MULTI-OBJECTIVE LEARNING FOR MULTI-MODAL NATURAL LANGUAGE GENERATION

Ramakanth Pasunuru

A thesis submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2021

ABSTRACT

Ramakanth Pasunuru: Multi-Objective Learning for Multi-Modal Natural Language Generation
(Under the direction of Mohit Bansal)

One of the important goals of Artificial Intelligence (AI) is to mimic the ability of humans to leverage the knowledge or skill from previously learned tasks to quickly learn a new task. For example, humans can reapply the learned skill of balancing the bicycle for learning to ride a motorbike. In a similar context, the field of Natural Language Processing (NLP) has several tasks including machine translation, textual summarization, image/video captioning, sentiment analysis, dialog systems, natural language inference, question answering, etc. While these different NLP tasks are often trained separately, leveraging the knowledge or skill from related tasks via joint training or training one task after another task in a sequential fashion, can have potential advantages. To this end, this dissertation explores various NLP tasks (especially multi-modal text generation and pair-wise classification tasks covering both natural language generation (NLG) and natural language understanding (NLU)) leveraging information from the related auxiliary tasks in an effective way via novel multi-objective learning strategies.

These proposed novel learning strategies can be broadly classified into three paradigms: multi-task learning, multi-reward reinforcement learning, and continual learning. In multi-task learning, we mainly focus on intuitively finding what related auxiliary tasks can benefit the multi-modal video caption generation task and textual summarization task. We explore effective ways of sharing the parameters across these related tasks via joint training. In multi-reward reinforcement learning, we teach various skills to multi-modal text generation models in the form of rewards. For example, we try to teach the entailment skill to the video captioning model with entailment rewards. Further, we propose novel and effective ways of inducing multiple skills by ‘dynamically’ choosing the auxiliary tasks (in MTL) or rewards (in RL) during the training in an automatic way using multi-armed bandits based approaches. Finally, in continual learning, we explore sharing of information across various tasks in a sequential way, where the model continually evolves during the sequential training without losing the performance on previously learned tasks. This kind of sharing allows the later tasks to benefit from previously trained tasks and vice-versa in some cases. For this, we propose a novel method that continually changes the model architecture to accommodate new tasks while retaining performance on old tasks. We empirically evaluate our method on three natural language inference tasks.

To my parents.

"The only person who is educated is the one who has learned how to learn and change."

– Carl Rogers

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof. Mohit Bansal for all the support and encouragement over these five years. I can't imagine myself in this position without his guidance. This dissertation is the result of his constant support and freedom to explore various challenging problems. I am grateful for all the feedback and the skills that I learned from him which I believe will carry for the rest of my life.

I had many fruitful collaborations with Han Guo and learned a lot working with him and also got the opportunity to mentor him. I would like to thank him for all the fun deep brainstorming sessions and maintaining collaborations for a significant part of my PhD time.

Throughout these five years, I have got various opportunities to work with a lot of amazing people through internships and collaborations. I want to thank all of them for sharing their valuable knowledge and experience with me. Especially, I would like to mention about few of them. I thank David Rosenberg for deepening my interest in better understanding the theoretical aspects of many machine learning concepts. I thank Prof. Ido Dagan (with his students Ori Shapira and Ori Ernst) for providing an opportunity to work on many interesting projects and thereby providing a valuable collaboration experience. I would like to thank Asli Celikyilmaz and Michel Galley for all the wonderful experiences I had during my internship at Microsoft Research. I deeply enjoyed the brainstorming sessions on awesome projects with them and significantly changed my thinking process. I also thank Markus Dreyer and Mengwen Liu for giving an exposure on research application to practical scenarios during my internship at Amazon. I would also like to thank Ves Stoyanov for helping me to better understand the importance of deeper thinking on a research problem.

I would like to thank my committee members (Mohit Bansal, Asli Celikyilmaz, Marc Niethammer, Shahriar Nirjon, and Shahriar Nirjon) for their constant support and for providing valuable feedback in completing this dissertation. I also thank the UNC-NLP lab members for always there to help me out and also thank them for having a fun and interactive PhD experience.

I would like to thank my parents: Subhash Reddy and Sabitha, my two support pillars. They have always encouraged me to pursue whatever I want to do. Without their support, I wouldn't have pursued PhD. I would also like to thank my other family members for always believing in me (my brother Srinath, pinni, babai, and amamma).

PhD is a long journey and there are ups and downs, and without my partner Divya and my roommate Praneeth, it would have been harder to navigate. I would like to thank them for always being there to support me and share my ups and downs. Also, I am fortunate to have many friends over these five years who have at some scale contributed to a better life as a Phd Student. I would like to thank those friends, specifically, Srihari, Sandeep, Adithya, Sowmya, Rakesh, Swathi, Pranav, Divya, Aatish, Tanmay, Bhavana, Lakshita, Sheetal, Vindhya, Anirudh, Arkopal, Anwesha, Deepthi, Ravi Teja, Hao, Lisa, and Spandana.

I am grateful to be supported by a Microsoft PhD Fellowship. I am also thankful for the support from other grants: NSF, DARPA, ARO, ONR, IBM, Nvidia, Google, Bloomberg, Salesforce, Facebook, Baidu, and Amazon.

TABLE OF CONTENTS

LIST OF TABLES ... xiv

LIST OF FIGURES ... xvi

LIST OF ABBREVIATIONS ... xviii

CHAPTER 1: INTRODUCTION ... 1

1.1 Thesis Statement ... 5

1.2 Overview of Chapters ... 5

CHAPTER 2: BACKGROUND AND RELATED WORK ... 7

2.1 Multi-Task Learning ... 7

2.1.1 Overview ... 7

2.1.2 Why does MTL work? ... 8

2.1.3 Earlier works in MTL ... 8

2.1.4 MTL for Deep Neural Networks ... 9

2.2 Reinforcement Learning ... 11

2.2.1 Overview ... 11

2.2.2 Details on REINFORCE Algorithm ... 12

2.3 Continual Learning ... 14

2.4 Multi-Modal Text Generation and Text Classification Tasks ... 15

2.4.1 Video Captioning ... 15

2.4.2 Abstractive Summarization ... 17

2.4.3 Sentence Simplification ... 18

2.4.4 Question Generation ... 18

2.4.5 Data-to-Text Generation 19

2.4.6 Text Classification Tasks 19

CHAPTER 3: MULTI-TASK LEARNING FOR VIDEO CAPTIONING 20

3.1 Introduction 20

3.2 Models 22

3.2.1 Baseline Sequence-to-Sequence Model 22

3.2.2 Attention-based Model 23

3.2.3 Unsupervised Video Prediction 25

3.2.4 Entailment Generation 25

3.2.5 Multi-Task Learning 26

3.3 Experimental Setup 27

3.3.1 Datasets 27

3.3.2 Evaluation (Automatic and Human) 29

3.3.3 Training Details 29

3.4 Experimental Results 30

3.4.1 Video Captioning on YouTube2Text 30

3.4.2 Video Captioning on MSR-VTT, M-VAD 32

3.4.3 Entailment Generation Results 33

3.4.4 Human Evaluation 34

3.4.5 Insights 35

3.5 Conclusion 36

CHAPTER 4: MULTI-TASK LEARNING FOR TEXTUAL SUMMARIZATION 38

4.1 Introduction 38

4.2 Models 39

4.2.1 Baseline Pointer+Coverage Model 40

4.2.2 Two Auxiliary Tasks 41

4.3 Multi-Task Learning 42
4.3.1 Layer-Specific Sharing Mechanism 43
4.3.2 Soft vs. Hard Parameter Sharing 44
4.3.3 Fast Multi-Task Training 45
4.4 Experimental Setup 45
4.5 Experimental Results 47
4.5.1 Summarization (Primary Task) Results 47
4.5.2 Human Evaluation 49
4.5.3 Generalizability Results (DUC-2002) 50
4.5.4 Auxiliary Task Results 50
4.6 Ablations and Insights 51
4.7 Conclusion 54
CHAPTER 5: REINFORCEMENT LEARNING FOR VIDEO CAPTIONING 56
5.1 Introduction 56
5.2 Models 57
5.3 Reward Functions 59
5.4 Experimental Setup 61
5.5 Experimental Results 62
5.5.1 Primary Results 62
5.5.2 Other Datasets 64
5.5.3 Other Metrics as Reward 64
5.5.4 Insights 65
5.6 Conclusion 66
CHAPTER 6: REINFORCEMENT LEARNING FOR TEXTUAL SUMMARIZATION . . . 67
6.1 Introduction 67
6.2 Models 68

6.2.1 Baseline Sequence-to-Sequence Model ... 68
6.2.2 Policy Gradient Reinforce ... 68
6.2.3 Multi-Reward Optimization ... 69
6.3 Rewards ... 70
6.4 Experimental Setup ... 72
6.4.1 Datasets and Training Details ... 72
6.4.2 Evaluation Metrics ... 73
6.5 Experimental Results ... 73
6.6 Output Analysis ... 76
6.7 Conclusion ... 77
CHAPTER 7: AUTOMATION METHODS FOR MTL AND RL WITH BANDITS ... 78
7.1 Multi-Armed Bandits (MAB) ... 78
7.2 Dynamic Multi-Task Learning with Bandits ... 79
7.2.1 Pointer-Copy Baseline Sentence Simplification Model ... 81
7.2.2 Auxiliary Tasks ... 82
7.2.3 Multi-Task Learning ... 83
7.2.4 Dynamic Mixing Ratio Learning ... 85
7.2.5 Evaluation Setup ... 87
7.2.6 Experimental Results ... 89
7.2.7 Ablations and Insights ... 92
7.2.8 AUTOSEM: Automatic Task Selection and Mixing in MTL ... 96
7.3 Optimizing Multiple RL Rewards with Bandits ... 97
7.3.1 Multi-Reward Optimization ... 99
7.3.2 Multi-Armed Bandit for Multi-Reward Optimization ... 101
7.3.3 Bandit Reward Settings ... 103
7.3.4 Single Bandit with Multi-Reward ... 103

7.3.5 Hierarchical Bandit with Multi-Reward 104
7.4 Tasks and Setup for DORB Framework 105
7.4.1 Question Generation 106
7.4.2 Data-to-Text Generation 107
7.4.3 Training Details 108
7.5 Experimental Results for DORB Framework 108
7.5.1 Results on Question Generation 109
7.5.2 Results on Data-to-Text Generation 110
7.5.3 Human Evaluation 111
7.5.4 Interpretable Bandit Analysis 113
7.6 Conclusion 114
CHAPTER 8: CAS: CONTINUAL ARCHITECTURE SEARCH 115
8.1 Introduction 115
8.2 Related Work 116
8.3 Architecture Search for Text Classification and Generation 117
8.3.1 ENAS Algorithm 117
8.3.2 ENAS for Bi-Text Classification 118
8.3.3 ENAS for Conditioned Generation 119
8.4 Continual Architecture Search (CAS) 120
8.5 Experimental Setup 123
8.5.1 Text Classification Datasets 123
8.5.2 Video Captioning Datasets 123
8.5.3 Evaluation 124
8.5.4 Training Details 124
8.6 Experimental Results and Insights 125
8.6.1 Continual Learning on GLUE Tasks 125

8.6.2 Continual Learning on Video Captioning ... 128
8.6.3 Insights ... 128
8.7 Conclusion ... 129
CHAPTER 9: SUMMARY, LIMITATIONS, AND FUTURE WORK ... 130
9.1 Summary of Contributions ... 130
9.2 Limitations and Future Work ... 130
APPENDIX A: EXPERIMENTAL SETUP AND TRAINING DETAILS ... 133
A.1 Full Training Details for MTL Video Captioning. ... 133
A.1.1 Video Captioning on YouTube2Text ... 133
A.1.2 Video Captioning on MSR-VTT ... 134
A.1.3 Video Captioning on M-VAD ... 134
A.1.4 Entailment Generation ... 135
A.2 Datasets and Training Details for MTL Summarization ... 135
A.2.1 Dataset Details ... 135
A.2.2 Training Details ... 136
A.3 Experimental Setup for RL Video Captioning ... 138
A.3.1 Datasets ... 138
A.3.2 Training Details ... 138
A.4 Additional Saliency Reward Details for RL Summarization ... 139
A.5 Training Details for Dynamic MTL ... 140
A.6 Training Details for DORB models ... 141
A.7 Training Details for CAS ... 142
REFERENCES ... 144

LIST OF TABLES

Table 3.1 – Primary video captioning results on Youtube2Text (MSVD). 31

Table 3.2 – Results on MSR-VTT dataset. 33

Table 3.3 – Results on M-VAD dataset. 33

Table 3.4 – Entailment generation results with the four metrics. 34

Table 3.5 – Human evaluation on YouTube2Text video captioning. 35

Table 3.6 – Human evaluation on entailment generation. 35

Table 3.7 – Examples of our multi-task model's generated entailment hypotheses. 36

Table 4.1 – CNN/DailyMail summarization results. 46

Table 4.2 – Summarization results on Gigaword. 47

Table 4.3 – Human Evaluation for 3-way multi-task model on CNN/DM. 48

Table 4.4 – Human Evaluation for 3-way multi-task model on Gigaword. 48

Table 4.5 – ROUGE F1 scores on DUC-2002. 49

Table 4.6 – Performance of our pointer-based entailment generation models. 50

Table 4.7 – Performance of our pointer-based question generation model. 50

Table 4.8 – Ablation studies on our final multi-task model. 51

Table 4.9 – Entailment classification results of our baseline vs. EG-multi-task model. 51

Table 4.10 –Saliency classification results of our baseline vs. QG-multi-task model. 52

Table 4.11 –Abstractiveness: novel n-gram percent. 53

Table 5.1 – Examples of captions sampled during policy gradient. 60

Table 5.2 – Our primary video captioning results on MSR-VTT with RL setup. 62

Table 5.3 – Human eval: Baseline-XE vs CIDEr-RL. 63

Table 5.4 – Human eval: CIDEr-RL vs CIDEnt-RL. 64

Table 5.5 – Results on YouTube2Text (MSVD) dataset with RL setup. 64

Table 6.1 – Results on CNN/Daily Mail with RL setup 74

Table 6.2 – Performance of cross-entropy and RL models on test-only DUC-2002. 75

Table 6.3 – Human evaluation comparing cross-entropy and RL models. 75

Table 6.4 – Abstractiveness: novel n-gram percentage. 76

Table 7.1 – MTL results on Newsela. 89

Table 7.2 – MTL results on WikiSmall/Large. 89

Table 7.3 – Results on dynamic vs. static mixing ratio. 90

Table 7.4 – Human evaluation results on Newsela. 91

Table 7.5 – Multi-task layer ablation results on Newsela. 93

Table 7.6 – Analysis of entailment and paraphrase classification results on Newsela. 94

Table 7.7 – Analysis: SARI's sub-operation scores on Newsela dataset. 94

Table 7.8 – Performance of various models on question generation task. 109

Table 7.9 – Performance of various models on WebNLG data-to-text task. 110

Table 7.10 –Human evaluation results on QG and WebNLG tasks. 111

Table 7.11 –Human evaluation results on WebNLG 'unseen' test set. 112

Table 8.1 – Test results on GLUE tasks for various models. 126

Table 8.2 – Ablation results on CAS conditions. 127

Table 8.3 – Video captioning results with Baseline, ENAS, and CAS models. 127

LIST OF FIGURES

Figure 3.1 – A video captioning example from the YouTube2Text dataset. 21

Figure 3.2 – Baseline sequence-to-sequence model for video captioning. 22

Figure 3.3 – Attention-based sequence-to-sequence baseline model for video captioning. 23

Figure 3.4 – Our many-to-many multi-task learning model. 24

Figure 3.5 – Examples of generated video captions on the YouTube2Text dataset. 36

Figure 4.1 – Overview of our multi-task model for abstractive summarization. 43

Figure 4.2 – Example summary from our 3-way MTL model. 55

Figure 5.1 – Example video caption generated by our CIDEnt-reward model. 57

Figure 5.2 – Reinforced video captioning using entailment-corrected CIDEr reward. 59

Figure 5.3 – Examples showing CIDEnt-RL model generating better entailed captions. 65

Figure 6.1 – Overview of sequence generator with RL training. 69

Figure 6.2 – Overview of our saliency predictor model. 71

Figure 7.1 – Overview of our 3-way multi-task model for sentence simplification. 83

Figure 7.2 – Overview of our MAB algorithm for dynamic mixing ratio learning. 83

Figure 7.3 – Task selection probability predicted by bandit controller. 95

Figure 7.4 – Overview of our DORB framework. 99

Figure 7.5 – Overview of our hierarchical multi-armed bandit approach. 105

Figure 7.6 – Plots showing the probability distribution of each child bandit of the HM-Bandit model on the QG task. 112

Figure 7.7 – Plot showing the probability distribution of each arm of the SM-Bandit. 113

Figure 8.1 – Architecture search models for bi-text classification and video captioning. 119

Figure 8.2 – Our continual architecture search (CAS) approach. 121

Figure 8.3 – Learned cell structures for GLUE tasks. 129

LIST OF ABBREVIATIONS

CAS	Continual Architecture Search
DORB	Dynamically Optimizing Multiple Rewards with Bandits
ENAS	Efficient Neural Architecture Search
GLUE	General Language Understanding Evaluation
LSTM	Long Short-Term Memory
MAB	Multi-Armed Bandit
MTL	Multi-Task Learning
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
RL	Reinforcement Learning
RNN	Recurrent Neural Network

CHAPTER 1: INTRODUCTION

One of the important goals of Artificial Intelligence (AI) is to mimic the ability of humans to leverage the knowledge or skill from previously learned tasks to quickly learn a new task. For example, humans can reapply the learned skill of balancing the bicycle for learning to ride a motorbike. Similarly, the field of Natural Language Processing (NLP) has several tasks including machine translation, textual summarization, image/video captioning, sentiment analysis, dialog systems, natural language inference, question answering, etc. While these different NLP tasks are often trained separately, leveraging the knowledge from related tasks via joint training (Caruana, 1998; Luong et al., 2016) or training one task after another task in a sequential fashion (Devlin et al., 2019; Parisi et al., 2019), can have potential advantages. Take for example abstractive summarization, the task of compressing and rewriting a long document into a short summary. One of the important aspects of a good summary is to be entailed by the input document, i.e., the summary should not contain any information that is contradictory or unrelated to the original document. Hence, this task could benefit by using the knowledge from entailment-related natural language inference task (Dagan et al., 2005). Similarly, sequential/continual learning of tasks is helpful in scenarios like enabling a robot to keep on learning new tasks via natural language instructions (She et al., 2014), adapting a conversational agent to adapt to new conversation topics (Lee, 2017), and improving a natural language inference system by adding new vocabulary and adapting it to various domains without retraining from scratch (Pasunuru and Bansal, 2019). To this end, this dissertation explores various NLP tasks (especially multi-modal text generation and pair-wise classification tasks covering both natural language generation (NLG) and natural language understanding (NLU)) leveraging information from the related auxiliary tasks in an effective way via novel multi-objective learning strategies in the context of multi-task learning, multi-reward reinforcement learning, and continual learning.

Towards exploring the shared knowledge across various tasks, we first focused on video captioning, the task of automatically generating natural language description of the content of a video clip. It has various applications such as assistance to a visually impaired person and improving the quality of online video search or retrieval. Previous work in video captioning have used sequence-to-sequence modeling, attention mechanism, hierarchical two-level RNNs for efficient video encoding (Venugopalan et al., 2015a; Pan et al., 2016b; Yao et al., 2015; Pan et al., 2016a; Yu et al., 2016). Despite these recent improvements, video captioning models still suffer from the lack of sufficient temporal and logical supervision to be able to correctly capture the action sequence and story-dynamic language in videos, especially in the case of short clips. Hence, they would benefit from incorporating such complementary directed knowledge, both visual and textual. We address this by jointly training the task of video captioning with two related directed-generation tasks: a temporally-directed unsupervised video prediction task and a logically-directed language entailment generation task (Pasunuru and Bansal, 2017a). We model this via many-to-many multi-task learning based sequence-to-sequence models (Luong et al., 2016) that allow the sharing of parameters among the encoders and decoders across the three different tasks, with additional shareable attention mechanisms. Additionally, we built the first state-of-the-art video captioning demo system (Guo et al., 2017), with the additional novel aspects of generating multi-sentence, paragraph-based captions, and allowing cooperative user feedback.

Another important NLP task, abstractive summarization, could also benefit from strong natural language inference skills, since a correct summary is logically entailed by the input document, i.e., it should not contain any contradictory or unrelated information. Despite the promising improvements with neural sequence-to-sequence models via machine translation inspired encoder-aligner-decoder approaches, further enhanced via convolutional encoders, pointer-copy mechanisms, and hierarchical attention (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), abstractive summaries suffer from generating contradictory or unrelated information. Towards this end, we incorporate natural language inference skills into an abstractive summarization model via multi-task learning, where we share its decoder parameters with those of an entailment generation model (Pasunuru et al., 2017). Further, an accurate abstractive summary of a document should also contain all the salient information from the input document. We improve this important aspect via multi-task learning with the task of question generation which teaches the summarization model the right questions to ask, which in turn is directly related to salient information in the input document (Pasunuru and Bansal, 2018). Jointly using both entailment generation and question generation tasks to improve abstractive summarization, we present novel multi-task learning architectures based on multi-layered encoder and decoder models, where we empirically show that it is substantially better to share the higher-level semantic layers between the three aforementioned tasks, while keeping the lower-level (lexico-syntactic) layers unshared. We also explore different ways to optimize the shared parameters and show that 'soft' parameter sharing achieves higher performance than hard sharing (Pasunuru and Bansal, 2018).

Exploring other ways of knowledge sharing among various tasks, we use reinforcement learning (RL), especially policy gradient-based RL (Williams, 1992), which enables to directly optimize the sentence-level evaluation metrics as opposed to traditional cross-entropy loss. One can leverage this by defining such metrics based on the properties of the related tasks, thus sharing their information. To this end, to improve video captioning, we introduce an entailment-enhanced reward function that guides the traditional phrase matching metric whenever the entailment score is low for the generated sample w.r.t. the ground-truth caption as premise (Pasunuru and Bansal, 2017b). Optimization of this reward function ensures that the generated caption is logically-entailed w.r.t. the ground-truth caption and does not contain spurious contradictory or extra (spurious, unrelated) words, which is a big problem in the current state-of-the-art systems.

Similarly, we introduce two novel reward functions for abstractive summarization, one for weighting more importance to salient words present in the generated summary, and another reward function which gives high (length-normalized) scores to logically-entailed summaries using an entailment classifier (Pasunuru and Bansal, 2018). We show that these reward functions help the summarization model get better, and we further show superior performance improvement when these rewards are combined with our novel and effective multi-reward approach of optimizing multiple rewards simultaneously in alternate mini-batches (Pasunuru and Bansal, 2018).

While the above approaches in the context of MTL and multi-reward RL are appealing, they need manual tuning of in what static proportions the auxiliary tasks (in MTL) or rewards (in RL) have to be mixed during the training. Further, this tuning can be computationally expensive with the increase in the number of tasks (or rewards) at hand. Addressing this issue, we introduce 'dynamic' mixing of auxiliary tasks or rewards in an automatic way using multi-armed bandits. Next, we briefly describe those approaches for MTL and multi-reward RL.

In the above-mentioned MTL works, the weights between tasks during MTL training are static and are considered hyperparameters. To avoid this, we move towards 'self-learned MTL' where we propose dynamic multi-armed bandit based training approach that automatically learns how to effectively switch across the primary and auxiliary tasks during multi-task training (Guo et al., 2018). In this setup, we improve the entailment and paraphrasing capabilities of a sentence simplification model via MTL with related auxiliary tasks of entailment and paraphrase generation.

In the context of multi-reward RL, the previously proposed multi-reward optimization approaches are not scalable to optimize multiple reward functions and one needs to manually decide the importance and scaling weights of these metric rewards. Further, it is important to consider using a dynamic combination and curriculum of metric rewards that flexibly changes over time. Considering the above aspects, in our work (Pasunuru et al., 2020), we automate the optimization of multiple metric rewards simultaneously via a multi-armed bandit approach (DORB), where at each round, the bandit chooses which metric reward to optimize next, based on expected arm gains. We use the Exp3 algorithm for bandits and formulate two approaches for bandit rewards: (1) Single Multi-reward Bandit (SM-Bandit); (2) Hierarchical Multi-reward Bandit (HM-Bandit). We empirically show the effectiveness of our approaches via various automatic metrics and human evaluation on two important NLG tasks: question generation and data-to-text generation, including on an unseen-test transfer setup.

We also explore sharing of information across various tasks in a sequential way via continual learning, where the model parameters continually evolve during the sequential training of related tasks without losing the performance on previously learned tasks. This kind of sharing allows the later tasks to benefit from previously trained tasks and vice-versa in some cases. In our work (Pasunuru and Bansal, 2019), we propose a novel approach called continual architecture search (CAS) by leveraging neural architecture search (NAS) (Pham et al., 2018). NAS is the process of automatically learning the neural model or cell structure that best suits the given task, and we leverage it for continual learning of various video captioning tasks and natural language inference tasks via block-sparsity and orthogonality constraints without losing performance.

With the above-proposed approaches of multi-objective learning across various NLP tasks (with multi-modal input), we can efficiently transfer/share knowledge and effectively learn complex behavior among multiple tasks.

1.1 Thesis Statement

Through various multi-objective learning strategies, it is possible to efficiently transfer/share knowledge and effectively learn complex behavior among multiple tasks in the domains of multi-modal text generation and pair-wise text classification.

1.2 Overview of Chapters

The remainder of this dissertation is organized into eight chapters. Chapter 2 discusses the related work around the three multi-objective paradigms and various NLP tasks used in our experiments. Chapter 3 presents our work on multi-task learning with video captioning. Chapter 4 presents a novel multi-task learning approach applied for textual summarization. Chapter 5 and Chapter 6 explore the idea of leveraging the skills of related tasks in the form of rewards for improving the tasks of video captioning and textual summarization, respectively. Chapter 7 presents our novel and effective ways of inducing multiple skills by ‘dynamically’ choosing the auxiliary tasks (in MTL) or rewards (in RL) during the training in an automatic way using multi-armed bandits based approaches. Next, Chapter 8 discusses our novel continual learning approach where the model can dynamically change its architecture to learn new tasks. Finally, Chapter 9 summarizes the contributions herein and discusses the potential opportunities for future work.

CHAPTER 2: BACKGROUND AND RELATED WORK

In this chapter, we discuss the background and related work of multi-task learning, reinforcement learning, and continual learning. We also discuss the related work of various multi-modal text generation tasks and text classification tasks used in our work.

2.1 Multi-Task Learning

2.1.1 Overview

Multi-task learning (MTL) is a useful learning paradigm to improve the supervision and the generalization performance of a task by jointly training it with related tasks (Caruana, 1998; Argyriou et al., 2007; Kumar and Daumé III, 2012). One can also motivate multi-task learning from a biological perspective on how humans learn new skills by often applying the knowledge that was acquired by learning related tasks. For example, humans can apply the skill of balancing learned through biking to learn motorcycle.

Multi-task learning has wide applications to natural language processing (Collobert and Weston, 2008), computer vision (Girshick, 2015), speech recognition (Deng et al., 2013), etc. The most common way of performing MTL is via hard or soft parameter sharing of hidden layers. In the hard parameter sharing, hidden layers between all tasks are shared, while keeping the task-specific layers separate. In soft parameter sharing, each task has its own parameters while the distance between the soft sharing parameters is regularized to encourage these parameters to be similar or close in their representation space (Duong et al., 2015; Yang and Hospedales, 2017).

2.1.2 Why does MTL work?

Below, we discuss some of the reasons why MTL works and how it achieves better generalizability.

- **Data Augmentation:** MTL provides more signals to learn a task in the form of additional data coming from related tasks. Essentially, this means MTL provides data augmentation (might be noisy) to improve a tasks' performance.

- **Representation Bias:** MTL not only learns representations that better fit the given task but also constraints the model to prefer representations that other tasks also prefer (Baxter, 2000). This will help the model to generalize better to new tasks.

- **Regularization:** MTL also acts as a good regularizer by introducing inductive bias. Different tasks have different noise patterns, and MTL enables simultaneous training of multiple tasks thereby averaging out the noise and avoiding the over-fitting problem.

