn a function $\hat{y}^t$ for each task t that accurately predicts the ground-truth label $\boldsymbol{y}^t$ of previously unseen images. While such a task-specific function

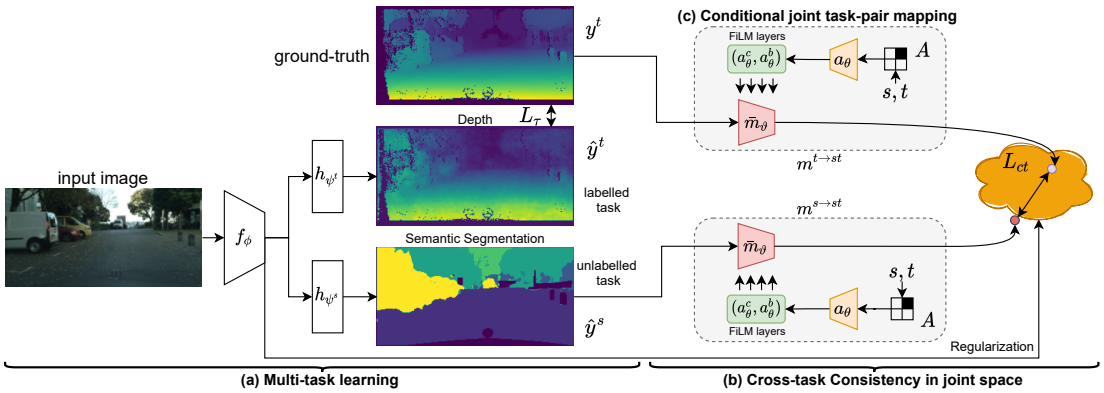


Figure 3.2: Illustration of our method for multi-task partially supervised learning. Given an image, our method uses a shared feature extractor f_ϕ taking in the input image and task-specific decoders (h_{ψ^s} and h_{ψ^t}) to produce predictions for all tasks (a). We compute the supervised loss L_t for labelled task. Besides, we regularize the cross-task consistency L_{ct} between the unlabelled task's prediction $\hat{\boldsymbol{y}}^s$ and the labelled task's ground-truth $\boldsymbol{y}^t$ in a joint space for the unlabelled task (b). To learn the cross-task consistency efficiently, we propose to use a shared mapping function whose output is conditioned on the task-pair (c) and regularize the learning of mapping function using the feature from f_ϕ to prevent trivial solution.

can be learned for each task independently, a more efficient design is to share most of the computations across the tasks via a common feature encoder, convolutional neural network $f_\phi : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{C\times H'\times W'}$ parameterized by ϕ that takes in an image and produces a $H' \times W'$ dimensional feature map with C channels, where typically $H' < H$ and $W' < W$. In this setting, f_ϕ is followed by multiple task-specific decoders $h_{\psi^t} : \mathbb{R}^{C\times H'\times W'} \rightarrow \mathbb{R}^{O^t\times H\times W}$, each with its own task-specific weights ψ^t that decodes the extracted feature to predict the label for the task t, *i.e.* $\hat{y}^t(\boldsymbol{x}) = h_{\psi^t} \circ f_\phi(\boldsymbol{x})$ (Fig. 3.2(a)).

Let $\mathcal{D}$ denote a set of N training images with their corresponding labels for K tasks. Assume that for each training image $\boldsymbol{x}$, we have ground-truth labels available only for some tasks where we use $\mathcal{T}$ and $\mathcal{U}$ to store the indices of labeled and unlabelled tasks respectively, where $|\mathcal{T}| + |\mathcal{U}| = K$, $\mathcal{U} = \varnothing$ indicates all labels available for $\boldsymbol{x}$ and $\mathcal{T} = \varnothing$ indicates no labels available for $\boldsymbol{x}$. In this chapter, we focus on the partially annotated setting, where each image is labelled at least for one task ($|\mathcal{T}| \geq 1$) and there exist at least few images with their ground-truth labels for each task available.

A naive way of learning $\hat{y}^t$ for each task on the partially annotated data $\mathcal{D}$ is to

jointly optimize its parameters on the labelled tasks as following:

$$\min_{\phi,\psi} \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathcal{T}_n|} \sum_{t \in \mathcal{T}_n} L^t(\hat{y}^t(\boldsymbol{x}_n), \boldsymbol{y}_n^t), \tag{3.1}$$

where n is the image index and L^t is the task-specific differentiable loss function. We denote this setting as the (vanilla) MTL. Here, thanks to the parameter sharing through the feature extractor, its task-agnostic weights are learned on all the images. However, the task-specific weights ψ^t are trained only on the labeled images.

A common strategy to exploit such information from unlabeled tasks is to formulate the problem in a semi-supervised learning (SSL) setting. Recent successful SSL techniques (Berthelot et al., 2019; Sohn et al., 2020) focus on learning models that can produce consistent predictions for unlabelled images when its input is perturbed in various ways.

$$\min_{\phi,\psi} \frac{1}{N} \sum_{n=1}^{N} \Big(\frac{1}{|\mathcal{T}_n|} \sum_{t \in \mathcal{T}_n} L^t(\hat{y}^t(\boldsymbol{x}_n), \boldsymbol{y}_n^t) + \frac{1}{|\mathcal{U}_n|} \sum_{t \in \mathcal{U}} L_u(e_r(\hat{y}^t(\boldsymbol{x}_n)), \hat{y}^t(e_r(\boldsymbol{x}_n))) \Big), \tag{3.2}$$

where L_u is the unsupervised loss function and e_r is a geometric transformation (*i.e.* cropping) parameterized by the random variable r (*i.e.* bounding box location). In words, for the unsupervised part, we apply our model to the original input $\boldsymbol{x}$ and also its cropped version $e_r(\boldsymbol{x})$, and then we also crop the prediction corresponding to the original input $e_r(\hat{y}^t(\boldsymbol{x}_n))$ before we measure the difference between two by using L_u. Note that we are aware of more sophisticated task-specific SSL methods for semantic segmentation (Olsson et al., 2021; Mendel et al., 2020), depth estimation (Kuznietsov et al., 2017; Guizilini et al., 2020), however, combining them for multiple tasks, each with different network designs and learning formulations is not trivial and here we focus on one SSL strategy that uses one perturbation type (*i.e.* random cropping) and L_u (*i.e.* mean square error) can be applied to several tasks.

3.3.2 Cross-task consistency learning

While optimizing Eq. (3.2) allows learning both task-agnostic and task-specific weights on the labeled and unlabelled data, it does not leverage cross-task relations, which can be used to further supervise unlabelled tasks. Prior works (Zamir et al., 2020; Lu et al., 2021) define the cross-task relations by a mapping function $m^{s \to t}$ for each task-pair (s,t) which maps the prediction for the source task s to target task t labels. The mapping function proposed by Lu et al. (2021) is analytical based on the assumption that target

task labels can be analytically computed from source labels. While such analytical relations is possible only for certain task pairs, each mapping function in the previous work (Zamir et al., 2020) is parameterized by a deep network and its weights are learned by minimizing $L_{ct}(m^{s\to t}(\boldsymbol{y}^s),\boldsymbol{y}^t)$, where L_{ct} is cross-task function that measures the distance between the mapped source labels and target labels. There are two limitations to this method in our setting. First the training set has a limited number of labelled images for both source and target tasks ($\boldsymbol{y}^s$ and $\boldsymbol{y}^t$). Second, learning such pairwise mappings accurately is not often possible in our case, as the labels of one task can only be partially recovered from another task (*e.g.* semantic segmentation to depth estimation). Note that this ill-posed problem can be solved accurately when strong prior knowledge about the data is available.

To employ cross-task consistency to our setting, we map each task pair (s,t) to a lower-dimensional joint pairwise task-space where only the common features of both tasks are encoded (Fig. 3.2(b)). Formally, each pairwise task-space for (s,t) is defined by a pair of mapping functions, $m_{\vartheta_s^{st}}:\mathbb{R}^{O^s\times H\times W}\to\mathbb{R}^D$ and $m_{\vartheta_t^{st}}:\mathbb{R}^{O^t\times H\times W}\to\mathbb{R}^D$ parameterized by ϑ_s^{st} and ϑ_t^{st} respectively. The cross-task consistency can be incorporated to Eq. (3.1) as following:

$$\begin{aligned}\min_{\phi,\psi,\vartheta}\frac{1}{N}\sum_{n=1}^{N}\Big(&\frac{1}{|\mathcal{T}_n|}\sum_{t\in\mathcal{T}_n}L^t\big(\hat{y}^t(\boldsymbol{x}_n),\boldsymbol{y}_n^t\big)+\\&\frac{1}{|\mathcal{U}_n|}\sum_{s\in\mathcal{U}_n,t\in\mathcal{T}_n}L_{ct}\big(m_{\vartheta_s^{st}}(\hat{y}^s(\boldsymbol{x}_n)),m_{\vartheta_t^{st}}(\boldsymbol{y}_n^t)\big)\Big),\end{aligned}\tag{3.3}$$

where L_{ct} is cosine distance (*i.e.* $L_{ct}(\mathbf{a},\mathbf{b})=1-(\mathbf{a}\cdot\mathbf{b})/(|\mathbf{a}||\mathbf{b}|)$. In words, along with the MTL optimization, Eq. (3.3) minimizes the cosine distance between the embeddings of the unlabelled task prediction $\hat{\boldsymbol{y}}_s$ and the annotated task label $\boldsymbol{y}^t$ in the joint pairwise task space. Here $m_{\vartheta_s^{st}}$ and $m_{\vartheta_t^{st}}$ are not necessarily equal to allow for treating the mapping from predicted and ground-truth labels differently. Note that one can also include the semi-supervised term L_u in Eq. (3.3). However we empirically found that it does not bring any tangible performance gain when used with the cross-task term L_{ct}.

There are two challenges to learn non-trivial pairwise mapping functions in a computationally efficient way. First the number of pairwise mappings to learn quadratically grows with the number of tasks. Although the mapping functions are only used in training, it can still be computationally expensive to train many of them jointly. In addition, learning an accurate mapping for each task-pair can be challenging in case of limited labels. Second, the mapping functions can simply learn a trivial solution such that each task is mapped to a fixed point (*e.g.* zero vector) in the shared space.

Conditional joint task-pair mapping. To address the first challenge, as shown in Fig. 3.2(c), we propose to use a task-agnostic mapping function $\bar{m}_\vartheta$ with one set of parameters ϑ whose output is conditioned both on the input task (s or t) and task-pair (s,t) through an auxiliary network (a_θ). Concretely, let A denote a variable that includes the input task (s or t) and target pair (s,t) for a pairwise mapping which in practice we encode with an asymmetric $K \times K$ dimensional matrix by setting the corresponding entry to 1 (*i.e.* $A[s,t] = 1$ or $A[t,s] = 1$) and the other entries to 0. Note that the diagonal entries are always zero, as we do not define any self-task relation. Let $\bar{m}_\vartheta$ be a multi-layer network and $\boldsymbol{h}_i$ denote a M channel feature map of its i-th layer for which the auxiliary network a_θ, parameterized by θ, takes in A and outputs two M-dimensional vectors $a^c_{\theta,i}$ and $a^b_{\theta,i}$. These vectors are applied to transform the feature map $\boldsymbol{h}_i$ in a similar way to the one proposed by Perez et al. (2018) as following:

$$\boldsymbol{h}_i \leftarrow a^c_{\theta,i}(A) \odot \boldsymbol{h}_i + a^b_{\theta,i}(A)$$

where $\odot$ denotes a Hadamard product. In other words, the auxiliary network alters the output of the task-agnostic mapping function $\bar{m}_\vartheta$ based on A. For brevity, we denote the conditional mapping from s to (s,t) as $m^{s\to st}$ which is a function of $\bar{m}_\vartheta$ and a_θ and hence parameterized with ϑ and θ.

We implement each a^c_i and a^b_i as an one layer fully-connected network. Hence, given the light-weight auxiliary network, the computational load for computing the conditional mapping function, in practice, does not vary with the number of task-pairs. Finally, as the dimensionality of each task label vary – *e.g.* while O^t is 1 for depth estimation and O^t equals to number of categories in semantic segmentation –, we use task-specific input layers and pass each prediction to the corresponding one before feeding it to the joint pairwise task mapping. In the formulation, we include these layers in our mapping $\bar{m}_\vartheta$ and explain their implementation details in Sec. 3.4.

Regularizing mapping function. To avoid learning trivial mappings, we propose a regularization strategy (Fig. 3.2) that encourages the mapping to retain high-level information about the input image. To this end, we penalize the distance between the output of the mapping function and a feature vector that is extracted from the input image. In particular, we use the output of the task-agnostic feature extractor $f_\phi(\boldsymbol{x})$ in

the regularization. Now we can add the regularizer to the formulation in Eq. (3.3):

$$\begin{aligned}\min_{\phi,\psi,\vartheta,\theta} \frac{1}{N}\sum_{n=1}^{N}\Big(&\frac{1}{|\mathcal{T}_n|}\sum_{t\in\mathcal{T}_n} L^t\big(\hat{y}^t(\boldsymbol{x}_n),\boldsymbol{y}_n^t\big)+\\ &\frac{1}{|\mathcal{U}_n|}\sum_{s\in\mathcal{U}_n,t\in\mathcal{T}_n} L_{ct}\big(m^{s\to st}(\hat{y}^s(\boldsymbol{x}_n)),m^{t\to st}(\boldsymbol{y}_n^t)\big)\\ &+R(f_\phi(\boldsymbol{x}_n),m^{s\to st}(\hat{y}_s(\boldsymbol{x}_n)))\\ &+R(f_\phi(\boldsymbol{x}_n),m^{t\to st}(\boldsymbol{y}_n^t))\Big),\end{aligned} \tag{3.4}$$

where $f_\phi(\boldsymbol{x})$ is the feature from feature encoder f_ϕ, R is the loss function and we use the cosine similarity loss for R in this chapter.

Alternative mapping strategies. Here we discuss two different mapping strategies to exploit cross-task consistency proposed by Zamir et al. (2020) and their adoption to our setting. As both require learning a mapping from one task's groundtruth label to another one and we have either no or few images with both groundtruth labels, here we approximate them by learning mappings from prediction of one task to another task's groundtruth. In the first case, one can substitute our cross-consistency loss and regularization terms with $L_{ct}(m^{s\to t}(\hat{y}^s(\boldsymbol{x})),\boldsymbol{y}^t)$ in Eq. (3.4), which denote as *Direct-Map*. In the second case, we replace our terms with $L_{ct}(m^{s\to t}(\hat{y}^s(\boldsymbol{x})),m^{s\to t}(\boldsymbol{y}^s))$ that maps both the groundtruth $\boldsymbol{y}^s$ and predicted labels $\hat{\boldsymbol{y}}^s$ and minimize their distance in task t's label space. We denote this setting as *Perceptual-Map* and compare our method to them in Sec. 3.4.

Alternative loss and regularization strategies. Alternatively, our cross-consistency loss and regularization terms can be replaced with another loss function only that does not allow for learning of trivial mappings. One such loss function is contrastive loss where one can define the predictions for two tasks on the same image as a positive pair (*i.e.* $m^{s\to st}(\hat{y}^s(\boldsymbol{x}_i))$ and $m^{t\to st}(\boldsymbol{y}_i^t)$) and on different images as a negative pair (*i.e.* $m^{s\to st}(\hat{y}^s(\boldsymbol{x}_j))$ and $m^{t\to st}(\boldsymbol{y}_i^t)$), and penalize when the distance from the positive one is bigger than the negative one. We denote this setting as *Contrastive-Loss*. Another method which also employs positive and negative pairs involves using a discriminator network. The discriminator (a convolutional neural network) takes in positive and negative pairs and predicts their binary labels, while the parameters of the MTL network and mapping functions are alternatively optimized. We denote this setting as *Discriminator-Loss* and compare our approach to the alternative methods in Sec. 3.4.

3.4 Experiments

Datasets. We evaluate all methods on three standard dense prediction benchmarks, Cityscapes (Cordts et al., 2016), NYU-V2 (Silberman et al., 2012), and PASCAL (Everingham et al., 2010). Cityscapes (Cordts et al., 2016) consists of street-view images, which are labeled for two tasks: 7-class semantic segmentation[1] and depth estimation. We resize the images to 128×256 to speed up the training as Liu et al. (2019). NYU-V2 (Silberman et al., 2012) contains RGB-D indoor scene images, where we evaluate performances on 3 tasks, including 13-class semantic segmentation, depth estimation, and surface normals estimation. We use the true depth data recorded by the Microsoft Kinect and surface normals provided by Eigen and Fergus (2015) for depth estimation and surface normal estimation. All images are resized to 288×384 resolution as in MTAN (Liu et al., 2019). PASCAL (Everingham et al., 2010) is a commonly used benchmark for dense prediction tasks. We use the data splits from PASCAL-Context (Chen et al., 2014) which has annotations for semantic segmentation, human part segmentation and semantic edge detection. Additionally, Following Vandenhende et al. (2021), we also consider the tasks of surface normals prediction and saliency detection and use the annotations provided by Vandenhende et al. (2021).

Experimental setting. For the evaluation of multi-task models learned in different partial label regimes, we design two settings: *(i) random* setting where, we randomly select and keep labels for at least 1 and at most $K-1$ tasks where K is the number of tasks, *(ii) one* label setting, where we randomly select and keep label only for 1 task for each training image.

In Cityscapes and NYU-v2, we follow the training and evaluation protocol in the prior work (Liu et al., 2019) and we use the SegNet (Badrinarayanan et al., 2017) as the MTL backbone for all methods. As Liu et al. (2019), we use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation in Cityscapes, and cosine similarity loss for surface normal estimation in NYU-v2. We use the exactly same hyper-parameters including learning rate, optimizer and also the same evaluation metrics, mean intersection over union (mIoU), absolute error (aErr) and mean error (mErr) in the predicted angles to evaluate the semantic segmentation, depth estimation and surface normals estimation task, respectively in MTAN (Liu et al., 2019). We use the encoder of SegNet for the joint pairwise task mapping ($\bar{m}_{\vartheta}$) and one convolutional layer as

[1]The original version of Cityscapes provides labels for 19-class semantic segmentation. We follow the evaluation protocol in the prior work (Liu et al., 2019), we use labels of 7-class semantic segmentation. Please refer to Liu et al. (2019) for more details.

task-specific input layer in $\bar{m}_{\vartheta}$. For `Direct-Map` and `Perceptual-Map`, as Zamir et al. (2020) we use the whole SegNet as the cross-task mapping functions.

In PASCAL, we follow the training, evaluation protocol and implementation in the prior work (Vandenhende et al., 2021) and employ the ResNet-18 (He et al., 2016) as the encoder shared across all tasks and Atrous Spatial Pyramid Pooling (ASPP) (Chen et al., 2018a) module as task-specific heads. We use the same hyper-parameters, *e.g.* learning rate, augmentation, loss functions, loss weights as Vandenhende et al. (2021). For evaluation metrics, we use the optimal dataset F-measure (odsF) (Martin et al., 2004) for edge detection, the standard mean intersection over union (mIoU) for semantic segmentation, human part segmentation and saliency estimation are evaluated, mean error (mErr) for surface normals. We modify the ResNet-18 to have task-specific input layers (one convolutional layer for each task) before the residual blocks as the mapping function $\bar{m}_{\vartheta}$ in our method. We refer to Appendix B.1 for more details.

3.4.1 Results

We compare our method to multiple baselines including the vanilla MTL Supervised Learning (SL) baseline in Eq. (3.1) on both all the labels and partial labels in Eq. (3.1), and the MTL Semi-supervised Learning (SSL) in Eq. (3.2), also variations of our method with Direct-Map, Perceptual-Map, Contrastive-Loss and Discriminator-Loss as described in Sec. 3.3. We use uniform weights for task-specific losses for all, unless stated otherwise. See Appendix B.2.1 for more results.

Results on Cityscapes. We first compare our method to the baselines on Cityscapes in Tab. 3.1 for only *one* label setting as there are two tasks in total. The results of the MTL model learned with SL when all task labels are available for training to serve as a strong baseline. In the partial label setting (one task label per image), the performance of the SL baseline drops substantially compared to its performance in full supervision setting. While the SSL baseline, by extracting task-specific information from unlabelled tasks, improves over SL, further improvements are obtained by exploiting cross-task consistency in various ways except Discriminator-Loss. The methods that learn mappings from one task to another one (Perceptual-Map and Direct-Map) surprisingly perform better than the ones learning joint space mapping functions (Contrastive-Loss and Discriminator-Loss), possibly due to insufficient number of negative samples. Due to the same reason, we exclude the further comparisons to Contrastive-Loss and Discriminator-Loss in NYU-v2 and PASCAL and include them in Appendix B.2.1.

Finally, the best results are obtained with our method that can exploit cross-task relations more efficiently through joint pairwise task mappings with the proposed regularization. Interestingly, our method also outperforms the SL baseline that has access to all the task labels, showing the potential information in the cross-task relations.

# label	Method	Seg. (IoU) ↑	Depth (aErr) ↓
full	Supervised Learning	73.36	0.0165
one	Supervised Learning	69.50	0.0186
	Semi-supervised Learning	71.67	0.0178
	Perceptual-Map	72.82	0.0169
	Direct-Map	72.33	0.0179
	Contrastive-Loss	71.79	0.0183
	Discriminator-Loss	68.94	0.0208
	Ours	**74.90**	**0.0161**

Table 3.1: Multi-task learning results on Cityscapes. 'one' indicates each image is randomly annotated with one task label.

Results on NYU-v2. We then evaluate our method along with the baselines on NYU-v2 in the *random* and *one* label settings in Tab. 3.2. While we observe a similar trend across different methods, overall the performances are lower in this benchmark possibly due to fewer training images than CityScapes. As expected, the performance in the random-label setting is better than the one in one-label setting, as there are more labels available in the former. While the best results are obtained with SL trained on the full supervision, our method obtains the best performance among the partially supervised methods. Here SSL improves over SL trained on the partial labels and cross-task consistency is beneficial except for Direct-Map in the one label setting, possibly because the dataset is too small to learn accurate mappings between two tasks, while our method is more data-efficient and more successful to exploit the cross-task relations.

Results on PASCAL-Context. We evaluate all methods on PASCAL-Context, in both label settings, which contains a wider variety of tasks than the previous benchmarks and report the results in Tab. 3.3. As the required number of pairwise mappings for Direct-Map and Perceptual-Map grows quadratically (20 mappings for 5 tasks), we omit these two due to their high computational cost and compare our method only to SL and SSL baselines. We see that the SSL baseline improves the performance over SL

# labels	Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓
full	Supervised learning	36.95	0.5510	29.51
random	Supervised Learning	27.05	0.6624	33.58
	Semi-supervised Learning	29.50	0.6224	33.31
	Perceptual-Map	32.20	0.6037	32.07
	Direct-Map	29.17	0.6128	33.63
	Ours	**34.26**	**0.5787**	**31.06**
one	Supervised Learning	25.75	0.6511	33.73
	Semi-supervised Learning	27.52	0.6499	33.58
	Perceptual-Map	26.94	0.6342	34.30
	Direct-Map	19.98	0.6960	37.56
	Ours	**30.36**	**0.6088**	**32.08**

Table 3.2: Multi-task learning results on NYU-v2. 'random' indicates each image is annotated with a random number of task labels and 'one' means each image is randomly annotated with one task.

in the random-label setting, however, it performs worse than the SL in one label setting, when there are 60% less labels. Again, by exploiting task relations, our method obtains better or comparable results to SSL, while the gains achieved over SL and SSL are more significant in the low label regime (one-label). Interestingly, SSL and our method obtain comparable results in the random-label setting which suggests that relations across tasks are less informative than the ones in CityScape and NYUv2.

# labels	Method	Seg. (IoU) ↑	H. Parts (IoU) ↑	Norm. (mErr) ↓	Sal. (IoU) ↑	Edge (odsF) ↑
full	Supervised Learning	63.9	58.9	15.1	65.4	69.4
random	Supervised Learning	58.4	55.3	16.0	63.9	**67.8**
	Semi-supervised Learning	**59.0**	**55.8**	**15.9**	**64.0**	66.9
	Ours	**59.0**	55.6	**15.9**	**64.0**	**67.8**
one	Supervised Learning	48.0	55.6	17.2	61.5	64.6
	Semi-supervised Learning	45.0	54.0	**16.9**	**61.7**	62.4
	Ours	**49.5**	**55.8**	17.0	**61.7**	**65.1**

Table 3.3: Multi-task learning results on PASCAL. 'random' indicates each image is annotated with a random number of task labels and 'one' means each image is randomly annotated with one task.

3.4.2 Further results

Learning from partial and imbalanced task labels. So far, we considered the partially annotated setting where the number of labels for each task is similar. We further evaluate all methods in an imbalanced partially supervised setting in Cityscapes, where we assume the ratio of labels for each task are imbalanced, *e.g*. we randomly sample 90% of images to be labeled for semantic segmentation and only 10% images having labels for depth and we denote this setting by the label ratio between segmentation and depth (Seg.:Depth = 9:1). The opposite case (Seg.:Depth = 1:9) is also considered.

#labels	Method	Seg. (IoU) ↑	Depth (aErr) ↓
full	Supervised Learning	73.36	0.0165
1:9	Supervised Learning	63.37	0.0161
	Semi-supervised Learning	64.40	0.0179
	Perceptual-Map	68.84	0.0141
	Direct-Map	67.04	0.0153
	Ours	**71.89**	**0.0131**
9:1	Supervised learning	72.77	0.0250
	Semi-supervised Learning	72.97	0.0395
	Perceptual-Map	73.36	0.0237
	Direct-Map	73.13	0.0288
	Ours	**74.23**	**0.0235**

Table 3.4: Multi-task learning results on Cityscapes. '#label' indicates the number ratio of labels for segmentation and depth, *e.g*. '1:9' means we have 10% of images annotated with segmentation labels and 90% of images have depth groundtruth.

We report the results in Tab. 3.4. The performance of supervised learning (SL) on the task with partial labels drops significantly. Though SSL improves the performance on segmentation, its performance on depth drops in both cases. In contrast to SL and SSL, our method and Perceptual-Map obtain better results on all tasks in both settings by learning cross-task consistency while our method obtains the best performance by joint space mapping. This demonstrates that our model can successfully learn cross-task relations from unbalanced labels thanks to its task-agnostic mapping function which can share parameters across multiple task pairs.

Cross-task consistency learning with full supervision. Our method can also be applied to the fully-supervised learning setting where all task labels are available for

each sample by mapping one task's prediction and another task's ground-truth to the joint space and measuring cross-task consistency in the joint space. We applied our method to NYU-v2 and compare it with the single task learning (STL) networks, vanilla MTL baseline, recent multi-task learning methods, *i.e*. MTAN (Liu et al., 2019), X-task (Zamir et al., 2020), and several methods focusing on loss weighting strategies, *i.e*. Uncertainty (Kendall et al., 2018), GradNorm (Chen et al., 2018b), MGDA (Sener and Koltun, 2018) and DWA (Liu et al., 2019) in Tab. 3.5.

Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓
STL	37.45	0.6079	25.94
MTL	36.95	0.5510	29.51
MTAN (Liu et al., 2019)	39.39	0.5696	28.89
X-task (Zamir et al., 2020)	38.91	0.5342	29.94
Uncertainty (Kendall et al., 2018)	36.46	0.5376	27.58
GradNorm (Chen et al., 2018b)	37.19	0.5775	28.51
MGDA (Sener and Koltun, 2018)	38.65	0.5572	28.89
DWA (Liu et al., 2019)	36.46	0.5429	29.45
Ours	41.00	0.5148	28.58
Ours + Uncertainty	**41.09**	**0.5090**	**26.78**

Table 3.5: Multi-task fully-supervised learning results on NYU-v2. 'STL' indicates standard single-task learning and 'MTL' means the standard multi-task learning network.

MTL, MTAN, X-task and Ours are trained with uniform loss weights. We see that our method (Ours) performs better than the other methods with uniform loss weights, *e.g*. MTAN and X-task, where X-task regularizes cross-task consistency by learning perceptual loss with pretrained cross-task mapping functions. This shows that cross-task consistency is informative even in the fully supervised case and our method is more effective for learning cross-task consistency. Compared to recent loss weighting strategies, our method (Ours) obtains better performance on segmentation and depth estimation than other methods while slightly worse on normal estimation compared with GradNorm and Uncertainty. This is because the loss weighting strategies enable a more balanced optimization of the multi-task learning model than uniformly loss weighting. Thus when we incorporate the loss weighing strategy of Uncertainty (Kendall et al., 2018) to our method, *i.e*. (Ours + Uncertainty), our method obtains further improvement and outperforms both GradNorm and Uncertainty.

3.4.3 Ablation study

Here, we conduct an ablation study to evaluate the effect of task-pair conditional mapping function and the regularization in Eq. (3.4). To this end, we report results of our method without task-pair condition network (a_θ), denoted as 'Ours (w/o cond)' where we use a single mapping ($\bar{m}_\vartheta$) for all task pairs, and also our method without the regularization in Eq. (3.4), denoted as 'Ours (w/o reg)' in Tab. 3.6. First our full model outperforms both Ours (w/o cond) and Ours (w/o reg) which shows that both the components are beneficial. Ours (w/o cond) which employs the same mapping for all the task pairs still achieves better performance than the SL baseline. Surprisingly, even after removing the regularization, despite the performance drop, the pairwise mappings can still be regulated with a lower learning rate to avoid learning trivial mappings and it still outperforms the SL baseline.

# labels	Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓
random	Supervised Learning	27.05	0.6624	33.58
	Ours (w/o cond)	34.13	0.5968	31.65
	Ours (w/o reg)	33.87	0.5887	31.24
	Ours	**34.26**	**0.5787**	**31.06**
one	Supervised Learning	25.75	0.6511	33.73
	Ours (w/o cond)	29.19	0.6181	32.62
	Ours (w/o reg)	28.36	0.6407	32.92
	Ours	**30.36**	**0.6088**	**32.08**

Table 3.6: Ablation study on NYU-v2. 'cond' indicates whether using conditional mapping function. 'reg' indicates whether we use regularization in Eq. (3.4).

3.4.4 Qualitative results

Here, we present some qualitative results and refer to Appendix B.2.2 for more results.

Mapped outputs. Here, we visualize the intermediate feature maps of $m^{s\to st}$ and $m^{t\to st}$ for one example in NYU-v2 in Fig. 3.3 where s and t correspond to segmentation and surface normal estimation respectively. We observe that the functions map both task labels to a joint pairwise space where the common information is around object boundaries, which in turn enables the model to produce more accurate predictions for both tasks.

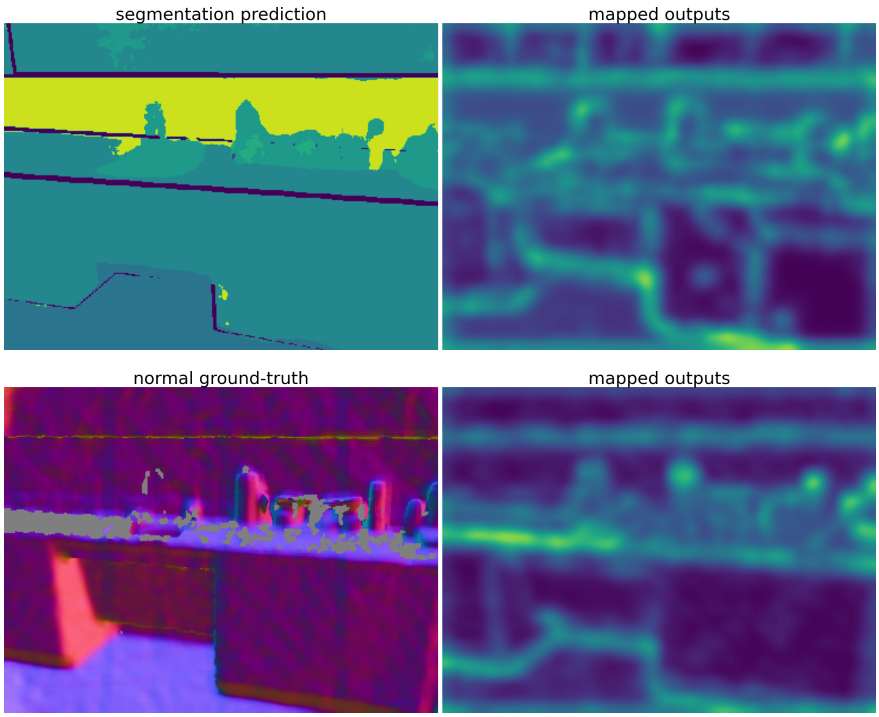


Figure 3.3: Intermediate feature map of the mapping function of the task-pair (segmentation to surface normal) of one example in NYU-v2. The first column shows the prediction or ground-truth and the second column present the corresponding mapped feature map (output of the mapping function's last second layer).

Predictions. Finally we show qualitative comparisons between our method, SL and SSL baselines on NYU-v2 in Fig. 3.4. We can see that our method produces more accurate predictions by leveraging cross-task consistency.

3.5 Conclusion and Limitations

In this chapter, we show that cross-task relations are crucial to learn multi-task dense prediction problems from partially annotated data in several benchmarks. We present a model agnostic method that learns relations between task pairs in joint latent spaces through mapping functions conditioned on the task pair in a computationally efficient way and also avoids learning trivial mappings with a regularization strategy.

Our method has limitations too. Despite the efficient learning of cross-task relations through a conditioned network, modeling cross-task relations for all task pairs may not be required. Thus it would be desirable to automatically identify which tasks are closely related and only learn such cross-task relations.