2.1.3 Earlier works in MTL

Two of the important ideas that were most pursued in earlier MTL works are block sparse regularization (Yuan and Lin, 2006; Argyriou et al., 2007; Zhang et al., 2008) and learning task relationships (Evgeniou et al., 2005; Chen et al., 2010; Thrun and O'Sullivan, 1996). Next, we will discuss some of the works around these two ideas.

For better understanding, let us consider a linear model. Let us assume that there are T tasks with the model parameters for the i^{th} task are represented by θ_i, where θ is an N-dimension vector. Let's also assume that these parameters are sparse in nature (many previous works considered this assumption (Yuan and Lin, 2006)). For MTL, let us assume that few parameters $\theta_{i,j}$ of i^{th} task with $j \subset [1, N]$ are shared across all tasks. To achieve both sparsity and also share the parameters across tasks, we need to have non-zero values at the shared positions. This means that if we put all the parameters of the T tasks in the form of a matrix A of size $N \times T$, then only a few rows are non-zeros, which is nothing but they have to be block sparse. We can achieve block-sparsity by computing ℓ_q norm for each row of A and then applying ℓ_1 norm. This is otherwise known as ℓ_1/ℓ_q mixed norms constraint. Zhang et al. (2008) used ℓ_1/ℓ_∞ mixed norm and Argyriou et al. (2007) used ℓ_1/ℓ_2 mixed norm to achieve block sparsity in the context of MTL. Note that if the features do not overlap much across the tasks, this mixed norm can perform worse than the element-wise regularization (Negahban and Wainwright, 2008). To address this issue, Jalali et al. (2010) decomposed the parameter matrix A into two matrices (B and S) such that $A = B + S$ with matrix B constrained with mixed norm and matrix S constrained with element-wise sparsity.

As mentioned above, features may no overlap much across tasks. In such scenarios, we could leverage prior knowledge on which tasks are related and group them together while doing MTL. (Evgeniou et al., 2005) proposed a clustering constraint by penalizing both the norms of the parameter matrix A. Further, it is extended to tasks that can be grouped into tree or graph structures (Chen et al., 2010; Thrun and O'Sullivan, 1996). Other works related to learning task relationships focused on Bayesian methods. Lawrence and Platt (2004) proposed to use Gaussian Processes to infer shared parameters in MTL. Daumé III (2009) proposed a hierarchical Bayesian model to learn a latent task hierarchy.

2.1.4 MTL for Deep Neural Networks

Several recent works have adopted MTL in neural models (Luong et al., 2016; Misra et al., 2016; Hashimoto et al., 2017; Ruder et al., 2019; Kaiser et al., 2017). Most of the earlier MTL approaches assume homogeneous setting, i.e., all the tasks are associated with single output, however, more recent works considered heterogeneous setting, i.e., having unique outputs for each task. MTL has been applied to sequence-to-sequence models, sharing parameters across the tasks' encoders and decoders and showed significant improvements on machine translation using parsing and image captioning as related auxiliary tasks (Luong et al., 2016). In computer vision tasks, MTL sharing is done at convolutional layers, while learning task specific fully-connected layers (Zhang et al., 2014). These are further improved by placing matrix priors on the fully-connected layers thereby allowing them to learn the relationship between tasks (Long and Wang, 2015). However, these approaches are restricted to pre-defined structure for sharing. Addressing this issue, a bottom-up approach has been proposed that dynamically allows to widen the network during the training based on a criterion that promotes grouping of related tasks (Lu et al., 2017). Similarly, cross-stitch networks (Misra et al., 2016) has been proposed to learn a linear combination of the task-specific previous layers' output as input to the current layer so that each task can leverage the knowledge of the other task. A hierarchical layer sharing for MTL has been proposed to reflect linguistic hierarchy of multiple tasks (POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment) by jointly training them with a strategy for successively growing the depth of the network to solve increasingly complex tasks (Hashimoto et al., 2017). Our work (Pasunuru and Bansal, 2017a) used hard parameter sharing at encoder-decoder level between video captioning, unsupervised video prediction, and entailment generation to share knowledge across them. We extended this to train an MTL setup with soft parameter training at layer level between textual summarization, question generation, and entailment generation. Further, Sluice Networks (Ruder et al., 2019) provided a unified MTL framework combining hard parameter sharing, cross-stitch networks, and block-sparse regularization approaches. Another way of sharing information with MTL is via the knowledge distillation (Hinton et al., 2014), where task-specific models teach a multi-task model (Clark et al., 2019) to effectively learn the joint representation of all tasks. This approach removes the restriction of using parameter sharing across tasks for MTL learning.

The success of MTL models also depends on the choice of the auxiliary tasks for a given primary task. On the problem of identifying task relatedness, a formal framework for task relatedness was provided by (Ben-David and Schuller, 2003) and derived generalization error bounds for learning of multiple tasks. Bingel and Søgaard (2017) explored task relatedness via exhaustively experimenting with all possible two task tuples in a non-automated multi-task setup. Further, our work (Guo et al., 2019a) proposed a two-stage MTL approach, where the first stage automatically selects the most useful auxiliary tasks via a Beta-Bernoulli multi-armed bandit with Thompson Sampling, and the second stage learns the training mixing ratio of these selected auxiliary tasks via a Gaussian Process based Bayesian optimization framework.

2.2 Reinforcement Learning

2.2.1 Overview

Reinforcement Learning (RL) is a training mechanism in which an agent or a policy is allowed to interact with a given environment in order to maximize a reward. It has successful application to many research areas such as continuous control (White and Sofge, 1992), dialogue systems (Singh et al., 2002; Peng et al., 2017; Srivastava et al., 2019), and games (Tesauro, 1995; Narasimhan et al., 2015). Recently, a special case of RL, called policy gradients based reinforcement learning, has been widely applied to text generation problems in NLP through REINFORCE algorithm (Williams, 1992). REINFORCE has already been used for other applications such as computer vision (Mnih et al., 2014; Xu et al., 2015a) and speech recognition (Graves and Jaitly, 2014). Text generation models are traditionally optimized to predict the next work using the cross-entropy loss, which does not correlate well with the sentence-level metrics that the task is finally evaluated on (e.g., BLEU, CIDEr). Moreover, these models suffer from exposure bias (Ranzato et al., 2016), which occurs when a model is only exposed to the training data distribution, instead of its own predictions. The idea of using the model's own predictions at training time was first advocated by Hal Daumé et al. (2009), where they cast the structure prediction problems as a particular instance of reinforcement learning. Motivated by this, Ranzato et al. (2016) proposed a mixed sequence level training paradigm that uses model predictions during training and also use non-differential metrics as reward using REINFORCE. Following this work, a few successful examples of REINFORCE for text generation include image captioning (Rennie et al., 2017; Ren et al., 2017), abstractive summarization (Paulus et al., 2018; Chen and Bansal, 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018), machine translation (Wu et al., 2016; Gu et al., 2017), sentence simplification (Zhang and Lapata, 2017a), as well as video captioning (Pasunuru and Bansal, 2017b; Wang et al., 2018). Further, some works have also explored the problem of optimizing multiple rewards simultaneously in the context of machine translation (Neubig and Watanabe, 2016), video captioning (Pasunuru and Bansal, 2017b), summarization (Pasunuru and Bansal, 2018), and question generation and data-to-text generation (Pasunuru et al., 2020).

2.2.2 Details on REINFORCE Algorithm

Traditional text generation systems minimize the cross-entropy loss during training, but typically evaluated using phrase-matching metrics: BLEU, METEOR, CIDEr, and ROUGE-L. This discrepancy can be addressed by directly optimizing the non-differentiable metric scores using policy gradients p_θ, where θ represents the model parameters. In text generation systems, model acts as an agent and interacts with its environment (multi-modal input and text output). At each time step, the agent generates a word (action), and the generation of the end-of-sequence token results in a reward r to the agent. Our training objective is to minimize the negative expected reward function given by:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)] \tag{2.1}$$

where $w^s = \{w_1^s, w_2^s, ..., w_m^s\}$, and w_t^s is the word sampled from the model at time step t. Based on the REINFORCE algorithm (Williams, 1992), the gradients of the non-differentiable, reward-based loss function can be computed as follows:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)\nabla_\theta \log p_\theta(w^s)] \tag{2.2}$$

The above gradients can be approximated from a single sampled word sequence w^s from p_θ as follows:

$$\nabla_\theta L(\theta) \approx -r(w^s)\nabla_\theta \log p_\theta(w^s) \tag{2.3}$$

However, the above approximation has high variance because of estimating the gradient with a single sample. Adding a baseline estimator reduces this variance (Williams, 1992) without changing the expected gradient. Hence, Eqn: 2.3 can be rewritten as follows:

$$\nabla_\theta L(\theta) \approx -(r(w^s) - b_t)\nabla_\theta \log p_\theta(w^s) \tag{2.4}$$

where b_t is the baseline estimator, where b_t can be a function of θ or time step t, but not a function of w^s. Using the chain rule, loss function can be written as:

$$\nabla_\theta L(\theta) = \sum_{t=1}^{m} \frac{\partial L}{\partial s_t} \frac{\partial s_t}{\partial \theta} \tag{2.5}$$

where s_t is the input to the *softmax* layer, where $s_t = W^T h_t^d$. $\frac{\partial L}{\partial s_t}$ is given by Zaremba and Sutskever (2015) as follows:

$$\frac{\partial L}{\partial s_t} \approx (r(w^s) - b_t)(p_\theta(w_t|h_t^d) - 1_{w_t^s}) \tag{2.6}$$

The overall intuition behind this gradient formulation is: if the reward $r(w^s)$ for the sampled word sequence w^s is greater than the baseline estimator b_t, the gradient of the loss function becomes negative, then model encourages the sampled distribution by increasing their word probabilities, otherwise the model discourages the sampled distribution by decreasing their word probabilities. Previous works have taken different approaches to calculate the baseline estimator. For example, the baseline estimator in Ranzato et al. (2016) is a simple linear regressor with hidden state of the decoder h_t^d at time step t as the input. They stop the back propagation of gradients before the hidden states for the baseline bias estimator. Rennie et al. (2017) proposed a self-critical sequence training approach to calculate the baseline estimator, which is based on the reward obtained by the current model using the test time inference algorithm, i.e., choosing the arg-max word w_t^a of the final vocabulary distribution at each time step t of the decoder. In our work, we explore both these baseline estimator approaches for various text generation tasks.

We also employ the joint cross-entropy (L_{XE}) and reinforce loss (L_{RL}) so as to optimize the non-differentiable evaluation metric as reward while also maintaining the readability of the generated sentence (Wu et al., 2016; Paulus et al., 2018; Pasunuru and Bansal, 2017b), which is defined as $L_{\text{Mixed}} = \gamma L_{\text{RL}} + (1-\gamma)L_{\text{XE}}$, where γ is a tunable hyperparameter.

2.3 Continual Learning

Continual learning is a long-standing challenge for machine learning (French, 1999; Hassabis et al., 2017), which is defined as an adaptive system capable of learning from a continuous stream of information, where such information progressively increases over time and there is no predefined number of tasks to be learned. The major problem in continual learning is catastrophic forgetting. Various works have tried to address this catastrophic forgetting problem, and they can be broadly classified into (1) architectural, (2) functional, and (3) structural approaches.

Architectural approaches mainly focus on altering the architecture of the network to reduce the interference between the tasks without changing the objective function, thereby reducing the catastrophic forgetting. A simple form of architectural regularization is by freezing weights that are important for the previously-learned tasks (Razavian et al., 2014). One can also reduce the learning rates of the shared layers with previously-learned tasks while fine-tuning to avoid dramatic changes (Donahue et al., 2014; Yosinski et al., 2014). Further, a dramatic architectural change approach is to copy the parameters of the previous-learned task and augment with new features while learning a new task (Rusu et al., 2016), but this will increase the complexity with the number of tasks. Functional approaches to reduce catastrophic forgetting focus on penalizing the changes in the input-output function of the neural network. An approach based on knowledge distillation (Hinton et al., 2014) was proposed (Li and Hoiem, 2017) to keep the predictions of the previous task's network while training with the data from the new task. Another regularization approach is to minimize the final hidden activations while moving from one task to another task (Jung et al., 2016). Structural approaches involve penalties on the parameters training for the new task such that they are close to the parameters for the old task. Recently, elastic weight consolidation (EWC) approach (Kirkpatrick et al., 2017) has been proposed with a quadratic penalty on the difference between the parameters of new and old tasks.

Some other notable works in the very recent years are based on using intelligent synapses to accumulate task-related information over time (Zenke et al., 2017), using online variational inference with neural networks for continual learning (Nguyen et al., 2017), and dynamically expandable network that can expand its parameter capacity by splitting/duplicating based on incoming new data in sequence (Yoon et al., 2018). Similar to Yoon et al. (2018), Sarwar et al. (2019) proposed to grow the network to learn new tasks while sharing a base network among all tasks. Further, Xu and Zhu (2018) proposed to expand each layer in the network based on new incoming data/task using reinforcement learning. Our work (Pasunuru et al., 2020) leveraged neural architecture search to continually evolve the model parameters during the sequential training of several tasks, without losing performance on previously learned tasks (via block-sparsity and orthogonality constraints). Along similar lines, Li et al. (2019) used neural architecture search to find optimal networks for each task in the continual learning setup.

2.4 Multi-Modal Text Generation and Text Classification Tasks

2.4.1 Video Captioning

Early video captioning work (Guadarrama et al., 2013; Thomason et al., 2014; Huang et al., 2013) used a two-stage pipeline to first extract a subject, verb, and object (S,V,O) triple and then generate a sentence based on it. An end-to-end neural model was proposed to fed mean-pooled static frame-level visual features (from convolution neural networks pre-trained on image recognition) of the video as input to the language decoder (Venugopalan et al., 2015b). To harness the important frame sequence temporal ordering, Venugopalan et al. (2015a) proposed a sequence-to-sequence model with video encoder and language decoder RNNs. Further, linguistic improvements are explored to the caption decoder by fusing it with external language models (Venugopalan et al., 2016). Moreover, an attention or alignment mechanism was added between the encoder and the decoder to learn the temporal relations (matching) between the video frames and the caption words (Yao et al., 2015; Pan et al., 2016a). In contrast to static visual features, temporal video features are considered from a 3D-CNN model pre-trained on an action recognition task (Yao et al., 2015).

To explore long range temporal relations, Pan et al. (2016a) proposed a two-level hierarchical RNN encoder which limits the length of input information and allows temporal transitions between segments. A hierarchical RNN generating sentences at the first level and the second level capturing inter-sentence dependencies in a paragraph was proposed by (Yao et al., 2015). Pan et al. (2016b) proposed to simultaneously learn the RNN word probabilities and a visual-semantic joint embedding space that enforces the relationship between the semantics of the entire sentence and the visual content. Our works (Pasunuru and Bansal, 2017a,b) improved video captioning by inducing various skills via multi-task learning and reinforcement learning approaches.

Following are some other notable works in recent years. Wang et al. (2018) proposed a hierarchical reinforcement learning method to generate more detailed captions to a video. This method has a high-level manager module that learns to design sub-level goals, and a low-level module to complete these sub-goals. Chen et al. (2018) proposed a strategy to choose minimal video frames by maximizing the diversity of frames and minimizing the difference between generated and ground-truth captions. This is done using a reinforcement learning setup. Wang et al. (2019b) introduced a gated mechanism that fuses different types of representations in the video to create a global syntactic structure. Together, they aid for better video captioning generator. By controlling the syntactic structure, this method could control the diversity of the generated captions. For better modeling of object interactions in videos, Zhang et al. (2020b) proposed object relational graph-based encoder and also tried to leverage external language model knowledge into the caption model. Further, Perez-Martin et al. (2021) created a joint embedding space of visual, semantic, and syntactic representations for better caption generation.

2.4.2 Abstractive Summarization

Automatic text summarization has been progressively improving over the time, initially more focused on extractive and compressive models (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015; Kedzie et al., 2015), and moving more towards compressive and abstractive summarization based on graphs and concept maps (Giannakopoulos, 2009; Ganesan et al., 2010; Falke and Gurevych, 2017) and discourse trees (Gerani et al., 2014), syntactic parse trees (Cheung and Penn, 2014; Wang et al., 2013), and Abstract Meaning Representations (AMR) (Liu et al., 2015; Dohare and Karnick, 2017). Recent work has also adopted machine translation inspired neural seq2seq models for abstractive summarization with advances in hierarchical, distractive, saliency, and graph-attention modeling (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Chen et al., 2016; Tan et al., 2017). Paulus et al. (2018) and Henß et al. (2015) incorporated recent advances from reinforcement learning. Also, See et al. (2017) further improved results via pointer-copy mechanism and addressed the redundancy with coverage mechanism. In the direction of directly optimizing the evaluation metrics rather than the cross-entropy loss, Paulus et al. (2018) and Celikyilmaz et al. (2018) have explored the use of policy gradients based reinforcement learning for abstractive summarization. On top of See et al. (2017)'s model, our works (Pasunuru and Bansal, 2018; Pasunuru et al., 2018) employed multi-task learning and reinforcement learning approaches to induce external knowledge (saliency and entailment capabilities) into textual summarization models. In order to make the summary generation fast, Chen and Bansal (2018) proposed a model that first selects salient sentence and then generates an abstractive summary from these salient sentences. Further, they also use reinforcement learning approach. Similarly, (Gehrmann et al., 2018b) followed a bottom-up approach where a content selector first selects candidate phrases in the source document that could be part of the summary, and decode summary from these pre-selected phrases using copy mechanism. Recent works (Liu and Lapata, 2019; Lewis et al., 2019; Zhang et al., 2020a) further leveraged various large pre-trained models to significantly improve the performance on textual summarization.

2.4.3 Sentence Simplification

Sentence simplification is the task of improving the readability and understandability of an input text. This challenging task has been the subject of research interest because it can address automatic ways of improving reading aids for people with limited language skills, or language impairments such as dyslexia (Rello et al., 2013), autism (Evans et al., 2014), and aphasia (Carroll et al., 1999). It also has wide applications in NLP tasks as a preprocessing step, for example, to improve the performance of parsers (Chandrasekar et al., 1996), summarizers (Klebanov et al., 2004), and semantic role labelers (Vickrey and Koller, 2008; Woodsend and Lapata, 2014). Previous approaches to sentence simplification systems range from hand-designed rules (Siddharthan, 2006), to syntactic and lexical simplification via synonyms and paraphrases (Siddharthan, 2014; Kaji et al., 2002; Horn et al., 2014; Glavaš and Štajner, 2015), as well as treating simplification as a monolingual MT task, where operations are learned from examples of complex-simple sentence pairs (Specia, 2010; Koehn et al., 2007; Coster and Kauchak, 2011; Zhu et al., 2010; Wubben et al., 2012; Narayan and Gardent, 2014). Recently, Xu et al. (2016b) trained a syntax-based MT model using the newly proposed SARI as a simplification-specific objective. Further, Zhang and Lapata (2017b) used reinforcement learning in a sequence-to-sequence approach to directly optimize simplification metrics. In this dissertation, we first introduce the pointer-copy mechanism (See et al., 2017) as a novel application to sentence simplification, and then use multi-task learning to bring in auxiliary entailment and paraphrasing skills.

2.4.4 Question Generation

The goal of the question generation (QG) task is to generate a natural question that can be answered by the given answer span in a context. Recent works have applied seq2seq neural models for QG, e.g., generating the question given answer sentence (Du et al., 2017; Zhou et al., 2017), or the whole paragraph (Du and Cardie, 2018; Song et al., 2018b; Liu et al., 2019a; Zhao et al., 2018; Kim et al., 2019; Sun et al., 2018). Many works also used RL to optimize specific metrics (Song et al., 2018a; Kumar et al., 2019; Yuan et al., 2017). Recently, Zhang and Bansal

(2019) proposed semantics-enhanced rewards to improve the QG model, and also used the multi-reward approach proposed by Pasunuru and Bansal (2018) in their RL models.

2.4.5 Data-to-Text Generation

Data-to-text is the task of expressing the components (attributes and values) of meaning representation (MR) as human-readable natural sentences. Previous work in this area include templates (Reiter, 1995), rules (Reiter et al., 2005), pipelines (Reiter, 2007; Reiter and Dale, 1997), probabilistic models (Liang et al., 2009) and more recently end-to-end as well as neural-based methods (Wen et al., 2015; Mei et al., 2016; Dušek and Jurcicek, 2016; Lampouras and Vlachos, 2016; Dušek et al., 2020; Wiseman et al., 2017; Gong, 2018; Chen and Mooney, 2008; Reiter, 2017; Lebret et al., 2016; Distiawan et al., 2018; Gehrmann et al., 2018a; Marcheggiani and Perez-Beltrachini, 2018; Guo et al., 2019b; Zhao et al., 2020). In our work, we use the state-of-the-art model from Zhao et al. (2020) as our baseline.

2.4.6 Text Classification Tasks

Recently, GLUE benchmark (Wang et al., 2019a) has been proposed to evaluate models on multiple text classification tasks which are meant to cover a diverse and difficult range of NLP problems. These tasks are categorized into single sentence tasks, natural language inference tasks (NLI), and similarity and paraphrasing tasks. Many recent large-scale pre-trained language models are evaluated on this benchmark to show the generalizability of these models to diverse text classification tasks (Devlin et al., 2019; Liu et al., 2019b). In this dissertation, we primarily focus on the NLI tasks, where, given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither. Some of the popular NLI tasks are: Question-Answering NLI (Rajpurkar et al., 2016), Recognizing Textual Entailment (RTE) (Dagan et al., 2005), and Winograd NLI (Levesque et al., 2012).

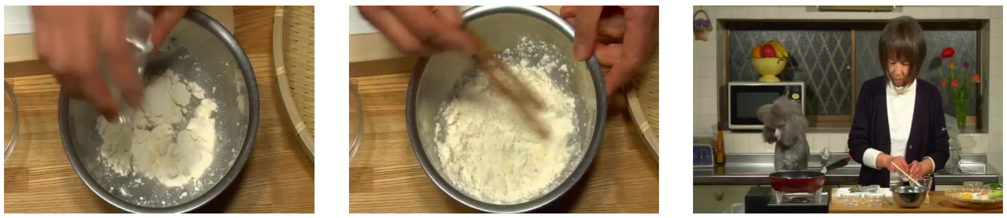
CHAPTER 3: MULTI-TASK LEARNING FOR VIDEO CAPTIONING

3.1 Introduction

Video captioning is the task of automatically generating a natural language description of the content of a video, as shown in Fig. 3.1. It has various applications such as assistance to a visually impaired person and improving the quality of online video search or retrieval. This task has gained recent momentum in the natural language processing and computer vision communities, esp. with the advent of powerful image processing features as well as sequence-to-sequence LSTM models. It is also a step forward from static image captioning, because in addition to modeling the spatial visual features, the model also needs to learn the temporal across-frame action dynamics and the logical storyline language dynamics.

Previous work in video captioning (Venugopalan et al., 2015a; Pan et al., 2016b) has shown that recurrent neural networks (RNNs) are a good choice for modeling the temporal information in the video. A sequence-to-sequence model is then used to 'translate' the video to a caption. Venugopalan et al. (2016) showed linguistic improvements over this by fusing the decoder with external language models. Furthermore, an attention mechanism between the video frames and the caption words captures some of the temporal matching relations better (Yao et al., 2015; Pan et al., 2016a). More recently, hierarchical two-level RNNs were proposed to allow for longer inputs and to model the full paragraph caption dynamics of long video clips (Pan et al., 2016a; Yu et al., 2016).

Despite these recent improvements, video captioning models still suffer from the lack of sufficient temporal and logical supervision to be able to correctly capture the action sequence and story-dynamic language in videos, esp. in the case of short clips. Hence, they would benefit from incorporating such complementary directed knowledge, both visual and textual. We



Ground truth: A person is mixing powdered ingradients with water.
A woman is mixing flour and water in a bowl.
Our model: A person is mixing ingredients in a bowl.

Figure 3.1: A video captioning example from the YouTube2Text dataset, with the ground truth captions and our many-to-many multi-task model's predicted caption.

address this by jointly training the task of video captioning with two related directed-generation tasks: a temporally-directed unsupervised video prediction task and a logically-directed language entailment generation task. We model this via many-to-many multi-task learning based sequence-to-sequence models (Luong et al., 2016) that allow the sharing of parameters among the encoders and decoders across the three different tasks, with additional shareable attention mechanisms.

The unsupervised video prediction task, i.e., video-to-video generation (adapted from Srivastava et al. (2015)), shares its encoder with the video captioning task's encoder, and helps it learn richer video representations that can predict their temporal context and action sequence. The entailment generation task, i.e., premise-to-entailment generation (based on the image caption domain SNLI corpus (Bowman et al., 2015)), shares its decoder with the video captioning decoder, and helps it learn better video-entailed caption representations, since the caption is essentially an entailment of the video, i.e., it describes subsets of objects and events that are logically implied by (or follow from) the full video content. The overall many-to-many multi-task model combines all three tasks.

Our three novel multi-task models show statistically significant improvements over the state-of-the-art, and achieve the best-reported results (and rank) on multiple datasets, based on several automatic and human evaluations. We also demonstrate that video captioning, in turn, gives mutual improvements on the new multi-reference entailment generation task.

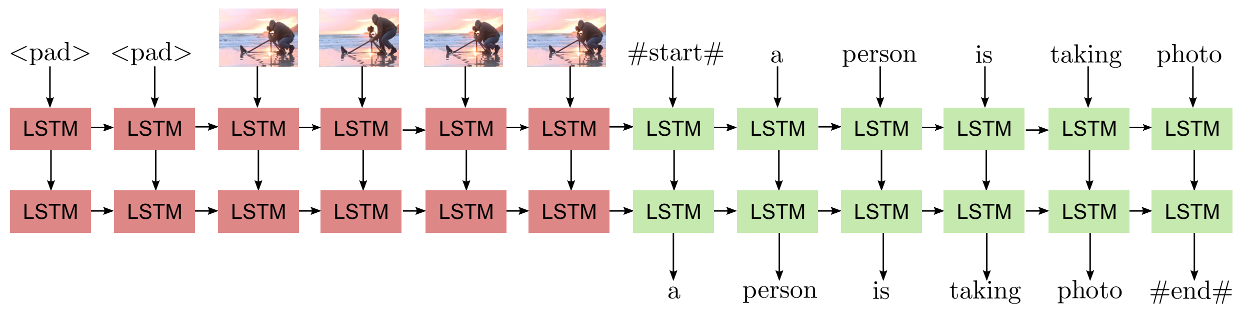


Figure 3.2: Baseline sequence-to-sequence model for video captioning: standard encoder-decoder LSTM-RNN model.

3.2 Models

We first discuss a simple encoder-decoder model as a baseline reference for video captioning. Next, we improve this via an attention mechanism. Finally, we present similar models for the unsupervised video prediction and entailment generation tasks, and then combine them with video captioning via the many-to-many multi-task approach.

3.2.1 Baseline Sequence-to-Sequence Model

Our baseline model is similar to the standard machine translation encoder-decoder RNN model (Sutskever et al., 2014) where the final state of the encoder RNN is input as an initial state to the decoder RNN, as shown in Fig. 3.2. The RNN is based on Long Short Term Memory (LSTM) units, which are good at memorizing long sequences due to forget-style gates (Hochreiter and Schmidhuber, 1997). For video captioning, our input to the encoder is the video frame features[1] $\{f_1, f_2, ..., f_n\}$ of length n, and the caption word sequence $\{w_1, w_2, ..., w_m\}$ of length m is generated during the decoding phase. The distribution of the output sequence w.r.t. the input sequence is:

$$p(w_1, ..., w_m | f_1, ..., f_n) = \prod_{t=1}^{m} p(w_t | h_t^d) \tag{3.1}$$

[1]We use several popular image features such as VGGNet, GoogLeNet and Inception-v4. Details in Sec. 3.3.1.

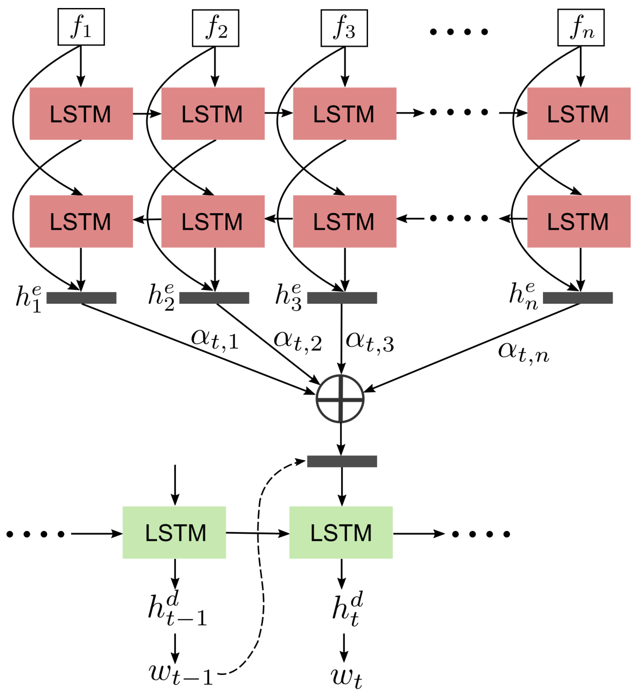


Figure 3.3: Attention-based sequence-to-sequence baseline model for video captioning (similar models also used for video prediction and entailment generation).

where h_t^d is the hidden state at the t^{th} time step of the decoder RNN, obtained from h_{t-1}^d and w_{t-1} via the standard LSTM-RNN equations. The distribution $p(w_t|h_t^d)$ is given by *softmax* over all the words in the vocabulary.

3.2.2 Attention-based Model

Our attention model architecture is similar to Bahdanau et al. (2015), with a bidirectional LSTM-RNN as the encoder and a unidirectional LSTM-RNN as the decoder, see Fig. 3.3. At each time step t, the decoder LSTM hidden state h_t^d is a non-linear recurrent function of the previous decoder hidden state h_{t-1}^d, the previous time-step's generated word w_{t-1}, and the context vector c_t:

$$h_t^d = S(h_{t-1}^d, w_{t-1}, c_t) \tag{3.2}$$

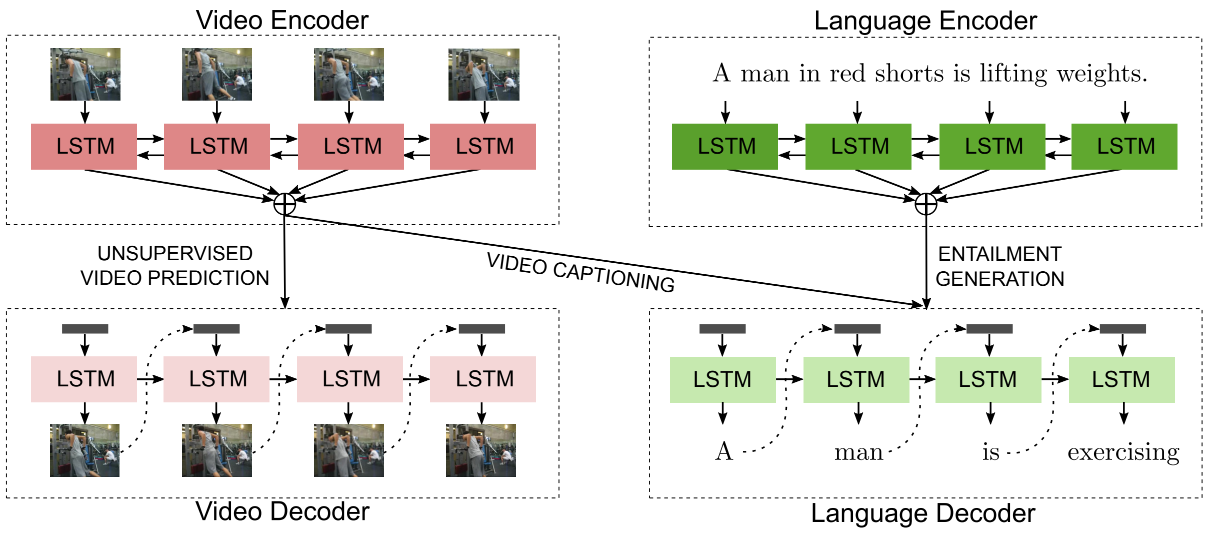


Figure 3.4: Our many-to-many multi-task learning model to share encoders and decoders of the video captioning, unsupervised video prediction, and entailment generation tasks.

where c_t is a weighted sum of encoder hidden states $\{h_i^e\}$:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i^e \tag{3.3}$$

These attention weights $\{\alpha_{t,i}\}$ act as an alignment mechanism by giving higher weights to certain encoder hidden states which match that decoder time step better, and are computed as:

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{k=1}^{n} exp(e_{t,k})} \tag{3.4}$$

where the attention function $e_{t,i}$ is defined as:

$$e_{t,i} = w^T tanh(W_a^e h_i^e + W_a^d h_{t-1}^d + b_a) \tag{3.5}$$

where w, W_a^e, W_a^d, and b_a are learned parameters. This attention-based sequence-to-sequence model (Fig. 3.3) is our enhanced baseline for video captioning. We next discuss similar models for the new tasks of unsupervised video prediction and entailment generation and then finally share them via multi-task learning.

3.2.3 Unsupervised Video Prediction

We model unsupervised video representation by predicting the sequence of future video frames given the current frame sequence. Similar to Sec. 3.2.2, a bidirectional LSTM-RNN encoder and an LSTM-RNN decoder is used, along with attention. If the frame level features of a video of length n are $\{f_1, f_2, ..., f_n\}$, these are divided into two sets such that given the current frames $\{f_1, f_2, .., f_k\}$ (in its encoder), the model has to predict (decode) the rest of the frames $\{f_{k+1}, f_{k+2}, .., f_n\}$. The motivation is that this helps the video encoder learn rich temporal representations that are aware of their action-based context and are also robust to missing frames and varying frame lengths or motion speeds. The optimization function is defined as:

$$\underset{\phi}{\text{minimize}} \sum_{t=1}^{n-k} ||f_t^d - f_{t+k}||_2^2 \tag{3.6}$$

where ϕ are the model parameters, f_{t+k} is the true future frame feature at decoder time step t and f_t^d is the decoder's predicted future frame feature at decoder time step t, defined as:

$$f_t^d = S(h_{t-1}^d, f_{t-1}^d, c_t) \tag{3.7}$$

similar to Eqn. 3.2, with h_{t-1}^d and f_{t-1}^d as the previous time step's hidden state and predicted frame feature respectively, and c_t as the attention-weighted context vector.

3.2.4 Entailment Generation

Given a sentence (premise), the task of entailment generation is to generate a sentence (hypothesis) which is a logical deduction or implication of the premise. Our entailment generation model again uses a bidirectional LSTM-RNN encoder and LSTM-RNN decoder with an attention mechanism (similar to Sec. 3.2.2). If the premise s^p is a sequence of words $\{w_1^p, w_2^p, ..., w_n^p\}$ and the hypothesis s^h is $\{w_1^h, w_2^h, ..., w_m^h\}$, the distribution of the entailed hypothesis w.r.t. the premise is:

$$p(w_1^h, ..., w_m^h | w_1^p, ..., w_n^p) = \prod_{t=1}^{m} p(w_t^h | h_t^d) \tag{3.8}$$

where the distribution $p(w_t^h|h_t^d)$ is again obtained via softmax over all the words in the vocabulary and the decoder state h_t^d is similar to Eqn. 3.2.

3.2.5 Multi-Task Learning

Multi-task learning helps in sharing information between different tasks and across domains. Our primary aim is to improve the video captioning model, where visual content translates to a textual form in a directed (entailed) generation way. Hence, this presents an interesting opportunity to share temporally and logically directed knowledge with both visual and linguistic generation tasks. Fig. 3.4 shows our overall many-to-many multi-task model for jointly learning video captioning, unsupervised video prediction, and textual entailment generation. Here, the video captioning task shares its video encoder (parameters) with the encoder of the video prediction task (one-to-many setting) so as to learn context-aware and temporally-directed visual representations (see Sec. 3.2.3).

Moreover, the decoder of the video captioning task is shared with the decoder of the textual entailment generation task (many-to-one setting), thus helping generate captions that can be 'entailed' by, i.e., are logically implied by or follow from the video content (see Sec. 3.2.4).[2] In both the one-to-many and the many-to-one settings, we also allow the attention parameters to be shared or separated. The overall many-to-many setting thus improves both the visual and language representations of the video captioning model.

We train the multi-task model by alternately optimizing each task in mini-batches based on a mixing ratio. Let α_v, α_f, and α_e be the number of mini-batches optimized alternately from each

[2]Empirically, logical entailment helped captioning more than simple fusion with language modeling (i.e., partial sentence completion with no logical implication), because a caption is also 'entailed' by a video in a logically-directed sense and hence the entailment generation task matches the video captioning task better than language modeling. Moreover, a multi-task setup is more suitable to add directed information such as entailment (as opposed to pretraining or fusion with only the decoder). Details in Sec. 3.4.1.

of these three tasks – video captioning, unsupervised video future frames prediction, and entailment generation, resp. Then the mixing ratio is defined as $\frac{\alpha_v}{(\alpha_v+\alpha_f+\alpha_e)} : \frac{\alpha_f}{(\alpha_v+\alpha_f+\alpha_e)} : \frac{\alpha_e}{(\alpha_v+\alpha_f+\alpha_e)}$.

3.3 Experimental Setup

3.3.1 Datasets

Video Captioning Datasets We report results on three popular video captioning datasets. First, we use the YouTube2Text or MSVD (Chen and Dolan, 2011) for our primary results, which contains 1970 YouTube videos in the wild with several different reference captions per video (40 on average). We also use MSR-VTT (Xu et al., 2016a) with $10,000$ diverse video clips (from a video search engine) – it has $200,000$ video clip-sentence pairs and around 20 captions per video; and M-VAD (Torabi et al., 2015) with $49,000$ movie-based video clips but only 1 or 2 captions per video, making most evaluation metrics (except paraphrase-based METEOR) infeasible. We use the standard splits for all three datasets. Further details about all these datasets are provided in Appendix A.1.

Video Prediction Dataset For our unsupervised video representation learning task, we use the UCF-101 action videos dataset (Soomro et al., 2012), which contains $13,320$ video clips of 101 action categories, and suits our video captioning task well because it also contains short video clips of a single action or few actions. We use the standard splits – further details in Appendix A.1.

Entailment Generation Dataset For the entailment generation encoder-decoder model, we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), which contains human-annotated English sentence pairs with classification labels of entailment, contradiction and neutral. It has a total of $570,152$ sentence pairs out of which $190,113$ correspond to true entailment pairs, and we use this subset in our multi-task video captioning model. For improving video captioning, we use the same training/validation/test splits as provided by Bowman et al.

(2015), which is $183,416$ training, $3,329$ validation, and $3,368$ testing pairs (for the entailment subset).

However, for the entailment generation multi-task results (see results in Sec. 3.4.3), we modify the splits so as to create a multi-reference setup which can afford evaluation with automatic metrics. A given premise usually has multiple entailed hypotheses but the original SNLI corpus is set up as single-reference (for classification). Due to this, the different entailed hypotheses of the same premise land up in different splits of the dataset (e.g., one in train and one in test/validation) in many cases. Therefore, we regroup the premise-entailment pairs and modify the split as follows: among the $190,113$ premise-entailment pairs subset of the SNLI corpus, there are $155,898$ unique premises; out of which $145,822$ have only one hypothesis and we make this the training set, and the rest of them $(10,076)$ have more than one hypothesis, which we randomly shuffle and divide equally into test and validation sets, so that each of these two sets has approximately the same distribution of the number of reference hypotheses per premise.

These new validation and test sets hence contain premises with multiple entailed hypotheses as ground truth references, thus allowing for automatic metric evaluation, where differing generations still get positive scores by matching one of the multiple references. Also, this creates a more challenging dataset for entailment generation because of zero premise overlap between the training and val/test sets. We will make these split details publicly available.

Pre-trained Visual Frame Features For the three video captioning and UCF-101 datasets, we fix our sampling rate to $3fps$ to bring uniformity in the temporal representation of actions across all videos. These sampled frames are then converted into features using several state-of-the-art pre-trained models on ImageNet (Deng et al., 2009) – VGGNet (Simonyan and Zisserman, 2015), GoogLeNet (Szegedy et al., 2015; Ioffe and Szegedy, 2015), and Inception-v4 (Szegedy et al., 2016).

3.3.2 Evaluation (Automatic and Human)

For our video captioning as well as entailment generation results, we use four diverse automatic evaluation metrics that are popular for image/video captioning and language generation in general: METEOR (Denkowski and Lavie, 2014a), BLEU-4 (Papineni et al., 2002), CIDEr-D (Vedantam et al., 2015), and ROUGE-L (Lin, 2004). Particularly, METEOR and CIDEr-D have been justified to be better for generation tasks, because CIDEr-D uses consensus among the (large) number of references and METEOR uses soft matching based on stemming, paraphrasing, and WordNet synonyms. We use the standard evaluation code from the Microsoft COCO server (Chen et al., 2015) to obtain these results and also to compare the results with previous papers.[3]

We also present human evaluation results based on relevance (i.e., how related is the generated caption w.r.t. the video contents such as actions, objects, and events; or is the generated hypothesis entailed or implied by the premise) and coherence (i.e., a score on the logic, readability, and fluency of the generated sentence).

3.3.3 Training Details

We tune all hyperparameters on the dev splits: LSTM-RNN hidden state size, learning rate, weight initializations, and mini-batch mixing ratios (tuning ranges in Appendix A.1). We use the following settings in all of our models (unless otherwise specified): we unroll video encoder/decoder RNNs to 50 time steps and language encoder/decoder RNNs to 30 time steps. We use a 1024-dimension RNN hidden state size and 512-dim vectors to embed visual features and word vectors. We use Adam optimizer (Kingma and Ba, 2015). We apply a dropout of 0.5. See Appendix A.1 for full details.

[3]We use avg. of these four metrics on validation set to choose the best model, except for single-reference M-VAD dataset where we only report and choose based on METEOR.

3.4 Experimental Results

3.4.1 Video Captioning on YouTube2Text

Table 3.1 presents our primary results on the YouTube2Text (MSVD) dataset, reporting several previous works, all our baselines and attention model ablations, and our three multi-task models, using the four automated evaluation metrics. For each subsection below, we have reported the important training details inline, and refer to Appendix A.1 for full details (e.g., learning rates and initialization).

Baseline Performance We first present all our baseline model choices (ablations) in Table 3.1. Our baselines represent the standard sequence-to-sequence model with three different visual feature types as well as those with attention mechanisms. Each baseline model is trained with three random seed initializations and the average is reported (for stable results). The final baseline model $\otimes$ instead uses an ensemble (E), which is a standard denoising method (Sutskever et al., 2014) that performs inference over ten randomly initialized models, i.e., at each time step t of the decoder, we generate a word based on the avg. of the likelihood probabilities from the ten models. Moreover, we use beam search with size 5 for all baseline models. Overall, the final baseline model with Inception-v4 features, attention, and 10-ensemble performs well (and is better than all previous state-of-the-art), and so we next add all our novel multi-task models on top of this final baseline.

Multi-Task with Video Prediction (1-to-M) Here, the video captioning and unsupervised video prediction tasks share their encoder LSTM-RNN weights and image embeddings in a one-to-many multi-task setting. Two important hyperparameters tuned (on the validation set of captioning datasets) are the ratio of encoder vs decoder frames for video prediction on UCF-101 (where we found that 80% of frames as input and 20% for prediction performs best); and the mini-batch mixing ratio between the captioning and video prediction tasks (where we found $100 : 200$ works

Models	METEOR	CIDEr-D	ROUGE-L	BLEU-4
PREVIOUS WORK				
LSTM-YT (V) (Venugopalan et al., 2015b)	26.9	-	-	31.2
S2VT (V + A) (Venugopalan et al., 2015a)	29.8	-	-	-
Temporal Attention (G + C) (Yao et al., 2015)	29.6	51.7	-	41.9
LSTM-E (V + C) (Pan et al., 2016b)	31.0	-	-	45.3
Glove + DeepFusion (V) (E) (Venugopalan et al., 2016)	31.4	-	-	42.1
p-RNN (V + C) (Yu et al., 2016)	32.6	65.8	-	49.9
HNRE + Attention (G + C) (Pan et al., 2016a)	33.9	-	-	46.7
OUR BASELINES				
Baseline (V)	31.4	63.9	68.0	43.6
Baseline (G)	31.7	64.8	68.6	44.1
Baseline (I)	33.3	75.6	69.7	46.3
Baseline + Attention (V)	32.6	72.2	69.0	47.5
Baseline + Attention (G)	33.0	69.4	68.3	44.9
Baseline + Attention (I)	33.8	77.2	70.3	49.9
Baseline + Attention (I) (E) $\otimes$	35.0	84.4	71.5	52.6
OUR MULTI-TASK LEARNING MODELS				
$\otimes$ + Video Prediction (1-to-M)	35.6	88.1	72.9	54.1
$\otimes$ + Entailment Generation (M-to-1)	35.9	88.0	72.7	54.4
$\otimes$ + Video Prediction + Entailment Generation (M-to-M)	**36.0**	**92.4**	**72.8**	**54.5**

Table 3.1: Primary video captioning results on Youtube2Text (MSVD), showing previous works, our several strong baselines, and our three multi-task models. Here, V, G, I, C, A are short for VGGNet, GoogLeNet, Inception-v4, C3D, and AlexNet visual features; E = ensemble. The multi-task models are applied on top of our best video captioning baseline $\otimes$, with an ensemble. All the multi-task models are statistically significant over the baseline (discussed inline in the corresponding results sections).

well). Table 3.1 shows a statistically significant improvement[4] in all metrics in comparison to the best baseline (non-multitask) model as well as w.r.t. all previous works, demonstrating the effectiveness of multi-task learning for video captioning with video prediction, even with unsupervised signals.

Multi-Task with Entailment Generation (M-to-1) Here, the video captioning and entailment generation tasks share their language decoder LSTM-RNN weights and word embeddings in a many-to-one multi-task setting. We observe that a mixing ratio of $100 : 50$ alternating mini-batches (between the captioning and entailment tasks) works well here. Again, Table 3.1 shows

[4]Statistical significance of $p < 0.01$ for CIDEr-D and ROUGE-L, $p < 0.02$ for BLEU-4, $p < 0.03$ for METEOR, based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples.

statistically significant improvements[5] in all the metrics in comparison to the best baseline model (and all previous works) under this multi-task setting. Note that in our initial experiments, our entailment generation model helped the video captioning task significantly more than the alternative approach of simply improving fluency by adding (or deep-fusing) an external language model (or pre-trained word embeddings) to the decoder (using both in-domain and out-of-domain language models), again because a caption is also 'entailed' by a video in a logically-directed sense and hence this matches our captioning task better (also see results of Venugopalan et al. (2016) in Table 3.1).

Multi-Task with Video and Entailment Generation (M-to-M) Combining the above one-to-many and many-to-one multi-task learning models, our full model is the 3-task, many-to-many model (Fig. 3.4) where both the video encoder and the language decoder of the video captioning model are shared (and hence improved) with that of the unsupervised video prediction and entailment generation models, respectively.[6] A mixing ratio of $100 : 100 : 50$ alternate mini-batches of video captioning, unsupervised video prediction, and entailment generation, resp. works well. Table 3.1 shows that our many-to-many multi-task model again outperforms our strongest baseline (with statistical significance of $p < 0.01$ on all metrics), as well as all the previous state-of-the-art results by large absolute margins on all metrics. It also achieves significant improvements on some metrics over the one-to-many and many-to-one models.[7] Overall, we achieve the best results to date on YouTube2Text (MSVD) on all metrics.

3.4.2 Video Captioning on MSR-VTT, M-VAD

In Table 3.2, we also train and evaluate our final many-to-many multi-task model on two other video captioning datasets (using their standard splits; details in Appendix A.1). First, we evaluate

[5]Statistical significance of $p < 0.01$ for all four metrics.

[6]We found the setting with unshared attention parameters to work best, likely because video captioning and video prediction prefer very different alignment distributions.

[7]Many-to-many model's improvements have a statistical significance of $p < 0.01$ on all metrics w.r.t. baseline, and $p < 0.01$ on CIDEr-D w.r.t. both one-to-many and many-to-one models, and $p < 0.04$ on METEOR w.r.t. one-to-many.

Models	M	C	R	B
Venugopalan et al. (2015b)*	23.4	-	-	32.3
Yao et al. (2015)*	25.2	-	-	35.2
Xu et al. (2016a)	25.9	-	-	36.6
Rank1: v2t_navigator	28.2	44.8	**60.9**	**40.8**
Rank2: Aalto	26.9	45.7	59.8	39.8
Rank3: VideoLAB	27.7	44.1	60.6	39.1
Our Model (**New Rank1**)	**28.8**	**47.1**	60.2	**40.8**

Table 3.2: Results on MSR-VTT dataset on the 4 metrics. *Results are reimplementations as per Xu et al. (2016a). We also report the top 3 leaderboard systems – our model achieves the new rank 1 based on their ranking method.

Models	METEOR
Yao et al. (2015)	5.7
Venugopalan et al. (2015a)	6.7
Pan et al. (2016a)	6.8
Our M-to-M Multi-Task Model	**7.4**

Table 3.3: Results on M-VAD dataset.

on the new MSR-VTT dataset (Xu et al., 2016a). Since this is a recent dataset, we list previous works' results as reported by the MSR-VTT dataset paper itself.[8] We improve over all of these significantly. Moreover, they maintain a leaderboard[9] on this dataset and we also report the top 3 systems from it. Based on their ranking method, our multi-task model achieves the new rank 1 on this leaderboard. In Table 3.3, we further evaluate our model on the challenging movie-based M-VAD dataset, and again achieve improvements over all previous work (Venugopalan et al., 2015a; Pan et al., 2016a; Yao et al., 2015).[10]

3.4.3 Entailment Generation Results

Above, we showed that the new entailment generation task helps improve video captioning. Next, we show that the video captioning task also inversely helps the entailment generation task.

[8]In their updated supplementary at `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/cvpr16.supplementary.pdf`

[9]`http://ms-multimedia-challenge.com/leaderboard`

[10]Following previous work, we only use METEOR because M-VAD only has a single reference caption per video.

Models	M	C	R	B
Entailment Generation	29.6	117.8	62.4	40.6
+Video Caption (M-to-1)	**30.0**	**121.6**	**63.9**	**41.6**

Table 3.4: Entailment generation results with the four metrics.

Given a premise, the task of entailment generation is to generate an entailed hypothesis. We use only the entailment pairs subset of the SNLI corpus for this, but with a multi-reference split setup to allow automatic metric evaluation and a zero train-test premise overlap (see Sec. 3.3.1). All the hyperparameter details (again tuned on the validation set) are presented in Appendix A.1. Table 3.4 presents the entailment generation results for the baseline (sequence-to-sequence with attention, 3-ensemble, beam search) and the multi-task model which uses video captioning (shared decoder) on top of the baseline. A mixing ratio of $100 : 20$ alternate mini-batches of entailment generation and video captioning (resp.) works well.[11] The multi-task model achieves stat. significant ($p < 0.01$) improvements over the baseline on all metrics, thus demonstrating that video captioning and entailment generation both mutually help each other.

3.4.4 Human Evaluation

In addition to the automated evaluation metrics, we present pilot-scale human evaluations on the YouTube2Text (Table 3.1) and entailment generation (Table 3.4) results. In each case, we compare our strongest baseline with our final multi-task model (M-to-M in case of video captioning and M-to-1 in case of entailment generation). We evaluate a random sample of 300 generated captions (or entailed hypotheses) from the test set, across three human evaluators. We remove the model identity to anonymize the two models, and ask the human evaluators to choose the better model based on *relevance* and *coherence* (described in Sec. 3.3.2). As shown in Table 3.5 and Table 3.6, the multi-task models are always better than the strongest baseline

[11]Note that this many-to-one model prefers a different mixing ratio and learning rate than the many-to-one model for improving video captioning (Sec. 3.4.1), because these hyperparameters depend on the primary task being improved, as also discussed in previous work (Luong et al., 2016).

	Relevance	Coherence
Not Distinguishable	70.7%	92.6%
SotA Baseline Wins	12.3%	1.7%
Multi-Task Wins (M-to-M)	**17.0%**	**5.7%**

Table 3.5: Human evaluation on YouTube2Text video captioning.

	Relevance	Coherence
Not Distinguishable	84.6%	98.3%
SotA Baseline Wins	6.7%	0.7%
Multi-Task Wins (M-to-1)	**8.7%**	**1.0%**

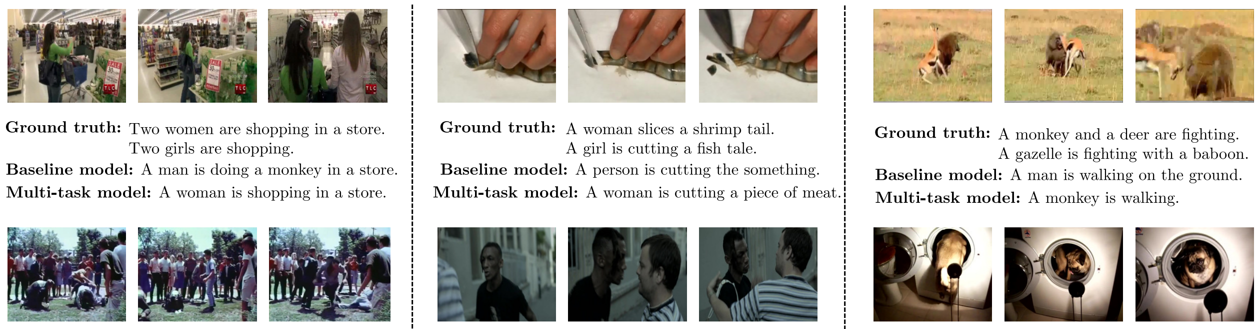
Table 3.6: Human evaluation on entailment generation.

for both video captioning and entailment generation, on both relevance and coherence, and with similar improvements (2-7%) as the automatic metrics (shown in Table 3.1).

3.4.5 Insights

Fig. 3.5 shows video captioning generation results on the YouTube2Text dataset where our final M-to-M multi-task model is compared with our strongest attention-based baseline model for three categories of videos: (a) complex examples where the multi-task model performs better than the baseline; (b) ambiguous examples (i.e., ground truth itself confusing) where multi-task model still correctly predicts one of the possible categories (c) complex examples where both models perform poorly. Overall, we find that the multi-task model generates captions that are better at both temporal action prediction and logical entailment (i.e., correct subset of full video premise) w.r.t. the ground truth captions.

On analyzing the cases where the baseline is better than the final M-to-M multi-task model, we find that these are often scenarios where the multi-task model's caption is also correct but the baseline caption is a bit more specific, e.g., "a man is holding a gun" vs "a man is shooting a gun".



(a)

Ground truth: Two women are shopping in a store.
Two girls are shopping.
Baseline model: A man is doing a monkey in a store.
Multi-task model: A woman is shopping in a store.

Ground truth: Two men are fighting.
A group of boys are fighting.
Baseline model: A group of men are dancing.
Multi-task model: Two men are fighting.

(b)

Ground truth: A woman slices a shrimp tail.
A girl is cutting a fish tale.
Baseline model: A person is cutting the something.
Multi-task model: A woman is cutting a piece of meat.

Ground truth: Two men are talking aggressively.
The boy is talking.
Baseline model: A man is crying.
Multi-task model: A man is talking.

(c)

Ground truth: A monkey and a deer are fighting.
A gazelle is fighting with a baboon.
Baseline model: A man is walking on the ground.
Multi-task model: A monkey is walking.

Ground truth: A dog climbs into a dryer.
A dog is in a washing machine.
Baseline model: A man is playing.
Multi-task model: A man is playing with a toy.

Figure 3.5: Examples of generated video captions on the YouTube2Text dataset: (a) complex examples where the multi-task model performs better than the baseline; (b) ambiguous examples (i.e., ground truth itself confusing) where multi-task model still correctly predicts one of the possible categories (c) complex examples where both models perform poorly.

Given Premise	Generated Entailment
a man on stilts is playing a tuba for money on the boardwalk	a man is playing an instrument
a child that is dressed as spiderman is ringing the doorbell	a child is dressed as a superhero
several young people sit at a table playing poker	people are playing a game
a woman in a dress with two children	a woman is wearing a dress
a blue and silver monster truck making a huge jump over crushed cars	a truck is being driven

Table 3.7: Examples of our multi-task model's generated entailment hypotheses given a premise.

Finally, Table 3.7 presents output examples of our entailment generation multi-task model (Sec. 3.4.3), showing how the model accurately learns to produce logically implied subsets of the premise.

3.5 Conclusion

We presented a multimodal, multi-task learning approach to improve video captioning by incorporating temporally and logically directed knowledge via video prediction and entailment generation tasks. We achieve the best reported results (and rank) on three datasets, based on multiple automatic and human evaluations. We also show mutual multi-task improvements on the new entailment generation task. In the next chapters, we apply our entailment-based multi-task paradigm to other directed language generation tasks such as textual summarization task.