This chapter improves the capacity of learning a deep network from partial super-

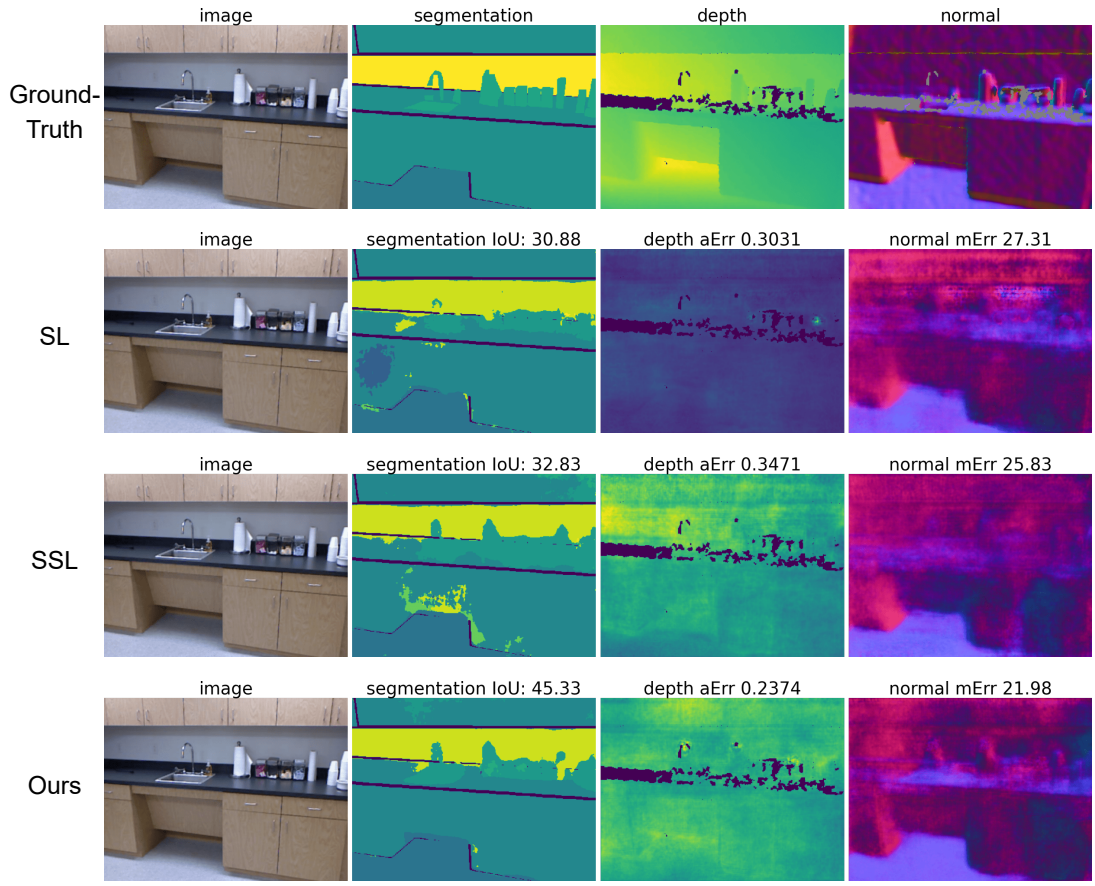


Figure 3.4: **Qualitative results on NYU-v2.** The fist column shows the RGB image, the second column plots the ground-truth or predictions with the IoU (↑) score of all methods for semantic segmentation, the third column presents the ground-truth or predictions with the absolute error (↓), and we show the prediction of surface normal with mean error (↓) in the last column.

vision for multiple tasks by utilizing cross-task relations. The next chapter aims at improving the ability of learning new domains and tasks from few (labelled) data, known as cross-domain few-shot learning, in which models are learned in a large training set and adapted for previously unseen domains and tasks with few labelled samples.

Chapter 4

Cross-domain Few-shot Classification

A fundamental shortcoming of deep neural networks is their limited ability to learn new concepts from small data. Cross-domain few-shot classification aims to learn a classifier from previously unseen classes and domains with few labeled samples. Recent approaches broadly solve this problem by parameterizing their few-shot classifiers with task-agnostic and task-specific weights where the former is typically learned on a large training set and the latter is dynamically predicted through an auxiliary network conditioned on a small support set. For learning task-agnostic weights, recent methods use various adaptation strategies for aligning their visual representations to new domains or select the relevant ones from multiple domain-specific feature extractors. This chapter proposes to learn well generalized representations by training a single task-agnostic network from multiple domains using the URL method introduced in Chapter 2. When learning task-specific weights, through a systematic analysis of various task adaptation strategies, this chapter shows that task-specific weights through parametric adapters in matrix form with residual connections to multiple intermediate layers of a task-agnostic backbone network significantly improves the performance of the state-of-the-art models in the Meta-Dataset benchmark with minor additional cost.

The chapter introduces challenges and approaches in few-shot learning Secs. 4.1 and 4.2 respectively. Sec. 4.3 explains how to learn well-generalized features through a task-agnostic network from multiple domains for cross-domain few-shot learning and how to efficiently adapt the task-agnostic network with task-specific weights for previously unseen domains and tasks with few samples. The experiments in Sec. 4.4 verify the effectiveness of two proposed methods on MetaDataset under different settings and systematically study and analyze various task adaptation strategies for cross-domain few-shot learning which has not been explored before.

4.1 Introduction

As deep neural networks progress to dramatically improve results in most of standard computer vision tasks/problems, there is a growing community interest for more ambitious goals. One of them is to improve the data efficiency of the standard supervised methods that rely on large amounts of expensive and time-consuming hand-labeled data. Just like the human intelligence is capable of learning concepts from few labeled samples, *few-shot learning* (Lake et al., 2011; Miller et al., 2000) is inspired from this limitation and aims at adapting a classifier to accommodate new classes not seen in training, given a few labeled samples from these classes (Fig. 4.1). In particular, the standard setting for learning few-shot classifiers involves two stages: (i) learning a model, typically from a large training set, (ii) adapting this model to learn new classes from a given small support set. These two stages are called meta-training and meta-testing respectively. The adapted model is finally evaluated on a query set where the task is to assign each query sample to one of the classes in the support set.

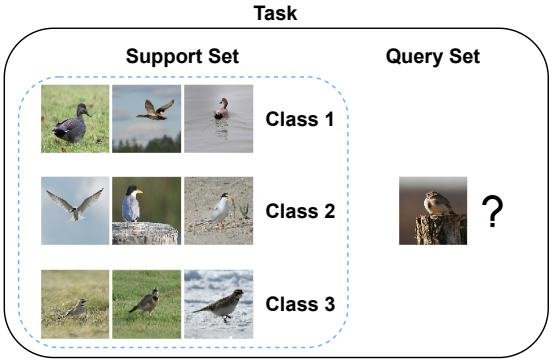


Figure 4.1: **Few-shot Learning** aims at learning a classifier for previously unseen tasks with few labeled samples (support set) such that the classifier accurately classifies the query samples.

Early methods (Vinyals et al., 2016; Ravi and Larochelle, 2016; Finn et al., 2017; Oreshkin et al., 2018; Rusu et al., 2020; Snell et al., 2017) pose the few-shot classification problem in a learning-to-learn formulation by training a deep network over a distribution of related tasks, which are sampled from the training set, and transfer this experience to improve its performance for learning new classes. Concretely, Vinyals *et al*. (Vinyals et al., 2016) learn a feature encoder that is conditioned on the support set in meta-training and does not require any further training in meta-test thanks to its non-parametric classifier. Ravi and Larochelle (Ravi and Larochelle, 2016) take the idea of learning a feature encoder in meta-train further by also learning an update

rule through an LSTM that produces the updates for a classifier in meta-test. Finn *et al*. (Finn et al., 2017) pose the task as a meta-learning problem and learn the parameters of a deep network in meta-training such that a network initialized with the learned parameters can be efficiently finetuned on a new task. We refer to Wang et al. (2020); Hospedales et al. (2020) for comprehensive review of early works.

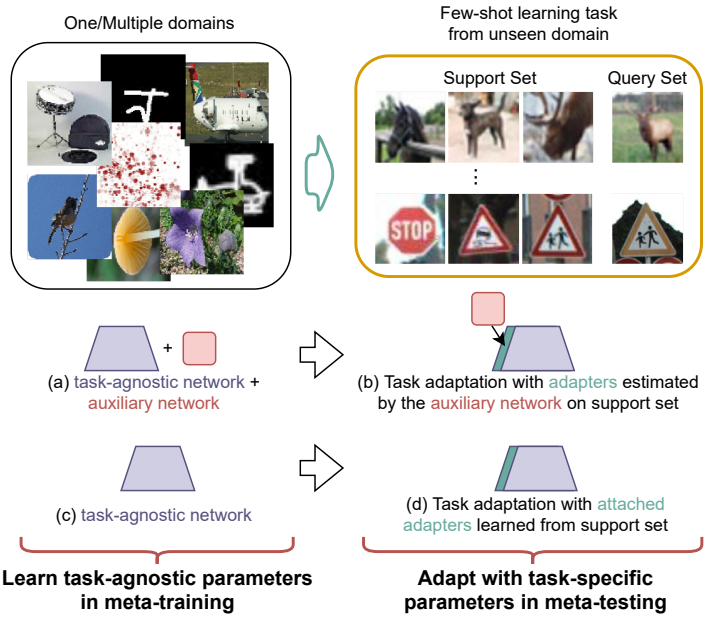


Figure 4.2: **Cross-domain Few-shot Learning** considers to learn a model from one or multiple domains to generalize to unseen domains with few samples. Prior works often learn a task-agnostic model with an auxiliary network during meta-training (a) and a set of adapters are generated by the auxiliary network to adapt to the given support set (b). While in this chapter, we propose to attach adapters directly to a pretrained task-agnostic model (c), which can be estimated from scratch during meta-testing (d). We also propose different architecture topologies of adapters and their efficient approximations.

Despite the significant progress, the scope of the early methods has been limited to a restrictive setting where training and test samples come from a single domain (or data distribution) such as Omniglot (Lake et al., 2015), miniImageNet (Vinyals et al., 2016) and tieredImageNet (Ren et al., 2018). They perform poorly in the more challenging cross-domain few-shot tasks, where test data is sampled from an unknown or previously unseen domain (Triantafillou et al., 2020). This setting poses an additional learning challenge, not only requires leveraging the limited information from the small support set for learning the target task but also *selectively transferring relevant knowledge* from previously seen domains to the target task.

Broadly, recent approaches address this challenge by parameterizing deep networks with a large set of task-agnostic and a small set of task-specific weights that encode generic representations valid for multiple tasks and private representations are specific to the target task respectively. While the task-agnostic weights are learned over multiple tasks, typically, from a large dataset in meta-training, the task-specific weights are estimated from a given small support set (*e.g.* 5 images per category) (Requeima et al., 2019; Bateni et al., 2020b; Lee et al., 2019; Dvornik et al., 2020; Liu et al., 2021b; Li et al., 2021b; Triantafillou et al., 2021).

In the literature, the task-agnostic weights are used to parameterize a single network that is trained on large data from one domain (Requeima et al., 2019; Bateni et al., 2020b; Doersch et al., 2020), or to be distributed over multiple networks, each trained on a different domain (Dvornik et al., 2020; Liu et al., 2021b; Triantafillou et al., 2021) [1]. Despite good performance obtained by Dvornik et al. (2020); Liu et al. (2021b); Triantafillou et al. (2021) that learn task-agnostic parameters distributed over multiple networks from multiple domains, they are computationally expensive and require multiple forward passes through multiple networks during inference time.

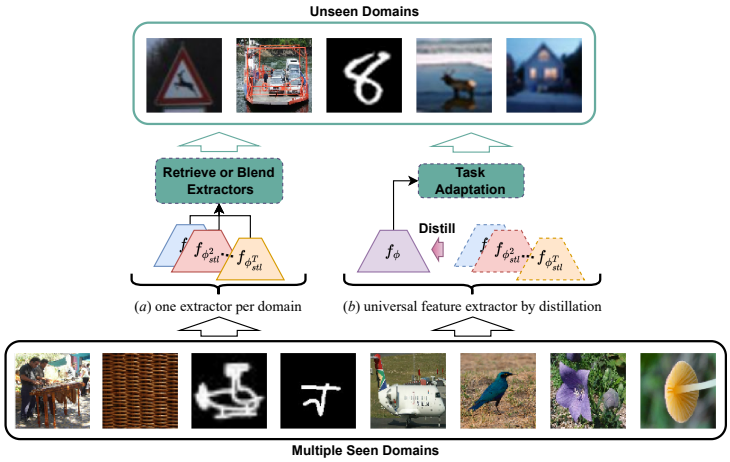


Figure 4.3: **Task-agnostic weight learning**. Unlike the previous methods (Dvornik et al., 2020; Liu et al., 2021b) (illustrated in (a)) that learn T feature extractors $\{f_{\phi^t_{stl}}\}_t^T$, one for each domain, and retrieve or combine their features for the target task during meta-test stage, our method (illustrated in (b)) learns a single universal feature extractor f_ϕ that is distilled from from multiple feature extractors $\{f_{\phi^t_{stl}}\}_t^T$. In meta-test stage, we show that the universal representations can be adapted to unseen domains.

To this end, in this chapter, we propose an efficient and high performance few-shot

[1] Note that the task-agnostic weights can also be finetuned on the target task (*e.g.* Chen et al. (2020b); Dhillon et al. (2020)).

method, called *URL* based on *U*niversal *R*epresentation *L*earning as introduced in Chapter 2 to learn a single task-agnostic network on multiple domains. Like Dvornik et al. (2020); Liu et al. (2021b), our method builds on multi-domain representations that are learned in an offline stage. However, we learn a single set of universal representations (a single feature extractor) over multiple domains which has a fixed computational cost regardless of the number of domains at inference unlike them. Similar to the adaptation based techniques (Bateni et al., 2020b; Requeima et al., 2019). In particular, we propose to *distill* knowledge from multiple domains to a single model, which can efficiently leverage useful information from multiple diverse domains. Learning multi-domain representations is a challenging task and requires to leverage commonalities in the domains while minimizing interference (negative transfer (Chen et al., 2018b; Rebuffi et al., 2017a; Yu et al., 2020)) between them. To mitigate this, we align the intermediate representations of our multi-domain network with the ones of the domain-specific networks after carefully aligning each space by using small task-specific adapters and Centered Kernel Alignment (CKA) (Kornblith et al., 2019) as in Sec. 2.3.

Given the task-agnostic weights, the task-specific weights are utilized to parameterize a linear classifier (Lee et al., 2019), and an ensemble of classifiers at each layer of a deep neural network (Adler et al., 2020). Recently, inspired from Perez et al. (2018), *task-specific adapters* (Requeima et al., 2019; Bateni et al., 2020b), small capacity transformations that are applied to multiple layers of a deep network, have been successfully used to steer the few-shot classifiers to new tasks and domains. Their weights are often estimated dynamically through an auxiliary network conditioned on the support set (Requeima et al., 2019; Bateni et al., 2020b; Liu et al., 2021b; Triantafillou et al., 2021) (see Fig. 4.2.(a,b)), in a similar spirit to Bertinetto et al. (2016); Jia et al. (2016). As the auxiliary network is trained on multiple tasks in meta-training, the premise of estimating the task-specific adapter weights with it is based on the principle of transfer learning such that it can transfer the knowledge from the previous tasks to better estimate them for unseen tasks. However, learning an accurate auxiliary network is a challenging task due to two reasons. First, it has to generalize to previously unseen tasks and especially to significantly different unseen domains. Second, learning to predict high-dimensional weights where each corresponds to a dimension of a highly nonlinear feature space is a difficult learning problem too.

Motivated by this shortcoming, as shown in Fig. 4.2, we propose to employ a set of light-weight task-specific adapters along with the task-agnostic weights for adapting the few-shot classifier to the tasks from unseen domains. Unlike the prior work, we

learn the weights of these adapters from scratch by directly optimizing them on a small support set (see Fig. 4.2.(c,d)). We also propose to attach a pre-classifier feature mapping on top of the feature extractor to transform the adapted features to a more discriminative space for the tasks from unseen domains. Moreover, we systematically study various combinations of several design choices for task-specific adaptation, which have not been explored before, including adapter connection types (serial or residual), parameterizations (matrix and its decomposed variations, channelwise operations) and estimation of task-specific parameters. Extensive experiments demonstrate that attaching parametric adapters in matrix form to convolutional layers with residual connections significantly boosts the state-of-the-art performance in most domains, especially resulting in superior performance in unseen domains on Meta-Dataset with negligible increase in computations.

4.2 Related Work

Few-shot learning is an extensively studied problem in computer vision. For the methods that focus on single-domain few-shot tasks, we refer to Wang et al. (2020); Hospedales et al. (2020) for comprehensive review. We focus on the more challenging multi-domain few-shot tasks on Meta-Dataset (Triantafillou et al., 2020). The previous work in this problem can be broadly grouped into two groups, learning well-generalized task-agnostic weights or efficiently learning task-specific weights with few labelled samples for previously unseen domains and tasks.

Task-agnostic learning for meta-training. CNAPS (Requeima et al., 2019) learns a task encoder to modulate the task-agnostic feature extractor with FiLM layers (Perez et al., 2018) and also adapts the classifier with new samples. Simple CNAPS (Requeima et al., 2019) improves CNAPS by replacing its classifier with a non-parametric classifier based on Mahalanobis distance. Transductive CNAP (Bateni et al., 2020b) extends the few-shot setting to a transductive setting where the query set is exploited to boost its performance. Both SUR (Dvornik et al., 2020) and URT (Liu et al., 2021b) first train an independent network for each domain and then a fusion strategy is further proposed to fuse features from all domains to adapt to unseen domains. FLUTE (Triantafillou et al., 2021) learns a shared backbone network for all domains with a specific modulator for each domain during meta-training. Domain modulators are later fused with an auxiliary "blending network" to initialize for new tasks in meta-test. tri-M (Liu et al., 2021d) adopts the same strategy of learning modulation parameters as CNAPS, where the

parameters are further grouped into the domain-specific set and the domain-cooperative set to explore the intra-domain information and inter-domain correlations, respectively. Unlike these methods, in this chapter, we In this chapter, we instead propose to use a more efficient method, the URL method introduced in Chapter 2, to learn a single multi-domain network by distilling the knowledge of domain-specific networks. It learns more general features that generalize to unseen tasks by leveraging data from multiple domains and it has a fixed computational cost regardless of the number of domains at inference which is more efficient.

Task-specific adaptation for meta-test. Non-parametric classifiers such as nearest centroid classifier (NCC) and its variants (Snell et al., 2017; Requeima et al., 2019) are commonly used in few-shot tasks due to their simplicity and do not require further adaptation at meta-test. Meta-Baseline (Chen et al., 2020b) and Baseline (Dhillon et al., 2020) fine-tune the whole model during meta-test, while Baseline++ (Chen et al., 2019) only updates a parametric classifier with the cosine distance, RFS (Tian et al., 2020b) learns a linear classifier with Logistic regression and MetaOptNet (Lee et al., 2019) optimizes a linear support vector machine (SVM). MAML (Finn et al., 2017) and LEO (Rusu et al., 2020) meta-learn how to update parameters during meta-test. TADAM (Oreshkin et al., 2018) employs a task embedding network to predict scaling and shifting parameters for each convolutional layer, similarly more dedicated methods are proposed for multi-domain few-shot generalization with a task encoder to generate conditioning parameters (Bateni et al., 2020a,b; Requeima et al., 2019) and the generated conditioning paramters can be finetuned for better adapting the task-agnostic network for previously unseen domains and tasks (Triantafillou et al., 2021).

There are also methods (*e.g*. Saikia et al. (2020); Doersch et al. (2020)) that do not fit into task-agnostic and task-specific parameterization grouping. BOHB (Saikia et al., 2020) proposes to use multi-domain data as validation objective for hyper-parameter optimization such that the feature learned on ImageNet with the optimized hyper-parameter generalizes well to multi-domain. CTX (Doersch et al., 2020) proposes to learn spatial correspondences from ImageNet and evaluates on the remaining (unseen) domains. We also compare our method to them in the setting where we use a standard single domain learning network learned from ImageNet and adapt its representations through residual adapters.

4.3 Method

Few-shot classification aims at learning to classify samples of new categories efficiently from few samples only. Each few-shot learning task consists of a support set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{S}|}$ with $|\mathcal{S}|$ sample and label pairs respectively and a query set $\mathcal{Q} = \{(\boldsymbol{x}_j)\}_{j=1}^{|\mathcal{Q}|}$ with $|\mathcal{Q}|$ samples to be classified. The goal is to learn a classifier on $\mathcal{S}$ that accurately predicts the labels of $\mathcal{Q}$. Note that this chapter focuses on the few-shot image classification problem, *i.e.* $\boldsymbol{x}$ and y denote an image and its label.

As in (Dvornik et al., 2020; Liu et al., 2021b), we solve this problem in two steps involving i) representation learning where we learn a task-agnostic feature extractor f from a large dataset $\mathcal{D}_{train}$, ii) task adaptation where we adapt the task-agnostic representations through various task-specific weights to the target tasks $(\mathcal{S}, \mathcal{Q})$ that are sampled from another large dataset $\mathcal{D}_{test}$ by taking the subsets of the dataset to build $\mathcal{S}$ and $\mathcal{Q}$. Note that $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ contain mutually exclusive classes.

4.3.1 Task-agnostic representation learning

Learning task-agnostic or universal representations (Bilen and Vedaldi, 2017) has been key to the success of cross-domain generalization. Representations learned from a large diverse dataset such as ImageNet (Deng et al., 2009) can be considered as universal and successfully transferred to tasks in different domains with minor adaptations (Rebuffi et al., 2017a; Liu et al., 2021b; Dvornik et al., 2020). We denote this setting as single domain learning (SDL).

More powerful and diverse representations can be obtained by learning the representations from multiple domains. One strategy of obtaining multi-domain representations is to employ multiple domain-specific feature extractors, one for each domain, and adaptively "fuse" their features for each task (Dvornik et al., 2020; Liu et al., 2021d; Triantafillou et al., 2021). While these methods are effective, they require computing features for each image through multiple feature extractors and are thus computationally expensive.

Alternatively, the multi-domain representations can be obtained in a more efficient way by training a single network over multiple domains. Let $\mathcal{D}_{train} = \{\mathcal{D}_t\}_{t=1}^{T}$ consists of K subdatasets, each sampled from a different domain. The vanilla multi-domain learning (MDL) strategy jointly optimizes network parameters over the images from all

K subdatasets:

$$\min_{\phi,\psi^t} \sum_{t=1}^{T} \frac{1}{|\mathcal{D}_t|} \sum_{\boldsymbol{x},y\in\mathcal{D}_t} \ell(g_{\psi^t} \circ f_{\phi}(\boldsymbol{x}), y), \tag{4.1}$$

where ℓ is cross-entropy loss, f is a feature extractor that takes an image as input and outputs a d dimensional feature. f is parameterized by ϕ which is shared across T domains. g_{ψ^t} is the classifier for domain t and parameterized by ψ^t which is discarded in the meta-test. We denote this setting as MDL.

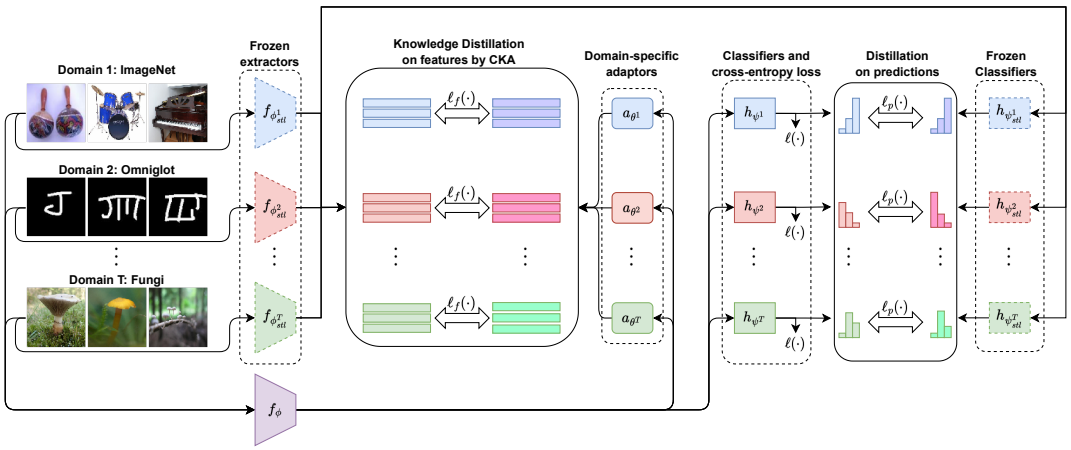


Figure 4.4: **Training pipeline for universal representation learning**. Given training images from T different domains, we first train T domain-specific networks $f_{\phi^1_{stl}}, \ldots, f_{\phi^T_{stl}}$ and their classifiers $h_{\psi^1_{stl}}, \ldots, h_{\psi^T_{stl}}$, freeze their weights and distill their knowledge to our multi-domain network by matching their features and predictions through two loss functions ℓ_f and ℓ_p respectively. As matching multiple features is challenging, we co-align all the features by using light-weight adaptors $a_{\theta^1}, a_{\theta^2}, \ldots, a_{\theta^T}$ and centered kernel alignment.

Universal Representation Learning (URL). The challenge in MDL is to efficiently share the knowledge across the domains while preventing negative transfer between them and also carefully balancing the individual loss functions ((Chen et al., 2018b)). In Chapter 2, we introduce a two stage procedure to learn multi-domain representations, a variant of MDL, inspired by the previous distillation methods (Hinton et al., 2014; Li and Bilen, 2020) as introduced in Chapter 2. Here, we briefly explain our URL method in the context of learning universal representation for cross-domain few-shot learning and illustrated in Fig. 4.4. As in Sec. 2.4.2, we first train a set of domain-specific deep networks where each consists of a specific feature extractor $f_{\phi^t_{stl}}$ and classifier $h_{\psi^t_{stl}}$ with parameters ϕ^t_{stl} and ψ^t_{stl} respectively, similarly to Dvornik et al. (2020); Liu et al. (2021b). However, instead of using T domain-specific feature extractors and selecting

the most relevant ones like them, we propose to learn a single multi-domain network that performs well in T domains by distilling the knowledge of T pretrained feature extractors. This has two key advantages over the prior works (Dvornik et al., 2020; Liu et al., 2021b). First, using a single feature extractor, which has the same capacity with each domain-specific one, is significantly more efficient in terms of run-time and number of parameters in the meta-test stage. Second learning to find the most relevant features for a given support and query set (Liu et al., 2021b) is not trivial and may also suffer from overfitting to the small number of datasets in the training set, while the multi-domain representations, by definition, automatically contain the required information from the relevant domains.

In the second stage, we freeze the pretrained weights of the domain-specific feature extractors $f_{\phi^t_{stl}}$ and transfer their knowledge into the multi-domain model at train time. Knowledge distillation can be performed at the prediction (Hinton et al., 2014) and feature level (Li and Bilen, 2020; Romero et al., 2015) by minimizing the distance between (i) the predictions of the multi-domain and corresponding single-domain network, and also between (ii) the multi-domain and single-domain features respectively for given training samples. While Kullback-Leibler (KL) divergence is the standard choice for the predictions in the original knowledge distillation article (Hinton et al., 2014), matching the multi-domain features to multiple single-domain ones simultaneously is an ill-posed problem, as the features from different domain-specific extractors for a given image $\boldsymbol{x}$ are not necessarily aligned and can vary significantly. To this end, as in Eq. (2.3), we propose to map each domain-specific feature into a common space by using adaptors $a_{\theta^t} \in \mathbb{R}^{d\times d}$ with parameters θ^t and jointly train them along with the parameters of the multi-domain network:

$$\min_{\phi,\psi^t,\theta^t} \sum_{t=1}^{T} \frac{1}{|\mathcal{D}^t|} \sum_{\boldsymbol{x},y\in\mathcal{D}^t} \Bigg(\ell(h_{\psi^t} \circ f_\phi(\boldsymbol{x}), y) + \\ \lambda^t_p \ell_p(h_{\psi^t} \circ f_\phi(\boldsymbol{x}), h_{\psi^t_{stl}} \circ f_{\phi^t_{stl}}(\boldsymbol{x})) + \lambda^t_f \ell^f(a_{\theta^t} \circ f_\phi(\boldsymbol{x}), f_{\phi^t_{stl}}(\boldsymbol{x})) \Bigg) \tag{4.2}$$

where ℓ_p is KL divergence, ℓ_f is a distance function in the feature space, λ^t_p and λ^t_f are their domain-specific weights for task t.

Due to its simplicity and effectiveness, we conduct experiments with the feature extractor of URL along with the SDL one learned from ImageNet as the task-agnostic network and study various task-adaptation strategies for cross-domain few-shot learning.

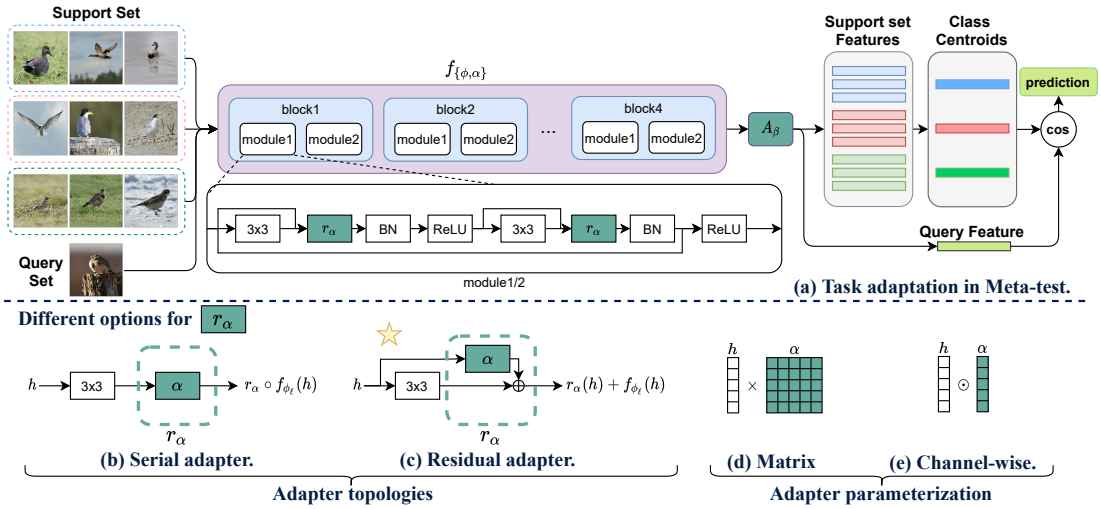


Figure 4.5: Illustration of our task adaptation for cross-domain few-shot learning. In meta-test stage (a), our method first attaches a parametric transformation r_α to each layer, where α can be constructed by (b) a serial or (c) a residual topology. They can be parameterized with matrix multiplication (d) or channel-wise scaling (e). We found that (c) is the best configuration with matrix parameterization which is further improved by attaching a linear transformation A_β to the end of the network. We adapt the network for a given task by optimizing α and A_β on a few labeled images from the support set, then map query images to the task-specific space and assign them to the nearest class center.

4.3.2 Task-specific weight learning

A good task-agnostic feature extractor f_ϕ is expected to produce representations that generalize to many previously unseen tasks and domains. However this gets more challenging when there is a large domain gap between the training set $\mathcal{D}_{train}$ and test set $\mathcal{D}_{test}$ which requires further adaptation to the target task. In this chapter, we propose to incorporate additional capacity to the task-agnostic feature extractor by adding task-specific weights to adapt the representations to the target task by using the support set. Specifically, we directly attach task-specific weights to a learned task-agnostic model, and estimate them from scratch given the support set. We denote the task-specific weights with ϑ and task-adapted classifier with $p_{(\phi,\vartheta)}$ that outputs a softmax probability vector whose dimensionality equals to the number of categories in the support set $\mathcal{S}$.

To obtain the task-specific weights, we freeze the task-agnostic weights ϕ and minimize the cross-entropy loss ℓ over the support samples in meta-test w.r.t. the

task-specific weights ϑ (Dvornik et al., 2020; Tian et al., 2020b; Li et al., 2021b):

$$\min_{\vartheta} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \ell(p_{(\phi,\vartheta)}(\boldsymbol{x}), y), \tag{4.3}$$

where $\mathcal{S}$ is sampled from the test set $\mathcal{D}_{test}$. Most previous works freeze the task-agnostic weights but estimate the task-specific weights through an auxiliary network (or a task encoder) (Requeima et al., 2019; Bateni et al., 2020b; Li et al., 2021b; Triantafillou et al., 2021), where inaccurate prediction of parameters can lead to noisy adaptation and wrong prediction.

4.3.3 Task-specific adapter parameterization (ϑ)

Task adaptation techniques can be broadly grouped into two categories that aims to adapt the feature extractor or classifier to a given target task. We use α and β to denote task-specific weights for adapting the feature extractor and classifier respectively where $\vartheta = \{\alpha, \beta\}$.

Feature extractor adaptation. A simple method to adapt f_ϕ is finetuning its parameters on the support set (Chen et al., 2020b; Dhillon et al., 2020). However, this strategy tends to suffer from the unproportionate optimization, *i.e*. updating very high-dimensional weights from a small number of support samples. In this chapter, we propose to attach task-specific adapters directly to the existing task-agnostic model, *e.g*. we attach the adapters to each module of a ResNet backbone in Fig. 4.5 (a), and the adapters can be efficiently learned/estimated from few samples. Concretely, let f_{ϕ_l} denote the l-th layer of the feature extractor f_ϕ (*i.e*. a convolutional layer) with the weights ϕ_l. Given a support set $\mathcal{S}$, the task-specific adapters r_α parameterized by α, can be incorporated to the output of the layer f_{ϕ_l} as

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha(f_{\phi_l}(\boldsymbol{h}), \boldsymbol{h}) \tag{4.4}$$

where $\boldsymbol{h} \in \mathrm{R}^{W\times H\times C}$ is the input tensor, f_{ϕ_l} is a convolutional layer in f_ϕ. Importantly, the number of the task-specific adaptation parameters α are significantly smaller than the task-agnostic ones. The adapters can be designed in different ways.

Next we propose two connection types for incorporating r_α to f_{ϕ_l}: i) serial connection by subsequently applying it to the output of layer $f_{\phi_l}(\boldsymbol{h})$ as

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha \circ f_{\phi_l}(\boldsymbol{h})$$

which is illustrated in Fig. 4.5(b), and ii) parallel connection by a residual addition as the residual adapters proposed by Rebuffi et al. (2018)

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha(\boldsymbol{h}) + f_{\phi_l}(\boldsymbol{h})$$

which is illustrated in Fig. 4.5(c). In our experiments, we found the parallel setting performing the best when α is learned on a support set during meta-test (illustrated in Fig. 4.5(c)) which we discuss in Sec. 4.4.