CHAPTER 4: MULTI-TASK LEARNING FOR TEXTUAL SUMMARIZATION

4.1 Introduction

Abstractive summarization is the challenging NLG task of compressing and rewriting a document into a short, relevant, salient, and coherent summary. It has numerous applications such as summarizing storylines, event understanding, etc. As compared to extractive or compressive summarization (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015; Henß et al., 2015), abstractive summaries are based on rewriting as opposed to selecting. Recent end-to-end, neural sequence-to-sequence models and larger datasets have allowed substantial progress on the abstractive task, with ideas ranging from copy-pointer mechanism and redundancy coverage, to metric reward based reinforcement learning (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017).

Despite these strong recent advancements, there is still a lot of scope for improving the summary quality generated by these models. A good rewritten summary is one that contains all the salient information from the document, is logically followed (entailed) by it, and avoids redundant information. The redundancy aspect was addressed by coverage models (Suzuki and Nagata, 2016; Chen et al., 2016; Nallapati et al., 2016; See et al., 2017), but we still need to teach these models about how to better detect salient information from the input document, as well as about better logically-directed natural language inference skills.

In this work, we improve abstractive text summarization via soft, high-level (semantic) layer-specific multi-task learning with two relevant auxiliary tasks. The first is that of document-to-question generation, which teaches the summarization model about what are the right questions to ask, which in turn is directly related to what the salient information in the input document is. The second auxiliary task is a premise-to-entailment generation task to teach it how to rewrite a summary which is a directed-logical subset of (i.e., logically follows from) the input document, and contains no contradictory or unrelated information. For the question generation task, we use the SQuAD dataset (Rajpurkar et al., 2016), where we learn to generate a question given a sentence containing the answer, similar to the recent work by Du et al. (2017). Our entailment generation task is based on the recent SNLI classification dataset and task (Bowman et al., 2015), converted to a generation task (Pasunuru and Bansal, 2017a).

Further, we also present novel multi-task learning architectures based on multi-layered encoder and decoder models, where we empirically show that it is substantially better to share the higher-level semantic layers between the three aforementioned tasks, while keeping the lower-level (lexico-syntactic) layers unshared. We also explore different ways to optimize the shared parameters and show that 'soft' parameter sharing achieves higher performance than hard sharing.

Empirically, our soft, layer-specific sharing model with the question and entailment generation auxiliary tasks achieves statistically significant improvements over the state-of-the-art on both the CNN/DailyMail and Gigaword datasets. It also performs significantly better on the DUC-2002 transfer setup, demonstrating its strong generalizability as well as the importance of auxiliary knowledge in low-resource scenarios. We also report improvements on our auxiliary question and entailment generation tasks over their respective previous state-of-the-art. Moreover, we significantly decrease the training time of the multi-task models by initializing the individual tasks from their pretrained baseline models. Finally, we present human evaluation studies as well as detailed quantitative and qualitative analysis studies of the improved saliency detection and logical inference skills learned by our multi-task model.

4.2 Models

First, we introduce our pointer+coverage baseline model and then our two auxiliary tasks: question generation and entailment generation (and finally the multi-task learning models in Sec. 4.3).

4.2.1 Baseline Pointer+Coverage Model

We use a sequence-attention-sequence model with a 2-layer bidirectional LSTM-RNN encoder and a 2-layer uni-directional LSTM-RNN decoder, along with Bahdanau et al. (2015) style attention. Let $x = \{x_1, x_2, ..., x_m\}$ be the source document and $y = \{y_1, y_2, ..., y_n\}$ be the target summary. The output summary generation vocabulary distribution conditioned over the input source document is $P_v(y|x;\theta) = \prod_{t=1}^{n} p_v(y_t|y_{1:t-1}, x; \theta)$. Let the decoder hidden state be s_t at time step t and let c_t be the context vector which is defined as a weighted combination of encoder hidden states. We concatenate the decoder's (last) RNN layer hidden state s_t and context vector c_t and apply a linear transformation, and then project to the vocabulary space by another linear transformation. Finally, the conditional vocabulary distribution at each time step t of the decoder is defined as:

$$p_v(y_t|y_{1:t-1}, x; \theta) = \mathrm{sfm}(V_p(W_f[s_t; c_t] + b_f) + b_p) \tag{4.1}$$

where, W_f, V_p, b_f, b_p are trainable parameters, and $\mathrm{sfm}(\cdot)$ is the softmax function.

Pointer-Generator Networks Pointer mechanism (Vinyals et al., 2015) helps in directly copying the words from the source sequence during target sequence generation, which is a good fit for a task like summarization. Our pointer mechanism approach is similar to See et al. (2017), who use a soft switch based on the generation probability $p_g = \sigma(W_g c_t + U_g s_t + V_g e_{w_{t-1}} + b_g)$, where $\sigma(\cdot)$ is a sigmoid function, W_g, U_g, V_g and b_g are parameters learned during training. $e_{w_{t-1}}$ is the previous time step output word embedding. The final word distribution is $P_f(y) = p_g \cdot P_v(y) + (1 - p_g) \cdot P_c(y)$, where P_v vocabulary distribution is as shown in Eq. 4.1, and copy distribution P_c is based on the attention distribution over source document words.

Coverage Mechanism Following previous work (See et al., 2017), coverage helps alleviate the issue of word repetition while generating long summaries. We maintain a coverage vector $\hat{c}_t = \sum_{t=0}^{t-1} \alpha_t$ that sums over all of the previous time steps attention distributions α_t, and this is added as input to the attention mechanism. Coverage loss is $L_{cov}(\theta) = \sum_t \sum_i min(\alpha_{t,i}, \hat{c}_{t,i})$.

Finally, the total loss is a weighted combination of cross-entropy loss and coverage loss:

$$L(\theta) = -\log P_f(y) + \lambda L_{cov}(\theta) \tag{4.2}$$

where λ is a tunable hyperparameter.

4.2.2 Two Auxiliary Tasks

Despite the strengths of the baseline model described above with attention, pointer, and coverage, a good summary should also contain maximal salient information and be a directed logical entailment of the source document. We teach these skills to the abstractive summarization model via multi-task training with two related auxiliary tasks: question generation task and entailment generation.

Question Generation The task of question generation is to generate a question from a given input sentence, which in turn is related to the skill of being able to find the important salient information to ask questions about. First the model has to identify the important information present in the given sentence, then it has to frame (generate) a question based on this salient information, such that, given the sentence and the question, one has to be able to predict the correct answer (salient information in this case). A good summary should also be able to find and extract all the salient information in the given source document, and hence we incorporate such capabilities into our abstractive text summarization model by multi-task learning it with a question generation task, sharing some common parameters/representations (see more details in Sec. 4.3). For setting up the question generation task, we follow Du et al. (2017) and use the SQuAD dataset to extract sentence-question pairs. Next, we use the same sequence-to-sequence model architecture as our summarization model. Note that even though our question generation task is generating one question at a time[1], our multi-task framework (see Sec. 4.3) is set up in such a way that the sentence-

[1]We also tried to generate all the questions at once from the full document, but we obtained low accuracy because of this task's challenging nature and overall less training data.

level knowledge from this auxiliary task can help the document-level primary (summarization) task to generate multiple salient facts – by sharing high-level semantic layer representations. See Sec. 4.6 and Table 4.10 for a quantitative evaluation showing that the multi-task model can find multiple (and more) salient phrases in the source document. Also see Sec. 4.6 (and supp) for challenging qualitative examples where baseline and SotA models only recover a small subset of salient information but our multi-task model with question generation is able to detect more of the important information.

Entailment Generation The task of entailment generation is to generate a hypothesis which is entailed by (or logically follows from) the given premise as input. In summarization, the generation decoder also needs to generate a summary that is entailed by the source document, i.e., does not contain any contradictory or unrelated/extraneous information as compared to the input document. We again incorporate such inference capabilities into the summarization model via multi-task learning, sharing some common representations/parameters between our summarization and entailment generation model (more details in Sec. 4.3). For this task, we use the entailment-labeled pairs from the SNLI dataset (Bowman et al., 2015) and set it up as a generation task (using the same strong model architecture as our abstractive summarization model). See Sec. 4.6 and Table 4.9 for a quantitative evaluation showing that the multi-task model is better entailed by the source document and has fewer extraneous facts. Also see Sec. 4.6 for qualitative example of how our multi-task model with the entailment auxiliary task is able to generate more logically-entailed summaries than the baseline and SotA models, which instead produce extraneous, unrelated words not present (in any paraphrased form) in the source document.

4.3 Multi-Task Learning

We employ multi-task learning for parallel training of our three tasks: abstractive summarization, question generation, and entailment generation. In this section, we describe our novel layer-specific, soft-sharing approaches and other multi-task learning details.

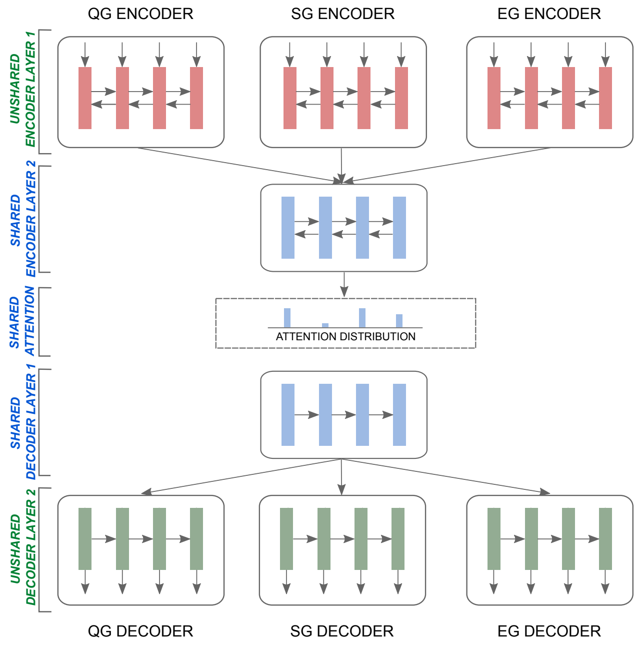


Figure 4.1: Overview of our multi-task model with parallel training of three tasks: abstractive summary generation (SG), question generation (QG), and entailment generation (EG). We share the 'blue' color representations across all the three tasks, i.e., second layer of encoder, attention parameters, and first layer of decoder.

4.3.1 Layer-Specific Sharing Mechanism

Simply sharing all parameters across the related tasks is not optimal, because models for different tasks have different input and output distributions, esp. for low-level vs. high-level parameters. Therefore, related tasks should share some common representations (e.g., high-level information), as well as need their own individual task-specific representations (esp. low-level information). To this end, we allow different components of model parameters of related tasks to be shared vs. unshared, as described next.

Encoder Layer Sharing: Belinkov et al. (2017) observed that lower layers (i.e., the layers closer to the input words) of RNN cells in a seq2seq machine translation model learn to represent word structure, while higher layers (farther from input) are more focused on high-level semantic meanings (similar to findings in the computer vision community for image features (Zeiler and Fergus, 2014)). We believe that while textual summarization, question generation, and entailment generation have different training data distributions and low-level representations, they can still benefit from sharing their models' high-level components (e.g., those that capture the skills of saliency and inference). Thus, we keep the lower-level layer (i.e., first layer closer to input words) of the 2-layer encoder of all three tasks unshared, while we share the higher layer (second layer in our model as shown in Fig. 4.1) across the three tasks.

Decoder Layer Sharing: Similarly for the decoder, lower layers (i.e., the layers closer to the output words) learn to represent word structure for generation, while higher layers (farther from output) are more focused on high-level semantic meaning. Hence, we again share the higher level components (first layer in the decoder far from output as shown in Fig. 4.1), while keeping the lower layer (i.e., second layer) of decoders of all three tasks unshared.

Attention Sharing: As described in Sec. 4.2.1, the attention mechanism defines an attention distribution over high-level layer encoder hidden states and since we share the second, high-level (semantic) layer of all the encoders, it is intuitive to share the attention parameters as well.

4.3.2 Soft vs. Hard Parameter Sharing

Hard-sharing: In the most common multi-task learning hard-sharing approach, the parameters to be shared are forced to be the same. As a result, gradient information from multiple tasks will directly pass through shared parameters, hence forcing a common space representation for all the related tasks. **Soft-sharing**: In our soft-sharing approach, we encourage shared parameters to be close in representation space by penalizing their l_2 distances. Unlike hard sharing, this approach gives more flexibility for the tasks by only loosely coupling the shared space representations. We minimize the following loss function for the primary task in soft-sharing approach:

$$L(\theta) = -\log P_f(y) + \lambda L_{cov}(\theta) + \gamma \|\theta_s - \psi_s\| \tag{4.3}$$

where γ is a hyperparameter, θ represents the primary summarization task's full parameters, while θ_s and ψ_s represent the shared parameter subset between the primary and auxiliary tasks.

4.3.3 Fast Multi-Task Training

During multi-task learning, we alternate the mini-batch optimization of the three tasks, based on a tunable 'mixing ratio' $\alpha_s : \alpha_q : \alpha_e$; i.e., optimizing the summarization task for α_s mini-batches followed by optimizing the question generation task for α_q mini-batches, followed by entailment generation task for α_e mini-batches (and for 2-way versions of this, we only add one auxiliary task at a time). We continue this process until all the models converge. Also, importantly, instead of training from scratch, we start the primary task (summarization) from a 90%-converged model of its baseline to make the training process faster. We observe that starting from a fully-converged baseline makes the model stuck in a local minimum. In addition, we also start all auxiliary models from their 90%-converged baselines, as we found that starting the auxiliary models from scratch has a chance to pull the primary model's shared parameters towards randomly-initialized auxiliary model's shared parameters.

4.4 Experimental Setup

Datasets: We use CNN/DailyMail dataset (Hermann et al., 2015; Nallapati et al., 2016) and Gigaword (Rush et al., 2015) datasets for summarization, and the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) datasets for our entailment and question generation tasks, resp. We also show generalizability/transfer results on DUC-2002 with our CNN/DM trained models. Appendix A.2 has full dataset details.

Evaluation Metrics: We use the standard ROUGE evaluation package (Lin, 2004) for reporting the results on all of our summarization models. Following previous work (Chopra et al., 2016; Nallapati et al., 2016), we use ROUGE full-length F1 variant for all our results. Following See et al. (2017), we also report METEOR (Denkowski and Lavie, 2014a) using the MS-COCO evaluation script (Chen et al., 2015).

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
PREVIOUS WORK				
Seq2Seq(50k vocab) (See et al., 2017)	31.33	11.81	28.83	12.03
Pointer (See et al., 2017)	36.44	15.66	33.42	15.35
Pointer+Coverage (See et al., 2017) $\star$	39.53	17.28	36.38	18.72
Pointer+Coverage (See et al., 2017) †	38.82	16.81	35.71	18.14
OUR MODELS				
Two-Layer Baseline (Pointer+Coverage) $\otimes$	39.56	17.52	36.36	18.17
$\otimes$ + Entailment Generation	39.84	17.63	36.54	18.61
$\otimes$ + Question Generation	39.73	17.59	36.48	18.33
$\otimes$ + Entailment Gen. + Question Gen.	39.81	17.64	36.54	18.54

Table 4.1: CNN/DailyMail summarization results. ROUGE scores are full length F-1 (as previous work). All the multi-task improvements are statistically significant over the state-of-the-art baseline.

Human Evaluation Criteria: We used Amazon MTurk to perform human evaluation of summary *relevance* and *readability*. We selected human annotators that were located in the US, had an approval rate greater than 95%, and had at least 10,000 approved HITs. For the pairwise model comparisons discussed in Sec. 4.5.2, we showed the annotators the input article, the ground truth summary, and the two model summaries (randomly shuffled to anonymize model identities) – we then asked them to choose the better among the two model summaries or choose 'Not-Distinguishable' if both summaries are equally good/bad. Instructions for relevance were defined based on the summary containing salient/important information from the given article, being correct (i.e., avoiding contradictory/unrelated information), and avoiding redundancy. Instructions for readability were based on the summary's fluency, grammaticality, and coherence.

Training Details All our soft/hard and layer-specific sharing decisions were made on the validation/development set. Details of RNN hidden state sizes, Adam optimizer, mixing ratios, etc. are provided in Appendix A.2.

Models	R-1	R-2	R-L
PREVIOUS WORK			
ABS+ (Rush et al., 2015)	29.76	11.88	26.96
RAS-El (Chopra et al., 2016)	33.78	15.97	31.15
lvt2k (Nallapati et al., 2016)	32.67	15.59	30.64
Pasunuru et al. (2017)	32.75	15.35	30.82
OUR MODELS			
2-Layer Pointer Baseline $\otimes$	34.26	16.40	32.03
$\otimes$ + Entailment Generation	35.45	17.16	33.19
$\otimes$ + Question Generation	35.48	17.31	32.97
$\otimes$ + Entailment + Question	35.98	17.76	33.63

Table 4.2: Summarization results on Gigaword. ROUGE scores are full length F-1. All the multi-task improvements are statistically significant over the state-of-the-art baseline.

4.5 Experimental Results

4.5.1 Summarization (Primary Task) Results

Pointer+Coverage Baseline We start from the strong model of See et al. (2017).[2] Table 4.1 shows that our baseline model performs better than or comparable to See et al. (2017).[3] On Gigaword dataset, our baseline model (with pointer only, since coverage not needed for this single-sentence summarization task) performs better than all previous works, as shown in Table 4.2.

Multi-Task with Entailment Generation We first perform multi-task learning between abstractive summarization and entailment generation with soft-sharing of parameters as discussed in Sec. 4.3. Table 4.1 and Table 4.2 shows that this multi-task setting is better than our strong baseline models and the improvements are statistically significant on all metrics[4] on both CNN/DailyMail ($p < 0.01$ in ROUGE-1/ROUGE-L/METEOR and $p < 0.05$ in ROUGE-2) and Gigaword ($p < 0.01$ on all metrics) datasets, showing that entailment generation task is inducing useful inference skills to the summarization task (also see analysis examples in Sec. 4.6).

[2]We use two layers so as to allow our high- versus low-level layer sharing intuition. Note that this does not increase the parameter size much (23M versus 22M for See et al. (2017)).

[3]As mentioned in the github for See et al. (2017), their publicly released pretrained model produces the lower scores that we represent by † in Table 4.1.

[4]Statistical significance is computed via bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples.

Models	Relevance	Readability	Total
MTL VS. BASELINE			
MTL wins	43	40	83
Baseline wins	22	24	46
Non-distinguish.	35	36	71
MTL VS. SEE ET AL. (2017)			
MTL wins	39	33	72
See et al. (2017) wins	29	38	67
Non-distinguish.	32	29	61

Table 4.3: CNN/DM Human Evaluation: pairwise comparison between our 3-way multi-task (MTL) model w.r.t. our baseline and See et al. (2017).

Models	Relevance	Readability	Total
MTL wins	33	32	65
Baseline wins	22	22	44
Non-distinguish.	45	46	91

Table 4.4: Gigaword Human Evaluation: pairwise comparison between our 3-way multi-task (MTL) model w.r.t. our baseline.

Multi-Task with Question Generation For multi-task learning with question generation, the improvements are statistically significant in ROUGE-1 ($p < 0.01$), ROUGE-L ($p < 0.05$), and METEOR ($p < 0.01$) for CNN/DailyMail and in all metrics ($p < 0.01$) for Gigaword, compared to the respective baseline models. Also, Sec. 4.6 presents quantitative and qualitative analysis of this model's improved saliency.[5]

Multi-Task with Entailment and Question Generation Finally, we perform multi-task learning with all three tasks together, achieving the best of both worlds (inference skills and saliency). Table 4.1 and Table 4.2 show that our full multi-task model achieves the best scores on CNN/DailyMail and Gigaword datasets, and the improvements are statistically significant on all metrics on both CNN/DailyMail ($p < 0.01$ in ROUGE-1/ROUGE-L/METEOR and $p < 0.02$ in ROUGE-2) and Gigaword ($p < 0.01$ on all metrics). Finally, our 3-way multi-task model (with both en-

[5]In order to verify that our improvements were from the auxiliary tasks' specific character/capabilities and not just due to adding more data, we separately trained word embeddings on each auxiliary dataset (i.e., SNLI and SQuAD) and incorporated them into the summarization model. We found that both our 2-way multi-task models perform significantly better than these models using the auxiliary word-embeddings, suggesting that merely adding more data is not enough.

Models	R-1	R-2	R-L
See et al. (2017)	34.30	14.25	30.82
Baseline	35.96	15.91	32.92
Multi-Task (EG + QG)	36.73	16.15	33.58

Table 4.5: ROUGE F1 scores on DUC-2002.

tailment and question generation) outperforms the publicly-available pretrained result (†) of the previous SotA (See et al., 2017) with stat. significance ($p < 0.01$), as well the higher-reported results ($\star$) on ROUGE-1/ROUGE-2 ($p < 0.01$).

4.5.2 Human Evaluation

We also conducted a blind human evaluation on Amazon MTurk for relevance and readability, based on 100 samples, for both CNN/DailyMail and Gigaword (see instructions in Sec. 4.4). Table. 4.3 shows the CNN/DM results where we do pairwise comparison between our 3-way multi-task model's output summaries w.r.t. our baseline summaries and w.r.t. See et al. (2017) summaries. As shown, our 3-way multi-task model achieves both higher relevance and higher readability scores w.r.t. the baseline. W.r.t. See et al. (2017), our MTL model is higher in relevance scores but a bit lower in readability scores (and is higher in terms of total aggregate scores). One potential reason for this lower readability score is that our entailment generation auxiliary task encourages our summarization model to rewrite more and to be more abstractive than See et al. (2017) – see abstractiveness results in Table 4.11.

We also show human evaluation results on the Gigaword dataset in Table 4.4 (again based on pairwise comparisons for 100 samples), where we see that our MTL model is better than our state-of-the-art baseline on both relevance and readability.[6]

[6]Note that we did not have output files of any previous work's model on Gigaword; however, our baseline is already a strong state-of-the-art model as shown in Table 4.2.

Models	M	C	R	B
Pasunuru and Bansal (2017a)	29.6	117.8	62.4	40.6
Our 1-layer pointer EG	32.4	139.3	65.1	43.6
Our 2-layer pointer EG	32.3	140.0	64.4	43.7

Table 4.6: Performance of our pointer-based entailment generation (EG) models compared with previous SotA work. M, C, R, B are short for Meteor, CIDEr-D, ROUGE-L, and BLEU-4, resp.

Models	M	C	R	B
Du et al. (2017)	15.2	-	38.0	10.8
Our 1-layer pointer QG	15.4	75.3	36.2	9.2
Our 2-layer pointer QG	17.5	95.3	40.1	13.8

Table 4.7: Performance of our pointer-based question generation (QG) model w.r.t. previous work.

4.5.3 Generalizability Results (DUC-2002)

Next, we also tested our model's generalizability/transfer skills, where we take the models trained on CNN/DailyMail and directly test them on DUC-2002. We take our baseline and 3-way multi-task models, plus the pointer-coverage model from See et al. (2017).[7] We only re-tune the beam-size for each of these three models separately (based on DUC-2003 as the validation set).[8] As shown in Table 4.5, our multi-task model achieves statistically significant improvements over the strong baseline ($p < 0.01$ in ROUGE-1 and ROUGE-L) and the pointer-coverage model from See et al. (2017) ($p < 0.01$ in all metrics). This demonstrates that our model is able to generalize well and that the auxiliary knowledge helps more in low-resource scenarios.

4.5.4 Auxiliary Task Results

In this section, we discuss the individual/separated performance of our auxiliary tasks.

[7]We use the publicly-available pretrained model from See et al. (2017)'s github for these DUC transfer results, which produces the † results in Table 4.1. All other comparisons and analysis in this chapter are based on their higher $\star$ results.

[8]We follow previous work which has shown that larger beam values are better and feasible for DUC corpora. However, our MTL model still achieves stat. significant improvements ($p < 0.01$ in all metrics) over See et al. (2017) without beam retuning (i.e., with beam $= 4$).

Models	R-1	R-2	R-L	M
Final Model	**39.81**	**17.64**	**36.54**	**18.54**
SOFT-VS.-HARD SHARING				
Hard-sharing	39.51	17.44	36.33	18.21
LAYER SHARING METHODS				
D1+D2	39.62	17.49	36.44	18.34
E1+D2	39.51	17.51	36.37	18.15
E1+Attn+D2	39.32	17.36	36.11	17.88

Table 4.8: Ablation studies comparing our final multi-task model with hard-sharing and different alternative layer-sharing methods. Here E1, E2, D1, D2, Attn refer to parameters of the first/second layer of encoder/decoder, and attention parameters. Improvements of final model upon ablation experiments are all stat. signif. with $p < 0.05$.

Models	Average Entailment Probability
Baseline	0.907
Multi-Task (EG)	0.912

Table 4.9: Entailment classification results of our baseline vs. EG-multi-task model ($p < 0.001$).

Entailment Generation We use the same architecture as described in Sec. 4.2.1 with pointer mechanism, and Table 4.6 compares our model's performance to Pasunuru and Bansal (2017a). Our pointer mechanism gives a performance boost, since the entailment generation task involves copying from the given premise sentence, whereas the 2-layer model seems comparable to the 1-layer model.

Question Generation Again, we use same architecture as described in Sec. 4.2.1 along with pointer mechanism for the task of question generation. Table 4.7 compares the performance of our model w.r.t. the state-of-the-art Du et al. (2017).

4.6 Ablations and Insights

Soft-sharing vs. Hard-sharing As described in Sec. 4.3.2, we choose soft-sharing over hard-sharing because of the more expressive parameter sharing it provides to the model. Empirical

Models	Average Match Rate
Baseline	27.75 %
Multi-Task (QG)	28.06 %

Table 4.10: Saliency classification results of our baseline vs. QG-multi-task model ($p < 0.01$).

results in Table. 4.8 prove that soft-sharing method is statistically significantly better than hard-sharing with $p < 0.001$ in all metrics.[9]

Comparison of Different Layer-Sharing Methods We also conducted ablation studies among various layer-sharing approaches. Table 4.8 shows results for soft-sharing models with decoder-only sharing (D1+D2; similar to Pasunuru et al. (2017)) as well as lower-layer sharing (encoder layer 1 + decoder layer 2, with and without attention shared). As shown, our final model (high-level semantic layer sharing E2+Attn+D1) outperforms these alternate sharing methods in all metrics with statistical significance ($p < 0.05$).[10]

Quantitative Improvements in Entailment We employ a state-of-the-art entailment classifier (Chen et al., 2017), and calculate the average of the entailment probability of each of the output summary's sentences being entailed by the input source document. We do this for output summaries of our baseline and 2-way-EG multi-task model (with entailment generation). As can be seen in Table 4.9, our multi-task model improves upon the baseline in the aspect of being entailed by the source document (with statistical significance $p < 0.001$). Further, we use the Named Entity Recognition (NER) module from CoreNLP (Manning et al., 2014) to compute the number of times the output summary contains extraneous facts (i.e., named entities as detected by the NER system) that are not present in the source documents, based on the intuition that a well-entailed summary should not contain unrelated information not followed from the input premise. We found that our 2-way MTL model with entailment generation reduces this extrane-

[9]In the interest of space, most of the analyses are shown for CNN/DailyMail experiments, but we observed similar trends for the Gigaword experiments as well.

[10]Note that all our soft and layer sharing decisions were strictly made on the dev/validation set (see Sec. 4.4).

Models	2-gram	3-gram	4-gram
See et al. (2017)	2.24	6.03	9.72
MTL (3-way)	2.84	6.83	10.66

Table 4.11: Abstractiveness: novel n-gram percent.

ous count by 17.2% w.r.t. the baseline. The qualitative examples below further discuss this issue of generating unrelated information.

Quantitative Improvements in Saliency Detection For our saliency evaluation, we used the answer-span prediction classifier from Pasunuru and Bansal (2018) trained on SQuAD (Rajpurkar et al., 2016) as the keyword detection classifier. We then annotate the ground-truth and model summaries with this keyword classifier and compute the % match, i.e., how many salient words from the ground-truth summary were also generated in the model summary. The results are shown in Table 4.10, where the 2-way-QG MTL model (with question generation) versus baseline improvement is stat. significant ($p < 0.01$). Moreover, we found 93 more cases where our 2-way-QG MTL model detects 2 or more additional salient keywords than the pointer baseline model (as opposed to vice versa), showing that sentence-level question generation task is helping the document-level summarization task in finding more salient terms.