For the parameterization of r_α, we consider two options. Matrix multiplication (illustrated in Fig. 4.5(d)) with $\alpha \in \mathrm{R}^{C\times C}$:

$$r_\alpha(\boldsymbol{h}) = \boldsymbol{h} * \alpha,$$

where $*$ denotes a convolution, $\alpha \in \mathbb{R}^{C\times C}$ and the transformation is implemented as a convolutional operation with 1×1 kernels in our code. And channelwise scaling (illustrated in Fig. 4.5(e)):

$$r_\alpha(\boldsymbol{h}) = \boldsymbol{h} \odot \alpha,$$

where $\odot$ is a Hadamard product and $\alpha \in \mathbb{R}^{C}$. Note that one can also use an additive bias weight in both settings, however, this has not resulted in any significant gains in our experiments. While the matrix multiplication is more powerful than the scaling operation, it also requires more parameters to be estimated or learned. Note that, in a deep neural network, the number of input C_{in} and output channels C_{out} for a layer can be different. In that case, one can still use a non-square matrix: $\alpha \in \mathrm{R}^{C_{out}\times C_{in}}$, however, it is not possible to use a scaling operator in the parallel setting. In our experiments, we use ResNet architecture (He et al., 2016) where most input and output channels are the same. r_α connected in parallel with matrix multiplication form, when its parameters α are learned on the support set, is known as residual adapter (Rebuffi et al., 2018) and r_α connected serial in channelwise is known as FiLM (Perez et al., 2018).

An alternative to reduce the dimensionality of α in case of matrix multiplication is matrix decomposition: $\alpha = V\gamma^\top$, where $V \in \mathrm{R}^{C\times B}$ and $\gamma \in \mathrm{R}^{C\times B}$, $B \ll C$. Using a bottleneck, *i.e.* setting $B < C/2$, reduces the number of parameters in the multiplication. In this chapter, we set $B = [C/N]$ and evaluate the performance for various N in Sec. 4.4.

Classifier learning. Finally, the adapted feature extractor $f_{(\phi,\alpha)}$ can be combined with a task-specific classifier c_β, parameterized by β to obtain the final model, *i.e.* $c \circ f_{(\phi,\alpha)}$. Based on the recent works, we investigate use of various linear classifiers (Dhillon et al., 2020; Lee et al., 2019; Chen et al., 2020b; Requeima et al., 2019), also nonparameteric

ones including nearest centroid classifier (NCC) (Mensink et al., 2013; Snell et al., 2017) and their variants based on Mahalanobis distance (MD) (Bateni et al., 2020b).

In this chapter, we also propose to combine the nonparametric classifiers with a pre-classifier transformation. Concretely, the transformation proposed in this chapter takes in the features computed from the network $f_{\{\phi,\alpha\}} \in \mathrm{R}^d$ and applies an affine transformation A_β parameterized by β to obtain the network embedding that is fed into the classifier, *i.e.* $p_{\phi,\vartheta} = c \circ A_\beta \circ f_{\{\phi,\alpha\}}$. Note that in the case of the non-parametric classifier, c is not parameterized by β and we use β to denote the transformation parameters.

More specifically, given a support set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{S}|}$ of a new learning task, we use the multi-domain model attached with task-specific adapters to extract features $\{f_{\{\phi,\alpha\}}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{S}|}$. We then apply a linear transformation $A_\beta : \mathrm{R}^d \to \mathrm{R}^d$ with learnable parameters β to the computed features, *i.e.* $\{\boldsymbol{z}_i\}_{i=1}^{|\mathcal{S}|} = \{A_\beta \circ f_{\{\phi,\alpha\}}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{S}|}$ where $\beta \in \mathrm{R}^{d\times d}$. Then we follow a similar pipeline to the one used by Dvornik et al. (2020); Mensink et al. (2013); Snell et al. (2017) to build a centroid classifier $c = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_C\}$ by averaging the embeddings belonging to this class:

$$\boldsymbol{c}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\boldsymbol{z}_i \in \mathcal{S}_j} \boldsymbol{z}_i, \mathcal{S}_j = \{\boldsymbol{z}_k : y_k = j\}, j = 1, \ldots, C \tag{4.5}$$

where C is the number of classes in the support set. Next the likelihood of a support sample $\boldsymbol{z}$ in Eq. (4.3) can be computed as:

$$p_{\{\phi,\vartheta\}}(y = l|\boldsymbol{z}) = \frac{\exp(-d(\boldsymbol{z}, \boldsymbol{c}_l))}{\sum_{j=1}^{C} \exp(-d(\boldsymbol{z}, \boldsymbol{c}_j))}, \tag{4.6}$$

where $d(\boldsymbol{z}, \boldsymbol{c}_l)$ is the negative cosine similarity and $\vartheta = \{\alpha, \beta\}$.

In our experiments, the best performing (adaptation) setting uses parallel adapters, whose parameters are in the matrix form, to adapt the feature extractor and followed by the pre-classifier transformation and NCC.

4.4 Experiments

Here we start with experimental setup, and then we compare our methods (URL and TSA) to state-of-the-art methods and rigorously evaluate various design decisions. We finally provide further analysis.

4.4.1 Experimental setup

Dataset. We use the Meta-dataset (Triantafillou et al., 2020) which is a few-shot classification benchmark that initially consists of ten datasets: ILSVRC_2012 (Russakovsky et al., 2015) (ImageNet), Omniglot (Lake et al., 2015), FGVC-Aircraft (Maji et al., 2013) (Aircraft), CUB-200-2011 (Wah et al., 2011) (Birds), Describable Textures (Cimpoi et al., 2014) (DTD), QuickDraw (Jongejan et al., 2016), FGVCx Fungi (Brigit and Yin, 2018) (Fungi), VGG Flower (Nilsback and Zisserman, 2008) (Flower), Traffic Signs (Houben et al., 2013) and MSCOCO (Lin et al., 2014) then further expands with MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009). We follow the standard procedure (Triantafillou et al., 2020) and consider both the 'Training on all datasets' (multi-domain learning) and 'Training on ImageNet only' (single-domain learning) settings. In 'Training on all datasets' setting, we follow the standard procedure and use the first eight datasets for meta-training, in which each dataset is further divided into train, validation and test set with disjoint classes. While the evaluation within these datasets is used to measure the generalization ability in the seen domains, the remaining five datasets are reserved as unseen domains in meta-test for measuring the cross-domain generalization ability. In 'Training on ImageNet only' setting, we follow the standard procedure and only use the train split of ImageNet for meta-training. The evaluation of models is in the test split of ImageNet and the rest 12 datasets which are reserved as unseen domains in meta-test. Following Triantafillou et al. (2020), we evaluate our method on 600 randomly sampled tasks for each dataset with varying number of ways and shots, and report average accuracy and 95% confidence score in all experiments.

Implementation details. We use PyTorch (Paszke et al., 2019) library to implement our method. In all experiments we build our method on ResNet-18 (He et al., 2016) backbone for both single-domain and multi-domain networks, unless stated otherwise. In the multi-domain network, we share all the layers but the last classifier across the domains.

For learning the task-agnostic network, we strictly follow the training protocol in the prior work (Dvornik et al., 2020), use a SGD optimizer with a momentum and the cosine annealing learning scheduler with the same hyperparameters to train single-domain models. For our multi-domain network (URL), we use the same optimizer and scheduler as before, training it for 240,000 iterations. We set λ^f and λ^p to 4 for ImageNet and 1 for other datasets and use early-stopping based on cross-validation over the validations

sets of 8 training datasets.

Once the task-agnostic network is learned, we freeze its paramters and attach the task-specific weights (ϑ) to it. For learning task-specific weights (ϑ), including the pre-classifier transformation β and the adapter parameters, we directly attach them to the task-agnostic weights and learn them on the support samples in meta-test by using Adadelta optimizer (Zeiler, 2012).

In the study of various task adaptation strategies in Section 4.4.3, we consider to only estimate the adapter parameters and learn the auxiliary network parameters by using Adam optimizer as Requeima et al. (2019); Bateni et al. (2020b) in meta-train. Note that estimation of pre-classifier and classifier weights via the auxiliary network leads to noisy and poor results and we do not report them. Similarly, we found that the auxiliary network fails to estimate very high-dimensional weights. Hence we only use it to estimate adapter weights that are parameterized with a vector for channelwise multiplication but not with a matrix. See Appendix C.1 for more details.

4.4.2 Comparison to state-of-the-art methods

We first evaluate our URL for task-agnostic weights learning in the multi-domain learning setting. We then compare our TSA in two settings with multi-domain or single-domain feature extractor to existing state-of-the-art methods. For evaluation, we follow the standard protocol (Triantafillou et al., 2020), randomly sample 600 tasks for each dataset, and report average accuracy and 95% confidence score in all experiments. As in Meta-Dataset (Triantafillou et al., 2020), we sample each task with a varying number of ways and shots and report the results in Tab. 4.1.

Task-agnostic Weight Learning. First we compare our URL method (with pre-classifier alignment (A_β)) for task-agnostic weight learning to our own baselines, i) the best single-domain model (Best SDL) where we use each single-domain network as the feature extractor and test it for few-shot classification in each dataset and pick the best performing model (see Appendix C.2.1 for the complete results). This involves evaluating 8 single-domain networks on 13 datasets, serves a very competitive baseline, ii) the vanilla multi-domain learning baseline (MDL) that is learning by optimizing Eq. (4.1) without the proposed distillation method. As an additional baseline, we include the state-of-the-art methods for task-agnostic weight learning, CNAPS (Requeima et al., 2019), SUR (Dvornik et al., 2020), URT (Liu et al., 2021b), and Simple CNAPS (Bateni

Test Dataset	CNAPS	Simple CNAPS	SUR	URT	Best SDL	MDL	Transductive CNAPS	triM	FLUTE	URL (Ours)	TSA (Ours)
ImageNet	50.8±1.1	58.4±1.1	56.2±1.0	56.8±1.1	55.8±1.0	53.4±1.1	57.9±1.1	58.6±1.0	51.8±1.1	58.8±1.1	**59.5±1.0**
Omniglot	91.7±0.5	91.6±0.6	94.1±0.4	94.2±0.4	93.2±0.5	93.8±0.4	94.3±0.4	92.0±0.6	93.2±0.5	94.5±0.4	**94.9±0.4**
Aircraft	83.7±0.6	82.0±0.7	85.5±0.5	85.8±0.5	85.7±0.5	86.6±0.5	84.7±0.5	82.8±0.7	87.2±0.5	89.4±0.4	**89.9±0.4**
Birds	73.6±0.9	74.8±0.9	71.0±1.0	76.2±0.8	71.2±0.9	78.5±0.8	78.8±0.7	75.3±0.8	79.2±0.8	80.7±0.8	**81.1±0.8**
Textures	59.5±0.7	68.8±0.9	71.0±0.8	71.6±0.7	73.0±0.6	71.4±0.7	66.2±0.8	71.2±0.8	68.8±0.8	77.2±0.7	**77.5±0.7**
Quick Draw	74.7±0.8	76.5±0.8	81.8±0.6	82.4±0.6	**82.8±0.6**	81.5±0.6	77.9±0.6	77.3±0.7	79.5±0.7	82.5±0.6	81.7±0.6
Fungi	50.2±1.1	46.6±1.0	64.3±0.9	64.0±1.0	65.8±0.9	61.9±1.0	48.9±1.2	48.5±1.0	58.1±1.1	**68.1±0.9**	66.3±0.8
VGG Flower	88.9±0.5	90.5±0.5	82.9±0.8	87.9±0.6	87.0±0.6	88.7±0.6	**92.3±0.4**	90.5±0.5	91.6±0.6	92.0±0.5	92.2±0.5
Traffic Sign	56.5±1.1	57.2±1.0	51.0±1.1	48.2±1.1	47.4±1.1	51.0±1.0	59.7±1.1	63.0±1.0	58.4±1.1	63.3±1.1	**82.8±1.0**
MSCOCO	39.4±1.0	48.9±1.1	52.0±1.1	51.5±1.1	53.5±1.0	49.6±1.1	42.5±1.1	52.8±1.1	50.0±1.0	57.3±1.0	**57.6±1.0**
MNIST	-	94.6±0.4	94.3±0.4	90.6±0.5	89.8±0.5	94.4±0.3	94.7±0.3	96.2±0.3	95.6±0.5	94.7±0.4	**96.7±0.4**
CIFAR-10	-	74.9±0.7	66.5±0.9	67.0±0.8	67.3±0.8	66.7±0.8	73.6±0.7	75.4±0.8	78.6±0.7	74.2±0.8	**82.9±0.7**
CIFAR-100	-	61.3±1.1	56.9±1.1	57.3±1.0	56.6±0.9	53.6±1.0	61.8±1.0	62.0±1.0	67.1±1.0	63.5±1.0	**70.4±0.9**
Average Seen	71.6	73.7	75.9	77.4	76.8	77.0	75.1	74.5	76.2	80.4	80.4
Average Unseen	-	67.4	64.1	62.9	62.9	63.1	66.5	69.9	69.9	70.6	78.1
Average All	-	71.2	71.4	71.8	71.5	71.6	71.8	72.7	73.8	76.6	79.5
Average Rank	-	7.1	6.7	6.4	6.7	6.5	6.3	5.6	5.3	2.5	1.8

Table 4.1: Comparison state-of-the-art methods on Meta-Dataset under multi-domain learning setting. Mean accuracy, 95% confidence interval are reported. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

et al., 2020b)[2]. We reproduce results by training and evaluating SUR (Dvornik et al., 2020), URT (Liu et al., 2021b), and Simple CNAPS (Bateni et al., 2020b) using their code for fair comparison as recommended by Meta-Dataset. To better analyze the results, we divide the table into two blocks that show the few-shot classification accuracy in previously seen domains and unseen domains along with their average accuracy. We also report average accuracy over all domains and the average rank as Triantafillou et al. (2021); Li et al. (2021b).

Comparing the state-of-the-art methods (CNAPS, Simple CNAPS, SUR, URT) for learning task-agnostic weight, our URL method obtains better performance in seven out of eight seen datasets and four out of five unseen datasets. We also compute average rank as recommended by Triantafillou et al. (2020), our method ranks 2.5 in average and the state-of-the-art methods SUR, URT rank 6.7 and 6.4, respectively. More specifically, we obtain significantly better results than the second best approach of task-agnostic weight learning methods (CNAPS, Simple CNAPS, SUR, URT, Best SDL and MDL) on Aircraft (+2.8), Birds (+2.1), Texture (+4.2), and VGG Flower (+1.5) for seen domains and Traffic Sign (+6.1)[3] and MSCOCO (+3.8). The results show that jointly learning a

[2]Results of CNAPS (Requeima et al., 2019) are obtained from Meta-Dataset.

[3]The accuracy of all methods on Traffic Sign is different from the one in the original papers as one bug has been fixed in Meta-Dataset repository. See `https://github.com/google-research/meta-dataset/issues/54` for more details. As mentioned in the Meta-Dataset repository, we further

single set of representations provides better generalization ability than fusing the ones from multiple single-domain feature extractors as done in SUR and URT. Notably, our method requires less parameters and computations to run during inference than SUR and URT, as it runs only one universal network to extract features, while both SUR and URT need to pass the query set to multiple single-domain networks.

We also see that our URL method outperforms two strong baselines, Best SDL and MDL in all datasets except in QuickDraw. This indicates that i) universal representations are superior to the single-domain ones while generalizing to new tasks in both seen and unseen domains, while requiring significantly less number of parameters (1 vs 8 neural networks), ii) our distillation strategy is essential to obtain good multi-domain representations. While MDL outperforms the best SDL in certain domains by transferring representations across them, its performance is lower in other domains than SDL, possibly due to negative transfer across the significantly diverse domains. Surprisingly, MDL achieves the third best in average rank among task-agnostic weight learning methods, indicating the benefit of multi-domain representations.

Task-specific Weight Learning. We then incorporate the proposed residual adapters in matrix form (TSA) to the multi-domain feature extractor of our URL method and compare its performance with the the state-of-the-art methods (CNAPS (Requeima et al., 2019), SUR (Dvornik et al., 2020), URT (Liu et al., 2021b), Simple CNAPS (Bateni et al., 2020b), and more recent state-of-the-art methods including Transductive CNAPS (Bateni et al., 2020a), FLUTE (Triantafillou et al., 2021), and tri-M (Liu et al., 2021d)) in Tab. 4.1. Simple CNAPS improves over CNAPS by adopting a simple Mahalanobis distance instead of learning an adapted linear classifier. Transductive CNAPS further improves by using unlabelled test images. SUR and URT fuse multi-domain features to get better performance. FLUTE improves URT by fusing FiLM parameters as initialization which is further finetuned on the support set in meta-test. tri-M adopts the same strategy of learning modulation parameters as CNAPS, where the parameters are further divided into the domain-specific set and the domain-cooperative set to explore the intra-domain information and inter-domain correlations, respectively. URL surpasses previous methods by learning a universal representation with distillation from multiple domains and can be further improved by our TSA.

From the results, our TSA method outperforms other methods on most domains (10 out of 13), especially obtaining significant improvement on 5 unseen datasets than

update the evaluation protocol and report the updated results of all methods in Appendix C.2.3.

the second best method, *i.e*. Average Unseen (+7.5). More specifically, our method obtains significantly better results than the second best approach on Traffic Sign (+19.5), CIFAR-10 (+8.7), and CIFAR-100 (+6.8). Achieving improvement on unseen domains is more challenging due to the large gap between seen and unseen domain and the scarcity of labeled samples for the unseen task. We address this problem by attaching light-weight adapters to the feature extractor residually and learn the attached adapters on the support set from scratch. This allows the model to learn more accurate and effective task-specific parameters (adapters) from the support set to efficiently steer the task-agnostic features for the unseen task, compared with predicting task-specific parameters by an auxiliary network learned in meta-train, *e.g*. Simple CNAPS, tri-M, or fusing representations from multiple feature extractors *e.g*. SUR, URT. Though FLUTE uses a hybrid approach which uses auxiliary networks learned from meta-train to initialize the FiLM parameters for further fine-tuning, their results are not better than URL, which achieves very competitive results as it learns a good universal representation that generalizes well to seen domains and can be further improved with the adaptation strategy proposed in this chapter, especially significant improvements on unseen domains.

	ResNet-18							ResNet-34		
Test Dataset	Finetune	ProtoNet	fo-Proto -MAML	ALFA+fo -Proto-MAML	BOHB	FLUTE	TSA (Ours)	ProtoNet	CTX	TSA (Ours)
ImageNet	45.8±1.1	50.5±1.1	49.5±1.1	52.8±1.1	51.9±1.1	46.9±1.1	**59.5±1.1**	53.7±1.1	62.8±1.0	**63.7±1.0**
Omniglot	60.9±1.6	60.0±1.4	63.4±1.3	61.9±1.5	67.6±1.2	61.6±1.4	**78.2±1.2**	68.5±1.3	82.2±1.0	**82.6±1.1**
Aircraft	68.7±1.3	53.1±1.0	56.0±1.0	63.4±1.1	54.1±0.9	48.5±1.0	**72.2±1.0**	58.0±1.0	79.5±0.9	**80.1±1.0**
Birds	57.3±1.3	68.8±1.0	68.7±1.0	69.8±1.1	70.7±0.9	47.9±1.0	**74.9±0.9**	74.1±0.9	80.6±0.9	**83.4±0.8**
Textures	69.0±0.9	66.6±0.8	66.5±0.8	70.8±0.9	68.3±0.8	63.8±0.8	**77.3±0.7**	68.8±0.8	75.6±0.6	**79.6±0.7**
Quick Draw	42.6±1.2	49.0±1.1	51.5±1.0	59.2±1.2	50.3±1.0	57.5±1.0	**67.6±0.9**	53.3±1.1	**72.7±0.8**	71.0±0.8
Fungi	38.2±1.0	39.7±1.1	40.0±1.1	41.5±1.2	41.4±1.1	31.8±1.0	**44.7±1.0**	40.7±1.1	**51.6±1.1**	51.4±1.2
VGG Flower	85.5±0.7	85.3±0.8	87.2±0.7	86.0±0.8	87.3±0.6	80.1±0.9	**90.9±0.6**	87.0±0.7	**95.3±0.4**	94.0±0.5
Traffic Sign	66.8±1.3	47.1±1.1	48.8±1.1	60.8±1.3	51.8±1.0	46.5±1.1	**82.5±0.8**	58.1±1.1	**82.7±0.8**	81.7±0.9
MSCOCO	34.9±1.0	41.0±1.1	43.7±1.1	48.1±1.1	48.0±1.0	41.4±1.0	**59.0±1.0**	41.7±1.1	59.9±1.0	**61.7±0.9**
MNIST	-	-	-	-	-	80.8±0.8	**93.9±0.6**	-	-	**94.6±0.5**
CIFAR-10	-	-	-	-	-	65.4±0.8	**82.1±0.7**	-	-	**86.0±0.6**
CIFAR-100	-	-	-	-	-	52.7±1.1	**70.7±0.9**	-	-	**78.3±0.8**
Average Seen	45.8	50.5	49.5	52.8	51.9	46.9	**59.5**	53.7	62.8	**63.7**
Average Unseen	58.2	56.7	58.4	62.4	60.0	53.2	**71.9**	61.1	75.6	**76.2**
Average All	57.0	56.1	57.5	61.4	59.2	52.6	**70.7**	60.4	74.3	**74.9**
Average Rank	7.9	8.3	7.0	5.3	6.0	8.9	**2.8**	5.5	1.8	**1.5**

Table 4.2: Comparison to state-of-the-art methods on Meta-Dataset (using a single-domain feature extractor which is trained only on ImageNet). Mean accuracy, 95% confidence interval are reported. Only ImageNet is seen during training and the rest datasets are unseen for test only.

Despite incorporating TSA with our URL multi-domain feature extractor, we also

evaluate our TSA method with a single-domain feature extractor trained on ImageNet only on ResNet-18 as Triantafillou et al. (2020) or ResNet-34 as Doersch et al. (2020). This setting is more challenging than the multi-domain one, as the model is trained only on one domain and tested on both the test split of ImageNet and the ones of other domains. We report the results of our method and state-of-the-art methods (BOHB (Saikia et al., 2020), FLUTE (Triantafillou et al., 2021), Finetune (Triantafillou et al., 2020), ProtoNet (Triantafillou et al., 2020), fo-Proto-MAML (Triantafillou et al., 2020), and ALFA+fo-Proto-MAML (Triantafillou et al., 2020), CTX (Doersch et al., 2020)) in Tab. 4.2. ALFA+fo-Proto-MAML achieves the prior best performance by combining the complementary strengths of Prototypical Networks and MAML (fo-Proto-MAML), with extra meta-learning of per-step hyperparameters: learning rate and weight decay coefficients. FLUTE fails to surpass it with one training source domain, probably due to the lack of FiLM parameters from multiple domains. Our TSA method, when using the ResNet18 backbone, outperforms other methods on all domains, especially obtaining significant improvement, *i.e.* Average Unseen (+9.5), on 12 unseen datasets than the second best method. We compare our method to CTX and ProtoNet, which use the ResNet-34 backbone. [4] CTX is very competitive by learning coarse spatial correspondence between the query and the support images with an attention mechanism. Ours is orthogonal to CTX and both CTX and our method can potentially be complementary, but we leave this as future work due to the high computational cost of CTX. Specifically, we see that our TSA method obtains the best average rank and outperforms CTX on most domains (6 out of 10) while our method being more efficient (We train our model on one single Nvidia GPU for around 33 hours while CTX requires 8 Nvidia V100 GPUs and 7 days for training. Please refer to Appendix C.2.4 for more details).

4.4.3 Analysis of task-specific parameterizations

Classifier learning. First we study the adaptation strategies for learning only a task-specific classifier on the pretrained feature extractor of our URL method. We evaluate non-parametric classifiers including nearest cetroid classifier (*NCC*) and NCC Mahalanobis Distance (*MD*) and parametric classifiers including logistic regression (*LR*), support vector machine *SVM* whose parameters are learned on support samples. We also

[4] Note that CTX also uses augmentation strategies such as AutoAugment (Cubuk et al., 2019) and other ones from SimClr (Chen et al., 2020a). We expect applying the same augmentation strategies to our method would yield further improvements, but we leave this for future work.

Test Dataset	classifier	Aux-Net or Ad	serial or residual	M or CW	β	#params	Image -Net	Omni -glot	Air- craft	Birds	Tex- tures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MS- COCO	MNIST	CIFAR -10	CIFAR -100
NCC	NCC	-	-	-	✗	-	57.0	94.4	88.0	80.3	74.6	81.8	66.2	91.5	49.8	54.1	91.1	70.6	59.1
MD	MD	-	-	-	✗	-	53.9	93.8	87.6	78.3	73.7	80.9	57.7	89.7	62.2	48.5	95.1	68.9	60.0
LR	LR	-	-	-	✗	-	56.0	93.7	88.3	79.7	74.7	80.0	62.1	91.1	59.7	51.2	93.5	73.1	60.1
SVM	SVM	-	-	-	✗	-	54.5	94.3	87.7	78.1	73.8	80.0	58.5	91.4	65.7	50.5	95.4	72.0	60.5
Finetune	NCC	-	-	-	✗	-	55.9	94.0	87.3	77.8	76.8	75.3	57.6	91.5	**86.1**	53.1	**96.8**	80.9	65.9
Aux-S-CW	NCC	Aux-Net	serial	CW	✗	76.98%	54.6	93.5	86.6	78.6	71.5	79.3	66.0	87.6	43.3	49.1	87.9	62.8	51.5
Aux-R-CW	NCC	Aux-Net	residual	CW	✗	76.98%	56.1	94.2	88.4	80.6	74.9	82.0	66.4	91.6	48.5	53.5	90.8	70.2	59.7
Aux-S-CW	MD	Aux-Net	serial	CW	✗	76.98%	55.1	93.8	86.8	77.4	73.2	79.9	57.4	88.1	58.4	50.1	92.7	66.5	55.7
Aux-R-CW	MD	Aux-Net	residual	CW	✗	76.98%	54.8	93.8	87.4	78.2	73.4	81.1	58.8	90.1	63.6	48.5	94.8	69.6	60.6
Ad-S-CW	NCC	Ad	serial	CW	✗	0.06%	56.8	94.8	89.3	80.7	74.5	81.6	65.8	91.3	73.9	53.6	95.7	78.4	64.3
Ad-R-CW	NCC	Ad	residual	CW	✗	1.57%	57.6	94.7	89.0	81.2	75.2	81.5	65.4	91.8	79.2	54.7	96.4	79.5	67.4
Ad-S-M	NCC	Ad	serial	M	✗	12.50%	56.2	94.4	89.1	80.6	75.8	81.6	**67.1**	92.1	67.6	54.8	95.9	78.9	66.6
Ad-R-M	NCC	Ad	residual	M	✗	10.93%	57.3	94.9	88.9	81.0	76.7	80.6	65.4	91.4	82.6	55.0	96.6	82.1	66.4
Ad-R-CW-PA	NCC	Ad	residual	CW	✓	3.91%	58.6	94.5	**90.0**	80.5	**77.6**	**81.9**	67.0	**92.2**	80.2	57.2	96.1	81.5	**71.4**
Ad-R-M-PA	NCC	Ad	residual	M	✓	13.27%	**59.5**	**94.9**	89.9	**81.1**	77.5	81.7	66.3	**92.2**	82.8	**57.6**	96.7	**82.9**	70.4

Table 4.3: Comparisons to methods that learn classifiers and model adaptation methods during meta-test stage based on URL model. NCC, MD, LR, SVM denote nearest centroid classifier, Mahalanobis distance, logistic regression, support vector machines respectively. 'Aux-Net or Ad' indicates using Auxiliary Network to predict α or attaching adapter α directly. 'M or CW' means using matrix multiplication or channel-wise scaling adapters. 'S' and 'R' denote serial adapter and residual adapter, respectively. 'β' indicates using the pre-classifier adaptation. The standard deviation results can be found in Appendix C.2.2. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

include another baseline with NCC that finetunes all the feature extractor parameters, and report the results in Tab. 4.3. We observe that NCC obtains the best results for the seen domains and its performance is further improved by MD, while SVM achieves the best for the unseen domains among other classifiers. Finetuning baseline provides competitive results especially for the unseen domains. However, it performs poorly in most seen domains.

Feature extractor adaptation. Next we analyze various design decisions for the feature extractor adaptation including connection types (serial, residual), *i.e*. Fig. 4.5(b), (c), its parameterization including channelwise modulation (*CW*) when they are estimated by an auxiliary network (*Aux-Net*), which has around 77% capacity of the feature extractor. We use with each combination with two nonparameteric classifier, either NCC or MD. *While the adaptation strategies using residual connections perform better than the serial one in almost all cases, the gains are more substantial when generalizing to unseen domains.* Learning adapter weights from few samples only can be very noisy. With residual addition, it is not necessary to change all connections for passing the information forward, which can improve the robustness of useful features

and reduce learning burdens for new tasks, hence increasing the generalization ability. While the serial connections may damage the previous learned structures. We also observe that NCC and MD obtain comparable performances. Note that Aux-S-CW with MD corresponds to our implementation of Simple CNAPS (Bateni et al., 2020b) with the more powerful feature extractor. We show that replacing its serial connection with a residual one leads to a strong performance boost.

Next we look at the adaptation strategy that learns the task-specific weights directly on the support set as in Eq. (4.3). We evaluate serial and residual connection types with channelwise and matrix parameterizations by using NCC. We denote this setting as Ad in Tab. 4.3. Note that we omit MD here, as it produces similar results to NCC. First we observe that learning the weights on the support set outperforms the strategy of estimating them through an auxiliary network in almost all cases. In addition, the learnable weights require less number of parameters per task, while the capacity of the auxiliary network is fixed. We again observe that the residual connections are more effective, especially when used with the matrix parameterization (Ad-R-M). However,the channelwise ones provide a good performance/computation tradeoff. Finally, using the pre-classifier alignment (Ad-R-CW-PA and Ad-R-M-PA) further boosts the performance of the best models and we use our best model Ad-R-M-PA to compare against the state-of-the-art.

4.4.4 Further results

Varying-way Five-shot. After evaluating our methods (URL and TSA) over a broad range of varying shots (*e.g.* up to 100 shots), we follow Doersch et al. (2020); Li et al. (2021b) to further analyze our method in 5-shot setting of varying number of categories. In this setting, we sample a varying number of ways with a fixed number of shots to form balanced support and query sets. As shown in Table 4.4, overall performance for all methods decreases in most datasets compared to results in Table 4.1 indicating that this is a more challenging setting. It is due to that five-shot setting samples much less support images per class than the standard setting. Our URL and TSA methods obtain better performance than other compared state-of-the-art methods and our TSA method improves over our URL method when the number of support images per class is fewer, especially on unseen domains (Average Unseen +6.2).

Five-way One-shot. A similar conclusion can be drawn from this challenging case. Note that there are extremely few samples available for training in this case. As we

	Varying-Way Five-Shot					Five-Way One-Shot				
Test Dataset	Simple CNAPS	SUR	URT	URL (Ours)	TSA (Ours)	Simple CNAPS	SUR	URT	URL (Ours)	TSA (Ours)
ImageNet	47.2±1.0	46.7±1.0	48.6±1.0	**49.4±1.0**	48.3±1.0	42.6±0.9	40.7±1.0	47.4±1.0	**49.6±1.1**	48.0±1.0
Omniglot	95.1±0.3	95.8±0.3	96.0±0.3	96.0±0.3	**96.8±0.3**	93.1±0.5	93.0±0.7	95.6±0.5	95.8±0.5	**96.3±0.4**
Aircraft	74.6±0.6	82.1±0.6	81.2±0.6	84.8±0.5	**85.5±0.5**	65.8±0.9	67.1±1.4	77.9±0.9	**79.6±0.9**	**79.6±0.9**
Birds	69.6±0.7	62.8±0.9	71.2±0.7	76.0±0.6	**76.6±0.6**	67.9±0.9	59.2±1.0	70.9±0.9	**74.9±0.9**	74.5±0.9
Textures	57.5±0.7	60.2±0.7	65.2±0.7	**69.1±0.6**	68.3±0.7	42.2±0.8	42.5±0.8	49.4±0.9	53.6±0.9	**54.5±0.9**
Quick Draw	70.9±0.6	79.0±0.5	**79.2±0.5**	78.2±0.5	77.9±0.6	70.5±0.9	**79.8±0.9**	79.6±0.9	79.0±0.8	79.3±0.9
Fungi	50.3±1.0	66.5±0.8	66.9±0.9	70.0±0.8	**70.4±0.8**	58.3±1.1	64.8±1.1	71.0±1.0	75.2±1.0	**75.3±1.0**
VGG Flower	86.5±0.4	76.9±0.6	82.4±0.5	89.3±0.4	**89.5±0.4**	79.9±0.7	65.0±1.0	72.7±0.0	79.9±0.8	**80.3±0.8**
Traffic Sign	55.2±0.8	44.9±0.9	45.1±0.9	57.5±0.8	**72.3±0.6**	55.3±0.9	44.6±0.9	52.7±0.9	**57.9±0.9**	57.2±1.0
MSCOCO	49.2±0.8	48.1±0.9	52.3±0.9	**56.1±0.8**	56.0±0.8	48.8±0.9	47.8±1.1	56.9±1.1	59.2±1.0	**59.9±1.0**
MNIST	88.9±0.4	90.1±0.4	86.5±0.5	89.7±0.4	**92.5±0.4**	**80.1±0.9**	77.1±0.9	75.6±0.9	78.7±0.9	**80.1±0.9**
CIFAR-10	66.1±0.7	50.3±1.0	61.4±0.7	66.0±0.7	**72.0±0.7**	50.3±0.9	35.8±0.8	47.3±0.9	54.7±0.9	**55.8±0.9**
CIFAR-100	53.8±0.9	46.4±0.9	52.5±0.9	57.0±0.9	**64.1±0.8**	53.8±0.9	42.9±1.0	54.9±1.1	61.8±1.0	**63.7±1.0**
Average Seen	69.0	71.2	73.8	76.6	**76.7**	65.0	64.0	70.6	73.4	**73.5**
Average Unseen	62.6	56.0	59.6	65.2	**71.4**	57.7	49.6	57.5	62.4	**63.4**
Average All	66.5	65.4	68.3	72.2	**74.6**	62.2	58.5	65.5	69.2	**69.6**
Average Rank	4.1	3.9	3.4	2.1	**1.5**	3.8	4.5	3.3	**1.7**	**1.7**

Table 4.4: Results of Varying-Way Five-Shot and Five-Way One-Shot scenarios. Mean accuracy, 95% confidence interval are reported.

can see, both TSA and URL outperforms other approaches and TSA achieves similar results with URL on seen domains but much better performance on unseen domains due to the learning of attached residual adapters is less over-fitting.

4.4.5 Ablation study for task-agnostic weight learning

Test Dataset	ImageNet		Omniglot		Aircraft		Birds		Textures		Quick Draw		Fungi		VGG Flower		Traffic Sign		MSCOCO		MNIST		CIFAR-10		CIFAR-100	
Recall@k	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Sum	22.1	30.3	84.7	91.8	69.7	80.7	45.9	59.7	66.3	78.2	77.4	84.3	31.9	42.9	85.1	92.1	94.6	97.2	62.6	71.2	98.3	99.2	54.0	68.9	27.8	37.4
Concate	20.2	28.0	84.4	91.5	44.3	58.1	35.5	48.8	68.8	78.2	73.0	80.8	30.7	40.4	83.4	91.3	**95.1**	**97.3**	60.7	69.8	**98.7**	**99.3**	49.7	65.3	25.4	34.6
MDL	29.8	39.6	**89.8**	**94.3**	80.3	87.1	63.2	75.9	67.0	77.1	79.5	85.4	40.2	51.7	86.9	93.3	89.5	94.1	63.6	72.6	97.6	98.8	58.9	72.9	31.6	42.0
Simple CNAPS	34.0	43.8	84.9	91.6	70.5	82.5	55.9	70.5	64.8	76.9	75.3	83.0	29.1	39.0	88.1	94.1	79.9	86.9	65.2	73.8	97.5	98.8	**66.2**	**79.3**	33.2	44.2
URL (Ours)	**36.1**	**46.2**	89.7	**94.3**	**83.3**	**90.4**	**66.7**	**78.9**	**70.2**	**80.8**	**79.9**	**86.5**	**44.5**	**56.2**	**90.0**	**94.6**	87.9	93.0	**67.4**	**76.3**	97.0	98.4	62.1	76.5	**35.1**	**46.1**

Table 4.5: **Global retrieval performance on MetaDataset**. Here we evaluate our method in a non-episodic retrieval task to further compare the generalization ability of our universal representations.

Global retrieval. Here we go beyond the few-shot classification experiments and evaluate the generalization ability of our representations (URL) that are learned in the multi-domain network in a retrieval task, inspired from metric learning literature (Oh Song et al., 2016; Yu et al., 2019). To this end, for each test image, we find the nearest images in the entire test set in the feature space and test whether they correspond to the same category. For evaluation metric, we use Recall@k which considers the

predictions with one of the k closest neighbors with the same label as positive. In Tab. 4.5, we compare our method with Simple CNAPS in Recall@1 and Recall@2 (see Appendix C.2.1 for more results). URT and SUR require adaptation using the support set and no such adaptation in retrieval task is possible, we replace them with two baselines that concatenate or sum features from multiple domain-specific networks. Our method achieves the best performance in ten out of thirteen domains with significant gains in Aircraft, Birds, Textures and Fungi. This strongly suggests that our multi-domain representations are the key to the success of our method in the previous few-shot classification tasks.

Effect of adaptors in knowledge distillation In this section, we evaluate our URL method with (linear) adaptors or without adaptors for aligning features when we use CKA for knowledge distillation. From Tab. 4.6, we can see that using adaptors can improve the performance, such as Birds (+1.7) and VGG Flower (+3.6), MSCOCO (+1.3). This indicates that the adaptors a_θ help align features between multi-domain and single-domain learning networks which are learned from very different domains.

Test Dataset	Ours (CKA w/o a_θ)	Ours (CKA)
ImageNet	58.3 ± 1.0	$\mathbf{59.0 \pm 1.0}$
Omniglot	94.4 ± 0.4	$\mathbf{94.7 \pm 0.4}$
Aircraft	$\mathbf{88.9 \pm 0.5}$	$\mathbf{88.9 \pm 0.4}$
Birds	78.7 ± 0.8	$\mathbf{80.4 \pm 0.7}$
Textures	$\mathbf{74.8 \pm 0.7}$	74.5 ± 0.7
Quick Draw	$\mathbf{82.1 \pm 0.6}$	81.9 ± 0.6
Fungi	65.4 ± 0.9	$\mathbf{66.4 \pm 0.9}$
VGG Flower	87.5 ± 0.6	$\mathbf{91.3 \pm 0.5}$
Traffic Sign	$\mathbf{63.3 \pm 1.1}$	63.2 ± 1.1
MSCOCO	55.3 ± 1.0	$\mathbf{56.6 \pm 1.0}$
MNIST	$\mathbf{94.9 \pm 0.4}$	94.7 ± 0.4
CIFAR-10	73.4 ± 0.7	$\mathbf{73.8 \pm 0.7}$
CIFAR-100	61.8 ± 1.0	$\mathbf{62.1 \pm 1.0}$

Table 4.6: Results of our URL method using CKA, CKA without adaptors (*i.e.* a_θ). Mean accuracy and 95% confidence interval are reported. Here, Ours (CKA w/o a_θ) indicates that adaptors are not applied for aligning features. All results are obtained with pre-classifier alignment during meta-test stage.

Different distillation loss functions. Compared to learning multiple dense prediction tasks in a single domain in Chapter 2, learning universal representations from multiple visually-diverse domains is a more challenging problem. Hence we use CKA as the loss function for representation distillation, *i.e.* ℓ_f. Here we evaluate the effect of CKA in MetaDataset and compare it to different distillation loss functions, and report their

performances in Tab. 4.7. In this study, we set λ_p to zero and do not match the prediction of the universal network with the one of the domain-specific ones. Among these loss functions, the best results are obtained with CKA loss in all domains. Although the universal representations are first mapped to the domain-specific spaces via adapters, L2 and cosine loss functions are not sufficient to match features from very diverse domains and further aligning features with CKA is significantly beneficial.

Test Dataset	L2	COSINE	CKA
ImageNet	55.7±1.1	57.0±1.1	**59.0±1.0**
Omniglot	94.0±0.4	94.1±0.4	**94.7±0.4**
Aircraft	87.4±0.5	88.3±0.5	**88.9±0.5**
Birds	78.5±0.7	77.5±0.8	**80.4±0.7**
Textures	72.8±0.6	73.2±0.7	**74.5±0.7**
Quick Draw	81.2±0.6	80.8±0.6	**81.9±0.6**
Fungi	65.7±0.9	65.9±0.9	**66.4±0.9**
VGG Flower	87.5±0.6	85.0±0.6	**91.3±0.5**
Traffic Sign	61.6±1.1	59.5±1.1	**63.2±1.1**
MSCOCO	53.4±1.0	53.8±1.1	**56.6±1.0**
MNIST	**94.7±0.3**	93.2±0.5	**94.7±0.4**
CIFAR-10	71.1±0.8	68.1±0.8	**73.8±0.7**
CIFAR-100	59.1±1.0	58.1±1.0	**62.1±1.0**

Table 4.7: **Quantitative analysis of knowledge distillation loss functions for ℓ_f.** Mean accuracy, 95% confidence interval are reported. COSINE denotes negative cosine similarity. All the loss functions are applied to measure the difference between intermediate representations of neural networks. All results are obtained with pre-classifier alignment during meta-test stage.

We then evaluate individual contributions of distillation through representations and predictions while using CKA and KL-divergence respectively in Tab. 4.8. Compared to only applying KL loss on predictions ('URL (Ours) w/o ℓ_f'), only aligning representations with CKA loss function ('URL (Ours) w/o ℓ_p') performs better in most domains. Finally, combining ℓ_f (CKA) with ℓ_p (KL divergence), *i.e.* 'URL (Ours) ($\ell_f + \ell_p$)', gives the best performance over the multi-domain models that are trained with the individual loss functions.

4.4.6 Ablation study for task-specific weight learning

Here, we conduct ablation study for task-specific weight learning of using residual adapters in matrix form with pre-classifier alignment, including the evaluation of our

Test Dataset	URL (Ours) w/o ℓ_p	URL (Ours) w/o ℓ_f	URL (Ours) ($\ell_f + \ell_p$)
ImageNet	**59.0 ± 1.0**	57.0 ± 1.1	58.8 ± 1.1
Omniglot	**94.7 ± 0.4**	94.5 ± 0.4	94.5 ± 0.4
Aircraft	88.9 ± 0.5	89.3 ± 0.4	**89.4 ± 0.4**
Birds	80.4 ± 0.7	78.6 ± 0.8	**80.7 ± 0.8**
Textures	74.5 ± 0.7	73.3 ± 0.7	**77.2 ± 0.7**
Quick Draw	81.9 ± 0.6	81.6 ± 0.6	**82.5 ± 0.6**
Fungi	66.4 ± 0.9	67.6 ± 0.9	**68.1 ± 0.9**
VGG Flower	91.3 ± 0.5	89.6 ± 0.5	**92.0 ± 0.5**
Traffic Sign	63.2 ± 1.1	62.5 ± 1.2	**63.3 ± 1.2**
MSCOCO	56.6 ± 1.0	55.6 ± 1.1	**57.3 ± 1.0**
MNIST	94.7 ± 0.4	**95.3 ± 0.4**	94.7 ± 0.4
CIFAR-10	73.8 ± 0.7	72.9 ± 0.8	**74.2 ± 0.8**
CIFAR-100	62.1 ± 1.0	60.8 ± 1.0	**63.6 ± 1.0**

Table 4.8: **Quantitative analysis of knowledge distillation loss functions on representations and predictions**. Mean accuracy, 95% confidence interval are reported. 'Ours w/o ℓ_p' and 'Ours w/o ℓ_f' means we only apply CKA function on representations and apply KL divergence on predictions for knowledge distillation, respectively. 'Ours ($\ell_f + \ell_p$)' is our model using both CKA on features and KL on predictions. All results are obtained with pre-classifier alignment during meta-test stage.

TSA incorporated with different feature extractors, *i.e*. SDL, MDL, and URL, the sensitivity analysis for number of iterations, initialization analysis of adapters, layer analysis for adapters, and decomposed residual adapters. We summarize results in figures and refer to Appendix C.2.4 for more detailed results.

TSA with different feature extractors Tab. 4.9 shows the results of our TSA (the proposed residual adapters in matrix form) when incorporated to different feature extractors, single domain model with ResNet-18 backbone (SDL-ResNet-18) pretrained on ImageNet, single domain model with ResNet-34 (SDL-ResNet-34) pretrained on ImageNet, vanilla multi-domain learning (MDL) and URL introduced in this chapter. We see that attaching and learning residual adapters can significantly improve the performance on all domains over SDL-ResNet-18, SDL-ResNet-34 and MDL and obtain better performance on most domains over URL (11 out of 13 domains). This strongly indicates that our method can efficiently adapt the model for unseen categories and domains with few support samples while being agnostic to the feature extractor with different backbone and resolution of images.

Sensitivity analysis for number of iterations. In our TSA (residual adapters in matrix form with pre-classifier alignment) method, we optimize the attached parameters (α, β) with 40 iterations. Figure 4.6 reports the results with 10, 20, 40, 60 iterations and

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
MDL	53.4±1.1	93.8±0.4	86.6±0.5	78.6±0.8	71.4±0.7	81.5±0.6	61.9±1.0	88.7±0.6	51.0±1.0	49.7±1.1	94.4±0.3	66.7±0.8	53.6±1.0
TSA (MDL)	**55.6±1.0**	**94.3±0.4**	**86.7±0.5**	**79.4±0.8**	**73.2±0.8**	**81.7±0.6**	**64.0±0.9**	**90.9±0.5**	**81.1±0.9**	**51.4±1.1**	**96.9±0.3**	**78.5±0.8**	**64.3±1.1**
URL (Ours)	58.8±1.1	94.5±0.4	89.4±0.4	80.7±0.8	77.2±0.7	**82.5±0.6**	**68.1±0.9**	92.0±0.5	63.3±1.2	57.3±1.0	94.7±0.4	74.2±0.8	63.6±1.0
TSA (URL)	**59.5±1.0**	**94.9±0.4**	**89.9±0.4**	**81.1±0.8**	**77.5±0.7**	81.7±0.6	66.3±0.9	**92.2±0.5**	**82.8±1.0**	**57.6±1.0**	**96.7±0.4**	**82.9±0.7**	**70.4±1.0**
SDL-ResNet-18	55.8±1.0	67.4±1.2	49.5±0.9	71.2±0.9	73.0±0.6	53.9±1.0	41.6±1.0	87.0±0.6	47.4±1.1	53.5±1.0	78.1±0.7	67.3±0.8	56.6±0.9
TSA (SDL-ResNet-18)	**59.5±1.1**	**78.2±1.2**	**72.2±1.0**	**74.9±0.9**	**77.3±0.7**	**67.6±0.9**	**44.7±1.0**	**90.9±0.6**	**82.5±0.8**	**59.0±1.0**	**93.9±0.6**	**82.1±0.7**	**70.7±0.9**
SDL-ResNet-34	62.2±1.1	72.8±1.1	62.9±0.9	79.6±0.8	75.6±0.6	64.5±0.8	47.4±1.1	90.4±0.6	54.8±1.0	56.1±1.0	79.3±0.6	83.0±0.6	74.8±0.8
TSA (SDL-ResNet-34)	**63.7±1.0**	**82.6±1.1**	**80.1±1.0**	**83.4±0.8**	**79.6±0.7**	**71.0±0.8**	**51.4±1.2**	**94.0±0.5**	**81.7±0.9**	**61.7±0.9**	**94.6±0.5**	**86.0±0.6**	**78.3±0.8**

Table 4.9: Results of attaching residual adapters to different baselines. 'SDL-ResNet-18' is the single domain model with ResNet-18 backbone pretrained on ImageNet. 'SDL-ResNet-34' is the single domain model with ResNet-34 backbone pretrained on ImageNet. 'MDL' is a vanilla Multi-Domain Learning (MDL) model trained on eight seen datasets jointly.

indicates that our method (solid green) converges to a stable solution after 20 iterations and achieves better average performance on all domains than the baseline URL (dash green).

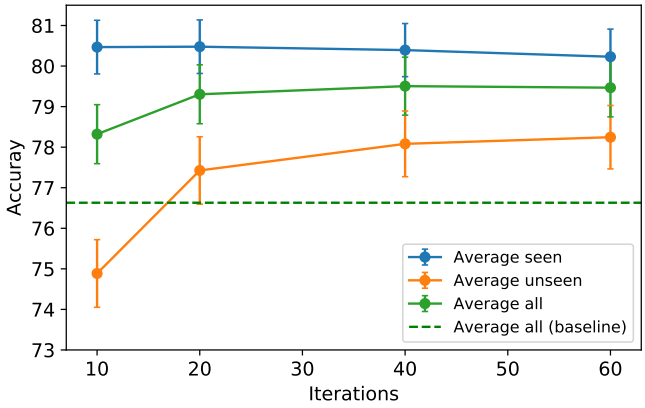


Figure 4.6: Sensitivity of performance to number of iterations.

Initialization analysis for adapters. Here, we investigate using different initialization strategies for adapters: i) Identity initialization: in this chapter we initialize each residual adapter as an identity matrix scaled by a scalar δ and we set $\delta = 1e-4$; ii) randomly initialization: alternatively, we can randomly initialize each residual adapter. As shown in Tab. 4.10, we can see that our methods with different initialization strategies obtain similar results, which indicates that our method works also with randomly initialization and again verifies the stability of our method. More detailed results with different backbones are shown in Appendix C.2.4.

Layer analysis for adapters. Here we investigate whether it is sufficient to attach the adapters only to the later layers. We evaluate this on ResNet18 which is composed of four blocks and attach the adapters to only later blocks (block4, block3,4, block2,3,4

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours(URL)-I	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0
Ours(URL)-R	58.8±1.1	94.9±0.4	90.5±0.4	81.8±0.6	77.7±0.7	82.3±0.6	66.8±0.9	92.6±0.5	83.7±0.8	57.7±1.1	96.9±0.4	82.5±0.7	72.0±0.9

Table 4.10: Initialization analysis of adapters. 'Ours(URL)-I' indicates our method using URL as the pretrained model and initializing residual adapters as identity matrix (scaled by $\delta = 0.0001$) while 'Ours(URL)-R' means our method initialize residual adapters randomly.

and block-all, see Fig. 4.5). Figure 4.7 shows that applying our adapters to only the last block (block4) obtains around 78% average accuracy on all domains which outperforms the URL. With attaching residual adapters to more layers, the performance on unseen domains is improved significantly while the one on seen domains remains stable.

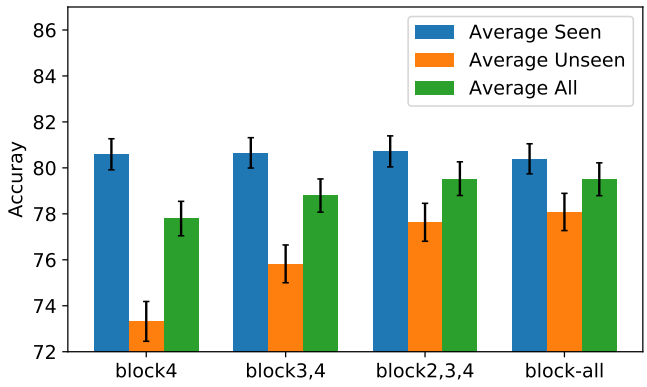


Figure 4.7: Block (layer) analysis for adapters.

Decomposing residual adapters. Here we investigate whether one can reduce the number of parameters in the adapters while retaining its performance by using matrix decomposition (see Sec. 4.3). As in deep neural networks, the adapters in earlier layers are relatively small; we then decompose the adapters in the last two blocks only where the adapter dimensionality goes up to 512×512. Figure 4.8 shows that our method can achieve good performance with less parameters by decomposing large residual adapters, (*e.g.* when $N = 32$ where the number of additional parameters equal to around 4% vs 13%, the performance is still comparable to the original form of residual adapters, *i.e.* N=0). We refer to Appendix C.2.4 for more details.

4.4.7 Qualitative results

We qualitatively analyze our methods (URL and TSA) and compare it to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b)

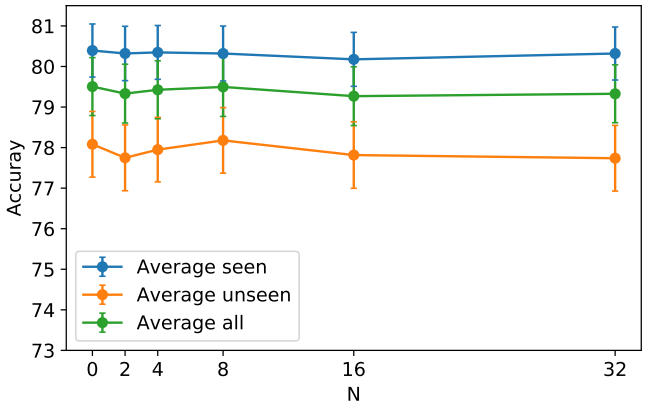


Figure 4.8: Decomposed residual adapters on block-3,4.

in all test datasets (See Appendix C.2.5) and here we report results in Birds (seen) dataset Fig. 4.9, CIFAR-100 (unseen) dataset and the Traffic Sign (unseen) dataset Fig. 4.10, by illustrating the nearest neighbors given a query image as in the prior work (Doersch et al., 2020). It is clear that our methods produce more correct neighbors than other methods. While other methods retrieve images with more similar colors, shapes and backgrounds, our methods are able to retrieve semantically similar images. More specifically, as shown in Fig. 4.9 (seen dataset), by learning task-agnostic weights by URL and adapting the task-agnostic weight by TSA, our whole method correctly produces neighbors of the bird in the query image while other methods pick images with similar appearances or similar background, *e.g.* images with twigs. In an unseen dataset (*e.g.* Fig. 4.11), other methods including SUR, URT are distracted by the blue background. While the URL learns more generalized features and is less distracted by the color and background. By using our TSA with the URL task-agnostic model, our TSA method selects the correct shark images. It again suggests that the URL method provides an efficient and effective way to learn a single task-agnostic network and generalized features from multiple domains and our TSA method is able to quickly adapt the features for unseen few-shot tasks. In Fig. 4.10, other methods and even URL mainly retrieve the triangle sign while our TSA method is able to retrieve the correct sign with illumination distortion.

4.5 Conclusion and Limitations

In this chapter, we first demonstrate that learning a single set of universal representations integrated with a feature refining step achieves superior performance than existing task-agnostic weight learning methods in the recent Meta-Dataset benchmark. We then

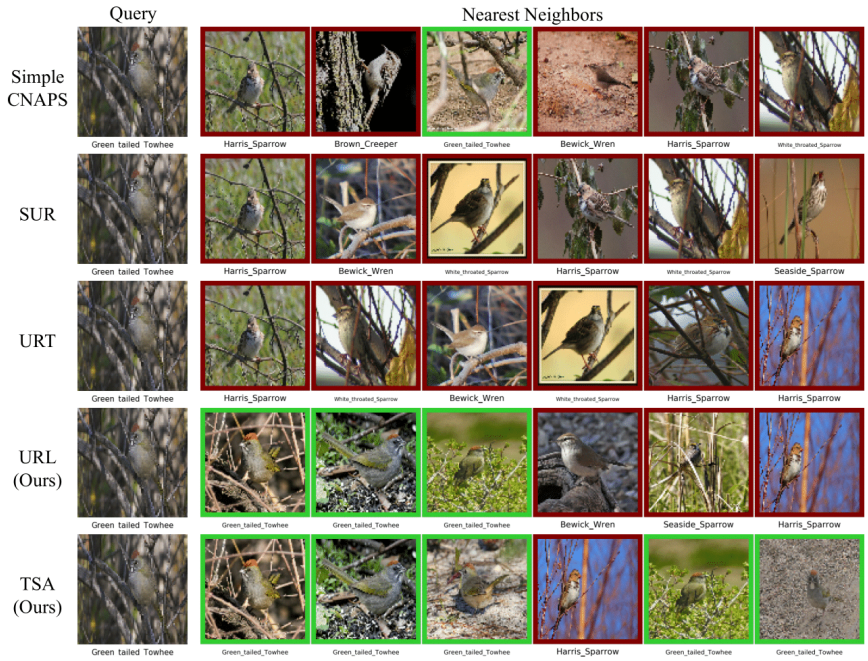


Figure 4.9: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Birds. Green and red colors indicate correct and false predictions respectively.

Figure 4.10: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Traffic Sign. Green and red colors indicate correct and false predictions respectively.

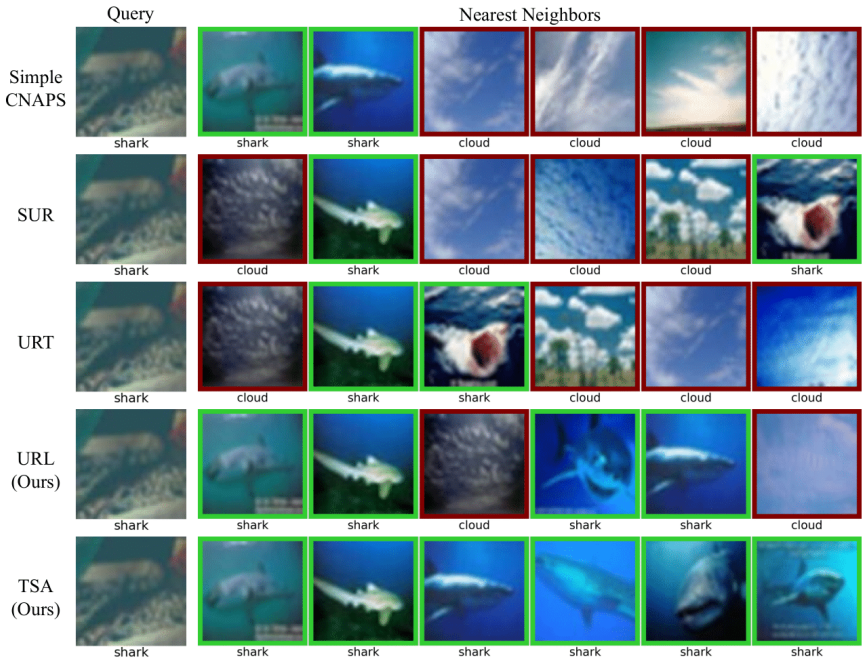


Figure 4.11: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in CIFAR-100. Green and red colors indicate correct and false predictions respectively.

investigate various strategies for adapting deep networks to few-shot classification tasks and show that light-weight adapters connected to a deep network with residual connections achieves strong adaptation to new tasks and domains only from few samples and obtains state-of-the-art performance while being efficient in the challenging Meta-Dataset benchmark. We demonstrate that the proposed solution can be incorporated to various feature extractors with a negligible increase in the number of parameters.

Our method has limitations too. We build our method on existing backbones such as ResNet-18 and ResNet-34, employ fixed adapter parameterizations and connection types which may not be optimal for every layer and task in multi-domain few-shot learning. Thus it would be desirable to have more flexible adapter structures that can be altered and tuned based on the target task.

Chapter 5

Conclusion and Future Work

Learning universal representations across tasks and domains presents several challenges for standard computer vision models. This thesis presented new methods for learning compact universal representations, *i.e*. a single deep network, that performs well on multiple vision tasks and various visual domains, formulating a more practical and realistic setting for multi-task learning and proposing a new method for learning MTL model from partially annotated data, as well as, novel approaches for cross-domain few-shot learning.

In the following, Sec. 5.1 discusses existing limitations and highlights future research directions for universal representation learning, cross-task relations learning, learning models from limited supervision, including learning MTL model from limited labels and cross-domain few-shot learning. This is followed by a discussion in Sec. 5.2 that aims to assess the broader impact of the methods and ideas proposed in this thesis.

5.1 Limitations and Future work

In this thesis, the common thread of the proposed solution for three related problems is learning universal representations from multiple tasks or domains through a single network. And the thesis shows that it is possible to learn universal representations within a single network for different problems by the proposed architecture-agnostic algorithms. This is more data efficient and important for platforms that have limited resources and are demanded to be versatile. In addition, this learns more complete representations and also allows knowledge transfer between tasks which can help improve performance of tasks that do not have enough data (*e.g*. medical problems). This will be helpful for better understanding the optimization in universal representation learning, studying

relations across tasks, and grouping tasks in the future. The solutions proposed in this thesis can also be useful for learning compact universal representations by leveraging existing resources of different problems (*e.g*. datasets that are collected for different problems) for these problems without collecting new datasets for all problems, which is more practical and more scalable for the community. It would be also beneficial for studying better ways of learning representations from multiple tasks and domains.

Despite the advance in universal representation or multi-task learning, the difficulties of learning a single network that works well for many tasks remain. It remains unclear whether the standard optimizers with adaptive gradient updates are optimal when learning multiple tasks, whether any network architecture provides more balanced solution when its weights are learned over multiple tasks, whether any pretraining strategies provide a better start point for learning multiple tasks, whether any regularization enables a more balanced optimization when learning multiple tasks. To this end, a more systematic analysis of unbalanced optimization in multi-task learning is needed and will be beneficial for better studying the interference/transferability among tasks.

Beyond these points, most existing benchmarks for evaluating universal representation learning are either too small (*e.g*. small dataset size and number of tasks) or noisy, *e.g*. labels in NYU-v2 (Silberman et al., 2012) and Pascal-Context (Chen et al., 2014) are noisy. Though some clean and large-scale synthetic datasets (Eftekhar et al., 2021) and large scale data, *e.g*. Taskonomy (Zamir et al., 2018), which is pseudo-labelled for multiple tasks are proposed, clean and large-scale real datasets are needed for better evaluating universal representation learning methods. Clean large-scale dataset for universal representation learning is inherently costly and challenging to collect and to annotate with clean groundtruth for many tasks. Alternatively, as discussed above, it would be interesting to leverage existing large-scale and clean datasets (different one for each problem) and learn universal representations from multiple datasets for multiple tasks.

Apart from this, existing evaluation metrics in universal representations are limited to testing accuracy (loss) for each task or domain. However, it would be useful to include other ways for the evaluation of universal representations as the goal is not only achieving good performance in individual tasks but also obtaining balanced solutions compared to single-task learning networks. To this end, the multi-task performance metric (Vandenhende et al., 2021) and multi-domain performance score (Rebuffi et al., 2017a) provide better ways of measuring how well the universal representations are by comparing the performance of universal network and task/domain-specific networks.

Another possible way is to include visual comparison of prediction errors made by different approaches to better study the improvement achieved by the proposed methods. Alternatively, similar to the analysis in MultiMAE (Bachmann et al., 2022), one can change parts of the input and visualize the predictions of different tasks to monitor the impact of the input changes (*e.g*. whether all tasks behave similarly (consistently)). Alternatively, one can evaluate the learned universal representations on downstream problems which are not seen during training without finetuning on the downstream problems and with finetuning using limited samples and so on.

In universal representation learning, tasks are defined manually (*e.g*. labels) and learned together without considering the intra-task and inter-task relations. It would be interesting to automatically detect and group related tasks and leverage the relations among related tasks for universal representation learning. In addition, the effect of architecture designs and optimization strategies are also important when detecting or grouping related tasks/problems (*e.g*. one can learn universal representations hierarchically). Apart from detecting tasks, designing architectures that can automatically and efficiently share related information among tasks and keep the rest task-specific while keeping less computational cost is important as well. Though recent efforts have been made by Neural Architecture Search (Guo et al., 2020; Vandenhende et al., 2020a), attention layers (Liu et al., 2019; Misra et al., 2016), learning relations using transformers (Bruggemann et al., 2021; Ye and Xu, 2022), ensembling tasks predictions (Yeo et al., 2021), factoring networks to different modulars (Mallya et al., 2018) or sub-networks (Yang et al., 2022), it would be interesting to explore more network architectures (*e.g*. Vision Transformers (Dosovitskiy et al., 2020)) that are shown to be more effective for universal representation learning (Ye and Xu, 2022).

Instead of learning a single universal network that requires data from multiple tasks to learn from them, one can also learn to merge pretrained task-specific networks for a downstream problem (Matena and Raffel, 2021). This provides a cheaper way for learning universal representations and for transferring knowledge for the downstream tasks (Matena and Raffel, 2021). Besides, recent task transferability analysis (Zamir et al., 2018; Pándy et al., 2022) shows that the success of positive knowledge transfer across tasks is heavily dependent on both the source and target task types. Given a target task, identifying related tasks and domains (data) for pretraining is important for learning useful representations for the target task. Instead of pretraining the representations with all possible combination candidates and testing them on the target tasks, which is costly and cumbersome, developing methods that can automatically and efficiently

identify related source tasks/domains can greatly reduce the cost and help study the transferability across tasks.

Universal representations learning. Learning universal representations that can generalize well to multiple tasks or various visual domains is one of important goals in computer vision and is crucial for platforms with limited resources such as mobile devices, autonomous cars and so on. Despite good performance in universal representations learning for related tasks and domains achieved by the URL method proposed in Chapter 2, it has limitations too. More specifically, it requires multiple models to learn universal representations at train time which can be computationally expensive in the presence of many tasks. This problem can be potentially alleviated by hierarchically grouping similar tasks (*e.g*. evaluating task grouping candidates and grouping tasks via early stopping or high order approximation (Standley et al., 2020) or grouping tasks through a inter-task affinity by examining the effect to which one task's gradient would increase or decrease another task's loss (Fifty et al., 2021)) and learning a single model on them, and then distilling their knowledge into universal representations.

So far, we focused on multi-task or multi-domain learning. Another interesting extension is to extend the proposed URL method to problems that involve multiple tasks and multiple domains at the same time (*i.e*. multi-domain multi-task learning) (He et al., 2022; Ghiasi et al., 2021) such as semantic segmentation and depth estimation from different cities, time of day (*e.g*. day or night time), weather (*e.g*. clear, rainy, snowy), or domains.

Existing multi-task and multi-domain learning also limited to the presence of data for all tasks and domains at the same time. Ours (Chapter 2) has the same too. However, one a new task or domain is added, standard methods requires to merge all tasks and domains for learning a new model for all tasks. This would result in large computational, storage and memory cost and some data of previous tasks may not be accessible when a new task is added. Thus, it is more desired to adapt the model for the new task without the access to previous data and prevent the catastrophic forgetting of previous tasks (Kirkpatrick et al., 2017; Lee et al., 2017) as this does not require the access to previous data and is more computationally efficient. Thus, one potential solution is to exploit the idea of continual learning to learn the model to accommodate new tasks without forgetting new tasks. Unlike conventional continual learning work (Castro et al., 2018; Rebuffi et al., 2017b) that focus on continually learning new image classification tasks, it would be interesting to focus on a more realistic setting where semantic tasks such as semantic segmentation, depth estimation are considered.

Multi-task partially supervised learning. Standard multi-task learning methods requires fully annotated dataset where all tasks labels are available for each image. To facilitate the research in multi-task learning, a more realistic and general setting, multi-task partially supervised learning is formulated in this thesis. Though it is shown that the propose method in Chapter 3 enables the efficient learning of cross-task relations through a conditioned network, modeling cross-task relations for all task pairs may not be required. Thus it would be desirable to automatically identify which tasks are closely related (*i.e*. the transferability between tasks and domains (Zamir et al., 2018; Mensink et al., 2022)) by measuring feature similarity (Dwivedi and Roig, 2019; Dwivedi et al., 2020), efficient transferability metrics (Fifty et al., 2021) such as H-score (Bao et al., 2019), LEEP (Nguyen et al., 2020), LogME (You et al., 2021), or GBC (Pándy et al., 2022) and learn such cross-task relations only between related tasks.

Another potential research direction is to extend multi-task partially supervised learning for multi-domain multi-task learning, where multiple tasks and multiple domains are presented at the same time as mentioned above. such as semantic segmentation and depth estimation from different cities (or domains). In MDMTL, for example, semantic segmentation and depth estimation from different cities (or domains) (He et al., 2022; Ghiasi et al., 2021), the data for depth estimation is unlabelled data for segmentation and the data may contains novel categories which are not seen in data for segmetnation. Instead of learning MTL model on MDMTL setting by only applying supervised loss over labelled tasks, one can explore the task-specific information of unlabelled task (Ghiasi et al., 2021). Alternatively, learning to augment the label space with novel categories for the unlabelled tasks is important and learning cross-task relations can be potential benefits the learning of MTL models in MDMTL.

Cross-domain few-shot learning. The method proposed in Chapter 4 aims to learn a well-generalized single task-agnostic model from multiple domains and efficiently adapt the task-agnostic model with light-weight adapters from very few samples to accommodate new domains and tasks. And the proposed method is built on existing backbones such as ResNet-18 and ResNet-34 with fixed adapter parameterizations and connection types which may not be optimal for every layer and task in cross-domain few-shot learning. It would be desirable to have more flexible adapter structures that can be altered and tuned based on the target task.