Qualitative Examples on Entailment and Saliency Improvements Fig. 4.2 presents an example of output summaries generated by See et al. (2017), our baseline, and our 3-way multi-task model. See et al. (2017) and our baseline models generate phrases like “john hartson” and “hampden injustice” that don’t appear in the input document, hence they are not entailed by the input.[11] Moreover, both models missed salient information like “josh meekings”, “leigh griffiths”, and “hoops”, that our multi-task model recovers.[12] Hence, our 3-way multi-task model generates summaries that are both better at logical entailment and contain more salient information.

[11]These extra, non-entailed unrelated/contradictory information are not present at all in any paraphrase form in the input document.

[12]We consider the fill-in-the-blank highlights annotated by human on CNN/DailyMail dataset as salient information.

Abstractiveness Analysis As suggested in See et al. (2017), we also compute the abstractiveness score as the number of novel n-grams between the model output summary and source document. As shown in Table 4.11, our multi-task model (EG + QG) is more abstractive than See et al. (2017).

4.7 Conclusion

We presented a multi-task learning approach to improve abstractive summarization by incorporating the ability to detect salient information and to be logically entailed by the document, via question generation and entailment generation auxiliary tasks. We propose effective soft and high-level (semantic) layer-specific parameter sharing and achieve significant improvements over the state-of-the-art on two popular datasets, as well as a generalizability/transfer DUC-2002 setup.

Input Document: *celtic have written to the scottish football association in order to gain an ‘ understanding óf the refereeing decisions during their scottish cup semi-final defeat by inverness on sunday . the hoops were left outraged by referee steven mclean ś failure to award a penalty or red card for a clear handball in the box by josh meekings to deny leigh griffith ś goal-bound shot during the first-half . caley thistle went on to win the game 3-2 after extra-time and denied rory delia ś men the chance to secure a domestic treble this season . celtic striker leigh griffiths has a goal-bound shot blocked by the outstretched arm of josh meekings . celtic ś adam matthews -lrb- right -rrb- slides in with a strong challenge on nick ross in the scottish cup semi-final . ‘ given the level of reaction from our supporters and across football , we are duty bound to seek an understanding of what actually happened , ćeltic said in a statement . they added , ‘ we have not been given any other specific explanation so far and this is simply to understand the circumstances of what went on and why such an obvious error was made . h́owever , the parkhead outfit made a point of congratulating their opponents , who have reached the first-ever scottish cup final in their history , describing caley as a ‘ fantastic club ánd saying ‘ reaching the final is a great achievement . ćeltic had taken the lead in the semi-final through defender virgil van dijk ś curling free-kick on 18 minutes , but were unable to double that lead thanks to the meekings controversy . it allowed inverness a route back into the game and celtic had goalkeeper craig gordon sent off after the restart for scything down marley watkins in the area . greg tansey duly converted the resulting penalty . edward ofere then put caley thistle ahead , only for john guidetti to draw level for the bhoys . with the game seemingly heading for penalties , david raven scored the winner on 117 minutes , breaking thousands of celtic hearts . celtic captain scott brown -lrb- left -rrb- protests to referee steven mclean but the handball goes unpunished . griffiths shows off his acrobatic skills during celtic ś eventual surprise defeat by inverness . celtic pair aleksandar tonev -lrb- left -rrb- and john guidetti look dejected as their hopes of a domestic treble end .*

Ground-truth: *celtic were defeated 3-2 after extra-time in the scottish cup semi-final . leigh griffiths had a goal-bound shot blocked by a clear handball. however, no action was taken against offender josh meekings . the hoops have written the sfa for an ’understanding’ of the decision .*

See et al. (2017): *john hartson was once on the end of a major hampden injustice while playing for celtic . but he can not see any point in his old club writing to the scottish football association over the latest controversy at the national stadium . hartson had a goal wrongly disallowed for offside while celtic were leading 1-0 at the time but went on to lose 3-2 .*

Our Baseline: *john hartson scored the late winner in 3-2 win against celtic . celtic were leading 1-0 at the time but went on to lose 3-2 . some fans have questioned how referee steven mclean and additional assistant alan muir could have missed the infringement .*

Multi-task: *celtic have written to the scottish football association in order to gain an ‘ understanding ’ of the refereeing decisions . the hoops were left outraged by referee steven mclean ’s failure to award a penalty or red card for a clear handball in the box by josh meekings . celtic striker leigh griffiths has a goal-bound shot blocked by the outstretched arm of josh meekings .*

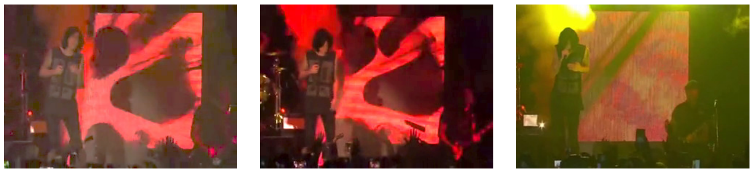
Figure 4.2: Example summary from our 3-way MTL model. The boxed-red highlights are extraneously-generated words not present/paraphrased in the input document. The unboxed-green highlights show salient phrases.

CHAPTER 5: REINFORCEMENT LEARNING FOR VIDEO CAPTIONING

5.1 Introduction

The task of video captioning (Fig. 5.1) is an important next step to image captioning, with additional modeling of temporal knowledge and action sequences, and has several applications in online content search, assisting the visually-impaired, etc. Advancements in neural sequence-to-sequence learning has shown promising improvements on this task, based on encoder-decoder, attention, and hierarchical models (Venugopalan et al., 2015a; Pan et al., 2016a). However, these models are still trained using a word-level cross-entropy loss, which does not correlate well with the sentence-level metrics that the task is finally evaluated on (e.g., CIDEr, BLEU). Moreover, these models suffer from exposure bias (Ranzato et al., 2016), which occurs when a model is only exposed to the training data distribution, instead of its own predictions. First, using a sequence-level training, policy gradient approach (Ranzato et al., 2016), we allow video captioning models to directly optimize these non-differentiable metrics, as rewards in a reinforcement learning paradigm. We also address the exposure bias issue by using a mixed-loss (Paulus et al., 2018; Wu et al., 2016), i.e., combining the cross-entropy and reward-based losses, which also helps maintain output fluency.

Next, we introduce a novel entailment-corrected reward that checks for logically-directed partial matches. Current reinforcement-based text generation works use traditional phrase-matching metrics (e.g., CIDEr, BLEU) as their reward function. However, these metrics use *undirected* n-gram matching of the machine-generated caption with the ground-truth caption, and hence fail to capture its *directed logical correctness*. Therefore, they still give high scores to even those generated captions that contain a single but critical wrong word (e.g., negation, unrelated action or object), because all the other words still match with the ground truth. We introduce CIDEnt,



Ground truth: A band is performing a song.
A rock concert performance by a band.
Baseline-XE: A person is playing a video game.
CIDEr-RL: A band is playing a video game.
CIDEnt-RL: A band is performing a song.

Figure 5.1: A correctly-predicted video caption generated by our CIDEnt-reward model.

which penalizes the phrase-matching metric (CIDEr) based reward, when the entailment score is low. This ensures that a generated caption gets a high reward only when it is a directed match with (i.e., it is logically implied by) the ground truth caption, hence avoiding contradictory or unrelated information (e.g., see Fig. 5.1). Empirically, we show that first the CIDEr-reward model achieves significant improvements over the cross-entropy baseline (on multiple datasets, and automatic and human evaluation); next, the CIDEnt-reward model further achieves significant improvements over the CIDEr-based reward. Overall, we achieve the new state-of-the-art on the MSR-VTT dataset.

5.2 Models

Attention Baseline (Cross-Entropy) Our attention-based seq-to-seq baseline model is similar to the Bahdanau et al. (2015) architecture, where we encode input frame level video features $\{f_{1:n}\}$ via a bi-directional LSTM-RNN and then generate the caption $w_{1:m}$ using an LSTM-RNN with an attention mechanism. Let θ be the model parameters and $w^*_{1:m}$ be the ground-truth caption, then the cross entropy loss function is:

$$L(\theta) = -\sum_{t=1}^{m} \log p(w^*_t | w^*_{1:t-1}, f_{1:n}) \tag{5.1}$$

where $p(w_t|w_{1:t-1}, f_{1:n}) = softmax(W^T h_t^d)$, W^T is the projection matrix, and w_t and h_t^d are the generated word and the RNN decoder hidden state at time step t, computed using the standard RNN recursion and attention-based context vector c_t.

Reinforcement Learning (Policy Gradient) In order to directly optimize the sentence-level test metrics (as opposed to the cross-entropy loss above), we use a policy gradient p_θ, where θ represent the model parameters. Here, our baseline model acts as an agent and interacts with its environment (video and caption). At each time step, the agent generates a word (action), and the generation of the end-of-sequence token results in a reward r to the agent. Our training objective is to minimize the negative expected reward function:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)] \tag{5.2}$$

where w^s is the word sequence sampled from the model. Based on the REINFORCE algorithm (Williams, 1992), the gradients of this non-differentiable, reward-based loss function are:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s) \cdot \nabla_\theta \log p_\theta(w^s)] \tag{5.3}$$

We follow Ranzato et al. (2016) approximating the above gradients via a single sampled word sequence. We also use a variance-reducing bias (baseline) estimator in the reward function.

Mixed Loss During reinforcement learning, optimizing for only the reinforcement loss (with automatic metrics as rewards) doesn't ensure the readability and fluency of the generated caption, and there is also a chance of gaming the metrics without actually improving the quality of the output (Liu et al., 2016a). Hence, for training our reinforcement based policy gradients, we use a mixed loss function, which is a weighted combination of the cross-entropy loss (XE) and the reinforcement learning loss (RL), similar to the previous work (Paulus et al., 2018; Wu et al., 2016). This mixed loss improves results on the metric used as reward through the reinforcement loss (and improves relevance based on our entailment-enhanced rewards) but also ensures better read-

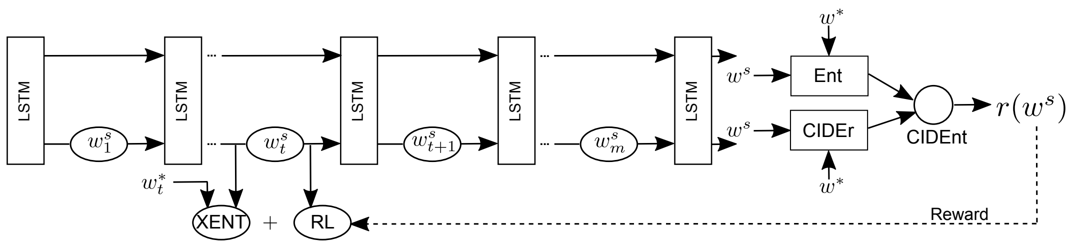


Figure 5.2: Reinforced (mixed-loss) video captioning using entailment-corrected CIDEr score as reward.

ability and fluency due to the cross-entropy loss (in which the training objective is a conditioned language model, learning to produce fluent captions). Our mixed loss is defined as:

$$L_{\text{MIXED}} = (1-\gamma)L_{\text{XE}} + \gamma L_{\text{RL}} \tag{5.4}$$

where γ is a tuning parameter used to balance the two losses. For annealing and faster convergence, we start with the optimized cross-entropy loss baseline model, and then move to optimizing the above mixed loss function.[1]

5.3 Reward Functions

Caption Metric Reward Previous image captioning papers have used traditional captioning metrics such as CIDEr, BLEU, or METEOR as reward functions, based on the match between the generated caption sample and the ground-truth reference(s). First, it has been shown by Vedantam et al. (2015) that CIDEr, based on a consensus measure across several human reference captions, has a higher correlation with human evaluation than other metrics such as METEOR, ROUGE, and BLEU. They further showed that CIDEr gets better with more number of human references (and this is a good fit for our video captioning datasets, which have 20-40 human references per video).

[1]We also experimented with the curriculum learning 'MIXER' strategy of Ranzato et al. (2016), where the XE+RL annealing is based on the decoder time-steps; however, the mixed loss function strategy (described above) performed better in terms of maintaining output caption fluency.

Ground-truth caption	Generated (sampled) caption	CIDEr	Ent
a man is spreading some butter in a pan	puppies is melting butter on the pan	140.5	0.07
a panda is eating some bamboo	a panda is eating some fried	256.8	0.14
a monkey pulls a dogs tail	a monkey pulls a woman	116.4	0.04
a man is cutting the meat	a man is cutting meat into potato	114.3	0.08
the dog is jumping in the snow	a dog is jumping in cucumbers	126.2	0.03
a man and a woman is swimming in the pool	a man and a whale are swimming in a pool	192.5	0.02

Table 5.1: Examples of captions sampled during policy gradient and their CIDEr vs Entailment scores.

More recently, Rennie et al. (2017) further showed that CIDEr as a reward in image captioning outperforms all other metrics as a reward, not just in terms of improvements on CIDEr metric, but also on all other metrics. In line with these above previous works, we also found that CIDEr as a reward ('CIDEr-RL' model) achieves the best metric improvements in our video captioning task, and also has the best human evaluation improvements (see Sec. 5.5.3 for result details, incl. those about other rewards based on BLEU, SPICE).

Entailment Corrected Reward Although CIDEr performs better than other metrics as a reward, all these metrics (including CIDEr) are still based on an undirected n-gram matching score between the generated and ground truth captions. For example, the wrong caption "a man is playing football" w.r.t. the correct caption "a man is playing basketball" still gets a high score, even though these two captions belong to two completely different events. Similar issues hold in case of a negation or a wrong action/object in the generated caption (see examples in Table 5.1).

We address the above issue by using an entailment score to correct the phrase-matching metric (CIDEr or others) when used as a reward, ensuring that the generated caption is logically implied by (i.e., is a paraphrase or directed partial match with) the ground-truth caption. To achieve an accurate entailment score, we adapt the state-of-the-art decomposable-attention model of Parikh et al. (2016) trained on the SNLI corpus (image caption domain). This model gives us a probability for whether the sampled video caption (generated by our model) is entailed by the ground truth caption as premise (as opposed to a contradiction or neutral case).[2] Similar to the traditional metrics, the overall 'Ent' score is the maximum over the entailment scores for a generated caption w.r.t. each reference human caption (around 20/40 per MSR-VTT/YouTube2Text video). CIDEnt is defined as:

$$\text{CIDEnt} = \begin{cases} \text{CIDEr} - \lambda, & \text{if} \quad \text{Ent} < \beta \\ \text{CIDEr}, & \text{otherwise} \end{cases} \tag{5.5}$$

which means that if the entailment score is very low, we penalize the metric reward score by decreasing it by a penalty λ. This agreement-based formulation ensures that we only trust the CIDEr-based reward in cases when the entailment score is also high. Using CIDEr$-\lambda$ also ensures the smoothness of the reward w.r.t. the original CIDEr function (as opposed to clipping the reward to a constant). Here, λ and β are hyperparameters that can be tuned on the dev-set; on light tuning, we found the best values to be intuitive: $\lambda =$ roughly the baseline (cross-entropy) model's score on that metric (e.g., 0.45 for CIDEr on MSR-VTT dataset); and $\beta = 0.33$ (i.e., the 3-class entailment classifier chose contradiction or neutral label for this pair). Table 5.1 shows some examples of sampled generated captions during our model training, where CIDEr was misleadingly high for incorrect captions, but the low entailment score (probability) helps us successfully identify these cases and penalize the reward.

5.4 Experimental Setup

Datasets We use 2 datasets: MSR-VTT (Xu et al., 2016a) has $10,000$ videos, 20 references/video; and YouTube2Text/MSVD (Chen and Dolan, 2011) has 1970 videos, 40 references/video. Standard splits and other details are in Appendix A.3.

[2]Our entailment classifier based on Parikh et al. (2016) is 92% accurate on entailment in the caption domain, hence serving as a highly accurate reward score. For other domains in future tasks such as new summarization, we plan to use the new multi-domain dataset by Williams et al. (2018b).

Models	BLEU-4	METEOR	ROUGE-L	CIDEr-D	CIDEnt	Human*
			PREVIOUS WORK			
Venugopalan et al. (2015b)*	32.3	23.4	-	-	-	-
Yao et al. (2015)*	35.2	25.2	-	-	-	-
Xu et al. (2016a)	36.6	25.9	-	-	-	-
Pasunuru and Bansal (2017a)	**40.8**	**28.8**	60.2	47.1	-	-
Rank1: v2t_navigator	**40.8**	28.2	60.9	44.8	-	-
Rank2: Aalto	39.8	26.9	59.8	45.7	-	-
Rank3: VideoLAB	39.1	27.7	60.6	44.1	-	-
			OUR MODELS			
Cross-Entropy (Baseline-XE)	38.6	27.7	59.5	44.6	34.4	-
CIDEr-RL	39.1	28.2	60.9	51.0	37.4	11.6
CIDEnt-RL (**New Rank1**)	**40.5**	**28.4**	**61.4**	**51.7**	**44.0**	**18.4**

Table 5.2: Our primary video captioning results on MSR-VTT. All CIDEr-RL results are statistically significant over the baseline XE results, and all CIDEnt-RL results are stat. signif. over the CIDEr-RL results. Human* refers to the 'pairwise' comparison of human relevance evaluation between CIDEr-RL and CIDEnt-RL models (see full human evaluations of the 3 models in Table 5.3 and Table 5.4).

Automatic Evaluation We use several standard automated evaluation metrics: METEOR, BLEU-4, CIDEr-D, and ROUGE-L (from MS-COCO evaluation server (Chen et al., 2015)).

Human Evaluation We also present human evaluation for comparison of baseline-XE, CIDEr-RL, and CIDEnt-RL models, esp. because the automatic metrics cannot be trusted solely. Relevance measures how related is the generated caption w.r.t, to the video content, whereas coherence measures readability of the generated caption.

Training Details All the hyperparameters are tuned on the validation set. All our results (including baseline) are based on a 5-avg-ensemble. See Appendix A.3 for extra training details, e.g., about the optimizer, learning rate, RNN size, Mixed-loss, and CIDEnt hyperparameters.

5.5 Experimental Results

5.5.1 Primary Results

Table 5.2 shows our primary results on the popular MSR-VTT dataset. First, our baseline attention model trained on cross entropy loss ('Baseline-XE') achieves strong results w.r.t. the

	Relevance	Coherence
Not Distinguishable	64.8%	92.8%
Baseline-XE Wins	13.6%	4.0%
CIDEr-RL Wins	**21.6%**	3.2%

Table 5.3: Human eval: Baseline-XE vs CIDEr-RL.

previous state-of-the-art methods.[3] Next, our policy gradient based mixed-loss RL model with reward as CIDEr ('CIDEr-RL') improves significantly[4] over the baseline on all metrics, and not just the CIDEr metric. It also achieves statistically significant improvements in terms of human relevance evaluation (see below). Finally, the last row in Table 5.2 shows results for our novel CIDEnt-reward RL model ('CIDEnt-RL'). This model achieves statistically significant[5] improvements on top of the strong CIDEr-RL model, on all automatic metrics (as well as human evaluation). Note that in Table 5.2, we also report the CIDEnt reward scores, and the CIDEnt-RL model strongly outperforms CIDEr and baseline models on this entailment-corrected measure. Overall, we are also the new Rank1 on the MSR-VTT leaderboard, based on their ranking criteria.

Human Evaluation We also perform small human evaluation studies (250 samples from the MSR-VTT test set output) to compare our 3 models pairwise.[6] As shown in Table 5.3 and Table 5.4, in terms of relevance, first our CIDEr-RL model stat. significantly outperforms the baseline XE model ($p < 0.02$); next, our CIDEnt-RL model significantly outperforms the CIDEr-RL model ($p < 0.03$). The models are statistically equal on coherence in both comparisons.

[3]We list previous works' results as reported by the MSR-VTT dataset paper itself, as well as their 3 leaderboard winners (`http://ms-multimedia-challenge.com/leaderboard`), plus the 10-ensemble video+entailment generation multi-task model of Pasunuru and Bansal (2017a).

[4]Statistical significance of $p < 0.01$ for CIDEr, METEOR, and ROUGE, and $p < 0.05$ for BLEU, based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994).

[5]Statistical significance of $p < 0.01$ for CIDEr, BLEU, ROUGE, and CIDEnt, and $p < 0.05$ for METEOR.

[6]We randomly shuffle pairs to anonymize model identity and the human evaluator then chooses the better caption based on relevance and coherence (see Sec. 5.4). 'Not Distinguishable' are cases where the annotator found both captions to be equally good or equally bad).

	Relevance	Coherence
Not Distinguishable	70.0%	94.6%
CIDEr-RL Wins	11.6%	2.8%
CIDEnt-RL Wins	**18.4%**	2.8%

Table 5.4: Human eval: CIDEr-RL vs CIDEnt-RL.

Models	B	M	R	C	CE	H*
Baseline-XE	52.4	35.0	71.6	83.9	68.1	-
CIDEr-RL	53.3	35.1	72.2	89.4	69.4	8.4
CIDEnt-RL	54.4	34.9	72.2	88.6	71.6	13.6

Table 5.5: Results on YouTube2Text (MSVD) dataset. CE = CIDEnt score. H* refer to the pairwise human comparison of relevance.

5.5.2 Other Datasets

We also tried our CIDEr and CIDEnt reward models on the YouTube2Text dataset. In Table 5.5, we first see strong improvements from our CIDEr-RL model on top of the cross-entropy baseline. Next, the CIDEnt-RL model also shows some improvements over the CIDEr-RL model, e.g., on BLEU and the new entailment-corrected CIDEnt score. It also achieves significant improvements on human relevance evaluation (250 samples).[7]

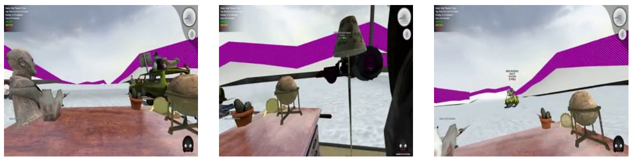
5.5.3 Other Metrics as Reward

As discussed in Sec. 5.3, CIDEr is the most promising metric to use as a reward for captioning, based on both previous work's findings as well as ours. We did investigate the use of other metrics as the reward. When using BLEU as a reward (on MSR-VTT), we found that this BLEU-RL model achieves BLEU-metric improvements, but was worse than the cross-entropy baseline on human evaluation. Similarly, a BLEUEnt-RL model achieves BLEU and BLEUEnt metric improvements, but is again worse on human evaluation. We also experimented with the new

[7]This dataset has a very small dev-set, causing tuning issues – we plan to use a better train/dev re-split in future work.

Ground truth: A man is playing a violin.
A man is playing the violin on stage.
Baseline-XE: A man is playing the drums.
CIDEr-RL: A man is playing a guitar.
CIDEnt-RL: A man is playing a violin.

Ground truth: Two men are wrestling.
Two guys are wrestling in a competition.
Baseline-XE: A group of people are playing a game.
CIDEr-RL: A man is playing a wrestling.
CIDEnt-RL: Two men are wrestling.



Ground truth: A person is playing a video game.
Someone is playing video game.
Baseline-XE: A man is riding a motorcycle.
CIDEr-RL: A man is talking about a plane.
CIDEnt-RL: A person is playing a video game.

Figure 5.3: Output examples where our CIDEnt-RL model produces better entailed captions than the phrase-matching CIDEr-RL model, which in turn is better than the baseline cross-entropy model.

SPICE metric (Anderson et al., 2016) as a reward, but this produced long repetitive phrases (as also discussed in Liu et al. (2016b)).

5.5.4 Insights

Fig. 5.1 shows an example where our CIDEnt-reward model correctly generates a ground-truth style caption, whereas the CIDEr-reward model produces a non-entailed caption because this caption will still get a high phrase-matching score. Further, Figure 5.3 shows several examples where our CIDEnt-reward model produces better entailed captions than the ones generated by the CIDEr-reward model. This is because the CIDEr-style captioning metrics achieve a high score even when the generation does not exactly entail the ground truth but is just a high phrase overlap. This can obviously cause issues by inserting a single wrong word such as a negation, contradiction, or wrong action/object. On the other hand, our entailment-enhanced CIDEnt score is only high when both CIDEr and the entailment classifier achieve high scores. The CIDEr-RL model, in turn, produces better captions than the baseline cross-entropy model, which is not aware of sentence-level matching at all.

5.6 Conclusion

We first presented a mixed-loss policy gradient approach for video captioning, allowing for metric-based optimization. We next presented an entailment-corrected CIDEnt reward that further improves results, achieving the new state-of-the-art on MSR-VTT. In the next chapter, we will apply our entailment-corrected rewards to other directed generation tasks such as textual summarization (using the new multi-domain NLI corpus (Williams et al., 2018b)).

CHAPTER 6: REINFORCEMENT LEARNING FOR TEXTUAL SUMMARIZATION

6.1 Introduction

Abstractive summarization, the task of generating a natural short summary of a long document, is more challenging than the extractive paradigm, which only involves selection of important sentences or grammatical sub-sentences (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015). Advent of sequence-to-sequence deep neural networks and large human summarization datasets (Hermann et al., 2015; Nallapati et al., 2016) made the abstractive summarization task more feasible and accurate, with recent ideas ranging from copy-pointer mechanism and redundancy coverage, to metric reward based reinforcement learning (Rush et al., 2015; Chopra et al., 2016; Ranzato et al., 2016; Nallapati et al., 2016; See et al., 2017).

A good abstractive summary requires several important properties, e.g., it should choose the most salient information from the input document, be logically entailed by it, and avoid redundancy. Coverage-based models address the latter redundancy issue (Suzuki and Nagata, 2016; Nallapati et al., 2016; See et al., 2017), but there is still a lot of scope to teach current state-of-the-art models about saliency and logical entailment. Towards this goal, we improve the task of abstractive summarization via a reinforcement learning approach with the introduction of two novel rewards: 'ROUGESal' and 'Entail', and also demonstrate that these saliency and entailment skills allow for better generalizability and transfer.

Our ROUGESal reward gives higher weight to the important, salient words in the summary, in contrast to the traditional ROUGE metric which gives equal weight to all tokens. These weights are obtained from a novel saliency scorer, which is trained on a reading comprehension dataset's answer spans to give a saliency-based probability score to every token in the sentence.

Our Entail reward gives higher weight to summaries whose sentences logically follow from the ground-truth summary. Further, we also add a length normalization constraint to our Entail reward, to importantly avoid misleadingly high entailment scores to very short sentences.

Empirically, we show that our new rewards with policy gradient approaches perform significantly better than a cross-entropy based state-of-the-art pointer-coverage baseline. We show further performance improvements by combining these rewards via our novel multi-reward optimization approach, where we optimize multiple rewards simultaneously in alternate mini-batches (hence avoiding complex scaling and weighting issues in reward combination), inspired from how humans take multiple concurrent types of rewards (feedback) to learn a task. Overall, our methods achieve the new state-of-the-art (including human evaluation) on the CNN/Daily Mail dataset as well as strong improvements in a test-only transfer setup on DUC-2002. Lastly, we present several analyses of our model's saliency, entailment, and abstractiveness skills.

6.2 Models

6.2.1 Baseline Sequence-to-Sequence Model

Our abstractive text summarization model is a simple sequence-to-sequence single-layer bidirectional encoder and unidirectional decoder LSTM-RNN, with attention (Bahdanau et al., 2015), pointer-copy, and coverage mechanism – please refer to See et al. (2017) for details.

6.2.2 Policy Gradient Reinforce

Traditional cross-entropy loss optimization for sequence generation has an exposure bias issue and the model is not optimized for the evaluated metrics (Ranzato et al., 2016). Reinforce-based policy gradient approach addresses both of these issues by using its own distribution during training and by directly optimizing the non-differentiable evaluation metrics as rewards. We use the REINFORCE algorithm (Williams, 1992; Zaremba and Sutskever, 2015) to learn a policy p_θ defined by the model parameters θ to predict the next action (word) and update its internal

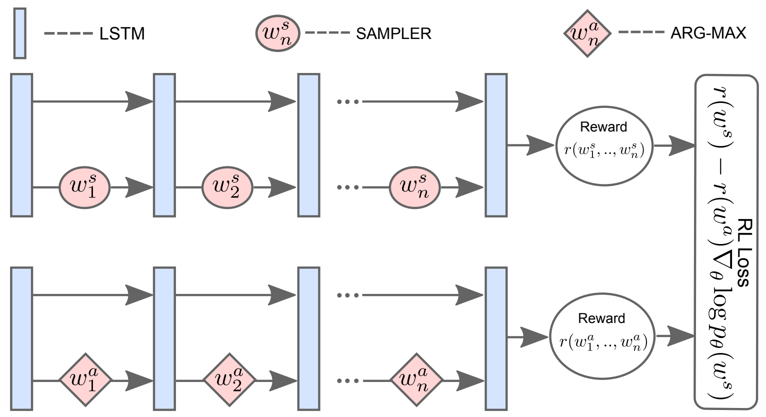


Figure 6.1: Overview of sequence generator with RL training.