In addition, instead of attaching adapters to all layers which can be computationally costly, a more flexible strategy is to learn to select which layers to be attach adapters. Another potential solution is to explore more powerful architecture such as vision

transformer (Dosovitskiy et al., 2020) with more effective pretraining strategies (Hu et al., 2022) to learn more general features in meta-training for generalizing to unseen domains and tasks.

5.2 Broader Impact

Universal representation learning over multiple tasks and domains are crutial for smart devices such as smart phones and autonomous vehicle that only have limited resource and are demanded to be versatile. Chapter 2 showed that learning general features from multiple tasks and domains is an important step for better generalization in various computer vision problems including multiple dense prediction, multi-domain image classification and cross-domain few-shot learning problems in Chapter 4. By distilling representations from multiple task-specific or domain-specific networks, the URL method can successfully learn a single set of universal representations after aligning them via small task/domain-specific adapters. These representations are compact and generalize better to unseen samples, tasks and domains in multiple benchmarks.

Our URL method obtains the best results in multiple standard benchmarks Visual Decathlon (Rebuffi et al., 2017a), MetaDataset (Triantafillou et al., 2020), NYU-v2 (Silberman et al., 2012), and Cityscapes (Cordts et al., 2016). The idea of our URL method has shown to be effective in Multilingual ASR (Li et al., 2021a), WiFi-based Sensing (Zhang et al., 2021), Stereo Matching (Liu et al., 2022a), Surgical Scene Understanding (Seenivasan et al., 2022), Object Detection (Zhang et al., 2022), Large-Scale Embedding Retrieval Systems (Ramanujan et al., 2022) and extended for Eye Authentication and Presentation Attack Detection (Dhar et al., 2022)

While existing literatures mainly focus on either learning multiple task from a single domain (Vandenhende et al., 2021; Caruana, 1997) or learning multiple domains for a single task (Rebuffi et al., 2017a; Zhou et al., 2022; Lambert et al., 2020; Ranftl et al., 2020), relatively less works focus on learning a single model in multi-domain multi-task learning (MDMTL) (He et al., 2022; Ghiasi et al., 2021), *e.g*. semantic segmentation and depth estimation from different domains (*e.g*. cities, scene, weather, time of day). The proposed URL can be a useful tool to learn compact and generalized representations in multi-domain multi-task learning.

Multi-task partially supervised learning and cross-task relations learning. In real-world scenarios, collecting a dataset that contains labels for all (dense prediction) tasks is very costly (*e.g*. medical imaging, self-driving) while it is more common

that the dataset collected dataset is partially annotated or the dataset consists subdatasets each annotated for different tasks. To facilate the research in this direction, this thesis formulates the multi-task partially supervised learning setting where not all tasks labels are available in each image and each image is annotated for at least one task.

Learning multi-task learning models from such partially annotated data is challenging and it is shown in Sec. 3.4 that good performance of the MTL model on multi-task partially supervised learning is achieved by the proposed method in Chapter 3 that learns consistency cross related tasks by the proposed regularized conditional joint space mapping in an efficient way. The proposed new setting and method can potentially be helpful for cancer tissue segmentation from medical images, landslide and earthquake damage estimation from satellite images where it is challenging and expensive to obtain labelled data.

Cross-domain few-shot learning which aims to adapt a model for previously unseen domains and tasks with limited data, is important for industry applications such as personalization (Massiceti et al., 2021), speech recognition (Xiong et al., 2018) which require efficient knowledge transfer across tasks with few samples. This is typically solved by learning a task-agnostic model (universal representations) over a diverse dataset or multiple datasets and adapting the task-agnostic model for unseen tasks by estimating task-specific weights from the support set for the target task.

Given that the proposed URL and TSA method in Chapter 4 perform better than existing few-shot learning methdos in recent challenging MetaDataset, it can potentially be used in many machine learning applications such as personalization (Massiceti et al., 2021) where it is costly and difficult to collect data for the target task and the paltforms only has limited resources for adapting the models.

Appendix A

Learning Universal Representations

A.1 Implementation Details

A.1.1 Multi-task Dense Prediction

We evaluate our method on learning universal representations for performing multiple dense prediction tasks on two standard multi-task learning benchmarks NYU-v2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) as Liu et al. (2019, 2021a). Here, we provide more details about our implementation.

We follow the training and evaluation settings in the prior works (Liu et al., 2019, 2021a) for both single-task and multi-task learning in both datasets. More specifically, *NYU-V2* (Silberman et al., 2012) contains RGB-D indoor scene images, where we evaluate performances on 3 tasks, including 13-class semantic segmentation, depth estimation, and surface normals estimation. We use the true depth data recorded by the Microsoft Kinect and surface normals provided by Eigen and Fergus (2015) for depth estimation and surface normal estimation as in Liu et al. (2019). All images are resized to 288×384 resolution as Liu et al. (2019). We follow the default setting in the prior work (Silberman et al., 2012; Liu et al., 2019) where 795 and 654 images are used for training and testing, respectively. *Cityscapes* (Cordts et al., 2016) consists of street-view images, which are labeled for two tasks: 7-class semantic segmentation[1] and depth estimation. We resize the images to 128×256 to speed up the training as Liu et al. (2019).

In both NYU-v2 and Cityscapes, we follow the training and evaluation protocol in

[1]The original version of Cityscapes provides labels 7&19-class semantic segmentation. We follow the 7-class semantic segmentation evaluation protocol as in MTAN (Liu et al., 2019) to be able to compare to the related works.

the prior works (Liu et al., 2019). We consider two backbone cases. For encoder-based ones, we use the SegNet (Badrinarayanan et al., 2017) as the backbone. Following Liu et al. (2019), we use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation in Cityscapes, and cosine similarity loss for surface normal estimation in NYU-v2. We use the exactly same hyper-parameters including learning rate, optimizer and also the same evaluation metrics, mean intersection over union (mIoU), absolute error (aErr) and mean error (mErr) in the predicted angles to evaluate the semantic segmentation, depth estimation and surface normals estimation task, respectively in the prior work (Liu et al., 2019). More specifically, we use Adam as the optimizer and train the model for 200 epochs as Liu et al. (2019) with the learning rate of 0.0001 which is halved at the 100-th epoch. The batch size is set to 2 and 8 for NYU-v2 and Cityscapes, respectively. We use the same augmentation as in MTAN (Liu et al., 2019), such as random crop, flipping.

For the decoder-based methods, as the MTI-Net (Vandenhende et al., 2020b) requires multi-scale feature extractor (encoder), we follow Vandenhende et al. (2020b, 2021), use the HRNet-18 backbone (Sun et al., 2019a) initialized with ImageNet pretrained weight as the feature encoder. As in the prior works (Vandenhende et al., 2020b, 2021), the batch size is set to 8 for both NYU-v2 and Cityscapes. We use the same loss functions, evaluation metrics, and training and evaluation protocol as the encoder-based methods.

In our method, we use the uniform loss weights (*i.e.* $\lambda^t = 1$ for all tasks) for task-specific losses, unless stated otherwise. As we do *not* minimize the difference between predictions of the universal and single-task networks, we set λ_p^t in Eq. (3) to zero. We then first split the train set as train and validation set to search $\lambda_f^t \in \{1,2\}$ by cross-validation and train our network on the whole training set. We set λ_f^t to 1 for semantic segmentation and depth and 2 for surface normal estimation. For the optimization of adapters, we use Adam as optimizer with the learning rate of 0.01 and weight decay of 0.0001 and anneal the learning rate to 0 using cosine scheduler.

A.1.2 Multi-domain Learning

Dataset. The *Visual Decathlon Benchmark* (Rebuffi et al., 2017a) consists of 10 different well-known datasets: including ILSVRC_2012 (ImNet) (Russakovsky et al., 2015), FGVC-Aircraft (Airc.) (Maji et al., 2013), CIFAR-100 (C100) (Krizhevsky et al., 2009), Daimler Mono Pedestrian Classification Benchmark (DPed) (Munder and Gavrila, 2006), Describable Texture Dataset (DTD) (Cimpoi et al., 2014), German

Traffic Sign Recognition (GTSR) (Houben et al., 2013), Flowers102 (Flwr) (Nilsback and Zisserman, 2008), Omniglot (OGlt) (Lake et al., 2015), Street View House Numbers (SVHN) (Netzer et al., 2011), UCF101 (UCF) (Soomro et al., 2012). In this benchmark (Rebuffi et al., 2017a), images are resized to a common resolution of roughly 72 pixels to accelerate evaluation.

Implementation details. Each dataset has train/val/test splits and we follow the standard training and evaluation protocol (Rebuffi et al., 2017a, 2018). For ImageNet, we random crop and center crop a 72×72 patch for training and evaluation respectively. For other datasets, as the aspect ratio in Airc. and DPed is quite different from other datasets, we first resize the images in both datasets to 72×72, and we random crop and center crop a 64×64 patch for training and evaluation as Rebuffi et al. (2017a).

We follow Rebuffi et al. (2017a, 2018), use the ResNet-26 (He et al., 2016) as the backbone for domain-specific network and universal network. In our universal network, the backbone (*i.e.* ResNet-26) is shared across all domains and followed by domain-specific linear classifiers. We use the same data augmentation as in the prior works (Rebuffi et al., 2017a, 2018), such as random crop, flipping. We use SGD as optimizer with weight decay and train domain-specific networks and our universal network for 120 epochs as Rebuffi et al. (2017a, 2018).

For domain-specific models, *i.e.*, learning one model per domain, we first train a model on ImageNet with learning rate of 0.1 and weight decay of 0.0005 and we finetune it for other datasets with weight decay of 0.0005. For finetuning on each dataset (except ImageNet), we set learning rate to 0.1 for Airc., Flwr, OGlt, SVHN, UCF and 0.01 for C100, DPed, DTD, GTSR.

Here we set the loss weights to 1 (*i.e.* $\lambda^t = 1$ for all tasks) and perform cross-validation to search loss weights $(\lambda^t_f, \lambda^t_p)$ in $\{0.1, 1, 10\}$ for knowledge distillations on features and predictions, and set λ_f and λ_p to 10 for ImageNet, 0.1 for DPed, and 1 for other datasets. We optimize our universal network and vanilla MDL using SGD as optimizer as Rebuffi et al. (2017a, 2018) with the learning rate of 0.01 and weight decay of 0.0001 for 120 epochs. The learning rate is scaled by 0.1 at 80-th and 100-th epoch as in the prior works (Rebuffi et al., 2017a, 2018). We optimize the parameters of adapters by using the Adam as optimizer with the learning rate of 0.01 which is annealed to zero using cosine scheduler and weight decay of 0.0005. We evaluate our method though the official online evaluation provided by Rebuffi et al. (2017a).

A.2 More results

Here, we analyze our method and qualitatively compare our method to STL, MTL with Uniform loss weights and the best compared method, *i.e*. IMTL-H (Liu et al., 2021c) and other related methods for multi dense prediction problem on NYU-v2 with SegNet backbone (see Fig. A.1). We can see that, Uniform baseline obtains improvement on segmentation and depth estimation over STL, while it performs worse in surface normal estimation. Though dynamically balancing the loss values with IMTL-H improves the overall performance, it still performs worse in surface normal estimation. Finally by distilling representations from single-task learning model to the universal network, our method can produce better or comparable results to STL, *i.e*. our method produces similar outputs for surface normal as STL and more accurate predictions for segmentation and depth estimation as our method enables a balanced optimization of universal network and a task can be benefited from another one. This indicates the effectiveness of our method on learning shared representations for multiple dense predictions.

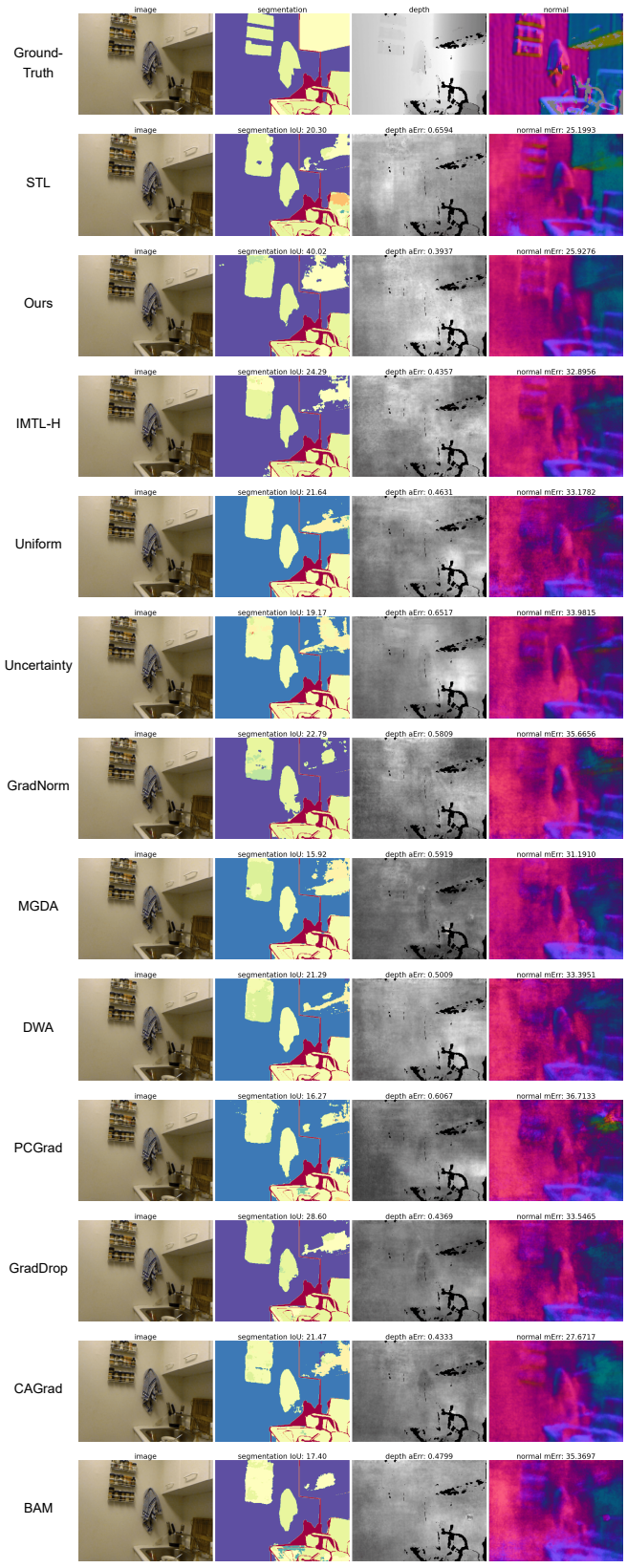


Figure A.1: **Qualitative results on NYU-v2**. The fist column shows the RGB image, the second column plots the ground-truth or predictions with the IoU (↑) score of all methods for semantic segmentation, the third column presents the ground-truth or predictions with the absolute error (↓), and we show the prediction of surface normal with mean error (↓) in the last column.

Appendix B

Multi-task Learning from Partially Annotated Data

B.1 Implementation Details

Tab. B.1 and Tab. B.2 provide an overview of the experimental settings, in particular report the number of train and test samples for each benchmark and number of labels used in different partially annotated settings respectively. Next we explain the implementation details for each dataset.

Cityscapes. The Cityscapes dataset (Cordts et al., 2016) contains 3475 labelled images. Following Liu et al. (2019), we use 2975 images for training and 500 images for testing. In multi-task partially supervised learning setting, we consider the one-label setting in Cityscapes, as there are only two tasks in total, *i.e*. we randomly select and keep label only for 1 task for each training image, resulting in 1487 training images annotated for segmentation and 1488 training images labelled for depth estimation, as shown in Tab. B.2.

We follow the training and evaluation protocol in the prior work (Liu et al., 2019) and we use SegNet (Badrinarayanan et al., 2017) as the MTL backbone for all methods, use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation. We use the exact same hyper-parameters including learning rate, optimizer as Liu et al. (2019). More specifically, we use Adam optimizer with a learning rate of 0.0001 and train all models for 200 epochs with a batch size of 8 and halve the learning rate at the 100-th epoch. We also employ the same evaluation metrics, mean intersection over union (mIoU) and absolute error (aErr) to evaluate the semantic segmentation and depth estimation task, respectively as in the prior work (Liu et al., 2019).

For our model, we use the encoder architecture of SegNet for instantiating the joint pairwise task mapping ($\bar{m}_{\vartheta}$) and include one convolutional layer as task-specific input layer in $\bar{m}_{\vartheta}$. For `Direct-Map` and `Perceptual-Map`, following Zamir et al. (2020) we use the whole SegNet as the cross-task mapping functions. We use the same data augmentations from the updated implementation in MTAN (Liu et al., 2019)[1], *i.e.* random crops and rand horizontal flips.

Dataset	# Train	# Test	Segmentation	Depth	Human Parts	Normals	Saliency	Edges
Cityscapes (Cordts et al., 2016)	2975	500	✓	✓	- -	- -	- -	- -
NYU-v2 (Silberman et al., 2012)	795	654	✓	✓	- -	✓	- -	- -
PASCAL (Chen et al., 2014)	4998	5105	✓	- -	✓	✓	✓	✓

Table B.1: Details of multi-task benchmarks.

NYU-v2. The dataset (Silberman et al., 2012) contains 795 training images and 654 test images. To evaluate the multi-task partially supervised learning, we consider one-label and random-label settings. For one-label setting, we randomly select and keep label for only 1 task for each training image, resulting in 265 images with annotation for segmentation, 265 images labelled for depth estimation and 265 images for surface normal. For random-label setting, we randomly select and keep labels for at least 1 and at most 2 tasks (1.49 labels per image), *i.e.* 392 images for semantic segmentation, 408 images for depth estimation, 385 images for surface normal, as shown in Tab. B.2.

We follow the training and evaluation protocol in the prior work (Liu et al., 2019) and we use the SegNet (Badrinarayanan et al., 2017) as the MTL backbone for all methods. Following Liu et al. (2019), we use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation and cosine similarity loss for surface normal estimation, use the same optimizer and hyper-parameters, *i.e.* Adam optimizer with a learning rate of 0.0001. We train the all model for 200 epochs with a batch size of 2 and halve the learning rate at the 100-th epoch and employ the same evaluation metrics, mean intersection over union (mIoU), absolute error (aErr) and mean error (mErr) in the predicted angles to evaluate the semantic segmentation, depth estimation and surface normals estimation task, respectively as in MTAN (Liu et al., 2019).

We use the encoder of SegNet architecture for the joint pairwise task mapping ($\bar{m}_{\vartheta}$) and one convolutional layer as task-specific input layer in $\bar{m}_{\vartheta}$. For `Direct-Map` and `Perceptual-Map`, following Zamir et al. (2020) we use the whole SegNet as the cross-task mapping functions. To regularize training, we use the exact same data

[1] https://github.com/lorenmt/mtan

Dataset	# label	# labelled images					
		Segmentation	Depth	Human Parts	Normals	Saliency	Edges
Cityscapes (Cordts et al., 2016)	one	1487	1488	- -	- -	- -	- -
NYU-v2 (Silberman et al., 2012)	random	392	408	- -	385	- -	- -
	one	265	265	- -	265	- -	- -
PASCAL (Chen et al., 2014)	random	2450	- -	2553	2480	2445	2557
	one	1000	- -	999	1000	1000	999

Table B.2: Details about multi-task partially supervised learning settings in three benchmarks used in this work. 'random' means the random-label setting where each training image has a random number of task labels and 'one' indicates the one-label setting where each training image is annotated with one task label. '# labelled images' shows the number of images containing labels for each task, *e.g*. segmentation.

augmentations from the updated implementation from the prior work (Liu et al., 2019), *e.g*. random crops and rand horizontal flips augmentations.

PASCAL-context. The dataset (Chen et al., 2014) contains 4998 training images and 5105 testing images for five tasks, *i.e*. semantic segmentation, human parts segmentation, surface normal, saliency detection and edge detection. We consider two partially supervised learning settings, random-label and one-label setting. For one-label setting, we have 1 label per image, *i.e*. 1000, 999, 1000, 1000, 999 labelled images for semantic segmentation, human parts, surface normal, saliency and edge detection, respectively. In random-label setting, we randomly sample and keep labels for at least 1 and at most 4 tasks (2.50 labels per image), resulting in 2450, 2553, 2480, 2445, 2557 labelled images for semantic segmentation, human parts, surface normal, saliency and edge detection, respectively, as shown in Tab. B.2.

We follow exactly the same training, evaluation protocol and implementation in the prior work (Vandenhende et al., 2021) and employ the ResNet-18 (He et al., 2016) as the encoder shared across all tasks and Atrous Spatial Pyramid Pooling (ASPP) (Chen et al., 2018a) module as task-specific heads. We use the same hyper-parameters, *e.g*. learning rate, augmentation, loss functions, loss weights in the prior work (Vandenhende et al., 2021). More specifically, we use Adam as the optimizer with a learning rate of 0.0001 and a weight decay of 0.0001. Following Vandenhende et al. (2021) all experiments are performed using pretrained ImageNet weights. We train all multi-task learning methods for 100 epochs with a batch size of 6 and we anneal the learning rate using the 'poly' learning rate scheduler as Vandenhende et al. (2021); Chen et al.

(2017). We follow Vandenhende et al. (2021) and use fixed loss weights for training all multi-task learning methods, *i.e*. the loss weight is 1, 2, 10, 5, 50 for semantic segmentation, human parts segmentation, surface normal estimation, saliency detection and edge detection, respectively. Please refer to Vandenhende et al. (2021) for more details. For evaluation metrics, we use the optimal dataset F-measure (odsF) (Martin et al., 2004) for edge detection, the standard mean intersection over union (mIoU) for semantic segmentation, human part segmentation and saliency estimation are evaluated, mean error (mErr) for surface normals. We modify the ResNet-18 to have task-specific input layers (one convolutional layer for each task) before the residual blocks as the mapping function $\bar{m}_{\vartheta}$ in our method.

Multi-task performance. Following prior work (Vandenhende et al., 2021), we also report the multi-task performance $\triangle$MTL of the multi-task learning model as the average per-task drop in performance w.r.t. the single-task baseline:

$$\triangle \text{MTL} = \frac{1}{T}\sum_{t=1}^{T}(-1)^{\ell_i}(P^t_{mtl} - P^t_{stl})/P^t_{stl}, \tag{B.1}$$

where $\ell_i = 1$ if a lower value of P_t means better performance for metric of task t, and 0 otherwise.

B.2 More results

Here, we report more results from the single-task learning (STL) model, Contrastive-Loss and Discriminator-Loss and also qualitative results.

B.2.1 Quantitative results

Results on Cityscapes. Here, we report the results on Cityscapes for only *one* label setting as there are two tasks in total in Tab. B.3. We also report results of single-task learning models which are used to compute the multi-task performance ($\triangle$MTL) to better analyze the results as Vandenhende et al. (2021). The performance of MTL methods are worse than single-task learning models for some tasks as the MTL models have less capacity and there is a problem of imbalanced optimization etc as discussed in the prior works (Kendall et al., 2018; Li and Bilen, 2020; Vandenhende et al., 2021).

The results show that the MTL model learned with SL when all task labels are available for training serves as a strong baseline for multi-task learning methods. In the partial label setting (one task label per image), the performance of the SL baseline drops

substantially compared to its performance in full supervision setting. While the SSL baseline, by extracting task-specific information from unlabelled tasks, improves over SL, further improvements are obtained by exploiting cross-task consistency in various ways except Discriminator-Loss. The methods that learn mappings from one task to another one (Perceptual-Map and Direct-Map) surprisingly perform better than the ones learning joint space mapping functions (Contrastive-Loss and Discriminator-Loss), possibly due to insufficient number of negative samples. Finally, the best results (*e.g.* the best multi-task performance △MTL) are obtained with our method that can exploit cross-task relations more efficiently through joint pairwise task mappings with the proposed regularization. Interestingly, our method also outperforms the SL baseline that has access to all the task labels, showing the potential information in the cross-task relations.

# label	Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	△MTL ↑
full	STL	Supervised Learning	74.19	0.0124	+0.00
	MTL	Supervised Learning	73.36	0.0165	-17.00
one	STL	Supervised Learning	70.26	0.0141	+0.00
	MTL	Supervised Learning	69.50	0.0186	-16.55
		Semi-supervised Learning	71.67	0.0178	-12.22
		Perceptual-Map	72.82	0.0169	-8.37
		Direct-Map	72.33	0.0179	-11.94
		Contrastive-Loss	71.79	0.0183	-13.77
		Discriminator-Loss	68.94	0.0208	-24.95
		Ours	**74.90**	**0.0161**	**-3.81**

Table B.3: Multi-task learning results on Cityscapes. 'one' indicates each image is randomly annotated with one task label. 'STL' means single-task learning and 'MTL' indicates multi-task learning.

Results on Cityscapes with larger images. We also provide results for 256×512 setting in Tab. B.4. Performance of all methods improve significantly compared to their ones using small images (in Tab. B.3) and our method achieves significant improvement over the baselines.

Results on NYU-v2 Here, we evaluate our method and related methods in the *random* and *one* label settings on NYU-v2 and we report the results in Tab. B.5. We also report results of single-task learning models which are used to compute the multi-task performance (△MTL) to better analyze the results as Vandenhende et al. (2021).

While we observe a similar trend across different methods, overall the performances

# label	Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	△MTL ↑
one	STL	Supervised Learning	77.97	0.0126	+0.00
	MTL	Supervised Learning	77.71	0.0165	-15.95
		Semi-supervised Learning	79.24	0.0161	-13.38
		Ours	**82.41**	**0.0143**	**-4.08**

Table B.4: Multi-task learning results on Cityscapes using 256×512 images. 'one' indicates each image is randomly annotated with one task label. 'STL' means single-task learning and 'MTL' indicates multi-task learning.

are lower in this benchmark possibly due to fewer training images than CityScapes. As expected, the performance in the random-label setting is better than the one in one-label setting, as there are more labels available in the former. While the best results are obtained with SL trained on full supervision, our method obtains the best performance (*e.g.* best results on all tasks and the best multi-task performance) among the partially supervised methods. Here SSL improves over SL trained on the partial labels and cross-task consistency is beneficial except for Direct-Map in the one label setting and Discriminator-Loss, possibly because the dataset is too small to learn accurate mappings between two tasks, while our method is more data-efficient and more successful to exploit the cross-task relations. In random-label setting, where images might have labels for more than one task, we also report our method also leveraging the labelled corss-task relations ('Ours+') in Tab. B.5 and it can indeed further boost the average performance.

Results on PASCAL. We evaluate all methods on PASCAL-Context, in both label settings, which contains a wider variety of tasks than the previous benchmarks and report the results in Tab. B.6. As in Cityscapes and NYU-v2, we also report results of single-task learning models which are used to compute the multi-task performance (△MTL) to better analyze the results as Vandenhende et al. (2021).

As the required number of pairwise mappings for Direct-Map and Perceptual-Map grows quadratically (20 mappings for 5 tasks), we omit these two due to their high computational cost and compare our method only to SL, SSL, Contrastive-Loss and Discriminator-Loss baselines. We see that the SSL baseline improves the performance over SL in the random-label setting, however, it performs worse than the SL in one label setting, when there are 60% less labels. By leveraging cross-task consistency, Contrastive-Loss and Discriminator-Loss obtains better performance than the SL baseline in one label setting while they get similar multi-task performance to the SL baseline

# labels	Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
full	STL	Supervised learning	37.45	0.6079	25.94	+0.00
	MTL	Supervised learning	36.95	0.5510	29.51	-1.92
random	STL	Supervised Learning	28.72	0.7540	28.95	+0.00
	MTL	Supervised Learning	27.05	0.6624	33.58	-3.23
		Semi-supervised Learning	29.50	0.6224	33.31	+1.70
		Perceptual-Map	32.20	0.6037	32.07	+7.10
		Direct-Map	29.17	0.6128	33.63	+1.38
		Contrastive-Loss	30.75	0.6143	32.05	+4.96
		Discriminator-Loss	26.76	0.6354	33.13	-1.84
		Ours	34.26	0.5787	**31.06**	+11.81
		Ours+	**34.91**	**0.5738**	31.20	**+12.57**
one	STL	Supervised Learning	24.71	0.7666	30.14	+0.00
	MTL	Supervised Learning	25.75	0.6511	33.73	+1.14
		Semi-supervised Learning	27.52	0.6499	33.58	+3.16
		Perceptual-Map	26.94	0.6342	34.30	+2.31
		Direct-Map	19.98	0.6960	37.56	-12.86
		Contrastive-Loss	26.65	0.6387	34.69	+1.31
		Discriminator-Loss	25.68	0.6566	34.02	+0.04
		Ours	**30.36**	**0.6088**	**32.08**	**+10.24**

Table B.5: Multi-task learning results on NYU-v2. 'random' indicates each image is annotated with a random number of task labels and 'one' means each image is randomly annotated with one task. 'STL' means single-task learning and 'MTL' indicates multi-task learning.

in random label setting. Again, by exploiting task relations, our method obtains better or comparable results to the second best method, *i.e*. SSL, while the gains achieved over SL and SSL are more significant in the low label regime (one-label). Interestingly, SSL and our method obtain comparable results in the random-label setting which suggests that relations across tasks are less informative than the ones in CityScape and NYUv2.

Learning from partial and imbalanced task labels. We also evaluate our method and baselines in an imbalanced partially supervised setting in Cityscapes, where we assume the ratio of labels for each task are imbalanced, *e.g*. we randomly sample 90% of images to be labeled for semantic segmentation and only 10% images having labels for depth and we denote this setting by the label ratio between segmentation and depth (Seg.:Depth = 9:1). The opposite case (Seg.:Depth = 1:9) is also considered. We report the results in Tab. B.7, where we also report results of single-task learning models which are used to compute the multi-task performance (△MTL) to better analyze the results as Vandenhende et al. (2021).

The performance of supervised learning (SL) on the task with partial labels drops

# labels	Type	Method	Seg. (IoU) ↑	H. Parts (IoU) ↑	Norm. (mErr) ↓	Sal. (IoU) ↑	Edge (odsF) ↑	△MTL ↑
full	STL	Supervised Learning	66.4	58.9	13.9	66.7	68.3	+0.00
	MTL	Supervised Learning	63.9	58.9	15.1	65.4	69.4	-2.75
random	STL	Supervised Learning	60.9	55.3	14.7	64.8	66.8	+0.00
	MTL	Supervised Learning	58.4	55.3	16.0	63.9	**67.8**	-2.67
		Semi-supervised Learning	**59.0**	**55.8**	**15.9**	**64.0**	66.9	-2.44
		Contrastive-Loss	**59.0**	55.3	16.0	63.8	**67.8**	-2.44
		Discriminator-Loss	57.9	55.2	16.2	63.4	67.4	-3.35
		Ours	**59.0**	55.6	**15.9**	**64.0**	**67.8**	**-2.15**
one	STL	Supervised Learning	47.7	56.2	16.0	61.9	64.0	+0.00
	MTL	Supervised Learning	48.0	55.6	17.2	61.5	64.6	-1.34
		Semi-supervised Learning	45.0	54.0	**16.9**	**61.7**	62.4	-3.02
		Contrastive-Loss	48.5	55.4	17.1	61.3	64.6	-1.25
		Discriminator-Loss	48.2	**56.0**	17.1	61.7	64.7	-1.04
		Ours	**49.5**	55.8	17.0	**61.7**	**65.1**	**-0.40**

Table B.6: Multi-task learning results on PASCAL. 'random' indicates each image is annotated with a random number of task labels and 'one' means each image is randomly annotated with one task. 'STL' means single-task learning and 'MTL' indicates multi-task learning.

significantly. Though SSL improves the performance on segmentation, its performance on depth drops in both cases. Different from SSL, Direct-Map, Contrastive-Loss and Discriminator-Loss improves the performance on both tasks in 1:9 setting while their performance on depth drops in the 9:1 case. In contrast to SL and the baselines, our method and Perceptual-Map obtain better results on all tasks in both settings by learning cross-task consistency while our method obtains the best performance (*i.e.* best results in all tasks and best multi-task performance, △MTL) by joint space mapping. This demonstrates that our model can successfully learn cross-task relations from unbalanced labels thanks to its task-agnostic mapping function which can share parameters across multiple task pairs.

Cross-task consistency learning in conventional semi-supervised learning. We evaluate our method and SSL baseline on conventional SSL setting where $\frac{1}{3}$ of training data in NYU-v2 are labeled for all tasks and $\frac{2}{3}$ are unlabeled, and report the results in Tab. B.8. In this setting, our method obtains better performance than SL and SSL.

Cross-task consistency learning with full supervision. Our method can also be applied to the fully-supervised learning setting where all task labels are available for each sample by mapping one task's prediction and another task's ground-truth to the joint space and measuring cross-task consistency in the joint space. We applied our method to NYU-v2 and compare it with the single-task learning (STL) networks, vanilla

#labels	Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	△MTL ↑
full	STL	Supervised learning	74.19	0.0124	+0.00
	MTL	Supervised Learning	73.36	0.0165	-17.00
1:9	STL	Supervised learning	62.23	0.0126	+0.00
	MTL	Supervised Learning	63.37	0.0161	-13.07
		Semi-supervised Learning	64.40	0.0179	-19.36
		Perceptual-Map	68.84	0.0141	-0.68
		Direct-Map	67.04	0.0153	-6.90
		Contrastive-Loss	67.12	0.0151	-5.95
		Discriminator-Loss	68.92	0.0144	-1.80
		Ours	**71.89**	**0.0131**	**+5.63**
9:1	STL	Supervised learning	72.62	0.0191	+0.00
	MTL	Supervised learning	72.77	0.0250	-15.25
		Semi-supervised Learning	72.97	0.0395	-53.11
		Perceptual-Map	73.36	0.0237	-11.34
		Direct-Map	73.13	0.0288	-19.38
		Contrastive-Loss	73.75	0.0243	-12.86
		Discriminator-Loss	72.97	0.0248	-14.65
		Ours	**74.23**	**0.0235**	**-10.23**

Table B.7: Multi-task learning results on Cityscapes. '#label' indicates the number ratio of labels for segmentation and depth, *e.g.* '1:9' means we have 10% of images annotated with segmentation labels and 90% of images have depth groundtruth. 'STL' means single-task learning and 'MTL' indicates multi-task learning.

Type	Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL ↑
MTL	Supervised Learning	24.78	0.6681	33.90	+1.48
	Semi-supervised Learning	26.09	0.6510	33.60	+4.37
	Ours	**28.43**	**0.6366**	**33.01**	**+8.83**

Table B.8: Multi-task learning results on NYU-v2 in SSL setting where $\frac{1}{3}$ of training data in NYU-v2 are labeled for all tasks and $\frac{2}{3}$ are unlabeled. 'MTL' indicates multi-task learning.

MTL baseline, recent multi-task learning methods, *i.e.* MTAN (Liu et al., 2019), X-task (Zamir et al., 2020), and several methods focusing on loss weighting strategies, *i.e.* Uncertainty (Kendall et al., 2018), GradNorm (Chen et al., 2018b), MGDA (Sener and Koltun, 2018) and DWA (Liu et al., 2019) in Tab. B.9. Here, we also report the multi-task performance (△MTL) of all MTL methods.

MTL, MTAN, X-task and Ours are trained with uniform loss weights. We see that our method (Ours) performs better than the other methods with uniform loss weights,

Method	Seg. (IoU) ↑	Depth (aErr) ↓	Norm. (mErr) ↓	△MTL
STL	37.45	0.6079	25.94	+0.00
MTL	36.95	0.5510	29.51	-1.92
MTAN (Liu et al., 2019)	39.39	0.5696	28.89	+0.03
X-task (Zamir et al., 2020)	38.91	0.5342	29.94	+0.89
Uncertainty (Kendall et al., 2018)	36.46	0.5376	27.58	+0.86
GradNorm (Chen et al., 2018b)	37.19	0.5775	28.51	-1.86
MGDA (Sener and Koltun, 2018)	38.65	0.5572	28.89	+0.06
DWA (Liu et al., 2019)	36.46	0.5429	29.45	-1.82
Ours	41.00	0.5148	28.58	+4.88
Ours + Uncertainty	**41.09**	**0.5090**	**26.78**	**+7.57**

Table B.9: Multi-task fully-supervised learning results on NYU-v2. 'STL' indicates standard single-task learning and 'MTL' means the standard multi-task learning network.

e.g. MTAN and X-task, where X-task regularizes cross-task consistency by learning perceptual loss with pretrained cross-task mapping functions. This shows that cross-task consistency is informative even in the fully supervised case and our method is more effective for learning cross-task consistency. Compared to recent loss weighting strategies, our method (Ours) obtains better multi-task performance (△MTL) and better performance on segmentation and depth estimation than other methods while slightly worse on normal estimation compared with GradNorm and Uncertainty. This is because the loss weighting strategies enable a more balanced optimization of the multi-task learning model than uniformly loss weighting. Thus when we incorporate the loss weighing strategy of Uncertainty (Kendall et al., 2018) to our method, *i.e*. (Ours + Uncertainty), our method obtains further improvement and outperforms both GradNorm and Uncertainty, *e.g*. 'Ours + Uncertainty' obtains the best multi-task performance (+7.57).

B.2.2 Qualitative results

Here, we present some qualitative results.

Mapped outputs. Here, we visualize the intermediate feature maps of $m^{s\to st}$ and $m^{t\to st}$ for one example in Cityscapes in Fig. B.1 where s and t correspond to segmentation and depth estimation respectively and one example in NYU-v2 in Fig. B.2 where s and t correspond to segmentation and surface normal estimation respectively. We observe that the functions map both task labels to a joint pairwise space where the common information is around object boundaries, which in turn enables the model to

produce more accurate predictions for both tasks.

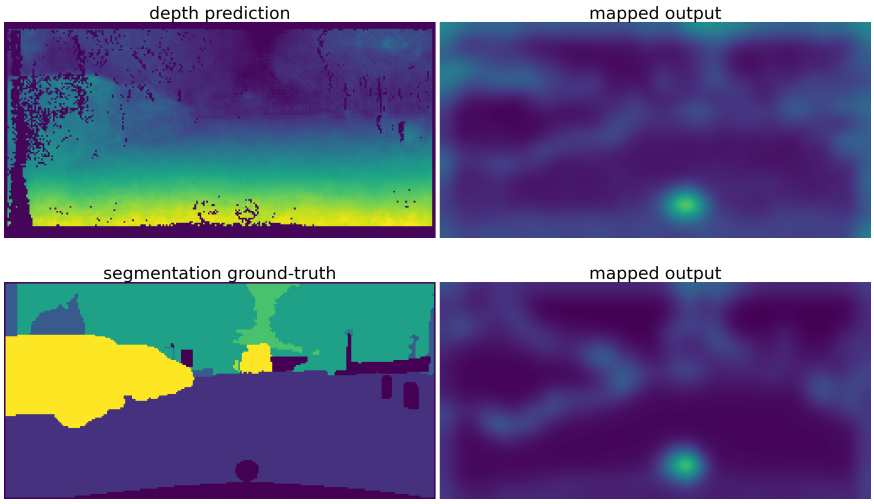


Figure B.1: Intermediate feature map of the mapping function of the task-pair (segmentation to depth) of one example in Cityscapes. The first column shows the prediction or ground-truth and the second column present the corresponding mapped feature map (output of the mapping function's last second layer).

Predictions. Finally we show qualitative comparisons between our method, SL and SSL baselines, Perceptual-Map (PM), Direct-Map (DM), Contrastive-Loss (CL) and Discriminator-Loss (DL) on Cityscapes in Fig. B.3 and on NYU-v2 in Fig. B.4. We can see that our method produces more accurate predictions by leveraging cross-task consistency. Specifically, in Fig. B.3, compared with methods that do not leverage cross-task consistency, the prediction of segmentation and depth are improved by our method (top left region) and our results are more accurate than related baselines (PM, DM, CL and DL). In Fig. B.4, we can see that SSL produces more accurate predictions on segmentation and surface normal than SL. And PM obtains more accurate results on depth and surface normal than SL. While they do not achieve consistent improvement on all three tasks, our method can improve the results consistently on three tasks which shows that our method is more effective on learning cross-task consistency for MTL from partially annotated data.

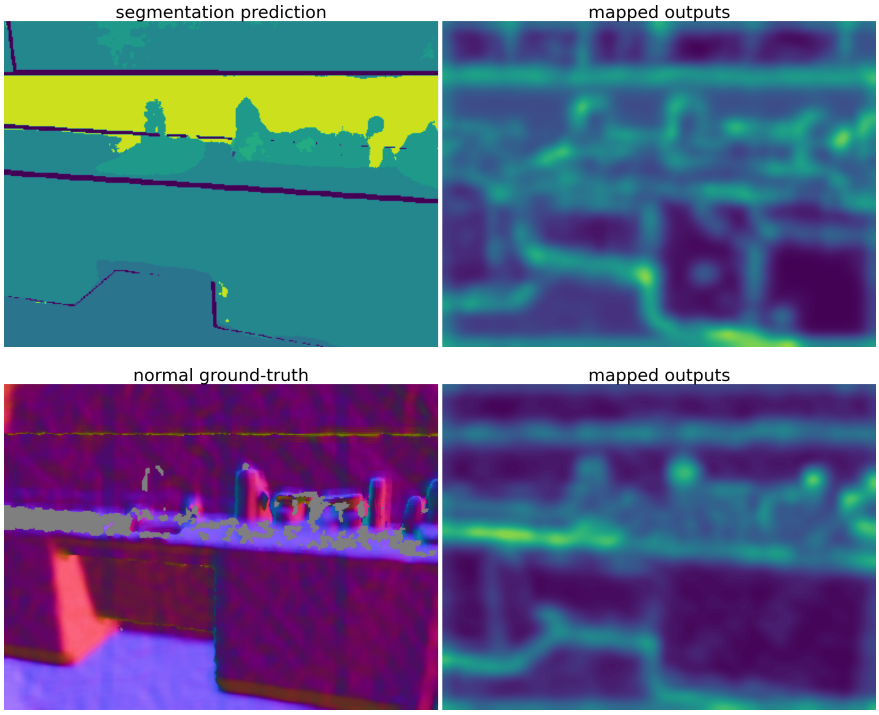


Figure B.2: Intermediate feature map of the mapping function of the task-pair (segmentation to surface normal) of one example in NYU-v2. The first column shows the prediction or ground-truth and the second column present the corresponding mapped feature map (output of the mapping function's last second layer).

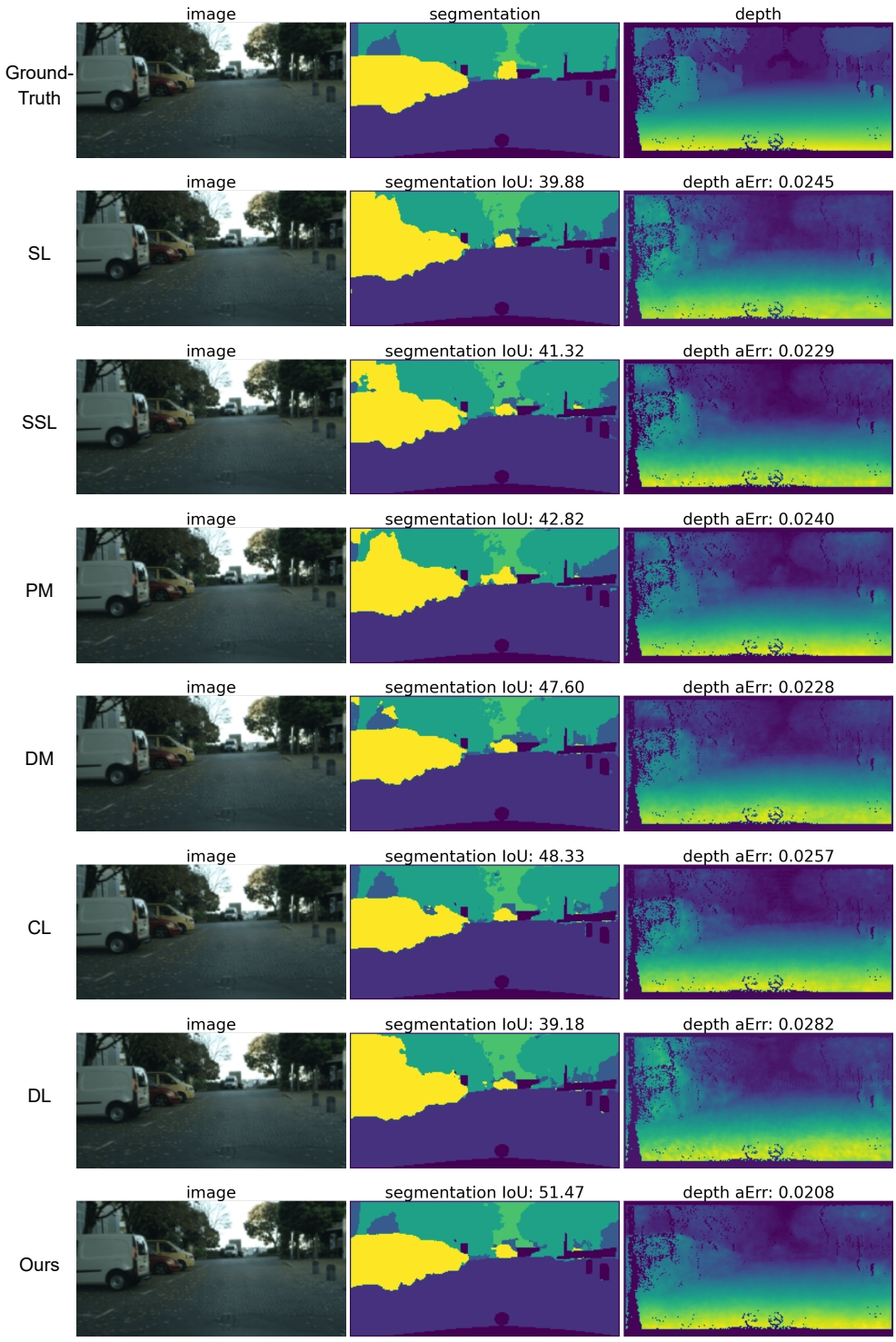


Figure B.3: **Qualitative results on Cityscapes.** The fist column shows the RGB image, the second column plots the ground-truth or predictions with the IoU (↑) score of all methods for semantic segmentation and we show the ground-truth or predictions with the absolute error (↓) in the last column.

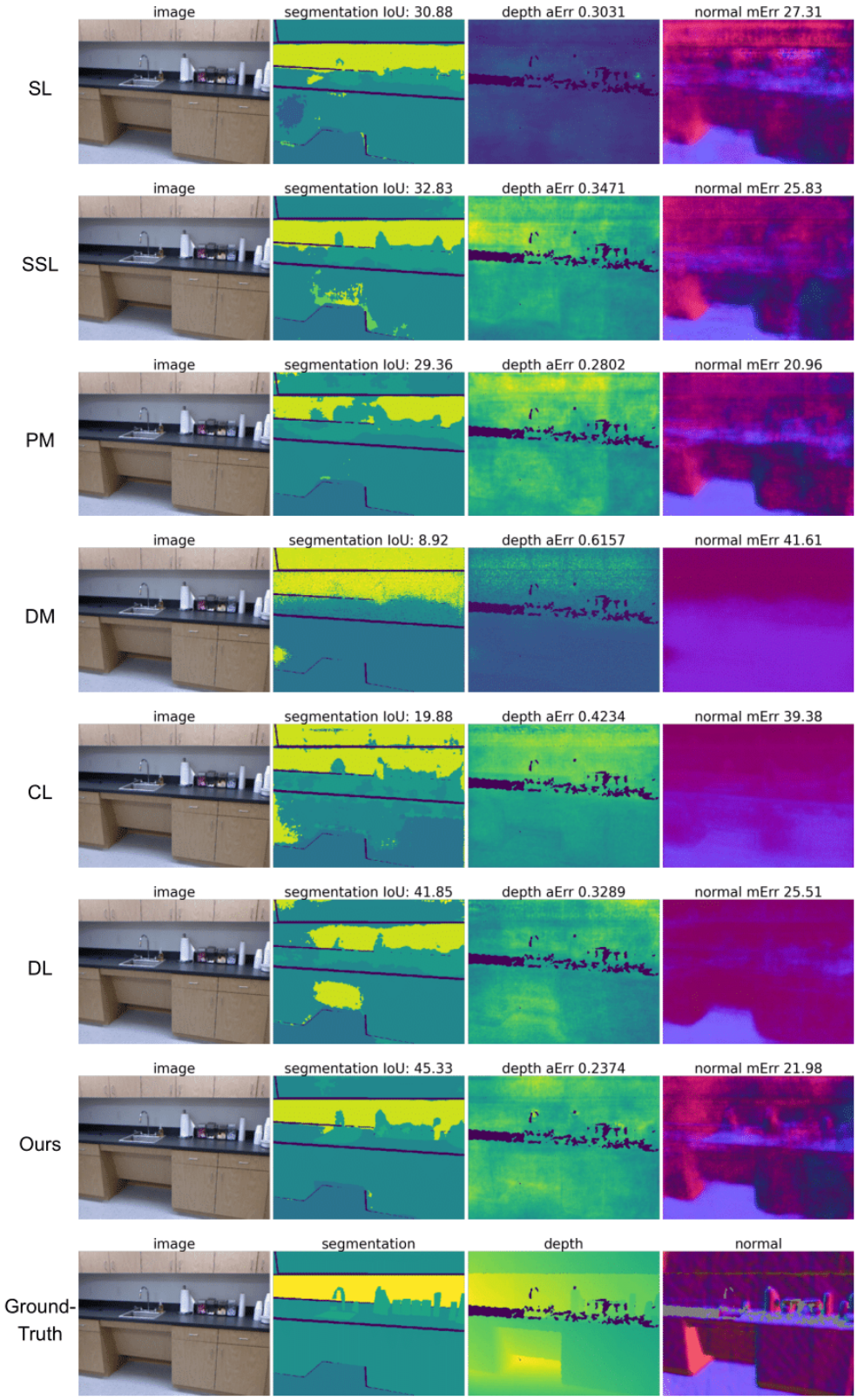


Figure B.4: **Qualitative results on NYU-v2.** The fist column shows the RGB image, the second column plots the ground-truth or predictions with the IoU (↑) score of all methods for semantic segmentation, the third column presents the ground-truth or predictions with the absolute error (↓), and we show the prediction of surface normal with mean error (↓) in the last column.

Appendix C

Cross-domain Few-shot Classification

C.1 Implementation details

In this section, we explain the details of task-agnostic (feature extractor) learning and then task-specific (adapter) learning.

C.1.1 Task-agnostic learning

Here we consider learning the parameters of the feature extractor from either multiple or single domains.

Best SDL. One way of learning universal representations over multiple domains for cross-domain few-shot learning is to use each single-domain network learned from one dataset as the feature extractor and test it for few-shot classification in each dataset and pick the best performing model. To this end, we train one ResNet-18 model for each training dataset. For optimization, we follow the training protocol in the prior work (Dvornik et al., 2020). Specifically, we use SGD optimizer and cosine annealing for all experiments with a momentum of 0.9 and a weight decay of 7×10^{-4}. The learning rate, batch size, annealing frequency, maximum number of iterations are shown in Tab. C.2. To regularize training, we also use the exact same data augmentations as in SUR (Dvornik et al., 2020), *e.g*. random crops and random color augmentations.

Vanilla multi-domain learning (MDL) and our URL. When we learn a single feature extractor from multiple domains, we consider two cases. In the first case, which we call vanilla multiple domain learning (or MDL), we design a deep network where we share all the layers across all domains and have domain-specific classifiers. This setting corresponds to Eq. (4.1). Second, in this thesis, we propose the URL method which

Dataset	learning rate	batch size	annealing freq.	max. iter.
ImageNet	3×10^{-2}	64	48,000	480,000
Omniglot	3×10^{-2}	16	3000	50,000
Aircraft	3×10^{-2}	8	3000	50,000
Birds	3×10^{-2}	16	3000	50,000
Textures	3×10^{-2}	32	1500	50,000
Quick Draw	1×10^{-2}	64	48,000	480,000
Fungi	3×10^{-2}	32	15,000	480,000
VGG Flower	3×10^{-2}	8	1500	50,000

Table C.1: Training hyper-parameters of single domain learning.

also learns a single network with shared and domain-specific layers as such, however, it is learned by distilling information from multiple domain-specific networks.

To train the multi-domain network for both MDL and URL, we use the same optimizer with a weight decay of 7×10^{-4} and a scheduler as single domain learning models for learning 240,000 iterations. The learning rate is 0.03 and the annealing frequency is 48,000. Similar to Triantafillou et al. (2020) that the training episodes have 50% probability coming from the ImageNet data source, each training batch for our multi-domain network consists of 50% data coming from ImageNet. In other words. The batch size for ImageNet is 64×7 and is 64 for the other 7 datasets.

To learn the URL, we set λ_f and λ_p as 4 for ImageNet and 1 for other datasets, respectively. And we linearly anneal λ by $\lambda \leftarrow \lambda \times (1 - \frac{k}{K})$, where, k is the current iteration and K is the total number of iterations to anneal λ to zero. Here, $K = e \times (anneal.\ freq.)$, where $anneal.\ freq.$ is 48, 000 in this work. We search the $u = \{1,2,3,4,5\}$ based on cross-validation over the validation sets of 8 training datasets and e is 5 (*i.e.* $K = 240,000$) for ImageNet, is 2 for Omniglot, Quick Draw, Fungi and is 1 for other datasets. For all experiments, early-stopping is performed based on cross-validation over the validations sets of 8 training datasets.

Single domain learning (SDL). We also evaluate our method on a feature extractor that is learned on a single domain which we call SDL. Here we evaluate our method on two backbones, ResNet-18 (SDL-ResNet-18) and ResNet-34 (SDL-ResNet-34).

SDL-ResNet-18. Following Triantafillou et al. (2020); Dvornik et al. (2020); Li et al. (2021b), we train a ResNet-18 on the train split of ImageNet and use 84×84 image size, which is denoted as SDL-ResNet-18. For optimization, we follow the training protocol in the prior works (Dvornik et al., 2020; Li et al., 2021b). Specifically, we

Backbone	learning rate	batch size	annealing freq.	max. iter.
SDL-ResNet-18	3×10^{-2}	64	48,000	480,000
SDL-ResNet-34	3×10^{-2}	128	48,000	480,000

Table C.2: Training hyper-parameters of single domain learning.

use SGD optimizer and cosine annealing for all experiments with a momentum of 0.9 and a weight decay of 7×10^{-4}. Some other hyperparameters are shown in Tab. C.2 as Dvornik et al. (2020); Li et al. (2021b). To regularize training, we also use the exact same data augmentations as in the prior works (Dvornik et al., 2020; Li et al., 2021b), *e.g*. random crops and random color augmentations.

SDL-ResNet-34. We also apply our method to the single domain learning model with ResNet-34 backbone learned on ImageNet only as Doersch et al. (2020). We follow Doersch et al. (2020) and use higher-resolution (224×224) images for meta-training and meta-testing. For optimization, we follow the training protocol as in the prior works (Dvornik et al., 2020; Li et al., 2021b). Specifically, we use SGD optimizer and cosine annealing with a momentum of 0.9, a weight decay of 1×10^{-4} with a batch size of 128. Other hyperparameters are the same as in SDL-ResNet-18 and are shown in Tab. C.2. To regularize training, we also use the exact same data augmentations as in the prior works (Dvornik et al., 2020; Li et al., 2021b), *e.g*. random crops and random color augmentations with an additional stage that randomly downsamples and upsamples images as Doersch et al. (2020).

C.1.2 Task-specific learning

Attaching and learning adapters. For the optimization of the adaptation parameters α which is attached directly and learned on support set and the pre-classifier adaptation β, we initialize β as an identity matrix and optimize both α and β for 40 iterations using Adadelta (Zeiler, 2012) as optimizer. The learning rate of β is 0.1 for first eight datasets and 1 for the last five datasets as in URL (Li et al., 2021b) and we set the learning rate of α as half of the learning rate of β, *i.e*. 0.05 for the first eight datasets and 0.5 for the last five datasets. Note that, we learn α and β on a per-task basis using the task's support set during the meta-test. That is, α and β are not re-used across the test tasks drawn from $\mathcal{D}_{test}$.

Predicting r_α. In the case of modulating α with the auxiliary network, we follow the auxiliary training protocols in CNAPS (Bateni et al., 2020b). We train for 10K

episodes to optimize the task encoder using Adam with a learning rate of 1×10^{-5} on eight training domains in meta-train. We validate every 5K iterations to save the best model for testing.

C.2 More results

C.2.1 Task-agnostic learning

Complete results of Best SDL To study the universal representation learning from multiple datasets, we train one network on each training dataset and use each single-domain network as the feature extractor and test it for few-shot classification in each dataset. This involves evaluating 8 single-domain networks on 13 datasets using Nearest Centroid Classifier (NCC). Tab. C.3 shows the results of single domain learning models, where each column present the mean accuracy and 95% confidence interval of a single-domain network trained on one dataset (*e.g*. ImageNet) and evaluated on 13 test datasets. The average accuracy and 95% confidence intervals computed over 600 few-shot tasks. The numbers in bold indicate that a method has the best accuracy per dataset.

As shown in Tab. C.3, the feature of the ImageNet model generalizes well and achieves the best results on four out of eight seen datasets, *e.g*. ImageNet, Birds, Texture, VGG Flower and four out of five previously unseen datasets, *e.g*. Traffic Sign, MSCOCO, CIFAR-10, CIFAR-100. The models trained on Omniglot, Aircraft, Quick Draw, and Fungi perform the best on the corresponding datasets while the Omniglot model also generalizes well to MNIST which has the similar style images to Omniglot. We then pick the best performing model, forming the best single-domain model (Best SDL) which serves a very competitive baseline for universal representation learning.

Complete global retrieval results. Here we go beyond the few-shot classification experiments and evaluate the generalization ability of our representations that are learned in the multi-domain network in a retrieval task, inspired from metric learning literature (Oh Song et al., 2016; Yu et al., 2019). To this end, for each test image, we find the nearest images in the entire test set in the feature space and test whether they correspond to the same category. For evaluation metric, we use Recall@k which considers the predictions with one of the k closest neighbors with the same label as positive. In Tabs. C.4 and C.5, we compare our method with Simple CNAPS in Recall@1, Recall@2, Recall@4 and Recall@8. URT and SUR require adaptation using the support set and no such adaptation in retrieval task is possible, we replace them

Test Dataset \ Train Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	Vgg Flower
ImageNet	**55.8±1.0**	17.1±0.6	21.7±0.7	25.4±0.8	24.2±0.8	24.1±0.8	32.9±0.9	25.0±0.8
Omniglot	67.4±1.2	**93.2±0.5**	58.2±1.2	58.7±1.4	57.3±1.4	78.4±1.0	57.6±1.3	54.6±1.3
Aircraft	49.5±0.9	16.8±0.5	**85.7±0.5**	31.4±0.8	26.0±0.7	23.8±0.6	31.0±0.7	24.6±0.6
Birds	**71.2±0.9**	13.0±0.6	19.9±0.7	65.0±0.9	19.6±0.7	16.7±0.7	42.8±1.0	28.9±0.8
Textures	**73.0±0.6**	25.0±0.5	38.6±0.7	42.2±0.7	54.9±0.7	38.6±0.6	54.1±0.7	42.3±0.7
Quick Draw	53.9±1.0	51.0±1.0	38.8±1.0	38.2±1.0	36.8±0.9	**82.8±0.6**	37.7±0.9	39.7±1.0
Fungi	41.6±1.0	9.1±0.5	14.9±0.7	25.5±0.8	15.6±0.7	12.5±0.6	**65.8±0.9**	23.3±0.8
VGG Flower	**87.0±0.6**	23.8±0.6	45.5±0.8	62.9±0.8	44.4±0.8	33.4±0.7	79.6±0.7	78.3±0.7
Traffic Sign	**47.4±1.1**	15.1±0.7	30.8±0.9	31.0±0.9	38.8±1.1	31.1±0.9	28.0±0.9	30.4±0.9
MSCOCO	**53.5±1.0**	12.9±0.6	22.5±0.8	25.1±0.9	23.7±0.8	21.3±0.8	32.5±1.0	25.7±0.8
MNIST	78.1±0.7	**89.8±0.5**	68.0±0.8	73.0±0.7	64.5±0.8	88.2±0.5	62.2±0.8	72.1±0.7
CIFAR-10	**67.3±0.8**	28.5±0.6	41.2±0.7	41.8±0.8	36.9±0.7	40.0±0.7	38.8±0.7	41.3±0.8
CIFAR-100	**56.6±0.9**	12.3±0.6	24.3±0.9	28.8±0.9	24.2±0.9	23.4±0.8	25.2±0.9	29.1±1.0

Table C.3: Results of all single domain learning models. Mean accuracy and 95% confidence interval are reported. The first eight datasets are seen during training and the last five datasets are unseen for test only.

with two baselines that concatenate or sum features from multiple domain-specific networks. Our method achieves the best performance in ten out of thirteen domains with significant gains in Aircraft, Birds, Textures and Fungi. This strongly suggests that our multi-domain representations are the key to the success of our method in the previous few-shot classification tasks.

Test Dataset	ImageNet				Omniglot				Aircraft				Birds				Textures				Quick Draw				Fungi				VGG Flower			
Recall@k	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8
Sum	22.1	30.3	39.6	50.0	84.7	91.8	95.8	97.8	69.7	80.7	88.6	94.5	45.9	59.7	72.0	84.1	66.3	78.2	87.3	94.0	77.4	84.3	89.1	92.1	31.9	42.9	54.0	65.4	85.1	92.1	96.7	98.6
Concate	20.2	28.0	36.9	47.8	84.4	91.5	95.8	97.8	44.3	58.1	71.1	82.9	35.5	48.8	62.8	76.0	68.8	78.2	87.3	93.9	73.0	80.8	86.2	90.6	30.7	40.4	51.8	63.0	83.4	91.3	95.2	98.2
MDL	29.8	39.6	49.9	60.9	**89.8**	**94.3**	96.8	98.2	80.3	87.1	92.5	95.9	63.2	75.9	84.7	91.6	67.0	77.1	85.4	92.9	79.5	85.4	89.7	92.8	40.2	51.7	63.0	72.4	86.9	93.3	96.6	98.4
Simple CNAPS (Bateni et al., 2020b)	34.0	43.8	54.4	65.1	84.9	91.6	95.5	97.5	70.5	82.5	91.3	96.1	55.9	70.5	82.0	90.2	64.8	76.9	**87.6**	**94.4**	75.3	83.0	88.0	91.7	29.1	39.0	49.6	61.5	88.1	94.1	**97.6**	**99.2**
Ours	**36.1**	**46.2**	**56.3**	**66.6**	89.7	**94.3**	**97.2**	**98.3**	**83.3**	**90.4**	**93.7**	**96.3**	**66.7**	**78.9**	**87.9**	**94.1**	**70.2**	**80.8**	87.5	93.8	**79.9**	**86.5**	**90.5**	**93.2**	**44.5**	**56.2**	**67.3**	**76.4**	**90.0**	**94.6**	97.5	98.9

Table C.4: Global retrieval performance on Meta-Dataset (seen datasets). In addition to few-shot learning experiments, we evaluate our method in a non-episodic retrieval task to further compare the generalization ability of our universal representations.

Test Dataset	Traffic Sign				MSCOCO				MNIST				CIFAR-10				CIFAR-100			
Recall@k	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8	1	2	4	8
Sum	94.6	97.2	98.5	**99.3**	62.6	71.2	78.9	85.0	98.3	99.2	**99.6**	**99.8**	54.0	68.9	81.9	90.6	27.8	37.4	48.4	60.4
Concate	**95.1**	**97.3**	**98.6**	99.2	60.7	69.8	77.4	83.6	**98.7**	**99.3**	**99.6**	**99.8**	49.7	65.3	79.4	88.9	25.4	34.6	45.3	57.2
MDL	89.5	94.1	96.6	98.3	63.6	72.6	79.9	86.0	97.6	98.8	99.2	99.6	58.9	72.9	84.1	92.2	31.6	42.0	53.4	64.8
Simple CNAPS (Bateni et al., 2020b)	79.9	86.9	92.6	96.2	65.2	73.8	81.1	86.6	97.5	98.8	99.3	99.7	**66.2**	**79.3**	**88.5**	**94.7**	33.2	44.2	57.3	68.7
Ours	87.9	93.0	96.1	98.2	**67.4**	**76.3**	**83.0**	**88.5**	97.0	98.4	99.1	99.5	62.1	76.5	86.0	93.3	**35.1**	**46.1**	**57.8**	**69.0**

Table C.5: Global retrieval performance on Meta-Dataset (unseen datasets). In addition to few-shot learning experiments, we evaluate our method in a non-episodic retrieval task to further compare the generalization ability of our universal representations.

Test Dataset	classifier	Aux-Net or Ad	serial or parallel	M or CW	β	#params	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
NCC	NCC	-	-	-	✗	-	57.0±1.1	94.4±0.4	88.0±0.5	80.3±0.7	74.6±0.7	81.8±0.6	66.2±0.9	91.5±0.5	49.8±1.1	54.1±1.0	91.1±0.4	70.6±0.7	59.1±1.0
MD	MD	-	-	-	✗	-	53.9±1.0	93.8±0.5	87.6±0.5	78.3±0.7	73.7±0.7	80.9±0.7	57.7±0.9	89.7±0.6	62.2±1.1	48.5±1.0	95.1±0.4	68.9±0.8	60.0±0.9
LR	LR	-	-	-	✗	-	56.0±1.1	93.7±0.5	88.3±0.6	79.7±0.8	74.7±0.7	80.0±0.7	62.1±0.8	91.1±0.5	59.7±1.1	51.2±1.1	93.5±0.5	73.1±0.8	60.1±1.1
SVM	SVM	-	-	-	✗	-	54.5±1.1	94.3±0.5	87.7±0.5	78.1±0.8	73.8±0.8	80.0±0.6	58.5±0.9	91.4±0.6	65.7±1.2	50.5±1.0	95.4±0.4	72.0±0.8	60.5±1.1
Softmax	Softmax	-	-	-	✗	-	42.2±1.0	85.3±0.7	71.9±0.8	59.6±1.0	62.0±0.8	61.2±1.0	37.3±0.9	66.7±1.0	51.4±1.1	48.2±1.1	93.5±0.5	70.4±0.8	59.3±1.0
KNN	KNN	-	-	-	✗	-	48.1±1.1	94.1±0.4	84.5±0.6	70.7±0.8	65.9±0.8	74.8±0.7	53.5±0.9	86.0±0.6	56.9±1.2	44.7±1.1	91.4±0.5	60.3±0.8	49.4±1.0
PA	NCC	-	-	-	✓	-	58.8±1.1	94.5±0.4	89.4±0.4	80.7±0.8	77.2±0.7	**82.5±0.6**	**68.1±0.9**	92.0±0.5	63.3±1.1	57.3±1.0	94.7±0.4	74.2±0.8	63.5±1.0
PA	Softmax	-	-	-	✓	-	53.4±1.2	92.7±0.5	85.7±0.6	76.1±0.9	73.9±0.8	76.5±0.8	51.1±0.9	86.9±0.7	52.5±1.1	48.2±1.1	94.3±0.4	69.7±0.8	60.4±1.0
Finetune	NCC	-	-	-	✗	-	55.9±1.2	94.0±0.5	87.3±0.6	77.8±0.9	76.8±0.8	75.3±0.9	57.6±1.1	91.5±0.6	**86.1±0.9**	53.1±1.2	**96.8±0.4**	80.9±0.8	65.9±1.1
Finetune	Softmax	-	-	-	✗	-	48.4±1.2	92.2±0.6	81.6±0.9	70.3±1.3	72.0±0.9	73.5±1.0	44.2±1.1	90.3±0.7	65.5±1.4	41.0±1.3	96.3±0.4	71.6±1.0	53.8±1.4
Aux-S-CW	NCC	Aux-Net	serial	CW	✗	-	54.6±1.1	93.5±0.5	86.6±0.5	78.6±0.8	71.5±0.7	79.3±0.6	66.0±0.9	87.6±0.6	43.3±0.9	49.1±1.0	87.9±0.5	62.8±0.8	51.5±1.0
Aux-R-CW	NCC	Aux-Net	residual	CW	✗	-	56.1±1.1	94.2±0.4	88.4±0.5	80.6±0.7	74.9±0.6	82.0±0.6	66.4±0.9	91.6±0.5	48.5±1.0	53.5±1.0	90.8±0.5	70.2±0.8	59.7±1.0
Aux-S-CW	MD	Aux-Net	serial	CW	✗	-	55.1±1.1	93.8±0.5	86.8±0.5	77.4±0.8	73.2±0.8	79.9±0.7	57.4±0.9	88.1±0.7	58.4±1.1	50.1±1.1	92.7±0.5	66.5±0.8	55.7±1.1
Aux-R-CW	MD	Aux-Net	residual	CW	✗	-	54.8±1.1	93.8±0.5	87.4±0.5	78.2±0.7	73.4±0.7	81.1±0.7	58.8±0.9	90.1±0.5	63.6±1.2	48.5±1.1	94.8±0.4	69.6±0.8	60.6±0.9
Ad-S-CW	NCC	Ad	serial	CW	✗	0.06%	56.8±1.1	94.8±0.4	89.3±0.5	80.7±0.7	74.5±0.7	81.6±0.6	65.8±0.9	91.3±0.5	73.9±1.1	53.6±1.1	95.7±0.4	78.4±0.7	64.3±1.0
Ad-R-CW	NCC	Ad	residual	CW	✗	1.57%	57.6±1.1	94.7±0.4	89.0±0.4	81.2±0.8	75.2±0.7	81.5±0.6	65.4±0.8	91.8±0.5	79.2±1.1	54.7±1.1	96.4±0.4	79.5±0.8	67.4±1.0
Ad-S-M	NCC	Ad	serial	M	✗	12.50%	56.2±1.1	94.4±0.4	89.1±0.5	80.6±0.7	75.8±0.7	81.6±0.6	67.1±0.9	92.1±0.4	67.6±1.2	54.8±1.1	95.9±0.4	78.9±0.7	66.6±1.1
Ad-R-M	NCC	Ad	residual	M	✗	10.93%	57.3±1.1	94.9±0.4	88.9±0.5	81.0±0.7	76.7±0.7	80.6±0.6	65.4±0.9	91.4±0.5	82.6±1.0	55.0±1.1	96.6±0.4	82.1±0.7	66.4±1.1
Ad-R-CW-PA	NCC	Ad	residual	CW	✓	3.91%	58.6±1.1	94.5±0.4	**90.0±0.4**	80.5±0.8	**77.6±0.7**	81.9±0.6	67.0±0.9	92.2±0.5	80.2±0.9	57.2±1.0	96.1±0.4	81.5±0.8	**71.4±0.9**
Ad-R-M-PA	NCC	Ad	residual	M	✓	13.27%	**59.5±1.0**	**94.9±0.4**	89.9±0.4	**81.1±0.8**	77.5±0.7	81.7±0.6	66.3±0.9	**92.2±0.5**	82.8±1.0	**57.6±1.0**	96.7±0.4	**82.9±0.7**	70.4±1.0

Table C.6: Comparisons to methods that learn classifiers and model adaptation methods during meta-test stage based on URL model. NCC, MD, LR, SVM, Softmax, KNN denote nearest centroid classifier, Mahalanobis distance, logistic regression, support vector machines, softmax classifier and k-nearest neighbors classifier respectively. PA indicates pre-classifier alignment. 'Aux-Net or Ad' indicates using Auxiliary Network to predict α or attaching adapter α directly. 'M or CW' means using matrix multiplication or channel-wise scaling adapters. 'S' and 'R' denote serial adapter and residual adapter, respectively. 'β' indicates using the pre-classifier adaptation. Mean accuracy, 95% confidence interval are reported. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

C.2.2 Task-specific parameterizations

In Tab. C.6, we report an additional 95% confidence interval of each dataset to Sec. 4.4 for the comparison of different r_α choices based on the URL model. The first eight datasets are seen during training and the last five datasets are unseen and used for testing only. We can see that the confidence intervals for different methods have marginal differences.