(LSTM) states. We minimize the loss function $L_{\text{RL}} = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)]$, where w^s is the sequence of sampled words with w_t^s sampled at time step t of the decoder. The derivative of this loss function with approximation using a single sample along with variance reduction with a bias estimator is:

$$\nabla_\theta L_{\text{RL}} = -(r(w^s) - b_e)\nabla_\theta \log p_\theta(w^s) \tag{6.1}$$

There are several ways to calculate the baseline estimator; we employ the effective SCST approach (Rennie et al., 2017), as depicted in Fig. 6.1, where $b_e = r(w^a)$, is based on the reward obtained by the current model using the test time inference algorithm, i.e., choosing the arg-max word w_t^a of the final vocabulary distribution at each time step t of the decoder. We use the joint cross-entropy and reinforce loss so as to optimize the non-differentiable evaluation metric as reward while also maintaining the readability of the generated sentence (Wu et al., 2016; Paulus et al., 2018; Pasunuru and Bansal, 2017b), which is defined as $L_{\text{Mixed}} = \gamma L_{\text{RL}} + (1-\gamma)L_{\text{XE}}$, where γ is a tunable hyperparameter.

6.2.3 Multi-Reward Optimization

Optimizing multiple rewards at the same time is important and desired for many language generation tasks. One approach would be to use a weighted combination of these rewards, but this has the issue of finding the complex scaling and weight balance among these reward combinations. To address this issue, we instead introduce a simple multi-reward optimization approach inspired from multi-task learning, where we have different tasks, and all of them share all the model parameters while having their own optimization function (different reward functions in this case). If r_1 and r_2 are two reward functions that we want to optimize simultaneously, then we train the two loss functions of Eqn. 6.2 in alternate mini-batches.

$$\begin{aligned} L_{\mathrm{RL}_1} &= -(r_1(w^s) - r_1(w^a))\nabla_\theta \log p_\theta(w^s) \\ L_{\mathrm{RL}_2} &= -(r_2(w^s) - r_2(w^a))\nabla_\theta \log p_\theta(w^s) \end{aligned} \tag{6.2}$$

6.3 Rewards

ROUGE Reward The first basic reward is based on the primary summarization metric of ROUGE package (Lin, 2004). Similar to Paulus et al. (2018), we found that ROUGE-L metric as a reward works better compared to ROUGE-1 and ROUGE-2 in terms of improving all the metric scores.[1] Since these metrics are based on simple phrase matching/n-gram overlap, they do not focus on important summarization factors such as salient phrase inclusion and directed logical entailment. Addressing these issues, we next introduce two new reward functions.

Saliency Reward ROUGE-based rewards have no knowledge about what information is salient in the summary, and hence we introduce a novel reward function called ‘ROUGESal’ which gives higher weight to the important, salient words/phrases when calculating the ROUGE score (which by default assumes all words are equally weighted). To learn these saliency weights, we train our saliency predictor on sentence and answer spans pairs from the popular SQuAD reading comprehension dataset (Rajpurkar et al., 2016)) (Wikipedia domain), where we treat the human-annotated answer spans (avg. span length 3.2) for important questions as representative salient information in the document. As shown in Fig. 6.2, given a sentence as input, the predictor assigns a saliency probability to every token, using a simple bidirectional encoder with a *softmax*

[1]For the rest of the chapter, we mean ROUGE-L whenever we mention ROUGE-reward models.

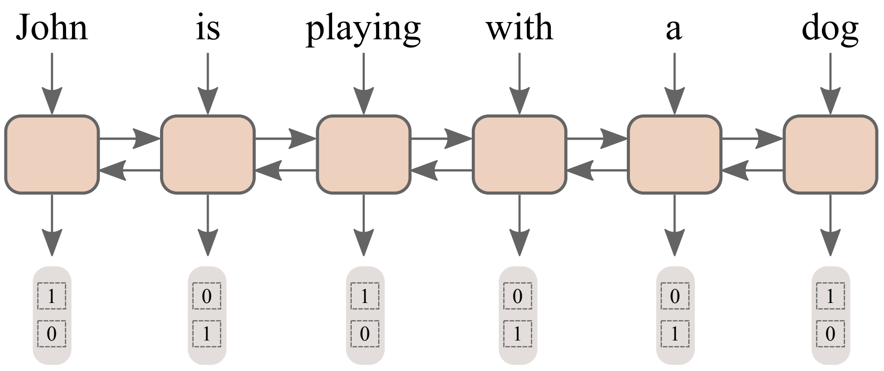


Figure 6.2: Overview of our saliency predictor model.

layer at every time step of the encoder hidden states to classify the token as salient or not. Finally, we use the probabilities given by this saliency prediction model as weights in the ROUGE matching formulation to achieve the final ROUGESal score (see Appendix A.4 for details about our ROUGESal weighted precision, recall, and F-1 formulations).

Entailment Reward A good summary should also be logically entailed by the given source document, i.e., contain no contradictory or unrelated information. Pasunuru and Bansal (2017b) used entailment-corrected phrase-matching metrics (CIDEnt) to improve the task of video captioning; we instead directly use the entailment knowledge from an entailment scorer and its multi-sentence, length-normalized extension as our 'Entail' reward, to improve the task of abstractive text summarization. We train the entailment classifier (Parikh et al., 2016) on the SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2018b) datasets and calculate the entailment probability score between the ground-truth (GT) summary (as premise) and each sentence of the generated summary (as hypothesis), and use avg. score as our Entail reward.[2] Finally, we add a length normalization constraint to avoid very short sentences achieving misleadingly high

[2]Since the GT summary is correctly entailed by the source document, we directly (by transitivity) use this GT as premise for easier (shorter) encoding. We also tried using the full input document as premise but this didn't perform as well (most likely because the entailment classifiers are not trained on such long premises; and the problem with the sentence-to-sentence avg. scoring approach is discussed below).
We also tried summary-to-summary entailment scoring (similar to ROUGE-L) as well as pairwise sentence-to-sentence avg. scoring, but we found that avg. scoring of ground-truth summary (as premise) w.r.t. each generated summary's sentence (as hypothesis) works better (intuitive because each sentence in generated summary might be a compression of multiple sentences of GT summary or source document).

entailment scores:

$$\text{Entail} = \text{Entail} \times \frac{\text{\#tokens in generated summary}}{\text{\#tokens in reference summary}} \tag{6.3}$$

6.4 Experimental Setup

6.4.1 Datasets and Training Details

CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016) is a collection of online news articles and their summaries. We use the non-anonymous version of the dataset as described in See et al. (2017). For test-only generalization experiments, we use the DUC-2002 single document summarization dataset[3]. For entailment reward classifier, we use a combination of the full Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and the recent Multi-NLI corpus (Williams et al., 2018b) training datasets. For our saliency prediction model, we use the Stanford Question Answering (SQuAD) dataset (Rajpurkar et al., 2016).

During training, all our LSTM-RNNs are set with hidden state size of 256. We use a vocabulary size of 50k, where word embeddings are represented in 128 dimension, and both the encoder and decoder share the same embedding for each word. We encode the source document using a 400 time-step unrolled LSTM-RNN and 100 time-step unrolled LSTM-RNN for decoder. We clip the gradients to a maximum gradient norm value of 2.0 and use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1×10^{-3} for pointer baseline and 1×10^{-4} while training along with coverage loss, and 1×10^{-6} for reinforcement learning. Following See et al. (2017), we add coverage mechanism to a converged pointer model. For mixed-loss (XE+RL) optimization, we use the following γ values for various rewards: 0.9985 for ROUGE, 0.9999 for Entail and ROUGE+Entail, and 0.9995 for ROUGESal and ROUGESal+Entail. For reinforcement learning, we only use 5000 training samples ($< 2\%$ of the actual data) to speed up convergence, but we found it to work well in practice. During inference time, we use a beam search of size 4.

[3]`http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html`

6.4.2 Evaluation Metrics

We use the standard ROUGE package (Lin, 2004) and Meteor package (Denkowski and Lavie, 2014a) for reporting the results on all of our summarization models. Following previous work (Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017), we use the ROUGE full-length F1 variant.

Human Evaluation Criteria: We also performed human evaluation of summary *relevance* and *readability*, via Amazon Mechanical Turk (AMT). We selected human annotators that were located in the US, had an approval rate greater than 98%, and had at least $10,000$ approved HITs. For the pairwise model comparisons discussed in Sec. 6.5, we showed the annotators the input article, the ground truth summary, and the two model summaries (randomly shuffled to anonymize model identities) – we then asked them to choose the better among the two model summaries or choose 'Not-Distinguishable' if both summaries are equally good/bad. Instructions for relevance were based on the summary containing salient/important information from the given article, being correct (i.e., avoiding contradictory/unrelated information), and avoiding redundancy. Instructions for readability were based on the summary's fluency, grammaticality, and coherence.

6.5 Experimental Results

Baseline Cross-Entropy Model Results Our abstractive summarization model has attention, pointer-copy, and coverage mechanism. First, we apply cross-entropy optimization and achieve comparable results on CNN/Daily Mail w.r.t. previous work (See et al., 2017).[4]

ROUGE Reward Results First, using ROUGE-L as RL reward (shown as ROUGE in Table 6.1) improves the performance on CNN/Daily Mail in all metrics with stat. significant scores ($p < 0.001$) as compared to the cross-entropy baseline (and also stat. signif. w.r.t. See et al. (2017)).

[4]Our baseline is statistically equal to the paper-reported scores of See et al. (2017) (see Table 6.1) on ROUGE-1, ROUGE-2, based on the bootstrap test (Efron and Tibshirani, 1994). Our baseline is stat. significantly better ($p < 0.001$) in all ROUGE metrics w.r.t. the github scores (R-1: 38.82, R-2: 16.81, R-3: 35.71, M: 18.14) of See et al. (2017).

Models	R-1	R-2	R-L	M
PREVIOUS WORK				
Nallapati et al. (2016)*	35.46	13.30	32.65	-
See et al. (2017)	39.53	17.28	36.38	18.72
Paulus et al. (2018) (XE)*	38.30	14.81	35.49	-
Paulus et al. (2018) (RL)*	39.87	15.82	36.90	-
OUR MODELS				
Baseline (XE)	39.41	17.33	36.07	18.27
ROUGE (RL)	39.99	17.72	36.66	18.93
Entail (RL)	39.53	17.51	36.44	20.15
ROUGESal (RL)	40.36	17.97	37.00	19.84
ROUGE+Ent (RL)	40.37	17.89	37.13	19.94
ROUGESal+Ent (RL)	40.43	18.00	37.10	20.02

Table 6.1: Results on CNN/Daily Mail (non-anonymous). * represents previous work on anonymous version. ‘XE’: cross-entropy loss, ‘RL’: reinforce mixed loss (XE+RL). Columns ‘R’: ROUGE, ‘M’: METEOR.

Similar to Paulus et al. (2018), we use mixed loss function (XE+RL) for all our reinforcement experiments, to ensure good readability of generated summaries.

ROUGESal and Entail Reward Results With our novel ROUGESal reward, we achieve stat. signif. improvements in all metrics w.r.t. the baseline as well as w.r.t. ROUGE-reward results ($p < 0.001$), showing that saliency knowledge is strongly improving the summarization model. For our Entail reward, we achieve stat. signif. improvements in ROUGE-L ($p < 0.001$) w.r.t. baseline and achieve the best METEOR score by a large margin. See Sec. 6.6 for analysis of the saliency/entailment skills learned by our models.

Multi-Reward Results Similar to ROUGESal, Entail is a better reward when combined with the complementary phrase-matching metric information in ROUGE; Table 6.1 shows that the ROUGE+Entail multi-reward combination performs stat. signif. better than ROUGE-reward in ROUGE-1, ROUGE-L, and METEOR ($p < 0.001$), and better than Entail-reward in all ROUGE metrics. Finally, we combined our two rewards ROUGESal+Entail to incorporate both saliency

Models	R-1	R-2	R-L	M
Baseline (XE)	35.50	14.57	32.19	14.36
ROUGE (RL)	35.97	15.45	32.72	14.50
ROUGESal+Ent (RL)	38.95	17.05	35.52	16.47

Table 6.2: ROUGE F1 full length scores of our models on test-only DUC-2002 generalizability setup.

Models	Relevance	Readability	Total
ROUGESal+Ent	55	54	109
See et al. (2017)	34	33	67
Non-distinguish.	11	13	24

Table 6.3: Human Evaluation: pairwise comparison of relevance and readability between our ROUGESal+Entail multi-reward model and See et al. (2017).

and entailment knowledge, and it gives the best results overall ($p < 0.001$ in all metrics w.r.t. both baseline and ROUGE-reward models), setting the new state-of-the-art.[5]

Human Evaluation Table. 6.3 shows the MTurk anonymous human evaluation study (based on 100 samples), where we do pairwise comparison between our ROUGESal+Entail multi-reward's output summaries w.r.t. See et al. (2017) summaries on CNN/Daily Mail (see setup details in Sec. 6.4.2). As shown, our multi-reward model is better on both relevance and readability.

Test-Only Transfer (DUC-2002) Results Finally, we also tested our model's generalizability/-transfer skills, where we take the models trained on CNN/Daily Mail and directly test them on DUC-2002 in a test-only setup. As shown in Table 6.2, our final ROUGESal+Entail multi-reward RL model is statistically significantly better than both the cross-entropy (pointer-generator + coverage) baseline as well as ROUGE reward RL model, in terms of all 4 metrics with a large margin (with $p < 0.001$). This demonstrates that our ROUGESal+Entail model learned better transferable and generalizable skills of saliency and logical entailment.

[5]Our last three rows in Table 6.1 are all stat. signif. better in all metrics with $p < 0.001$ compared to See et al. (2017).

Models	2-gram	3-gram	4-gram
See et al. (2017)	2.24	6.03	9.72
Baseline (XE)	2.23	5.58	8.81
ROUGE (RL)	2.69	6.57	10.23
ROUGESal (RL)	2.37	6.00	9.50
Entail (RL)	2.63	6.56	10.26

Table 6.4: Abstractiveness: novel n-gram percentage.

6.6 Output Analysis

Saliency Analysis We analyzed the output summaries generated by See et al. (2017), and our baseline, ROUGE-reward and ROUGESal-reward models, using our saliency prediction model (Sec. 6.3) as the keyword detection classifier. We annotated the ground-truth and model summaries with this keyword classifier and computed the $\%$ match, i.e., how many salient words from the ground-truth summary were also generated in the model summary[6], and the scores are 27.95%, 28.00%, 28.80%, and 30.86%. We also used the original CNN/Daily Mail Cloze Q&A setup (Hermann et al., 2015) with the fill-in-the-blank answers treated as salient information, and the results are 60.66%, 59.36%, 60.67%, and 64.66% for the four models. Further, we also calculated the ROUGESal scores (based on our reward formulation in Sec. 6.3), and the results are 42.04%, 42.14%, 43.05%, and 46.56% for the four models. All three of these saliency analysis experiments illustrate that our ROUGESal reward model is stat. signif. better in saliency than the See et al. (2017), our baseline, and ROUGE-reward models ($p < 0.001$ for all three experiments).

Entailment Analysis We also analyzed the entailment scores of the generated summaries from See et al. (2017), and our baseline, ROUGE-reward, and Entail-reward models, and the results are 27.33%, 27.21%, 28.23%, and 28.98%.[7] We observe that our Entail-reward model achieves stat. significant entailment scores ($p < 0.001$) w.r.t. all the other three models.

[6]In order to select the keywords for this analysis, we used a 0.2 probability threshold on the saliency classifier (based on the scale of the classifier's distribution).

[7]Based on our ground-truth summary to output summary sentences' average entailment score (see Sec. 6.3); similar trends hold for document-to-summary entailment scores.

Abstractiveness Analysis In order to measure the abstractiveness of our models, we followed the 'novel n-gram counts' approach suggested in See et al. (2017). First, we found that all our reward-based RL models have significantly ($p < 0.01$) more novel n-grams than our cross-entropy baseline (see Table 6.4). Next, the Entail-reward model 'maintains' stat. equal abstractiveness as the ROUGE-reward model, likely because it encourages rewriting to create logical subsets of information, while the ROUGESal-reward model does a bit worse, probably because it focuses on copying more salient information (e.g., names). Compared to previous work (See et al., 2017), our Entail-reward and ROUGE-reward models achieve statistically significant improvement ($p < 0.01$) while ROUGESal is comparable.

6.7 Conclusion

We presented a summarization model trained with novel RL reward functions to improve the saliency and directed logical entailment aspects of a good summary. Further, we introduced the novel and effective multi-reward approach of optimizing multiple rewards simultaneously in alternate mini-batches. We achieve the new state-of-the-art on CNN/Daily Mail and also strong test-only improvements on a DUC-2002 transfer setup. In the next chapter, we will introduce a better way of optimizing multiple reward using multi-armed bandits.

CHAPTER 7: AUTOMATION METHODS FOR MTL AND RL WITH BANDITS

In previous chapters (Chapters 3, 4, 5, 6), we presented various multi-task learning and reinforcement learning based approaches to share knowledge across tasks. While these approaches are appealing, they need manual tuning of in what static proportions the auxiliary tasks (in MTL) or rewards (in RL) have to be mixed during the training. Further, this tuning can be computationally expensive with the increase in the number of tasks (or rewards) at hand. Addressing this issue, in this chapter, we introduce 'dynamic' mixing of auxiliary tasks (in MTL) or rewards (in RL) in an automatic way using multi-armed bandits. First, we briefly discuss multi-armed bandits, and then discuss its application to automate MTL and multi-reward RL optimization.

7.1 Multi-Armed Bandits (MAB)

Many control problems can be cast as multi-armed bandit problems, where the goal is to select a sequence of arms/actions in order to optimize certain objective (e.g., expected future payoff) (Bubeck et al., 2012). One widely studied problem in the multi-armed bandit literature is finding the optimal trade-off between exploration and exploitation (Audibert et al., 2009; Macready and Wolpert, 1998; Auer et al., 2002a; Kveton et al., 2019; Bubeck et al., 2012). Some widely used bandit algorithms include ϵ-greedy (Sutton and Barto, 2018), Boltzmann exploration (Kaelbling et al., 1996), UCB (Auer et al., 2002a), Thompson sampling (Chapelle and Li, 2011), contextual bandit (Sharaf and Daumé III, 2019), as well as Exp3 adversarial bandit (Auer et al., 2002b).

Multi-armed bandit algorithms have been used in a wide range of applications, such as online advertising (Chen et al., 2013), recommendation (Li et al., 2010), multi-task task selection (Guo et al., 2019a), and hyper-parameter optimization (Li et al., 2018; Merentitis et al., 2018). Recently, Graves et al. (2017) apply a non-stationary multi-armed bandit (in particular, the Exp3.S algorithm) to select an adaptive policy (curriculum) that a neural network follows to maximize the learning efficiency. Sharma and Ravindran (2017) use multi-armed bandit sampling to choose which domain data (harder vs. easier) to feed as input to a single model (using different Atari games). To our knowledge, we are the first ones to apply multi-armed bandits approaches to automate MTL and multi-reward RL optimization in the context of text generation tasks. We will provide more details in the later sections on the MAB algorithms used for MTL and multi-reward RL scenarios.

7.2 Dynamic Multi-Task Learning with Bandits

In Chapter 3 and Chapter 4 we proposed various ways of sharing the parameters across tasks. However, we have to manually tune the static mixing ratio of various tasks during the training. The complexity of finding the right optimal mixing ratio is exponentially hard with the increase in the number of tasks at hand. Also, instead of a mixing ratio, a dynamic approach could be more beneficial. Addressing these issues, in this chapter, we propose a multi-armed bandit approach that dynamically learns an effective schedule (curriculum) of switching between tasks for optimization during multi-task learning, instead of the traditional approach with a manually-tuned, static (fixed) mixing ratio (Luong et al., 2016). This dynamic approach allows us to achieve not only equal, but in fact better results than the manual approach, while importantly avoiding the hassle of tuning on the large space of mixing ratios over several different tasks. Next, we briefly discuss the primary and auxiliary tasks used in this dynamic multi-task learning setup.

We choose sentence simplification as our primary task. Sentence simplification is the task of improving the readability and understandability of an input text. This challenging task has been the subject of research interest because it can address automatic ways of improving reading aids for people with limited language skills, or language impairments such as dyslexia (Rello et al., 2013), autism (Evans et al., 2014), and aphasia (Carroll et al., 1999). It also has wide applications in NLP tasks as a preprocessing step, for example, to improve the performance of parsers (Chandrasekar et al., 1996), summarizers (Klebanov et al., 2004), and semantic role labelers (Vickrey and Koller, 2008; Woodsend and Lapata, 2014).

Several sentence simplification systems focus on operations such as splitting a long sentence into shorter sentences (Siddharthan, 2006; Petersen and Ostendorf, 2007), deletion of less important words/phrases (Knight and Marcu, 2002; Clarke and Lapata, 2006; Filippova and Strube, 2008), and paraphrasing (Devlin, 1999; Inui et al., 2003; Kaji et al., 2002). Inspired from machine translation based neural models, recent work has built end-to-end sentence simplification models along with attention mechanism, and further improved it with reinforcement-based policy gradient approaches (Zhang and Lapata, 2017b). Our baseline is a novel application of the pointer-copy mechanism (See et al., 2017) for the sentence simplification task, which allows the model to directly copy words and phrases from the input to the output. We further improve this strong baseline by bringing in auxiliary entailment and paraphrasing knowledge via soft and dynamic multi-level, multi-task learning.

Apart from the three simplification operations discussed above, we also ensure that the simplified output is a directed logical entailment w.r.t. the input text, i.e., does not generate any contradictory or unrelated information. We incorporate this entailment skill via multi-task learning (Luong et al., 2016) with an auxiliary entailment generation task. Further, we also induce word/phrase-level paraphrasing knowledge via a paraphrase generation task, enabling parallel learning of these three tasks in a three-way multi-task learning setup. We employ a novel 'multi-level' layered, soft sharing approach, where the parameters between the tasks are loosely coupled at different levels of layers; we share higher-level semantic layers between the sentence simplification and entailment generation tasks (which teaches the model to generate outputs that are entailed by the full input), while sharing the lower-level lexico-syntactic layers between the sentence simplification and paraphrase generation tasks (which teaches the model to paraphrase only the smaller sub-sentence pieces).

Empirically, we evaluate our system on three standard datasets: Newsela, WikiSmall, and WikiLarge. First, we show that our pointer-copy baseline is significantly better than sequence-to-sequence models, and competitive w.r.t. the state-of-the-art. Next, we show that our multi-level, multi-task framework performs significantly better than our strong pointer baseline and other competitive sentence simplification models on both automatic evaluation as well as on human study simplicity criterion. Further, we show that the dynamic multi-armed bandit based switching of tasks during training improves over the traditional manually-tuned static mixing ratio. Lastly, we show several ablation studies based on different layer-sharing approaches (higher versus lower) with auxiliary tasks, hard versus soft sharing, dynamic mixing ratio sampling, as well as our model's learned entailment and paraphrasing skills.

Next, we first describe our sentence simplification baseline model with attention mechanism, which is further improved by pointer-copy mechanism. Later, we introduce our two auxiliary tasks (entailment and paraphrase generation) and discuss how they can share specific lower/higher-level layers/parameters to improve the sentence simplification task in a multi-task learning setting. Finally, we discuss our new multi-armed bandit based dynamic multi-task learning approach.

7.2.1 Pointer-Copy Baseline Sentence Simplification Model

Our baseline is a 2-layer sequence-to-sequence model with both attention (Bahdanau et al., 2015) and pointer-copy mechanism (See et al., 2017). Given the sequence of input/source tokens $x = \{x_1, ..., x_{T_s}\}$, the model learns an auto-regressive distribution over output/target tokens $y = \{y_1, ..., y_{T_o}\}$, which is defined as $P_{vocab}(y|x;\theta) = \prod_t p(y_t|y_{1:t-1}, x; \theta)$, where θ represents model parameters and $p(y_t|y_{1:t-1}, x; \theta)$ is probability of generating token y_t at decoder time step t given the previous generated tokens $y_{1:t-1}$ and input x. Given encoder hidden states $\{h_i\}$, and decoder's t^{th} time step hidden state (of last layer) s_t, the context vector $c_t = \sum_i \alpha_{t,i} h_i$, where the attention weights $\alpha_{t,i}$ define an attention distribution over encoder hidden states: $\alpha_{t,i} = \exp(e_{t,i}) / \sum_k \exp(e_{t,k})$, where $e_{t,i} = v_a^T \tanh(W_a s_t + U_a h_i + b_a)$. Finally, the conditional distribution at each time step t of the decoder is defined as $p(y_t|y_{1:t-1}, x; \theta) = \text{softmax}(W_s s'_t)$, where the final hidden state s'_t is a combination of context vector c_t and last layer hidden state s_t and is defined as $s'_t = \tanh(W_c[c_t, s_t])$, where W_s and W_c are trained parameters.

Pointer-Copy Mechanism: This helps in directly copying the words from the source inputs to the target outputs via merging the generative distribution and attention distribution (as a proxy of copy distribution). The goal of sentence simplification is to rewrite sentences more simply, while preserving important information; hence, it also involves significant amount of copying from the source. Our pointer mechanism approach is similar to See et al. (2017). At each time step of the decoder, the model makes a (soft) choice between words from the vocabulary distribution P_{vocab} and attention distribution P_{att} (based on words in the input) using the word generation probability $p_g = \sigma(W_g c_t + U_g s_t + V_g d_t + b_g)$, where $\sigma(\cdot)$ is sigmoid, W_g, U_g, V_g and b_g are trainable parameters, and d_t is decoder input. The final vocabulary distribution is defined as the weighted combination of vocabulary and attention distributions:

$$P_f(y) = p_g P_{vocab}(y) + (1 - p_g) P_{att}(y) \tag{7.1}$$

7.2.2 Auxiliary Tasks

Entailment Generation The task of entailment generation is to generate a hypothesis which is entailed by the given input premise. A good simplified sentence should be entailed by (follow from) the source sentence, and hence we incorporate such knowledge through an entailment generation task into our sentence simplification task. We share the higher-level semantic layers between the two tasks (see reasoning in Sec. 7.2.3 below). We use entailment pairs from SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2018b) datasets for training our entailment generation model, where we use the same architecture as our sentence simplification model.

Paraphrase Generation Paraphrase generation is the task of generating similar meaning phrases or sentences by reordering and modifying the syntax and/or lexicon. Paraphrasing is one of the

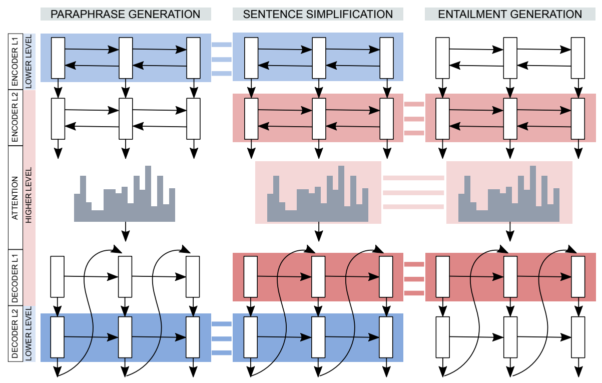


Figure 7.1: Overview of our 3-way multi-task model. Same color and dashed connections represent soft-shared parameters in different layers.