C.2.3 Results evaluated with updated evaluation protocol.

As the code from Meta-dataset has been updated, we evaluate all methods with the updated evaluation protocol from the Meta-dataset [1] and report the results [2] in Tab. C.7.

[1] As mentioned in `https://github.com/google-research/meta-dataset/issues/54`, we also set the shuffle_buffer_size as 1000 to evaluate all methods and report the results in Tab. C.7. This change does not affect much on the results as the datasets we used were shuffled using the latest data convert code from Meta-Dataset.

[2] The results of Simple CNAPS (Bateni et al., 2020b) and Transductive CNAPS (Bateni et al., 2020a) are reproduced by the authors and reported at `https://github.com/peymanbateni/simple-cnaps`.

As shown in Tab. C.7, the update does not affect much on the results and our method rank 1.5 in average and the state-of-the-art method URL rank 2.7. Our method outperforms other methods on most domains (9 out of 13), especially obtaining significant improvement on 5 unseen datasets than the second best method, *i.e*. Average Unseen (+7.9). More specifically, our method obtains significant better results than the second best approach (URL) on Traffic Sign (+20.2), CIFAR-10 (+8.7), and CIFAR-100 (+7.0).

Test Dataset	CNAPS (Requeima et al., 2019)	Simple CNAPS (Bateni et al., 2020b)	TransductiveCNAPS (Bateni et al., 2020a)	SUR (Dvornik et al., 2020)	URT (Liu et al., 2021b)	FLUTE (Triantafillou et al., 2021)	tri-M (Liu et al., 2021d)	URL (Ours)	Ours
ImageNet	50.8±1.1	56.5±1.1	57.9±1.1	54.5±1.1	55.0±1.1	51.8±1.1	**58.6±1.0**	57.5±1.1	57.4±1.1
Omniglot	91.7±0.5	91.9±0.6	94.3±0.4	93.0±0.5	93.3±0.5	93.2±0.5	92.0±0.6	94.5±0.4	**95.0±0.4**
Aircraft	83.7±0.6	83.8±0.6	84.7±0.5	84.3±0.5	84.5±0.6	87.2±0.5	82.8±0.7	88.6±0.5	**89.3±0.4**
Birds	73.6±0.9	76.1±0.9	78.8±0.7	70.4±1.1	75.8±0.8	79.2±0.8	75.3±0.8	80.5±0.7	**81.4±0.7**
Textures	59.5±0.7	70.0±0.8	66.2±0.8	70.5±0.7	70.6±0.7	68.8±0.8	71.2±0.8	76.2±0.7	**76.7±0.7**
Quick Draw	74.7±0.8	78.3±0.7	77.9±0.6	81.6±0.6	**82.1±0.6**	79.5±0.7	77.3±0.7	81.9±0.6	82.0±0.6
Fungi	50.2±1.1	49.1±1.2	48.9±1.2	65.0±1.0	63.7±1.0	58.1±1.1	48.5±1.0	**68.8±0.9**	67.4±1.0
VGG Flower	88.9±0.5	91.3±0.6	**92.3±0.4**	82.2±0.8	88.3±0.6	91.6±0.6	90.5±0.5	92.1±0.5	92.2±0.5
Traffic Sign	56.5±1.1	59.2±1.0	59.7±1.1	49.8±1.1	50.1±1.1	58.4±1.1	63.0±1.0	63.3±1.2	**83.5±0.9**
MSCOCO	39.4±1.0	42.4±1.1	42.5±1.1	49.4±1.1	48.9±1.1	50.0±1.0	52.8±1.1	54.0±1.0	**55.8±1.1**
MNIST	-	94.3±0.4	94.7±0.3	94.9±0.4	90.5±0.4	95.6±0.5	96.2±0.3	94.5±0.5	**96.7±0.4**
CIFAR-10	-	72.0±0.8	73.6±0.7	64.2±0.9	65.1±0.8	78.6±0.7	75.4±0.8	71.9±0.7	**80.6±0.8**
CIFAR-100	-	60.9±1.1	61.8±1.0	57.1±1.1	57.2±1.0	67.1±1.0	62.0±1.0	62.6±1.0	**69.6±1.0**
Average Seen	71.6	74.6	75.1	75.2	76.7	76.2	74.5	80.0	80.2
Average Unseen	-	65.8	66.5	63.1	62.4	69.9	69.9	69.3	77.2
Average All	-	71.2	71.8	70.5	71.2	73.8	72.7	75.9	79.0
Average Rank	-	6.3	4.9	5.8	5.7	4.3	4.8	2.7	1.5

Table C.7: Comparison state-of-the-art methods on Meta-Dataset (using a multi-domain feature extractor of (Li et al., 2021b)). Mean accuracy, 95% confidence interval are reported. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

C.2.4 Ablation study

Here, we conduct an ablation study of our method with the URL model, unless stated otherwise.

Sensitivity analysis for number of iterations. In our method, we optimize the attached parameters (α, β) with 40 iterations. Figure C.1 and Figure C.2 report the results with 10, 20, 40, 60 iterations and indicates that our method (solid green) converges to a stable solution after 20 iterations and achieves better average performance on all domains than the baseline URL (dash green). The mean accuracy with 95% confidence interval are reported in Tabs. C.8 and C.9

Influence of α and β. We evaluate different components of our method and report the results in Tab. C.10. The results show that both residual adapters α and the linear transformation β help adapt features to unseen classes while residual adapters significantly improve the performance on unseen domains. The best results are achieved by using both α and β.

Results of FLUTE (Triantafillou et al., 2021) and tri-M (Liu et al., 2021d) are from their papers. We reproduce the results of SUR (Dvornik et al., 2020) and URT (Liu et al., 2021b) with the updated evaluation protocol for fair comparison.

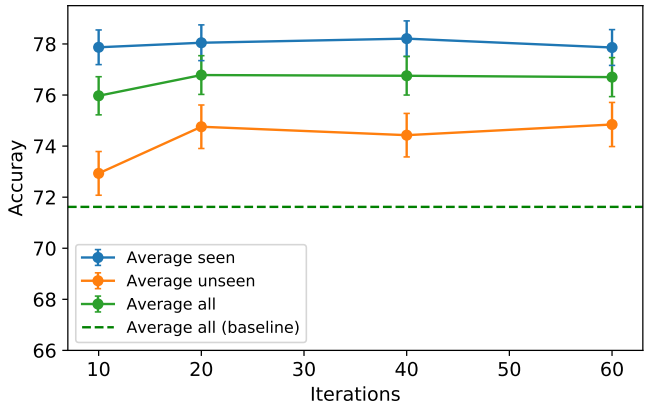


Figure C.1: Sensitivity of performance to number of iterations based on MDL model.

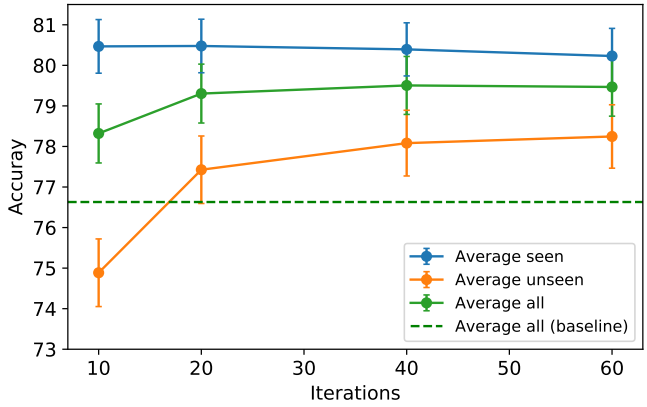


Figure C.2: Sensitivity of performance to number of iterations based on URL model.

Initialization analysis for adapters. Here, we investigate using different initialization strategies for adapters: i) Identity initialization: in this chapter we initialize each residual adapter as an identity matrix scaled by a scalar δ and we set $\delta = 1e-4$; ii) randomly initialization: alternatively, we can randomly initialize each residual adapter. The results of different initialization are summarized in Fig. C.3. We can see that our methods with different initialization strategies obtain similar results, which indicates that our method works also with randomly initialization and again verifies the stability of our method. Detailed results of each dataset are shown in Tab. C.11.

Layer analysis for adapters. Here we investigate whether it is sufficient to attach the adapters only to the later layers. We evaluate this on ResNet18 which is composed of four blocks and attach the adapters to only later blocks (block4, block3,4, block2,3,4 and block-all. Figure C.4 shows that applying our adapters to only the last block (block4) obtains around 78% average accuracy on all domains which outperforms the URL. With attaching residual adapters to more layers, the performance on unseen domains is improved significantly while the one on seen domains remains stable. The mean

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
10 iterations	55.5±1.1	93.9±0.5	86.4±0.5	78.6±0.7	73.3±0.7	81.9±0.6	63.1±0.9	90.3±0.5	77.6±1.0	50.6±1.1	96.9±0.3	77.0±0.8	62.6±1.1
20 iterations	56.2±1.1	94.7±0.4	86.3±0.5	78.3±0.8	73.9±0.7	81.6±0.6	63.4±0.9	90.1±0.6	79.4±1.0	52.8±1.1	97.2±0.3	78.6±0.8	65.9±1.1
40 iterations	55.6±1.0	94.3±0.4	86.7±0.5	79.4±0.8	73.2±0.8	81.7±0.6	64.0±0.9	90.9±0.5	81.1±0.9	51.4±1.1	96.9±0.3	78.5±0.8	64.3±1.1
60 iterations	55.9±1.1	95.1±0.4	85.9±0.6	77.5±0.8	74.7±0.7	80.9±0.6	62.1±0.9	90.7±0.6	82.2±0.9	52.2±1.1	97.0±0.4	78.4±0.8	64.4±1.1

Table C.8: Sensitivity of performance to number of iterations based on MDL model.

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
10 iterations	58.4±1.1	94.8±0.4	89.9±0.4	81.3±0.7	76.6±0.7	81.8±0.6	68.4±0.9	92.5±0.5	76.5±1.1	55.6±1.1	96.4±0.4	79.0±0.7	66.9±1.0
20 iterations	58.2±1.1	94.8±0.4	89.9±0.4	81.1±0.7	77.5±0.8	81.9±0.6	68.0±0.9	92.4±0.5	81.8±1.0	57.8±1.1	96.7±0.4	81.7±0.8	69.1±0.9
40 iterations	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0
60 iterations	58.7±1.1	94.9±0.4	89.5±0.5	80.8±0.7	77.4±0.8	81.8±0.6	66.2±0.9	92.5±0.5	83.7±0.9	56.9±1.0	96.6±0.3	82.0±0.8	72.0±0.9

Table C.9: Sensitivity of performance to number of iterations based on URL model.

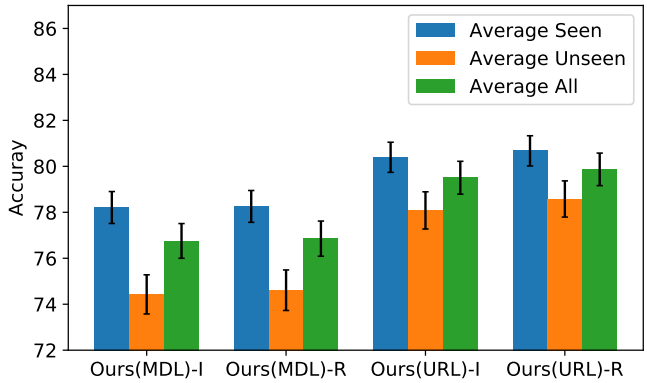


Figure C.3: Initialization analysis for adapters. '-I' indicates identity initialization and '-R' is randomly initialization.

accuracy with 95% confidence interval for layer analysis are shown in Tab. C.12.

Decomposing residual adapters. Here we investigate whether one can reduce the number of parameters in the adapters while retaining its performance by using matrix decomposition. As in deep neural networks, the adapters in earlier layers are relatively small; we then decompose the adapters in the last two blocks only where the adapter dimensionality goes up to 512×512. Figure C.5 shows that our method can achieve good performance with less parameters by decomposing large residual adapters, (*e.g.* when $N = 32$ where the number of additional parameters equal to around 4% vs 13%, the performance is still comparable to the original form of residual adapters, *i.e.* N=0). Results of each dataset in Tab. C.13, also show that, by decomposing large residual adapters, the performance of our method is still comparable to the original form of residual adapters (*i.e.* Ours) with less parameters.

The similar conclusion can be drawn from results (shown in Fig. C.6) of our method using decomposed residual adapters in all layers. When N increases, *i.e.*, smaller residual adapters, the average accuracy on all domains is still comparable to the original

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours w/o α & β	57.0±1.1	94.4±0.4	88.0±0.5	80.3±0.7	74.6±0.7	81.8±0.6	66.2±0.9	91.5±0.5	49.8±1.1	54.1±1.0	91.1±0.4	70.6±0.7	59.1±1.0
Ours w/o β	57.3±1.1	**94.9±0.4**	88.9±0.5	81.0±0.7	76.7±0.7	80.6±0.6	65.4±0.9	91.4±0.5	82.6±1.0	55.0±1.1	96.6±0.4	82.1±0.7	66.4±1.1
Ours w/o α	58.8±1.1	94.5±0.4	89.4±0.4	80.7±0.8	77.2±0.7	**82.5±0.6**	**68.1±0.9**	92.0±0.5	63.3±1.2	57.3±1.0	94.7±0.4	74.2±0.8	63.6±1.0
Ours	**59.5±1.0**	**94.9±0.4**	**89.9±0.4**	**81.1±0.8**	**77.5±0.7**	81.7±0.6	66.3±0.9	**92.2±0.5**	**82.8±1.0**	**57.6±1.0**	**96.7±0.4**	**82.9±0.7**	**70.4±1.0**

Table C.10: Effect of each component. We build our method on the URL model and 'Ours w/o α & β' means we remove both residual adapters α and the pre-classifier adaptation layer β in our method.

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours(SDL-ResNet-18)-I	59.5±1.1	78.2±1.2	72.2±1.0	74.9±0.9	77.3±0.7	67.6±0.9	44.7±1.0	90.9±0.6	82.5±0.8	59.0±1.0	93.9±0.6	82.1±0.7	70.7±0.9
Ours(SDL-ResNet-18)-R	58.2±1.0	78.4±1.2	71.1±1.1	74.4±1.0	77.1±0.7	67.2±1.0	45.9±1.0	90.7±0.6	81.9±1.0	57.7±1.1	94.1±0.5	81.9±0.7	70.5±0.9
Ours(MDL)-I	55.6±1.0	94.3±0.4	86.7±0.5	79.4±0.8	73.2±0.8	81.7±0.6	64.0±0.9	90.9±0.5	81.1±0.9	51.4±1.1	96.9±0.3	78.5±0.8	64.3±1.1
Ours(MDL)-R	56.0±1.1	94.1±0.4	87.1±0.5	79.7±0.8	74.0±0.7	82.0±0.6	62.6±0.9	90.6±0.6	80.9±0.9	51.7±1.1	96.9±0.4	77.7±0.9	65.8±1.1
Ours(URL)-I	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0
Ours(URL)-R	58.8±1.1	94.9±0.4	90.5±0.4	81.8±0.6	77.7±0.7	82.3±0.6	66.8±0.9	92.6±0.5	83.7±0.8	57.7±1.1	96.9±0.4	82.5±0.7	72.0±0.9

Table C.11: Initialization analysis of adapters. 'Ours(URL)-I' indicates our method using URL as the pretrained model and initializing residual adapters as identity matrix (scaled by $\delta = 0.0001$) while 'Ours(URL)-R' means our method initialize residual adapters randomly.

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours (block4)	59.0±1.1	95.0±0.4	90.0±0.4	80.6±0.8	77.8±0.7	82.3±0.6	68.2±0.9	91.8±0.6	70.6±1.1	57.1±1.1	95.9±0.4	77.2±0.8	65.9±1.0
Ours (block3,4)	60.4±1.1	94.7±0.4	90.0±0.5	80.4±0.7	77.8±0.7	82.2±0.6	67.2±0.8	92.5±0.5	77.2±1.0	57.9±1.0	96.7±0.3	78.8±0.9	68.6±0.9
Ours (block2,3,4)	59.6±1.1	94.9±0.4	89.9±0.5	81.0±0.8	78.2±0.7	82.4±0.6	67.6±0.9	92.3±0.5	81.5±1.0	57.9±1.0	96.6±0.4	81.5±0.8	70.6±1.0
Ours (block-all)	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0

Table C.12: Block (layer) analysis for adapters based on URL model.

form of residual adapters (*i.e.* N=0) with less parameters though the average accuracy on unseen domains drops slightly. From the results depicted in Tab. C.14, we can see that when N increases, the performance of most domains are still comparable to the original form of residual adapters (*i.e.* Ours) while the performance on Traffic Sign drops slightly as the adapters in earlier layers are small and when N is larger the decomposed residual adapters might be too small to tranform the features. Overall, our method can achieve good performance with less parameters by decomposing large residual adapters.

Training time. The training time (meta-train) of our method is equal to the one of URL (hence no additional cost), *i.e.* 48 hours in multi-domain setting, 6 hours for Resnet-18 and 33 hours for Resnet-34 in single-domain learning in one Nvidia V100 GPU. Whereas CTX meta-training requires 8 Nvidia V100 GPUs for 7 days and approximately 40 times more expensive than ours. During the meta-test stage, the model parameters are further trained using the support set of each episode. Meta-test training cost is depicted in Tab. C.15 for Meta-Dataset tasks. URL baseline only finetunes parameters of PA β. Finetune+NCC updates the entire backbone parameters. Ours

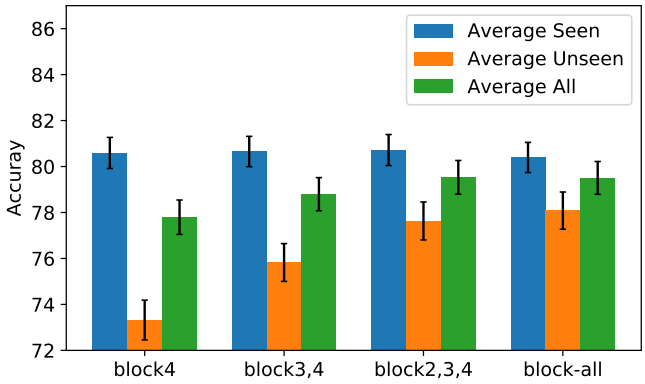


Figure C.4: Block (layer) analysis for adapters.

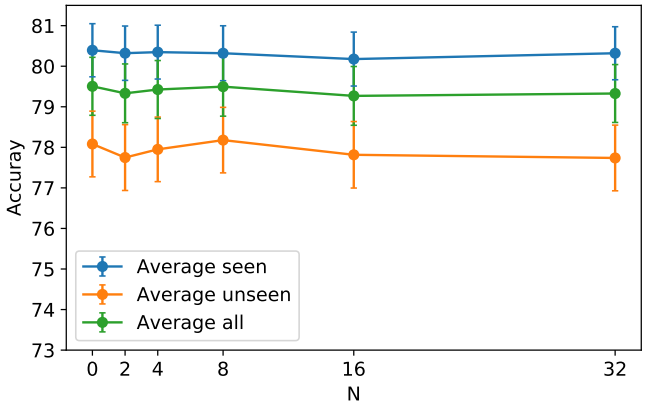


Figure C.5: Decomposed residual adapters on block-3,4.

learn RA and PA parameters. While URL is the fastest baseline, as it does not require backpropagating the error to early layers, ours is more efficient than finetuning all the backbone parameters.

C.2.5 Qualitative results

We qualitatively analyze our methods (URL and TSA) and compare it to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Figs. C.7 to C.19 by illustrating the nearest neighbors in all test datasets given a query image as Li et al. (2021b). It is clear that our methods produce more correct neighbors than other methods. While other methods retrieve images with more similar colors, shapes and backgrounds, *e.g.* in Figs. C.15, C.16, C.18 and C.19, our method is able to retrieve semantically similar images. More specifically, as shown in Fig. C.10, by learning task-agnostic weights by URL and adapting the task-agnostic weight by TSA, our whole method correctly produces neighbors of the bird in the query image while other methods pick images with similar appearances or similar background, *e.g.*

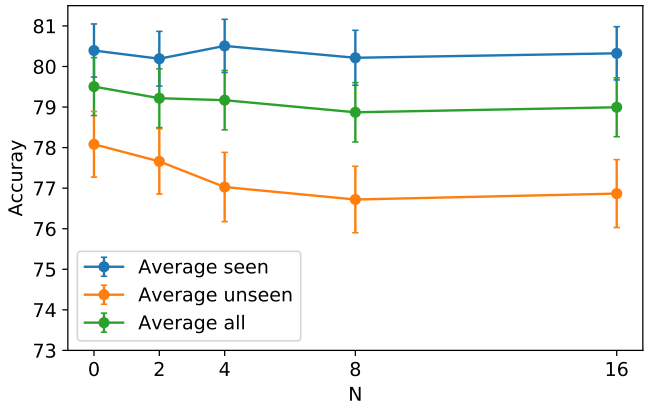


Figure C.6: Decomposed residual adapters on all layers.

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0
Ours(N=2)	58.9±1.1	95.2±0.4	89.7±0.5	80.9±0.7	76.7±0.7	81.4±0.6	67.7±0.9	92.2±0.5	82.4±1.0	57.1±1.0	96.5±0.4	82.4±0.7	70.3±1.0
Ours(N=4)	58.7±1.1	94.9±0.4	89.7±0.5	80.3±0.7	77.0±0.7	82.5±0.6	67.2±0.9	92.5±0.5	82.6±1.0	57.5±1.1	96.5±0.4	82.5±0.7	70.8±0.9
Ours(N=8)	59.1±1.1	95.0±0.4	89.8±0.5	80.2±0.8	77.2±0.7	82.1±0.6	67.0±0.9	92.2±0.5	82.5±1.0	57.2±1.1	96.8±0.4	82.6±0.7	71.8±0.9
Ours(N=16)	58.2±1.1	94.7±0.4	90.1±0.4	80.3±0.8	76.9±0.7	81.7±0.6	67.6±0.9	92.0±0.5	81.8±1.0	58.1±1.1	96.4±0.4	81.8±0.7	71.1±0.9
Ours(N=32)	59.2±1.1	94.8±0.4	89.6±0.5	80.0±0.8	77.3±0.6	82.4±0.6	67.2±0.9	92.1±0.5	82.1±1.0	57.1±1.0	96.7±0.3	81.6±0.8	71.1±0.9

Table C.13: Results of using decomposed RA on layer3,4.

Test Dataset	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO	MNIST	CIFAR-10	CIFAR-100
Ours	59.5±1.0	94.9±0.4	89.9±0.4	81.1±0.8	77.5±0.7	81.7±0.6	66.3±0.9	92.2±0.5	82.8±1.0	57.6±1.0	96.7±0.4	82.9±0.7	70.4±1.0
Ours(N=2)	58.1±1.1	94.8±0.4	89.7±0.5	80.2±0.8	76.9±0.7	82.1±0.6	67.8±0.9	92.0±0.6	82.5±0.9	56.9±1.1	96.7±0.3	82.0±0.8	70.3±1.0
Ours(N=4)	59.6±1.1	94.8±0.4	89.9±0.5	80.3±0.8	77.4±0.7	82.6±0.6	66.6±0.9	92.9±0.5	79.7±1.1	57.6±1.1	96.5±0.4	80.9±0.8	70.6±1.0
Ours(N=8)	58.2±1.1	94.6±0.4	89.6±0.5	81.2±0.8	76.6±0.7	82.7±0.6	66.5±0.9	92.3±0.5	78.1±1.1	57.3±1.0	96.3±0.3	81.0±0.8	70.9±0.9
Ours(N=16)	58.9±1.1	94.6±0.4	89.7±0.5	80.1±0.7	77.0±0.7	82.1±0.6	68.4±0.9	91.9±0.5	78.3±1.0	57.8±1.1	96.0±0.4	82.0±0.7	70.3±1.0

Table C.14: Results of using decomposed RA on all layers.

Test Dataset	Image -Net	Omni -glot	Air- craft	Birds	Tex- tures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MS- COCO	MNIST	CIFAR -10	CIFAR -100
URL	0.7	0.7	0.4	0.7	0.4	1.0	1.0	0.5	0.9	0.9	0.4	0.4	1.0
Finetune+NCC	7.7	2.5	7.4	7.0	5.8	9.3	8.7	6.6	9.1	9.0	6.5	6.7	9.3
Ours (URL+RA+PA)	7.2	2.4	6.1	6.8	4.8	8.9	7.4	5.2	8.8	8.3	6.0	6.2	8.6

Table C.15: Computation cost (# second per task) during meta-test.

images with twigs. In Fig. C.15, other methods and even URL mainly retrieve the triangle sign while our TSA method is able to retrieve the correct sign with illumination distortion. In Fig. C.19, other methods including SUR, URT are distracted by the blue background. While the URL learns more generalized features and is less distracted by the color and background. By using our TSA with the URL task-agnostic model, our TSA method selects the correct shark images. It again suggests that the URL method provides an efficient and effective way to learn a single task-agnostic network and generalized features from multiple domains and our TSA method is able to quickly adapt the features for unseen few-shot tasks.

Figure C.7: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in ImageNet. Green and red colors indicate correct and false predictions respectively.

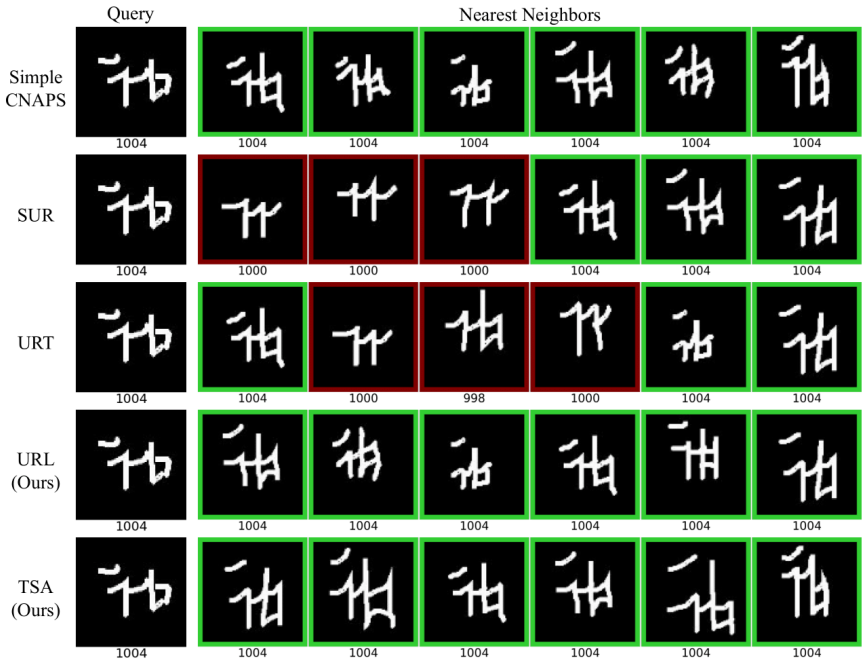


Figure C.8: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Omniglot. Green and red colors indicate correct and false predictions respectively.

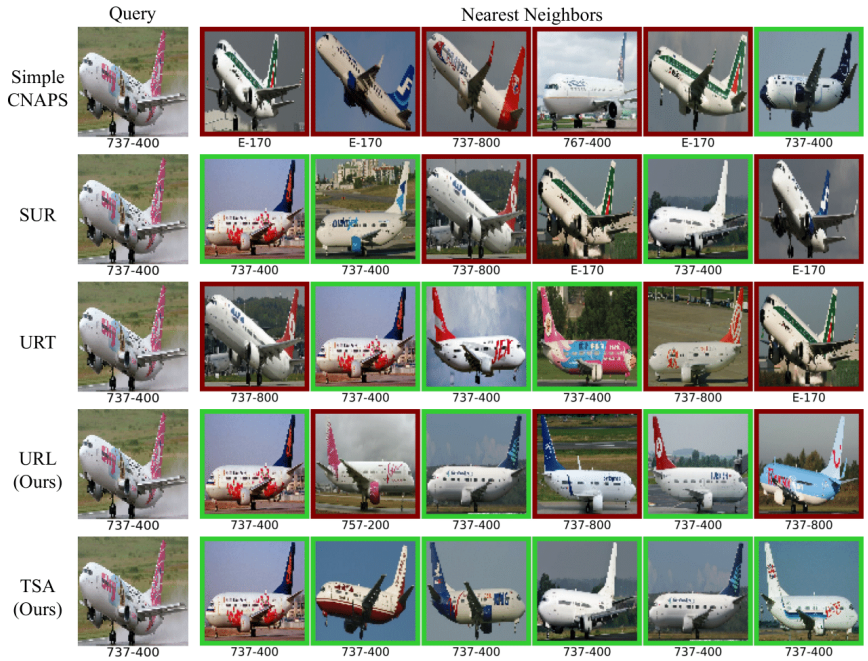


Figure C.9: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Aircraft. Green and red colors indicate correct and false predictions respectively.

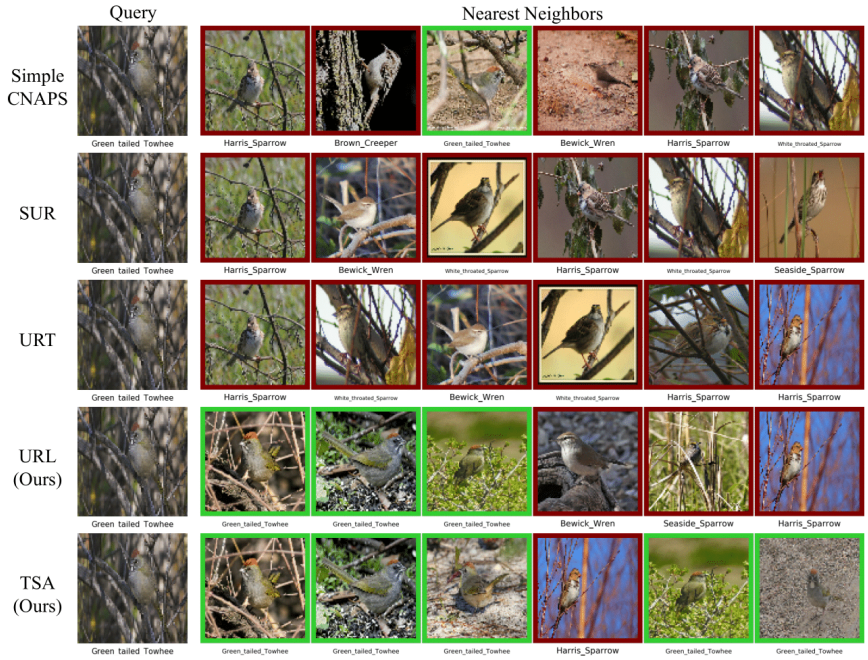


Figure C.10: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Birds. Green and red colors indicate correct and false predictions respectively.