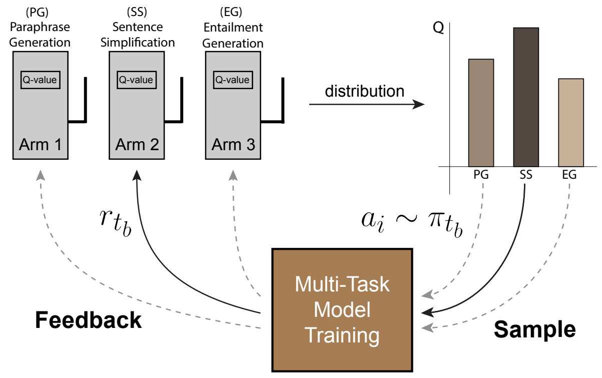

(PG) Paraphrase Generation
(SS) Sentence Simplification
(EG) Entailment Generation
Q-value
Arm 1
Arm 2
Arm 3
distribution
Q
PG
SS
EG
r_{t_b}
$a_i \sim \pi_{t_b}$
Multi-Task Model Training
Feedback
Sample

Figure 7.2: Overview of our multi-armed bandits algorithm for dynamic mixing ratio learning. It consists of a controller with 3 arms/tasks.

common operations used in sentence simplification, i.e, by substituting complex words and phrases with their simpler paraphrase forms. Hence, we add this knowledge to the sentence simplification task via multi-task learning, by sharing the lower-level lexico-syntactic layers between the two tasks (see reasoning in Sec. 7.2.3 below). For this, we use the paraphrase pairs from ParaNMT (Wieting and Gimpel, 2017a). Here, again, we use the same architecture as our sentence simplification model.

7.2.3 Multi-Task Learning

In this subsection, we discuss our multi-task, multi-level soft sharing strategy with parallel training of sentence simplification and related auxiliary tasks (entailment and paraphrase generation).

The predominant approach for multi-task learning in sequence-to-sequence models is to directly hard-share all encoder/decoder layers/parameters (Luong et al., 2016; Johnson et al., 2016; Pasunuru and Bansal, 2017a; Kaiser et al., 2017). However, this approach places very strong constraints/priors on the primary model to compress knowledge from diverse tasks. We believe that while the auxiliary tasks considered in this work share many similarities with the primary sentence simplification task, they are still different in either lower-level or higher-level representations (e.g., entailment will deal with higher-level, full-sentence logical inference, while paraphrasing will handle the lower-level intermediate word/phrase simplifications). In this section, we propose to relax the priors in two ways: (1) we share the model parameters in a finer-grained scale, i.e. layer-specific sharing, by keeping some of their parameters private, while sharing related representations; and (2) we encourage shared parameters to be close in certain distance metrics with a penalty term instead of hard-parameter-tying (Luong et al., 2016).

Multi-Level Sharing Mechanism Fig. 7.1 shows our multi-task model with parallel training of three tasks: sentence simplification (primary task), entailment generation (auxiliary task), and paraphrase generation (auxiliary task). Recently, Belinkov et al. (2017) observed that different layers in a sequence-to-sequence model (trained on translation) exhibit different functionalities: lower-layers (closer to inputs) of the encoder learn to represent word structure while higher layers (farther from inputs) are more focused on semantics and meanings (Zeiler and Fergus (2014) observed similar findings for convolutional image features). Based on these findings, we share the higher-level layers[1] between the entailment generation and sentence simplification tasks, since they share higher semantic-level language inference skills (for full sentence-to-sentence logical directedness). On the other hand, we share the lower-level lexico-syntactic layers[2] between the paraphrase generation and sentence simplification tasks, since they share more word/phrase and syntactic level paraphrasing knowledge to simplify the smaller, intermediate sentence pieces. Sec. 7.2.7 present empirical ablations to support our intuitive layer sharing.[3]

Soft Sharing In multi-task learning, we can do either hard sharing or soft sharing of parameters. Hard sharing directly ties the parameters to be shared, and receives gradient information from multiple tasks. On the other hand, soft sharing only loosely couples the parameters, and encourages them to be close in representation space. Hence the soft sharing approach gives more

[1]We found that sharing higher-level semantic layers (farther from input/output), i.e., encoder layer 2, attention, and decoder layer 1 (in Fig. 7.1), to work well. See Sec. 7.2.7 for ablations on alternative layer sharing methods.

[2]We found that sharing lower-level lexico-syntactic layers (closer to input/output), i.e., encoder layer 1 and decoder layer 2 (in Fig. 7.1), to work well. See Sec. 7.2.7 for ablations on alternative layer sharing methods.

[3]Note that even though entailment just tries to generate shorter, logical-subset sub-sentences, the overall saliency and quality of the simplified output is still balanced because the entailment task is flexibly (softly) shared with the paraphrasing and sentence simplification tasks, and the final model mixture is chosen based on simplification task metrics.

flexibility for parameter sharing, hence allowing different tasks to choose what parts of their parameters space to share. We minimize the l_2 distance between shared parameters as a regularization along with the cross entropy loss. Hence, the final loss function of the primary task with a related auxiliary task is defined as follows:

$$L(\theta) = -\log P_f(y|x;\theta) + \lambda||\theta_s - \phi_s|| \tag{7.2}$$

where θ represents the full parameters of the primary task (sentence simplification), θ_s and ϕ_s are the subsets of shared parameters between the primary and auxiliary task resp., and λ is a hyperparameter.

Multi-Task Training We employ multi-task learning with parallel training of related tasks in alternate mini-batches based on a mixing ratio α_{ss}:α_{eg}:α_{pp}, where we alternatively optimize α_{ss}, α_{eg}, α_{pp} mini-batches of sentence simplification, entailment generation, and paraphrase generation, respectively, until all models converge. In the next section, we discuss a new approach to replace this static mixing ratio with dynamically-learned task switching.

7.2.4 Dynamic Mixing Ratio Learning

Current multi-task models are trained via alternate mini-batch optimization based on a task 'mixing ratio' (Luong et al., 2016; Pasunuru and Bansal, 2017a), i.e., how many iterations on each task relative to other tasks (see end of Sec. 7.2.3). This is usually treated as a very important hyperparameter to be tuned, and the search space scales exponentially with the number of tasks. Hence, we importantly replace this manually-tuned and static mixing ratio with a 'dynamic' mixing ratio learning approach, where a controller automatically switches between the tasks during training, based on the current state of the multi-task model. Specifically, we use a multi-armed bandits based controller with Boltzmann exploration (Kaelbling et al., 1996) with an exponential moving average update rule.

We view the problem of learning the right mixing of tasks as a sequential control problem, where the controller's goal is to decide the next task/action after every n^s training steps in each task-sampling round t_b.[4] Let $\{a_1, ..., a_M\}$ represent the set of 3 tasks in our multi-task setting, i.e., sentence simplification, entailment generation, and paraphrase generation. We model the controller as a M-armed bandits, where it selects a sequence of actions/arms over the current training trajectory to maximize the expected future payoffs (see Fig. 7.2). At each round t_b, the controller selects an arm based on noisy value estimates and observes rewards r_{t_b} for the selected arm (we use the negative validation loss of the primary task as the reward in our setup). One problem in bandits learning is the trade-off between exploration and exploitation, where the agent needs to make a decision between taking the action that yields the best payoff on current estimates, or explore new actions whose payoffs are not yet certain. For this, we use the Boltzmann exploration (Kaelbling et al., 1996) with exponentially moving action value estimates. Let π_{t_b} be the policy of the bandit controller at round t_b, we define this to be:

$$\pi_{t_b}(a_i) = \exp(Q_{t_b,i}/\tau) \Big/ \sum_{j=1}^{M} \exp(Q_{t_b,j}/\tau) \tag{7.3}$$

where $Q_{t_b,i}$ is the estimated action value of each arm i at round t_b, and τ is the temperature.[5] If $Q_{0,i}$ is the initial value estimate of arm i, then $Q_{t_b,i}$ is the exponentially weighted mean with the decay rate α:

$$Q_{t_b,i} = (1-\alpha)^{t_b} Q_{0,i} + \sum_{k=1}^{t_b} \alpha(1-\alpha)^{t_b-k} r_k \tag{7.4}$$

To further help the exploration process, we follow the principle of optimism under uncertainty (Sutton and Barto, 1998) and set $Q_{0,i}$ to be above the maximum empirical rewards. Empirically, we show that this approach of 'dynamic mixing ratio' is equal or better than the traditional static mixing ratio (see Table 7.3). Also, we further show ablation study in Sec. 7.2.7 to show that this switching approach is better than the alternative approach of first using multi-armed bandits

[4]We set n^s to 10 to reduce variance of estimates, i.e., the bandit controller's task/action will be trained for 10 mini-batches.

[5]We tried decaying the temperature variable, but we didn't find this to very beneficial, so we instead fix this to 1.0.

for finding an optimal 'final' mixing ratio and then re-training the model based on this bandits-selected mixing ratio.

7.2.5 Evaluation Setup

Datasets We first describe the three standard sentence simplification datasets we evaluate on: Newsela, WikiSmall, and WikiLarge; next, we describe datasets for our auxiliary entailment and paraphrase generation tasks. **Newsela** (Xu et al., 2015b) is acknowledged as a higher-quality dataset for studying sentence simplifications, as opposed to Wikipedia-based datasets which automatically align complex-simple sentence pairs and have generalization issues (Zhang and Lapata, 2017b; Xu et al., 2015b; Amancio and Specia, 2014; Hwang et al., 2015; Štajner et al., 2015). Newsela consists of $1,130$ news articles, and we follow previous work (Zhang and Lapata, 2017b) to use the first $1,070$ documents for training, and 30 documents each for development and test. **WikiSmall** (Zhu et al., 2010) contains automatically-aligned complex-simple sentences from the ordinary-simple English Wikipedias. The data has $89,042$ pairs for training and 100 for test. We use the 205-pairs validation set from Zhang and Lapata (2017b). **WikiLarge** (Zhang and Lapata, 2017b) is a larger Wikipedia corpus aggregating pairs from Kauchak (2013), Woodsend and Lapata (2011), and WikiSmall. We use the exact training/evaluation sets provided by Zhang and Lapata (2017b). **SNLI and MultiNLI**: For the task of entailment generation, we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and MultiNLI (Williams et al., 2018b). We use their entailment labeled pairs for our entailment generation task, following previous work (Pasunuru and Bansal, 2017a). The combined SNLI and MultiNLI dataset has $302,879$ entailment pairs, out of which we use $276,720$ pairs for training, and the rest are divided into validation and test sets. **ParaNMT**: For the task of paraphrase generation, we use the back-translated paraphrase dataset provided by Wieting and Gimpel (2017a).

The filtered version of the dataset has 5.3 million pairs of paraphrases.[6] We use 99% for training, and the rest are evenly divided into validation and test sets.

Evaluation Metrics Following previous work (Zhang and Lapata, 2017b), we report all the standard evaluation metrics: SARI (Xu et al., 2016b), FKGL (Kincaid et al., 1975), and BLEU (Papineni et al., 2002). However, several studies have shown that BLEU is poorly correlated w.r.t. simplicity (Zhu et al., 2010; Štajner et al., 2015; Xu et al., 2016b). Moreover, Shardlow (2014) argues that FKGL (Kincaid et al., 1975), which measures readability of simpler output (lower is better), favors very short sentences even though longer/less coarse counterparts can be simpler. Further, Xu et al. (2016b) argues that BLEU tends to favor conservative systems that do not make many changes, and proposes SARI metric which explicitly measures the quality of words that are added and deleted. SARI is shown to correlate well with human judgment in simplicity (Xu et al., 2016b), and hence we primarily focus on this metric in our models' performance analysis.[7] Further, we also do human evaluation based on: Fluency ('is the output grammatical and well formed?'), Adequacy ('to what extent is the meaning expressed in the original sentence preserved in the output?') and Simplicity ('is the output simpler than the original sentence?'), following guidelines suggested by Xu et al. (2016b) and Zhang and Lapata (2017b).

Training Details All our soft/hard and layer-specific sharing decisions (Sec.7.2.7) were made on the validation/dev set. Our model selection (tuning) criteria is based on the average of our 3 metrics (SARI, BLEU, 1/FKGL) on the validation set. Please refer to the Appendix A.5 for full training details (vocabulary overlap, mixing ratios and bandit sampler decay rates and reward, WikiLarge pre-training, etc.).

[6]We chose ParaNMT over other paraphrase datasets (e.g. the phrase-to-phrase PPDB dataset (Ganitkevitch et al., 2013)), because ParaNMT is a sentence-to-sentence dataset and hence is a more natural fit for sentence-level multi-task RNN-layer sharing with our sentence-to-sentence simplification task.

[7]We use the JOSHUA package for calculating SARI and BLEU score following Zhang and Lapata (2017b) and Xu et al. (2016b). Our FKGL implementation is based on `https://github.com/mmautner/readability`.

Models	BLEU	FKGL	SARI
PREVIOUS WORK			
PBMT-R	18.19	7.59	15.77
Hybrid	14.46	4.01	30.00
EncDecA	21.70	5.11	24.12
DRESS	23.21	4.13	27.37
DRESS-LS	24.30	4.21	26.63
OUR MODELS			
Baseline ⊗	23.72	3.25	28.31
⊗ + Ent.	16.82	2.21	31.55
⊗ + Paraphr.	16.29	2.03	31.71
⊗+Ent+Par	11.86	1.38	32.98

Table 7.1: Newsela (FKGL: lower is better). Note that SARI is the primary, human-correlated metric for sentence simplification (Xu et al., 2016b).

	WIKISMALL			WIKILARGE		
Models	BLEU	FKGL	SARI	BLEU	FKGL	SARI
PREVIOUS WORK						
PBMT-R	46.31	11.42	15.97	81.11	8.33	38.56
Hybrid	53.94	9.21	30.46	48.97	4.56	31.40
SBMT-SARI	-	-	-	73.08	7.29	39.96
EncDecA	47.93	11.35	13.61	88.85	8.41	35.66
DRESS	34.53	7.48	27.48	77.18	6.58	37.08
DRESS-LS	36.32	7.55	27.24	80.12	6.62	37.27
OUR MODELS						
Baseline ⊗	36.18	7.69	25.67	82.37	7.84	36.68
⊗+Ent+Par	29.70	6.93	28.24	81.49	7.41	37.45

Table 7.2: WikiSmall/Large results (FKGL: lower is better). Note that SARI is the primary, human-correlated metric for sentence simplification (Xu et al., 2016b).

7.2.6 Experimental Results

We evaluate our models on three datasets and via several automatic metrics plus human evaluation.[8]

Pointer Baseline First, we compare our pointer baseline with various previous works: PBMT-R (Wubben et al., 2012), Hybrid (Narayan and Gardent, 2014), SBMT-SARI (Xu et al., 2016b)[9], and EncDecA, DRESS, and DRESS-LS (Zhang and Lapata, 2017b). As shown in Table 7.1, our pointer baseline already achieves the best score in FKGL and the second-best score in SARI on Newsela, and also achieves overall comparable results on both WikiSmall and WikiLarge (see Table 7.2).

Multi-Task Models We further improve our strong pointer-based sentence simplification baseline model by multi-task learning it with entailment and paraphrase generation. First, we show that our 2-way multi-task models with auxiliary tasks (entailment and paraphrase generation) are

[8]As described in Sec. 7.2.5, Newsela is considered as a higher quality dataset for text simplification, and thus we report ablation-style results (e.g., 2-way multi-task models and different layer-sharing ablations) and human evaluation on Newsela (since Wikipedia datasets are automatically-aligned). Moreover, we report SARI, FKGL, and BLEU for completeness, but as described in Sec. 7.2.5, SARI is the primary human-correlated metric for sentence simplification.

[9]We borrow the SBMT-SARI results for WikiLarge from Zhang and Lapata (2017b).

Models	BLEU	FKGL	SARI
NEWSELA			
Static Mixing Ratio	11.86	1.38	32.98
Dynamic Mixing Ratio	11.14	1.32	33.22
WIKISMALL			
Static Mixing Ratio	29.70	6.93	28.24
Dynamic Mixing Ratio	27.23	5.86	29.58

Table 7.3: Results on dynamic vs. static mixing ratio (FKGL: lower is better).

statistically significantly better than our pointer baseline and previous works in both SARI and FKGL on Newsela (see Table 7.1).[10] Next, Table 7.1 and Table 7.2 summarize the performance of our final 3-way multi-level, multi-task models with entailment generation and paraphrase generation on all three datasets. Here, our 3-way multi-task models are statistically significantly better than our pointer baselines in both SARI and FKGL (with $p < 0.01$) on Newsela and WikiSmall, and in SARI ($p < 0.01$) on WikiLarge. Also, our 3-way multi-task model is statistically significantly better than the 2-way multi-task models in SARI and FKGL with $p < 0.01$ (see Table 7.1). In Sec. 7.2.7, we further provide a set of detailed ablation experiments investigating the effects of different (higher-level versus lower-level) layer sharing methods and soft- vs. hard-sharing in our multi-level, multi-task models; and we show the superiority of our final choice of higher-level semantic sharing for entailment generation and lower-level lexico-syntactic sharing for paraphrase generation.

Dynamic Mixing Ratio Models Finally, we present results on our 3-way multi-task model with the new approach of using ‘dynamic’ mixing ratios based on multi-armed bandits sampling (see Sec. 7.2.4). As shown in Table 7.3, this dynamic multi-task approach achieves a stat. significant improvement in SARI as compared to the traditional fixed and manually-tuned mixing ratio based 3-way multi-task model: 33.22 vs. 32.98 ($p < 0.05$) on Newsela, and 29.58 vs. 28.24 ($p < 0.001$) on WikiSmall. Hence, this allows us to achieve not only equal, but in fact better results

[10]Stat. significance is computed via bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994). Both our 2-way multi-task models are statistically significantly better in SARI and FKGL with $p < 0.01$ w.r.t. our pointer baseline and previous works. Note the discussion in Sec. 7.2.5 about why BLEU is not a good sentence simplification metric.

	HUMAN EVALUATION				MATCH-WITH-INPUT		
Models	Fluency	Adequacy	Simplicity	Average	BLEU (%)	ROUGE (%)	Exact Match (%)
Ground-truth	4.97	4.08	3.83	4.29	18.25	43.74	0.00
Hybrid	3.88	3.82	3.92	3.87	25.74	56.20	3.34
DRESS-LS	4.84	4.18	3.21	4.08	42.93	67.61	14.48
Pointer Baseline	4.61	3.94	3.99	4.18	30.80	60.56	10.68
3-way Multi-task	4.73	3.18	4.62	4.18	8.74	37.82	2.41

Table 7.4: Human evaluation results (on left) and closeness-to-input source results (on right), for Newsela. In Sec. 7.2.6 'Human Evaluation', we discuss the issue of high adequacy scores for outputs that are very similar to the input (see right part of the table).

than the manual approach, while importantly avoiding the hassle of tuning on the large space of mixing ratios over several different tasks. In Sec. 7.2.7, we further provide ablation analysis to study whether the improvements come from the bandit learning this dynamic curriculum or from the bandit finding the final optimal mixing-ratio at the end of the sampling procedure (and also compare it to a random curriculum).

Human Evaluation We also perform an anonymized human study comparing our pointer baseline, our multi-task model, some previous works (Hybrid (Narayan and Gardent, 2014) and state-of-the-art DRESS-LS (Zhang and Lapata, 2017b)), and ground-truth references (see left part of Table 7.4), based on fluency, adequacy, and simplicity (see Sec. 7.2.5 for more details about these criteria) using 5-point Likert scale. We asked annotators to evaluate the models (randomly shuffled to anonymize model identity) based on 200 samples from the representative and cleaner Newsela test set, and their scores are reported in Table 7.4. Our 3-way multi-task model achieves a significantly higher ($p < 0.001$) simplicity score compared to DRESS-LS, Hybrid, and our pointer baseline models. However, we next observe that our 3-way multi-task model has lower adequacy score as compared to DRESS-LS and the pointer model, but this is because our 3-way multi-task model focuses more strongly on simplification, which is the goal of the given task. Moreover, based on the overall average score of the three human evaluation criteria, our 3-way multi-task model is also significantly better ($p < 0.03$) than the state-of-the-art DRESS-LS model (and $p < 0.001$ w.r.t. Hybrid model).[11] Also, on further investigation, we found that a problem with the adequacy metric is that it gets artificially high scores for output sentences which are exact match (or a very close match) with the input source sentence, i.e., they have very little simplification and hence almost fully retain the exact meaning. In the right part of Table 7.4, we analyzed the matching scores of the outputs from different models w.r.t. the source input text, based on BLEU, ROUGE (Lin, 2004) and exact match. First, this shows that the ground-truth sentence-simplification references are in fact (as expected) very different from the input source (0% exact match, 18% BLEU, 44% ROUGE). Next, we find that our multi-task model also has low match-with-input scores (2% exact match, 9% BLEU, 38% ROUGE), similar to the behavior of the ground-truth references. On the other hand, DRESS-LS (and pointer baseline) model is generating output sentences which are substantially closer to the input and hence is not making enough changes (14% exact match, 43% BLEU, 68% ROUGE) as compared to the references (which explains their higher adequacy but lower simplicity scores).

7.2.7 Ablations and Insights

In this section, we conduct several ablation analyses to study the different layer-sharing mechanisms (higher semantic vs. lower lexico-syntactic), soft- vs. hard-sharing, two dynamic multi-armed bandit approaches, and our model's learned entailment and paraphrasing skills. We also present and analyze some output examples from several models.[12] Note that all our soft and layer sharing decisions were strictly made on the dev/validation set (see Sec. 7.2.5).

Different Layer Sharing Approaches We empirically show that our final multi-level layer sharing method (i.e., higher-level semantic layer sharing with entailment generation, while lower-level lexico-syntactic layer sharing with paraphrase generation) performs better than the follow-

[11]Note that our multi-task model is stat. equal to our pointer baseline on the overall-average score, showing the available trade-off between systems that simplify conservatively vs. strongly, based on one's desired downstream task application. Also refer to the high 'match-with-input' issue with the adequacy metric discussed next.

[12]Since Newsela is considered as the more representative dataset for sentence simplification with lesser noise and human quality (Xu et al., 2015b; Zhang and Lapata, 2017b), we conduct our ablation studies on this dataset, but we observed similar patterns on the other two datasets as well.

Models	BLEU	FKGL	SARI
Final (High Ent + Low PP)	**11.86**	**1.38**	**32.98**
Both lower-layer	11.94	1.47	31.92
Both higher-layer	12.26	1.38	32.02
Swapped (Low Ent + High PP)	21.64	2.97	29.07
Hard-sharing	13.01	1.38	32.36

Table 7.5: Multi-task layer ablation results on Newsela.

ing alternative layer sharing methods: (1) both auxiliary tasks with high-level layer sharing, (2) both with low-level layer sharing, and (3) reverse/swapped sharing (i.e., lower-level layer sharing for entailment, and higher-level layer sharing for paraphrasing). Results in Table 7.5 show that our approach of high-level sharing for entailment generation and low-level sharing for paraphrase generation is statistically significantly better than all other alternative approaches in SARI ($p < 0.01$) (and statistically better or equal in FKGL).

Soft- vs. Hard-Sharing In this work, we use soft-sharing instead of hard-sharing approach (benefits discussed in Sec. 7.2.3) in all of our models. Table 7.5 also presents empirical results comparing soft- vs. hard-sharing on our final 3-way multi-task model, and we observe that soft-sharing is statistically significantly better than hard-sharing in SARI with $p < 0.01$.

Quantitative Improvements in Entailment We employ a state-of-the-art entailment classifier (Chen et al., 2017) to calculate the entailment probabilities of output sentence being entailed by the ground-truth.[13] Table 7.6 summaries the average entailment scores for the Hybrid, DRESS-LS, Pointer baseline, and 2-way multi-task model (with entailment generation auxiliary task), showing that the 2-way multi-task model improves in the aspect of logical entailment ($p < 0.001$), demonstrating the inference skill acquired by the simplification model via the auxiliary knowledge from the entailment generation task.

[13]For this entailment analysis, we use ground-truth output as premise instead of input source, because: (1) entailment w.r.t. input source can give artificially high scores even when the output doesn't simplify enough and just copies the source (see the discussion in Sec. 7.2.6 and Table 7.4); (2) By transitivity, if output is entailed by ground-truth, which in turn is entailed by source, then output should also be entailed by source (plus, we want the output to be closer to ground-truth than to input source).

Models	Entailment	Paraphrasing
Ground-truth	N/A	62.1
Hybrid	34.8	74.1
DRESS-LS	30.7	77.9
Pointer Baseline	36.9	76.6
2-way Multi-Task	41.4	63.9

Table 7.6: Analysis: Entailment and paraphrase classification results (avg. probability scores as %) on Newsela.

Models	Deletions	Additions
Hybrid	95.18	0.000
DRESS-LS	85.37	0.047
Pointer Baseline	88.91	0.026
3-way Multi-Task	97.54	0.049

Table 7.7: Analysis: SARI's sub-operation scores on Newsela dataset.

Quantitative Improvements in Paraphrasing We use the paraphrase classifier from Wieting and Gimpel (2017b) to compute the paraphrase probability score between the generated output and the input source. The results in Table 7.6 show that our 2-way multi-task model (with paraphrasing generation auxiliary task) is closer to the ground-truth in terms of the amount of paraphrasing (w.r.t. input) required by the sentence-simplification task, while the pointer baseline and previous models have higher scores due to higher amount of copying from input source (see 'Match-with-Input' discussion in Sec. 7.2.6, Table 7.4).

Addition/Deletion Operations We also measured the performance of the various models in terms of the addition and deletion operations using SARI's sub-operation scores computed w.r.t. both the ground-truth and source (Xu et al., 2016b). Table 7.7 shows that our multi-task model is equal or better in terms of both operations.

Two Multi-Armed-Bandit Approaches As described in Sec. 7.2.4, our multi-armed bandit approach with dynamic mixing ratio during multi-task training learns a sufficiently good curriculum to improve the sentence simplification task (see Sec. 7.2.6). Here, we further show an ablation study on another alternative approach of using multi-armed bandits, where we record the last 10% of the actions from the bandit controller[14], then calculate the corresponding mixing ratio based on this 10%, and run another independent model from scratch with this fixed mixing ratio. We found that the curriculum-style dynamic switching of tasks is in fact very effective as compared to this other 2-stage approach (33.22 versus 32.58 in SARI with $p < 0.01$). This is intuitive be-

[14]We choose the last 10% to avoid the noisy action-value estimates at the start of the training.

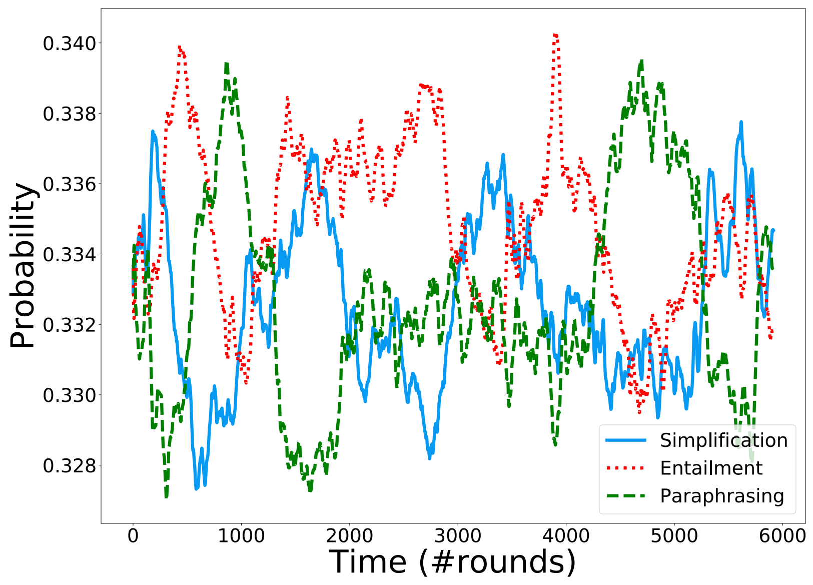


Figure 7.3: Task selection probability over training trajectory, predicted by bandit controller.

cause the dynamic switching of tasks during multi-task training allows the model to choose the best next task to run based on the current state (as well as the previous curriculum path) of the model, as opposed to a fixed/static single mixing ratio for the full training period. In Fig. 7.3, we visualize the (moving averages of) probabilities of selecting each task, which shows that in the 0-1000 #rounds range, the bandit initially gives higher weight to the main task, but gradually redistributes the probabilities to the auxiliary tasks; and beyond 1000 #rounds, it then alternates switching among the three different tasks periodically. We also experimented with replacing the bandit controller with random task choices, and our bandit-controller achieves statistically significantly better results than this approach in both SARI and FKGL with $p < 0.01$, which shows that the path learned by the bandit controller is meaningful.