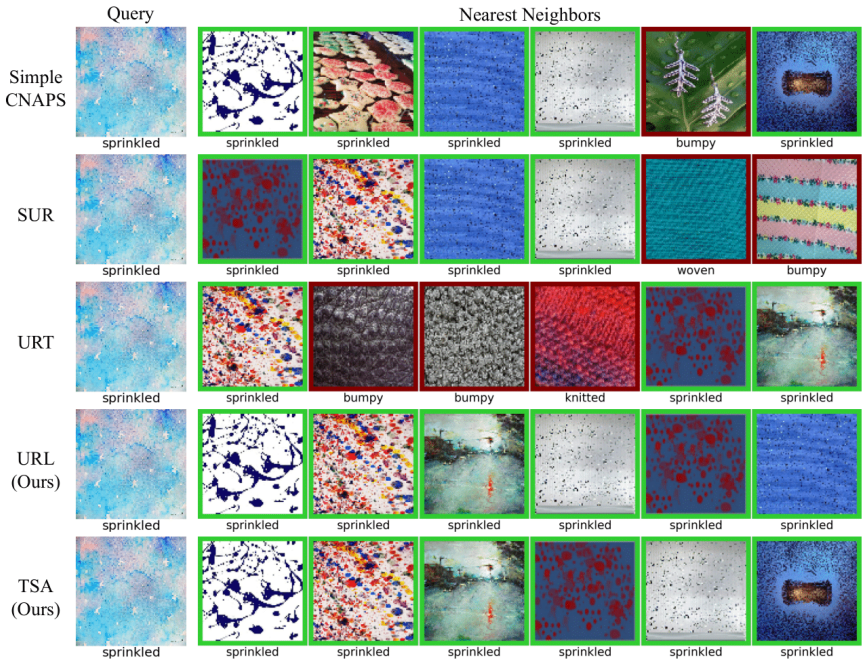


Figure C.11: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Textures. Green and red colors indicate correct and false predictions respectively.

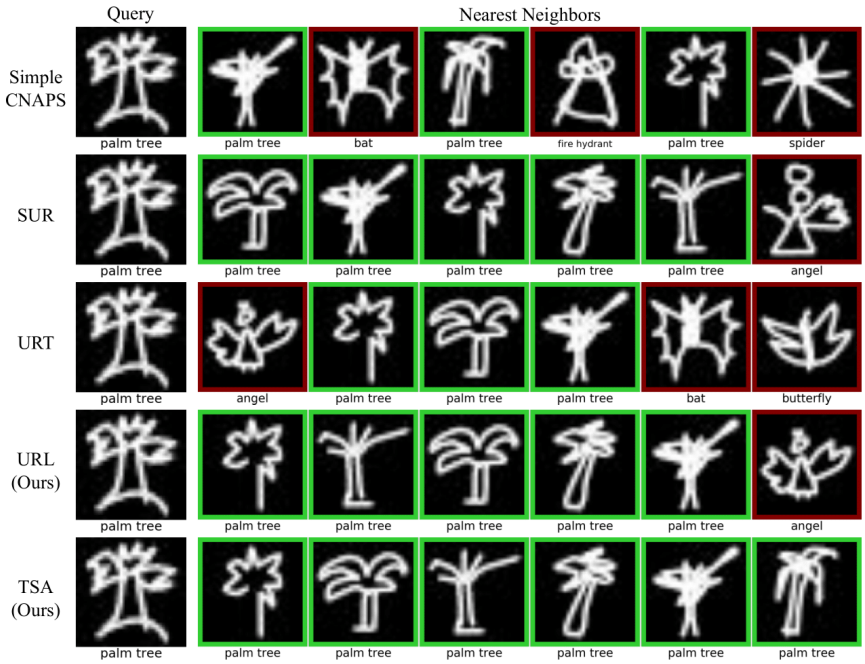


Figure C.12: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Quick Draw. Green and red colors indicate correct and false predictions respectively.

Figure C.13: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Fungi. Green and red colors indicate correct and false predictions respectively.

Figure C.14: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in VGG Flower. Green and red colors indicate correct and false predictions respectively.

Figure C.15: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in Traffic Sign. Green and red colors indicate correct and false predictions respectively.

Figure C.16: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in MSCOCO. Green and red colors indicate correct and false predictions respectively.

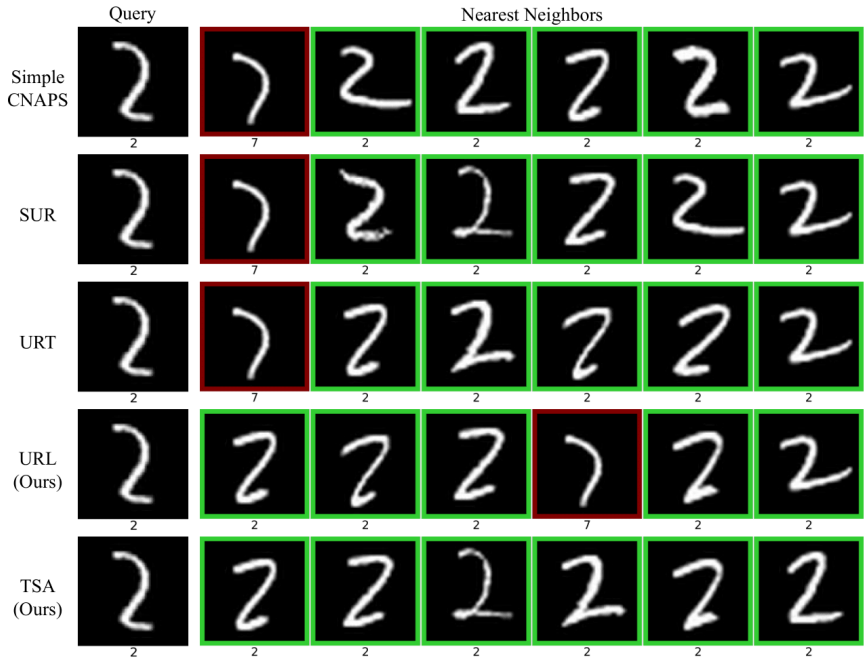


Figure C.17: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in MNIST. Green and red colors indicate correct and false predictions respectively.

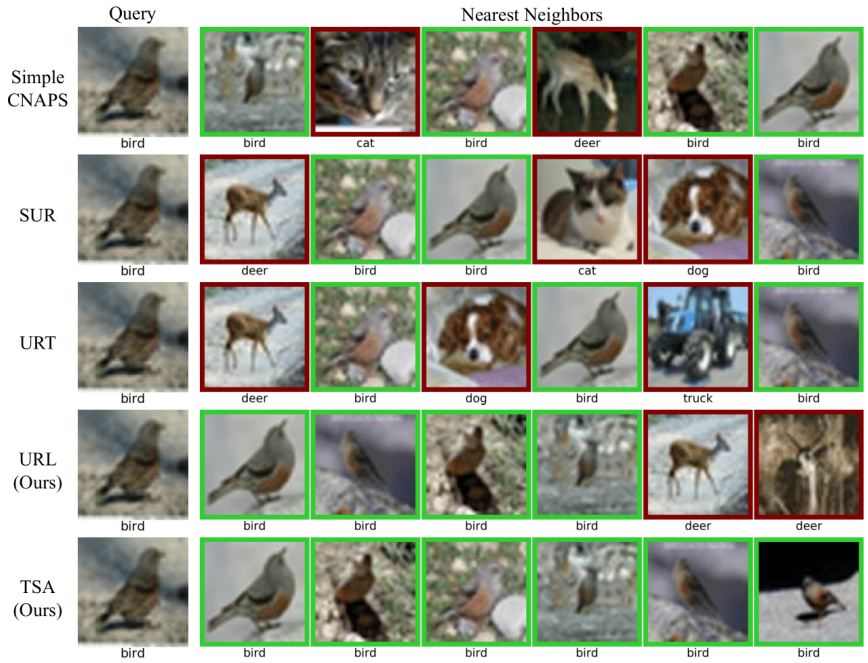


Figure C.18: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in CIFAR-10. Green and red colors indicate correct and false predictions respectively.

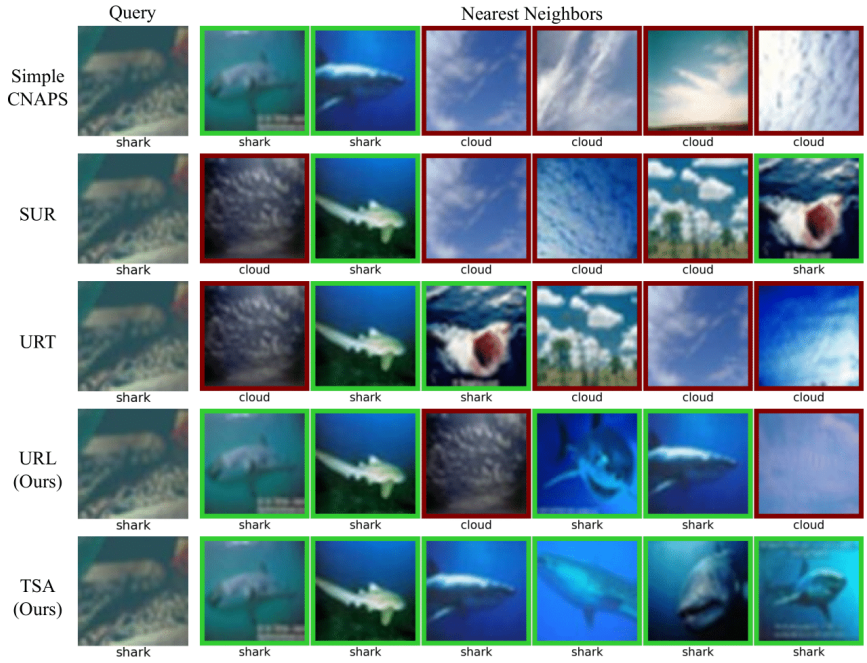


Figure C.19: Qualitative comparison to Simple CNAPS (Bateni et al., 2020b), SUR (Dvornik et al., 2020), and URT (Liu et al., 2021b) in CIFAR-100. Green and red colors indicate correct and false predictions respectively.

Bibliography

Adler, T., Brandstetter, J., Widrich, M., Mayr, A., Kreil, D., Kopp, M., Klambauer, G., and Hochreiter, S. (2020). Cross-domain few-shot learning by representation fusion. *arXiv preprint arXiv:2010.06498*. 61

Atkinson, J. (2002). The developing visual brain. 10

Bachmann, R., Mizrahi, D., Atanov, A., and Zamir, A. (2022). Multimae: Multi-modal multi-task masked autoencoders. In *ECCV*. 91

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 39(12):2481–2495. 23, 24, 34, 47, 98, 103, 104

Bao, Y., Li, Y., Huang, S.-L., Zhang, L., Zheng, L., Zamir, A., and Guibas, L. (2019). An information-theoretic approach to transferability in task transfer learning. In *ICIP*, pages 2309–2313. IEEE. 93

Bateni, P., Barber, J., van de Meent, J.-W., and Wood, F. (2020a). Enhancing few-shot image classification with unlabelled examples. *arXiv preprint arXiv:2006.12245*. 4, 63, 74, 122, 123

Bateni, P., Goyal, R., Masrani, V., Wood, F., and Sigal, L. (2020b). Improved few-shot visual classification. In *CVPR*, pages 14493–14502. 4, 60, 61, 62, 63, 68, 70, 72, 73, 74, 78, 84, 86, 87, 119, 121, 122, 123, 127, 129, 130, 131, 132, 133, 134, 135

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*. 43

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. (2016). Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531. 61

Bilen, H. and Vedaldi, A. (2016). Integrated perception with recurrent multi-task neural networks. In *Advances in Neural Information Processing Systems*, pages 235–243. 10, 13, 41

Bilen, H. and Vedaldi, A. (2017). Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*. 10, 15, 64

Bragman, F. J., Tanno, R., Ourselin, S., Alexander, D. C., and Cardoso, J. (2019). Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *ICCV*, pages 1385–1394. 13, 37

Brigit, S. and Yin, C. (2018). Fgvcx fungi classification challenge. *online*. 1, 9, 71

Bruggemann, D., Kanakis, M., Georgoulis, S., and Van Gool, L. (2020). Automated search for resource-efficient branched multi-task networks. *arXiv preprint arXiv:2008.10292*. 10, 13, 37

Bruggemann, D., Kanakis, M., Obukhov, A., Georgoulis, S., and Van Gool, L. (2021). Exploring relational context for multi-task dense prediction. In *ICCV*. 13, 91

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75. 1, 13, 40, 94

Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2019). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, volume 33, pages 8001–8008. 41

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *ECCV*, pages 233–248. 92

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 40(4):834–848. 105

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018a). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818. 48, 105

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR. 76

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. (2019). A closer look at few-shot classification. In *ICLR*. 63

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, pages 1971–1978. 47, 90, 104, 105

Chen, Y., Wang, X., Liu, Z., Xu, H., and Darrell, T. (2020b). A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*. 60, 63, 68, 69

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018b). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, pages 794–803. PMLR. 2, 11, 13, 18, 24, 26, 27, 38, 52, 61, 65, 111, 112

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. (2020c). Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *NeurIPS*. 13, 24, 26, 27, 38

Chen, Z., Zhu, L., Wan, L., Wang, S., Feng, W., and Heng, P.-A. (2020d). A multi-task mean teacher for semi-supervised shadow detection. In *CVPR*, pages 5611–5620. 39, 40

Chennupati, S., Sistu, G., Yogamani, S., and A Rawashdeh, S. (2019). Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *CVPR Workshop*. 13

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *CVPR*, pages 3606–3613. 9, 30, 71, 98

Clark, K., Luong, M.-T., Khandelwal, U., Manning, C. D., and Le, Q. V. (2019). Bam! born-again multi-task networks for natural language understanding. In *ACL*. 16, 20, 24, 26, 27

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223. 2, 22, 23, 38, 47, 94, 97, 103, 104, 105

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123. 76

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, pages 3150–3158. 9

Deecke, L., Hospedales, T., and Bilen, H. (2022). Visual representation learning over latent domains. In *ICLR*. 10, 15

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. 12, 15, 64

Dhar, P., Kumar, A., Kaplan, K., Gupta, K., Ranjan, R., and Chellappa, R. (2022). Eyepad++: A distillation-based approach for joint eye authentication and presentation attack detection using periocular images. In *CVPR*, pages 20218–20227. 94

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. (2020). A baseline for few-shot image classification. In *ICLR*. 60, 63, 68, 69

Doersch, C., Gupta, A., and Zisserman, A. (2020). Crosstransformers: spatially-aware few-shot transfer. In *NeurIPS*. 60, 63, 76, 78, 85, 119

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 13, 91, 94

Dvornik, N., Schmid, C., and Mairal, J. (2020). Selecting relevant features from a multi-domain representation for few-shot classification. In *ECCV*, pages 769–786. 4, 11, 12, 60, 61, 62, 64, 65, 66, 68, 70, 71, 72, 73, 74, 84, 86, 87, 117, 118, 119, 123, 127, 129, 130, 131, 132, 133, 134, 135

Dwivedi, K., Huang, J., Cichy, R. M., and Roig, G. (2020). Duality diagram similarity: a generic framework for initialization selection in task transfer learning. In *ECCV*, pages 497–513. Springer. 93

Dwivedi, K. and Roig, G. (2019). Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR*, pages 12387–12396. 93

Eftekhar, A., Sax, A., Malik, J., and Zamir, A. (2021). Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, pages 10786–10796. 90

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658. 23, 47, 97

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*. 1, 9

Eitz, M., Hays, J., and Alexa, M. (2012). How do humans sketch objects? *TOG*, 31(4):1–10. 9

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338. 47

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. *NeurIPS*, 34:27503–27516. 92, 93

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICLR*, pages 1126–1135. 58, 59, 63

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born again neural networks. In *ICML*. 16

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030. 15

Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. (2019). Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR*, pages 3205–3214. 13, 37

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE. 38

Ghiasi, G., Zoph, B., Cubuk, E. D., Le, Q. V., and Lin, T.-Y. (2021). Multi-task self-training for learning general representations. In *ICCV*, pages 8856–8865. 92, 93, 94

Gong, T., Lee, T., Stephenson, C., Renduchintala, V., Padhy, S., Ndirango, A., Keskin, G., and Elibol, O. H. (2019). A comparison of loss weighting strategies for multi task learning in deep neural networks. *IEEE Access*, 7:141627–141632. 2, 38

Guizilini, V., Li, J., Ambrus, R., Pillai, S., and Gaidon, A. (2020). Robust semi-supervised monocular depth estimation with reprojected distances. In *Conference on robot learning*, pages 503–512. PMLR. 43

Guo, M., Haque, A., Huang, D.-A., Yeung, S., and Fei-Fei, L. (2018). Dynamic task prioritization for multitask learning. In *ECCV*, pages 270–287. 2, 11, 13, 14, 38

Guo, P., Lee, C.-Y., and Ulbricht, D. (2020). Learning to branch for multi-task learning. In *ICML*, pages 3854–3863. PMLR. 10, 13, 37, 91

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. 21, 31, 48, 69, 71, 99, 105

He, Y., Huang, G., Chen, S., Teng, J., Kun, W., Yin, Z., Sheng, L., Liu, Z., Qiao, Y., and Shao, J. (2022). X-learner: Learning cross sources and tasks for universal visual representation. *arXiv preprint arXiv:2203.08764*. 92, 93, 94

Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NeurIPS Deep Learning Workshop*. 11, 16, 19, 20, 65, 66

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, pages 1989–1998. PMLR. 15

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*. 59, 62

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. (2013). Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, pages 1–8. Ieee. 30, 71, 99

Hoyer, L., Dai, D., Wang, Q., Chen, Y., and Van Gool, L. (2021). Improving semi-supervised and domain-adaptive semantic segmentation with self-supervised depth estimation. *arXiv preprint arXiv:2108.12545*. 41

Hu, S. X., Li, D., Stühmer, J., Kim, M., and Hospedales, T. M. (2022). Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *CVPR*, pages 9068–9077. 94

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Xiao, Y., Hao, D., Qian, Z., and Terzopoulos, D. (2020). Partly supervised multitask learning. *arXiv preprint arXiv:2005.02523*. 39, 40, 41

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. (2016). Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675. 61

Jongejan, J., Henry, R., Takashi, K., Jongmin, K., and Nick, F.-G. (2016). The quick, draw! a.i. experiment. *online*. 71

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491. 2, 11, 12, 13, 14, 24, 26, 27, 28, 38, 52, 106, 111, 112

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526. 92

Komodakis, N. and Zagoruyko, S. (2017). Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *ICLR*. 20, 33

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *ICML*, pages 3519–3529. PMLR. 5, 16, 21, 22, 61

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Citeseer*. 30, 71, 98

Kuznietsov, Y., Stuckler, J., and Leibe, B. (2017). Semi-supervised deep learning for monocular depth map prediction. In *CVPR*, pages 6647–6655. 43

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. (2011). One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33. 4, 58

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338. 9, 12, 31, 59, 71, 99

Lambert, J., Liu, Z., Sener, O., Hays, J., and Koltun, V. (2020). Mseg: A composite dataset for multi-domain semantic segmentation. In *CVPR*, pages 2879–2888. 94

Latif, S., Rana, R., Khalifa, S., Jurdak, R., Epps, J., and Schuller, B. W. (2019). Multi-task semi-supervised adversarial autoencoding for speech emotion recognition. *arXiv preprint arXiv:1907.06078*. 39, 40

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 71

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665. 60, 61, 63, 69

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017). Overcoming catastrophic forgetting by incremental moment matching. *NeurIPS*, 30. 92

Lewis, T. L. and Maurer, D. (2005). Multiple sensitive periods in human visual development: evidence from visually deprived children. *Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology*, 46(3):163–183. 10

Li, B., Pang, R., Sainath, T. N., Gulati, A., Zhang, Y., Qin, J., Haghani, P., Huang, W. R., Ma, M., and Bai, J. (2021a). Scaling end-to-end models for large-scale multilingual asr. In *ASRU*, pages 1011–1018. IEEE. 94

Li, W.-H. and Bilen, H. (2020). Knowledge distillation for multi-task learning. In *ECCV Workshop on Imbalance Problems in Computer Vision*, pages 163–176. Springer. 16, 38, 65, 66, 106

Li, W.-H., Liu, X., and Bilen, H. (2021b). Universal representation learning from multiple domains for few-shot classification. *ICCV*. 60, 68, 73, 78, 118, 119, 123, 127

Li, W.-H., Liu, X., and Bilen, H. (2022). Learning multiple dense prediction tasks from partially annotated data. In *CVPR*. 24

Liang, J., Meyerson, E., and Miikkulainen, R. (2018). Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 466–473. 10

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer. 71

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. (2019). Pareto multi-task learning. *NeurIPS*, 32:12060–12070. 13, 38

Liu, B., Gould, S., and Koller, D. (2010). Single image depth estimation from predicted semantic labels. In *CVPR*, pages 1253–1260. IEEE. 41

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. (2021a). Conflict-averse gradient descent for multi-task learning. *NeurIPS*. 2, 12, 13, 14, 23, 24, 26, 27, 28, 97

Liu, B., Yu, H., and Qi, G. (2022a). Graftnet: Towards domain generalized stereo matching with a broad-spectrum and task-oriented feature. In *CVPR*, pages 13012–13021. 94

Liu, L., Hamilton, W., Long, G., Jiang, J., and Larochelle, H. (2021b). A universal representation transformer layer for few-shot image classification. In *ICLR*. 4, 11, 12, 60, 61, 62, 64, 65, 66, 72, 73, 74, 84, 86, 87, 123, 127, 129, 130, 131, 132, 133, 134, 135

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., and Zhang, W. (2021c). Towards impartial multi-task learning. In *ICLR*. 4, 12, 13, 24, 26, 27, 28, 35, 100

Liu, Q., Liao, X., and Carin, L. (2008). Semi-supervised multitask learning. In *NeurIPS*, pages 937–944. 39, 40

Liu, S., James, S., Davison, A. J., and Johns, E. (2022b). Auto-lambda: Disentangling dynamic task relationships. *TMLR*. 13, 14

Liu, S., Johns, E., and Davison, A. J. (2019). End-to-end multi-task learning with attention. In *CVPR*, pages 1871–1880. 2, 10, 11, 13, 23, 24, 26, 27, 37, 38, 47, 52, 91, 97, 98, 103, 104, 105, 111, 112

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer. 9

Liu, Y., Lee, J., Zhu, L., Chen, L., Shi, H., and Yang, Y. (2021d). A multi-mode modulator for multi-domain few-shot classification. In *ICCV*, pages 8453–8462. 62, 64, 74, 123

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440. 1

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. (2017). Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, pages 5334–5343. 13, 37

Lu, Y., Pirk, S., Dlabal, J., Brohan, A., Pasad, A., Chen, Z., Casser, V., Angelova, A., and Gordon, A. (2021). Taskology: Utilizing task relations at scale. In *CVPR*, pages 8700–8709. 39, 41, 43

Ma, J. and Mei, Q. (2019). Graph representation learning via multi-task knowledge distillation. In *NeurIPS GRL Workshop*. 16

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*. 12, 30, 71, 98

Mallya, A., Davis, D., and Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, pages 67–82. 30, 31, 91

Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549. 48, 106

Massiceti, D., Zintgraf, L., Bronskill, J., Theodorou, L., Harris, M. T., Cutrell, E., Morrison, C., Hofmann, K., and Stumpf, S. (2021). Orbit: A real-world few-shot dataset for teachable object recognition. In *CVPR*, pages 10818–10828. 95

Matena, M. and Raffel, C. (2021). Merging models with fisher-weighted averaging. *arXiv preprint arXiv:2111.09832*. 91

Maurer, D. and Lewis, T. L. (2001). Visual acuity: the role of visual input in inducing postnatal change. *Clinical Neuroscience Research*, 1(4):239–247. 10

Mendel, R., De Souza, L. A., Rauber, D., Papa, J. P., and Palm, C. (2020). Semi-supervised segmentation based on error-correcting supervision. In *ECCV*, pages 141–157. Springer. 43

Mensink, T., Uijlings, J., Kuznetsova, A., Gygli, M., and Ferrari, V. (2022). Factors of influence for transfer learning across diverse appearance domains and task types. *PAMI*. 93

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 35(11):2624–2637. 70

Miller, E. G., Matsakis, N. E., and Viola, P. A. (2000). Learning from one example through shared densities on transforms. In *CVPR*, volume 1, pages 464–471. IEEE. 4, 58

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *CVPR*, pages 3994–4003. 2, 10, 13, 37, 91

Munder, S. and Gavrila, D. M. (2006). An experimental study on pedestrian classification. *PAMI*, 28(11):1863–1868. 30, 98

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 31, 99

Nguyen, C., Hassner, T., Seeger, M., and Archambeau, C. (2020). Leep: A new measure to evaluate transferability of learned representations. In *ICML*, pages 7294–7305. PMLR. 93

Nguyen, T., Raghu, M., and Kornblith, S. (2021). Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *ICLR*. 22

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE. 9, 30, 71, 99

Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012. 79, 120

Olsson, V., Tranheden, W., Pinto, J., and Svensson, L. (2021). Classmix: Segmentation-based data augmentation for semi-supervised learning. In *WACV*, pages 1369–1378. 43

Oreshkin, B. N., Rodriguez, P., and Lacoste, A. (2018). Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*. 58, 63

Pándy, M., Agostinelli, A., Uijlings, J., Ferrari, V., and Mensink, T. (2022). Transferability estimation using bhattacharyya class separability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9172–9182. 91, 93

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 41.1–41.12. BMVA Press. 9

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., and Garnett, R., editors, *NeurIPS*, pages 8024–8035. Curran Associates, Inc. 71

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415. 15

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 45, 61, 62, 69

Phuong, M. and Lampert, C. (2019). Towards understanding knowledge distillation. In *ICML*, pages 5142–5151. 16

Poggi, M., Aleotti, F., Tosi, F., and Mattoccia, S. (2020). On the uncertainty of self-supervised monocular depth estimation. In *CVPR*, pages 3227–3237. 2

Ramanujan, V., Vasu, P. K. A., Farhadi, A., Tuzel, O., and Pouransari, H. (2022). Forward compatible training for large-scale embedding retrieval systems. In *CVPR*, pages 19386–19395. 94

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *PAMI*. 94

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. 58

Raychaudhuri, D. S., Suh, Y., Schulter, S., Yu, X., Faraki, M., Roy-Chowdhury, A. K., and Chandraker, M. (2022). Controllable dynamic multi-task architectures. In *CVPR*, pages 10955–10964. 13

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2017a). Learning multiple visual domains with residual adapters. In *NeurIPS*. 1, 2, 10, 15, 17, 21, 22, 30, 31, 61, 64, 90, 94, 98, 99

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2018). Efficient parametrization of multi-domain deep neural networks. In *CVPR*, pages 8119–8127. 10, 12, 15, 30, 31, 34, 69, 99

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017b). icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010. 92

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. In *ICLR*. 59

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28. 9

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes. In *NeurIPS*. 4, 60, 61, 62, 63, 68, 69, 72, 73, 74, 123

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *ICLR*. 11, 16, 19, 20, 66

Rosenfeld, A. and Tsotsos, J. K. (2018). Incremental learning through deep adaptation. *PAMI*, 42(3):651–663. 10, 15, 30, 31

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*. 40

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *AAAI*, volume 33, pages 4822–4829. 10, 13, 37

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252. 1, 30, 71, 98

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2020). Meta-learning with latent embedding optimization. In *ICLR*. 58, 63

Saha, S., Obukhov, A., Paudel, D. P., Kanakis, M., Chen, Y., Georgoulis, S., and Van Gool, L. (2021). Learning to relate depth and semantics for unsupervised domain adaptation. In *CVPR*, pages 8197–8207. 39, 41

Saikia, T., Brox, T., and Schmid, C. (2020). Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*. 63, 76

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. 9

Seenivasan, L., Mitheran, S., Islam, M., and Ren, H. (2022). Global-reasoned multi-task learning model for surgical scene understanding. *IEEE Robotics and Automation Letters*, 7(2):3858–3865. 94

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. *NeurIPS*. 11, 13, 14, 24, 26, 27, 38, 52, 111, 112

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer. iii, 1, 2, 22, 23, 47, 90, 94, 97, 104, 105

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *NeurIPS*. 58, 63, 70

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*. 43

Soomro, K., Zamir, A. R., and Shah, M. (2012). A dataset of 101 human action classes from videos in the wild. *arXiv preprint arXiv:1212.0402*. 1, 31, 99

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *ICML*, pages 9120–9132. PMLR. 92

Sun, B., Xing, J., Blum, H., Siegwart, R., and Cadena, C. (2021). See yourself in others: Attending multiple tasks for own failure detection. *arXiv preprint arXiv:2110.02549*. 41

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019a). Deep high-resolution representation learning for human pose estimation. In *CVPR*, pages 5693–5703. 24, 28, 98

Sun, Y., Tzeng, E., Darrell, T., and Efros, A. A. (2019b). Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*. 15

Suteu, M. and Guo, Y. (2019). Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*. 13

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708. 9

Terzopoulos, D. et al. (2019). Semi-supervised multi-task learning with chest x-ray images. In *International Workshop on Machine Learning in Medical Imaging*, pages 151–159. Springer. 39, 40

Tian, Y., Krishnan, D., and Isola, P. (2020a). Contrastive representation distillation. In *ICLR*. 16

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020b). Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*. 16, 63, 68

Triantafillou, E., Larochelle, H., Zemel, R., and Dumoulin, V. (2021). Learning a universal template for few-shot dataset generalization. In *ICML*. 4, 60, 61, 62, 63, 64, 68, 73, 74, 76, 123

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., et al. (2020). Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*. 4, 6, 15, 17, 22, 34, 59, 62, 71, 72, 73, 76, 94, 118

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176. 15

Vandenhende, S., Georgoulis, S., De Brabandere, B., and Van Gool, L. (2020a). Branched multi-task networks: deciding what layers to share. In *BMVC*. 2, 10, 13, 37, 91

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. *PAMI*. 1, 11, 19, 23, 24, 28, 38, 40, 47, 48, 90, 94, 98, 105, 106, 107, 108, 109

Vandenhende, S., Georgoulis, S., and Van Gool, L. (2020b). Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV*, pages 527–543. Springer. 2, 10, 12, 13, 24, 28, 29, 30, 37, 98

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *NeurIPS*. 58, 59

Voges, R. and Wagner, B. (2018). Timestamp offset calibration for an imu-camera system under interval uncertainty. In *IROS*. IEEE. 38

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*. 1, 9, 71

Wallingford, M., Li, H., Achille, A., Ravichandran, A., Fowlkes, C., Bhotika, R., and Soatto, S. (2022). Task adaptive parameter sharing for multi-task learning. In *CVPR*, pages 7561–7570. 13

Wang, F., Wang, X., and Li, T. (2009). Semi-supervised multi-task learning with task regularizations. In *ICDM*, pages 562–568. IEEE. 40

Wang, Q., Dai, D., Hoyer, L., Van Gool, L., and Fink, O. (2021). Domain adaptive semantic segmentation with self-supervised depth estimation. In *ICCV*, pages 8515–8525. 41

Wang, X., Fouhey, D., and Gupta, A. (2015). Designing deep networks for surface normal estimation. In *CVPR*, pages 539–547. 1, 9

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34. 59, 62

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A. (2018). The microsoft 2017 conversational speech recognition system. In *ICASSP*, pages 5934–5938. IEEE. 95

Xu, D., Ouyang, W., Wang, X., and Sebe, N. (2018a). Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, pages 675–684. 10, 13, 24, 28, 29, 30, 37

Xu, R., Chen, Z., Zuo, W., Yan, J., and Lin, L. (2018b). Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, pages 3964–3973. 15

Xu, Y., Li, X., Yuan, H., Yang, Y., Zhang, J., Tong, Y., Zhang, L., and Tao, D. (2022). Multi-task learning with multi-query transformer for dense prediction. *arXiv preprint arXiv:2205.14354*. 13

Yang, X., Ye, J., and Wang, X. (2022). Factorizing knowledge in neural networks. In *ECCV*. 91

Ye, H. and Xu, D. (2022). Inverted pyramid multi-task transformer for dense scene understanding. *arXiv preprint arXiv:2203.07997*. 13, 91

Yeo, T., Kar, O. F., and Zamir, A. (2021). Robustness via cross-domain ensembles. In *ICCV*, pages 12189–12199. 91

You, K., Liu, Y., Wang, J., and Long, M. (2021). Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, pages 12133–12143. PMLR. 93

Yu, L., Yazici, V. O., Liu, X., Weijer, J. v. d., Cheng, Y., and Ramisa, A. (2019). Learning metrics from teachers: Compact networks for image embedding. In *CVPR*, pages 2907–2916. 79, 120

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020). Gradient surgery for multi-task learning. *NeurIPS*. 2, 11, 12, 13, 14, 18, 24, 26, 27, 38, 61

Zamir, A. R., Sax, A., Cheerla, N., Suri, R., Cao, Z., Malik, J., and Guibas, L. J. (2020). Robust learning through cross-task consistency. In *CVPR*, pages 11197–11206. 39, 41, 43, 44, 46, 48, 52, 104, 111, 112

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722. 41, 90, 91, 93

Zamir, A. R., Wekel, T., Agrawal, P., Wei, C., Malik, J., and Savarese, S. (2016). Generic 3d representation via pose estimation and matching. In *ECCV*, pages 535–553. Springer. 41

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*. 72, 119

Zhang, H., Mao, F., Xue, M., Fang, G., Feng, Z., Song, J., and Song, M. (2022). Knowledge amalgamation for object detection with transformers. *arXiv preprint arXiv:2203.03187*. 94

Zhang, X., Tang, C., An, Y., and Yin, K. (2021). Wifi-based multi-task sensing. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 169–189. Springer. 94

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*. 40

Zhang, Y. and Yeung, D.-Y. (2009). Semi-supervised multi-task regression. In *ECML PKDD*, pages 617–631. Springer. 40

Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., and Yang, J. (2018). Joint task-recursive learning for semantic segmentation and depth estimation. In *ECCV*, pages 235–251. 10, 13, 37

Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., and Yang, J. (2019). Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *CVPR*, pages 4106–4115. 13, 37

Zhong, Y., Arandjelović, R., and Zisserman, A. (2016). Faces in places: Compound query retrieval. In *BMVC*. 9

Zhou, L., Cui, Z., Xu, C., Zhang, Z., Wang, C., Zhang, T., and Yang, J. (2020). Pattern-structure diffusion for multi-task learning. In *CVPR*, pages 4514–4523. 13, 37

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858. 41

Zhou, X., Koltun, V., and Krähenbühl, P. (2022). Simple multi-dataset detection. In *CVPR*, pages 7571–7580. 94