Multi-Task Learning vs. Data Augmentation To verify that our improvements come indeed from the auxiliary tasks' specific character/capabilities and not just due to adding more data, we separately trained word embeddings on each auxiliary dataset (i.e., SNLI+MultiNLI and ParaNMT) and incorporated them into the primary simplification model. We found that both our 2-way multi-task models perform stat. significantly better than these models (which use the auxiliary word-embeddings), suggesting that merely adding more data is not enough. Moreover, Table 7.5 shows that only specific intuitive (syntactic vs. semantic) layer sharing between the primary and auxiliary tasks helps results and not just adding data.

7.2.8 AUTOSEM: Automatic Task Selection and Mixing in MTL

In the previous section, we have presented an effective way of dynamically switching across auxiliary tasks during MTL training avoiding the hassle of manual tuning of the mixing ratio of tasks. Another issue is the success of MTL models also depend on the correct choice of auxiliary tasks. One can achieve this via manual intuition or hyper-parameter tuning over all combinatorial task choices, but this introduces human inductive bias or is not scalable when the number of candidate auxiliary tasks is considerably large.

Addressing this issue, we proposed AUTOSEM (Guo et al., 2019a) framework, a two-stage Bayesian optimization pipeline. The first stage addresses automatic task selection from a pool of auxiliary tasks. For this, we use a non-stationary multi-armed bandit controller (MAB) (Bubeck et al., 2012; Raj and Kalyani, 2017) that dynamically alternates among task choices within the training loop, and eventually returns estimates of the utility of each task w.r.t. the primary task. We model the utility of each task as a Beta distribution, whose expected value can be interpreted as the probability of each task making a non-negative contribution to the training performance of the primary task. Further, we model the observations as Bernoulli variables so that the posterior distribution is also Beta-distributed. We use Thompson sampling (Chapelle and Li, 2011; Russo et al., 2018) to trade off exploitation and exploration.

The second stage then takes the auxiliary tasks selected in the first stage and automatically learns the training mixing ratio of these tasks, through the framework of Bayesian optimization, by modeling the performance of each mixing ratio as a sample from a Gaussian Process (GP) to sequentially search for the optimal values (Rasmussen, 2004; Snoek et al., 2012). For the covariance function in the GP, we use the Matern kernel which is parameterized by a smoothness hyperparameter so as to control the level of differentiability of the samples from GP. Further, following Hoffman et al. (2011), we use a portfolio of optimistic and improvement-based policies as acquisition functions (Shahriari et al., 2016) for selecting the next sample point from the GP search space.

We conducted several experiments on the GLUE natural language understanding benchmark (Wang et al., 2019a), where we choose each of RTE, MRPC, QNLI, CoLA, and SST-2 as the primary task, and treat the rest of the classification tasks from the GLUE benchmark as candidate auxiliary tasks. Empirical results suggest that our AUTOSEM framework can successfully find useful auxiliary tasks and automatically learn their mixing ratio, achieving significant performance boosts on top of strong baselines for several primary tasks, e.g., 5.2% improvement on QNLI, 4.7% improvement on RTE, and 2.8%/0.8% improvement on MRPC.

7.3 Optimizing Multiple RL Rewards with Bandits

Recent advancements in end-to-end neural networks-based approaches have shown wide success in various sequence generation tasks: machine translation (Sutskever et al., 2014; Luong et al., 2015), dialogue systems (Vinyals and Le, 2015; Serban et al., 2016), textual summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), image/video captioning (Bahdanau et al., 2015; Venugopalan et al., 2015a; Pasunuru and Bansal, 2017a), question generation (Du et al., 2017; Du and Cardie, 2018; Zhang and Bansal, 2019), etc. In all of these tasks, cross-entropy loss optimization has been widely used as a standard optimization approach (Sutskever et al., 2014), but this approach suffers from exposure-bias issue (Ranzato et al., 2016) and does not optimize for the non-differentiable automatic evaluation metrics that measure the quality of the generated sequence. Recent introduction of policy gradient-based reinforcement learning approaches address these issues for sequence generation tasks by directly optimizing the non-differentiable evaluation metrics (Zaremba and Sutskever, 2015; Ranzato et al., 2016; Rennie et al., 2017).

However, optimizing for a particular metric/reward via policy gradient-based approaches often leads to improvement in mostly that specific metric, suggesting that this approach is gaming the metrics (Paulus et al., 2018). The weighted average of multiple metrics or surrogate rewards have been explored (Liu et al., 2017b), but these approaches have to deal with finding the optimal scale balance across different metrics. One can alternatively optimize multiple metrics via a mixing ratio (Pasunuru and Bansal, 2018) (Chapter 6), but this still needs careful tuning of the mixing ratio. Moreover, all these reward approaches are fixed and do not change over training, and all the metrics may not be important over every stage of the training. Thus, it might be useful to consider using a dynamic combination of metrics, which rewards to use early vs. later, or which rewards might be useful to come back later in training, and consider the context of the full history of rewards, as well as the model's current state and the nature of the metric.

To this end, we present a multi-armed bandit approach (which we name the DORB framework) where the arms of the bandit are the choices of the metrics that we want to optimize as rewards. At every round, the bandit chooses the next possible metric to optimize based on its previous performance history over these metrics, hence allowing the automatic learning of an optimal curriculum of rewards. We explore this approach in the context of exploration vs. exploitation via Exp3 algorithm (Auer et al., 2002b) with two novel approaches for bandit rewards: (1) Single Multi-reward Bandit (SM-Bandit); (2) Hierarchical Multi-reward Bandit (HM-Bandit). First, we present a reward scaling approach to maintain the metric rewards range in $[0, 1]$. Next, we present our SM-Bandit, where at each round, the bandit's reward is based on the performance improvement from multiple sources. Here, we use the average of all the scaled metric rewards from multiple sources as the final reward to the bandit. Finally, we present our HM-Bandit, which consists of a single first-level controller, as well as K second-level multi-armed bandits. The first-level controller's goal is to find the under-performing reward metric, while the second-level bandits' goal is to trigger the specific metric optimizer that will lead to a promising improvement in this specific metric.

We validate the effectiveness of our approaches on two important generation tasks: question generation and data-to-text generation (including an unseen-test transfer setup) via both automatic evaluation metrics and human evaluation. For question generation, we present results on the SQuAD QG dataset (Du et al., 2017), and for data-to-text NLG, we choose the WebNLG

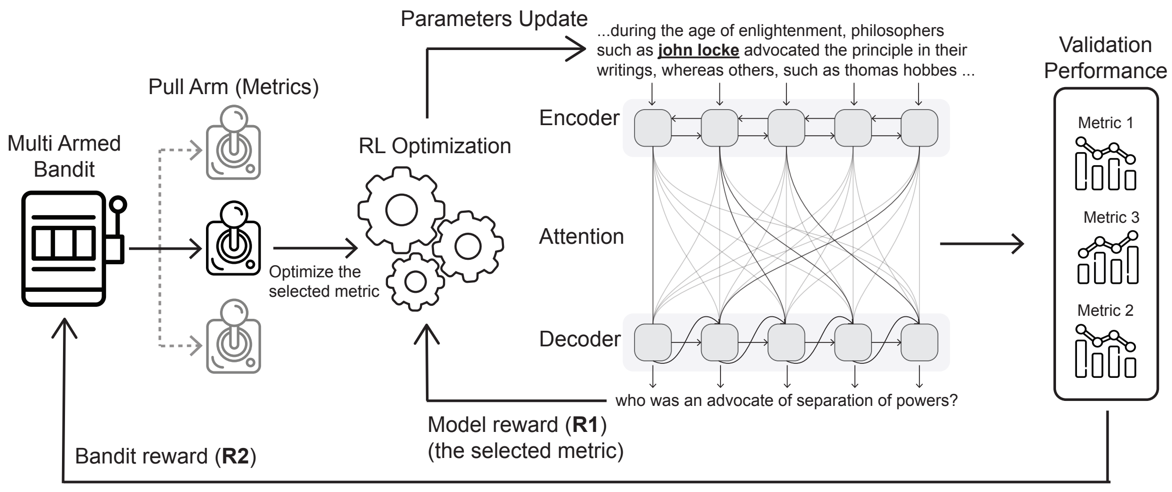

Parameters Update
...during the age of enlightenment, philosophers such as john locke advocated the principle in their writings, whereas others, such as thomas hobbes ...
Validation Performance
Pull Arm (Metrics)
Multi Armed Bandit
RL Optimization
Encoder
Attention
Decoder
Optimize the selected metric
Metric 1
Metric 3
Metric 2
who was an advocate of separation of powers?
Model reward (R1) (the selected metric)
Bandit reward (R2)

Figure 7.4: Overview of our multi-armed bandit reward selection framework DORB. At each step, the model outputs are scored based on a reward function (metric), where the choice of the reward function is dynamically controlled by the multi-armed bandit. Then the corresponding optimization is executed based on the chosen reward function. Finally, the observed validation performance metrics are given as feedback to the bandit.

dataset (Gardent et al., 2017). We show that our bandit-based approaches perform statistically significantly better (based on human evaluation) than strong single-reward based RL models as well as non-bandits based multi-reward methods such as the multi-task approach of Pasunuru and Bansal (2018). We further present various interpretable analyses of our bandit progress and learned rewards curriculum over different bandit approaches.

7.3.1 Multi-Reward Optimization

In this section, we first describe the policy gradients-based reinforcement learning (RL) approach for text generation tasks, and then discuss the need for a better multi-reward optimization approach for RL in the context of generation tasks. Lastly, we introduce our novel methods for multi-reward optimization via multi-armed bandits.

Glossary: Agent: RL policy gradients; Bandit: multi-armed bandit; Controller: controller in HM-Bandit (see Fig. 7.5).

Policy Gradient Background. Cross-entropy loss based optimization is traditionally used for the sequence generation tasks. However, recent policy gradient-based reinforcement learning approach has shown two advantages over the cross-entropy loss optimization approach: (1) avoiding *exposure bias* issue which is about the mismatch in the output distributions created by different train and test time decoding approaches in cross-entropy loss optimization; (2) able to directly optimize the non-differentiable evaluation metrics.

To this end, REINFORCE algorithm (Williams, 1992; Zaremba and Sutskever, 2015) is used to learn a policy p_θ defined by the model parameters θ to predict the next action (tokens in our setup). Specifically, instead of minimizing the negative log-likelihood, we minimize the following loss:

$$L_{\text{RL}} = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)] \tag{7.5}$$

where w^s is the sequence of sampled tokens and $r(\cdot)$ is the reward function that measures the quality of w^s. The derivative of this loss function can then be approximated using a single sample along with a bias estimator $\hat{b}$ to reduce variance:

$$\nabla_\theta L_{\text{RL}} = -(r(w^s) - \hat{b})\nabla_\theta \log p_\theta(w^s) \tag{7.6}$$

There are several ways to calculate the baseline estimator, and in this work we use the SCST mechanism (Rennie et al., 2017).

Need for a better multi-reward optimization. Often, an RL agent can improve the policy p_θ via multiple reward sources. However, efficient ways of optimizing multiple rewards in a policy gradient-based reinforcement learning setup have been less explored. Previous works have either explored using a weighted combination of multiple rewards (Zhang and Lapata, 2017a; Li et al., 2016) or alternate fashion of optimizing multiple rewards inspired via multi-task learning setup (Pasunuru and Bansal, 2018). However, these approaches have a disadvantage of tuning the weights of the rewards combination or using a static tunable mixing ratio while optimizing in an alternate fashion. On the other hand, previous work (Shi et al., 2018) have tried to pose the text generation problem in inverse reinforcement learning framework (IRL) to directly learn a reward function using training data, instead of coming up with sparse reward signals. However, this approach does not have the flexibility to consider user intended rewards, hence may not be always desirable. To this end, we explore multi-reward optimization via a multi-armed bandit approach (Bubeck et al., 2012; Lattimore and Szepesvári, 2019; Burtini et al., 2015). During the training, the bandit explores/exploits the choice of reward functions in order to improve the overall performance of the model. In the remaining part of this section, we discuss various multi-armed bandit-based models for multi-reward optimization (Sec. 7.3.2), and reward settings (Sec. 7.3.3). Then, we present the two novel approaches, namely Single Multi-reward Bandit (SM-Bandit, Sec. 7.3.4) and Hierarchical Multi-reward Bandit (HM-Bandit, Sec. 7.3.5).

7.3.2 Multi-Armed Bandit for Multi-Reward Optimization

Given a set of K candidate actions (arms) $\{a_1, a_2, ..., a_K\}$, the objective of a multi-armed bandit problem is to maximize rewards earned through a sequence of lever pulls (actions). We call this reward as bandit reward. We view the problem of optimizing multiple rewards as a sequential design of experiments (Robbins, 1952), where the bandit's goal is to decide the next arm (loss function) to pull after each round in order to maximize the rewards it earns.

Let $\{R_1, R_2, .., R_K\}$ be a set of different rewards from K sources which can measure the model/policy's performance. To directly maximize the performance of these K rewards, we need to use K different reinforcement learning-based loss functions. Let the loss function for R_i be:

$$L_{\mathrm{RL}_i} = -\mathbb{E}_{w^s \sim p_\theta}[R_i(w^s)] \tag{7.7}$$

Each of these K loss functions is considered as an arm of the multi-armed bandit (i.e., the arms/joysticks in Fig. 7.4), where pulling the i^{th} arm will result in optimizing for reinforcement based loss function L_{RL_i} (i.e., in Fig. 7.4, main model parameters get updated). The goal of the bandit is to explore and exploit different loss functions and maximize its reward (the validation performance of the model, see Fig. 7.4). One widely studied problem is the trade-off between "exploitation" of the arm with the highest estimated payoff and "exploration" of less known arms. For this, we use the popular Exp3 bandit algorithm (Auer et al., 2002b).

Exp3 Bandit Algorithm A stochastic bandit is completely determined by the distribution of rewards of respective actions. However, it will be hard to argue that rewards are truly randomly generated, and even if they are randomly generated, the rewards could be correlated over time (e.g., the validation performance at the next step will be correlated with validation performance at this time step). Taking all these factors into account makes the algorithm overly complicated, and thus an alternative is to assume nothing about the underlying mechanism that generates the rewards while still trying to achieve the lowest possible regret. This is called the adversarial bandit problem, where the goal is to design an algorithm that keeps the regret small regardless of what rewards are assigned to actions.

Exponential-weight algorithm for Exploration and Exploitation, or Exp3 (Auer et al., 2002b), was created to handle the non-stochastic adversarial bandit problem. We use this algorithm in our DORB framework. Exp3 works by maintaining a set of weights for each candidate action, and the weights are used to decide randomly which action to take next. The empirical observation is fed back to the bandit to either increase or decrease the relevant weights. The algorithm also has a hyper-parameter $\gamma \in [0, 1]$ that decides the probability to take action uniformly at random. Specifically, at round t, the bandit picks action (arm) i among K arms based on the arm selection probability which is defined as follows:

$$p_t(i) = (1 - \gamma)\frac{w_{t,i}}{\sum_{j=1}^{K} w_{t,j}} + \frac{\gamma}{K} \tag{7.8}$$

where the weights $w_{t,i}$ are updated based on the observed bandit reward r_t^B:

$$\hat{r}_{t,j}^B = \begin{cases} r_t^B / p_t(i) & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

$$w_{t+1,i} = w_{t,i} \exp(\gamma \hat{r}^B_{t,i}/K) \tag{7.10}$$

7.3.3 Bandit Reward Settings

Note that in this work, we have two sets of rewards: rewards used for optimizing the sequence generation model via policy gradients-based reinforcement learning (R1 in Fig. 7.4, Sec. 7.3.1), and rewards used for the bandit (R2 in Fig. 7.4). The rewards for the generation model are used to optimize the model w.r.t. the metric of interest, while the rewards for the bandit help the bandit decide which "metric of interest" the generation model should optimize.

In order to maintain consistent magnitude/scale across metric rewards while using them for bandits, we use scaled rewards via the quantiles of rewards history following Graves et al. (2017). Let $\mathbf{R}^t = \{R^i\}_{i=1}^{t-1}$ be the history of unscaled rewards up to time step t. Let q_t^{lo} and q_t^{hi} be the lower and upper quantiles of $\mathbf{R}^t$, respectively.[15] Then, the scaled reward, $\hat{r}^t$ is defined as follows:

$$\hat{r}^t = \begin{cases} 0 & \text{if } R^t < q_t^{lo} \\ 1 & \text{if } R^t > q_t^{hi} \\ \frac{R^t - q_t^{lo}}{q_t^{hi} - q_t^{lo}}, & \text{otherwise} \end{cases} \tag{7.11}$$

Instead of keeping the entire history of rewards, we use past n rewards from the history.

7.3.4 Single Bandit with Multi-Reward

Often, we want to optimize multiple metrics in our RL approach. For this, we have to give a joint reward coming from multiple sources (metrics in our case) to the bandit as a bandit reward. One can easily give the weighted combination of these rewards coming from multiple sources as a reward to the bandit. However, tuning these weights is intractable if the number of reward sources is large. Here, we introduce a new approach called Single Multi-reward Bandit (SM-

[15]We set q_t^{lo} and q_t^{hi} to be 20^{th} and 80^{th} quantiles.

Algorithm 1 SM-Bandit Training

```
 1: Inputs: #rewards: $K$, #train steps: $n_{\text{train}}$, #steps in bandit round: $n_{\text{bandit}}$
 2: Initialize the Exp3 bandit $B$ with $K$ arms
 3: $a \leftarrow$ chooseArm($B$)                                        ▷ Based on Eqn. 7.8
 4: $i \leftarrow 0$
 5: while $i < n_{\text{train}}$ do
 6:     Sample word sequence $w^s$ from model
 7:     Calculate rewards $R^{\text{train}}$ based on $w^s$
 8:     Optimize model's $L_{\text{RL}_a}$ loss using $R_a^{\text{train}}$
 9:     if $i \mod n_{\text{bandit}} == 0$ then
10:         Evaluate model to get $R^{\text{val}}$
11:         $r \leftarrow \frac{1}{K}\sum_{k=1}^{K}$ scaled($R_k^{\text{val}}$)                ▷ Based on Eqn. 7.11
12:         updateBandit($B, a, r$)                                  ▷ Based on Eqn. 7.10
13:         $a \leftarrow$ chooseArm($B$)
14:     $i \leftarrow i + 1$
```

Bandit), which avoids tuning and uses rewards from multiple sources as feedback to the bandit. Let L_{RL_1}, L_{RL_2}, and L_{RL_3} be the reinforcement learning-based loss functions corresponding to three arms of the bandit: arm_1, arm_2, and arm_3, respectively. If arm_2 is selected at round t, then we optimize for L_{RL_2} and measure the performance of all the unscaled metric scores on the validation set and then calculate the corresponding scaled rewards for each metric. We average over these scaled rewards and give that as a reward to the bandit. The generalization of this reward for K-armed bandit is: $r^t = \frac{1}{K}\sum_{i=1}^{K}\hat{r}_i^t$, where r^t is the bandit reward at round t and $\hat{r}_i^t$ is the scaled reward (Eq. 7.11) for the metric corresponding to arm_i at round t. This approach allows us to avoid tuning the balancing weights across the metrics that we optimize, and ensure that the bandit is improving all metrics, as the bandit goal is to maximize the average of all metrics. A detailed procedure of SM-Bandit is described in Algorithm 1.

7.3.5 Hierarchical Bandit with Multi-Reward

The SM-bandit's goal in the previous approach described in Sec. 7.3.4 is to improve all metrics using a single bandit. In this section, we introduce another bandit-based variant to improve all metrics but by using multiple bandits which are controlled by a controller, called Hierarchical Multi-reward Bandits (HM-Bandit, Fig. 7.5). The HM-Bandit consists of a single first-level con-

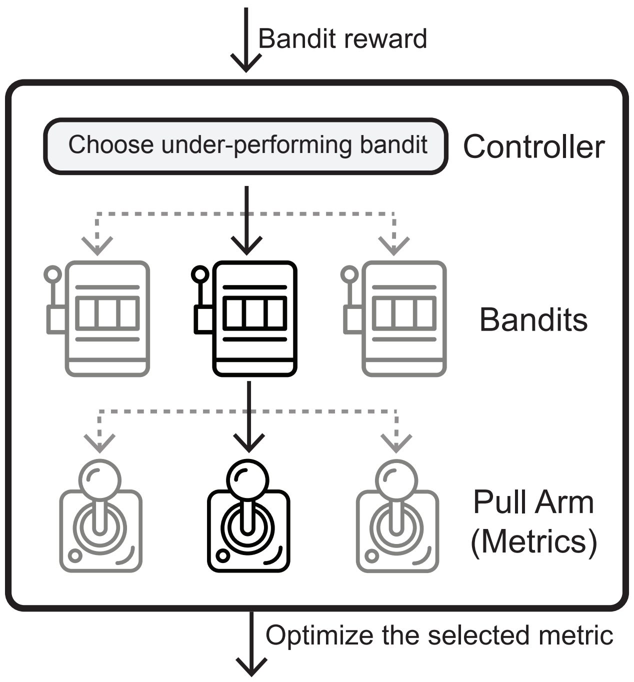


Figure 7.5: Overview of the hierarchical multi-armed bandit. The first-level has a controller and the second-level has bandits. The controller decides which bandit of the second-level will be pulled. The second-level bandits then decide which metric to use as the reward function during RL optimization.

troller (not a bandit, top row in Fig. 7.5), and K second-level multi-armed bandits (middle row in Fig. 7.5). The first-level controller's goal is to find the under-performing reward metric, while the second-level bandits' goal is to trigger a specific metric optimizer that will lead to a promising improvement in this specific metric. More intuitively, the first-level controller sets the objective (e.g., ROUGE needs to be improved), while the second-level bandit decides which specific reward function can help accomplish the objective. A detailed procedure of our HM-bandit is described in Algorithm 2. This concept is also loosely related to Bayesian model selection, where it's common to use a hierarchical specification of models (Rasmussen and Williams, 2005).

7.4 Tasks and Setup for DORB Framework

We use question generation and data-to-text generation tasks in our experiments. In this section, we discuss the details on these two tasks along with the experimental setup.

Algorithm 2 HM-Bandit Training

```
 1: Inputs: #rewards: $K$, #train steps: $n_{\text{train}}$, #steps in bandit round: $n_{\text{bandit}}$, #steps in controller round: $n_{\text{controller}}$
 2: Create the controller $C$ with $K$ bandits
 3: Initialize all bandits, and set $j \leftarrow 0$
 4: $B \leftarrow$ chooseBandit$(C, j)$                         ▷ choose bandit at index $j$
 5: $a \leftarrow$ chooseArm$(B)$                               ▷ Based on Eqn. 7.8
 6: $i \leftarrow 0$
 7: while $i < n_{\text{train}}$ do
 8:     Sample word sequence $w^s$ from model
 9:     Calculate rewards $R^{\text{train}}$ based on $w^s$
10:     Optimize model's $L_{\text{RL}_a}$ loss using $R_a^{\text{train}}$
11:     if $i \mod n_{\text{bandit}} == 0$ then
12:         Evaluate model to get $R^{\text{val}}$
13:         $r \leftarrow$ scaled$(R_j^{\text{val}})$
14:         updateBandit$(B, a, r)$                             ▷ Based on Eqn. 7.10
15:         $a \leftarrow$ chooseArm$(B)$
16:     if $i \mod n_{\text{controller}} == 0$ then
17:         Evaluate model to get $R^{\text{val}}$
18:         $j \leftarrow \text{argmin}_k\{$scale$(R_k^{\text{val}})\}_{k=1}^{K}$
19:         $B \leftarrow$ chooseBandit$(C, j)$
20:         $a \leftarrow$ chooseArm$(B)$
21:     $i \leftarrow i + 1$
```

7.4.1 Question Generation

Baseline. Given a paragraph p, and an answer span a, the goal of the QG model is to generate a question q answering a. We follow the encoder-attention-decoder style architecture (see Fig. 7.4). The encoder is a bi-directional LSTM-RNN (Hochreiter and Schmidhuber, 1997) with self-attention (Wang et al., 2017), and the decoder is a uni-directional LSTM-RNN with attention (Luong et al., 2015) and pointer (Gu et al., 2016) mechanism, similar to Zhang and Bansal (2019). The input to the model is a concatenation of contextualized word representations (BERT (Devlin et al., 2019)), answer tag embedding (BIO tagging scheme), Part-of-Speech (POS) tag embedding, and Named-Entity (NER) tag embedding.

Rewards. We use ROUGE-L, QPP, and QAP (Zhang and Bansal, 2019) as rewards for this task. QPP is calculated as the probability of the generated question being the paraphrase of the ground-truth question via a classifier trained on Quora Question Pairs. QAP is calculated as the probability of a pre-trained QA model to correctly answer the given generated question as input.

Dataset & Evaluation. We use the SQuAD QG English dataset from Du et al. (2017) for the QG task, derived from SQuAD v1.1 (Rajpurkar et al., 2016), and the test set consists of 10% sampled examples from the training set, as the SQuAD test set is not open. For pre-processing, we do standard tokenization. We report on evaluation metrics including BLEU-4, METEOR, ROUGE-L, Q-BLEU1 (Nema and Khapra, 2018), as well as QPP and QAP (Zhang and Bansal, 2019).

7.4.2 Data-to-Text Generation

Baseline. Given a set of Resource Description Framework (RDF) triples,[16] the task is to generate a natural language text describing the facts in the RDF data. Following Zhao et al. (2020), we serialize and reorder the RDF data as an intermediate planning setup, and feed the plan into a seq2seq model with attention and copy mechanism.

Rewards. We use BLEU, ROUGE-L, and Entailment-Score (Pasunuru and Bansal, 2018) as rewards. Entailment-Score is calculated based on the probability that the generated sentence is classified as an entailment w.r.t. the ground truth.[17]

Dataset & Evaluation. We use the WebNLG dataset (Gardent et al., 2017) - a widely used English benchmark for data-to-text generation which focuses on micro-planning involving several subtasks like referring expression generation, aggregation, lexicalization, sentence segmentation, and surface realization. It contains 9,674 unique RDF triple-sets and 25,298 text references, which is divided into train, dev, and test sets.[18] We report all our results on the 'seen' and 'unseen' part of the test set. For each sample, the input is a set of up to 7 RDF triples from DBPedia, and the output is their text descriptions. The standard evaluation metrics for this dataset include

[16]Each triple contains a subject, a predicate, and an object.

[17]We use a RoBERTa classifier (Liu et al., 2019b) trained on MultiNLI (Williams et al., 2018a) as entailment scorer.

[18]`https://webnlg-challenge.loria.fr/`

METEOR[19] (Denkowski and Lavie, 2014b), BLEU (Papineni et al., 2002), and TER[20] (Snover et al., 2006). We also report ROUGE-L (Lin, 2004) and Entailment-Score (Pasunuru and Bansal, 2018).

7.4.3 Training Details

All the hyperparameters are tuned on the validation set for both question generation and data-to-text tasks. We use TITAN X and GeForce GTX 1080 GPUs for all our experiments. For the question generation task, we use two layers for both encoder and decoder. We set the hidden size of LSTM-RNN to 600 and use BERT-based contextual embeddings as input. We use a batch size of 32, encoder maximum length of 512 and decoder maximum length of 50, and maximum gradient clipping of 5. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-3 and 1e-6 for the cross-entropy and RL models, respectively. For data-to-text task, we use the same hyperparameters as discussed in Zhao et al. (2020) for the cross-entropy model, e.g., we use Adam with a batch size of 64 and an initial learning rate of 0.001. All RL models are initialized with the best cross-entropy model checkpoint, and use Adam with a learning rate of 1e-6. We refer to Appendix A.6 for full training details.

7.5 Experimental Results for DORB Framework

In this section, we present the performance of previous work, our cross-entropy baselines, our RL-based baselines, and finally our multi-arm bandit-based models. We start with results on automatic evaluation (Sec. 7.5.1-7.5.2). Next, we present results on human evaluation (Sec. 7.5.3). Finally, we present an interpretable analysis on the bandits (Sec. 7.5.4).

[19] `http://www.cs.cmu.edu/~alavie/METEOR/`

[20] `http://www.cs.umd.edu/~snover/tercom/`

Models	BLEU-4	METEOR	ROUGE-L	Q-BLEU1	QPP	QAP
BASELINES						
Cross-Entropy (Zhang and Bansal, 2019)	17.88	22.38	46.39	49.01	28.83	54.25
ROUGE-RL	18.03	22.55	46.64	49.52	29.09	55.07
QPP-RL	17.90	22.55	46.68	49.50	30.10	55.50
QAP-RL	18.22	22.69	46.65	49.72	30.03	**57.60**
MULTI-REWARD MODELS						
Pasunuru and Bansal (2018)†	18.36	22.55	46.75	49.66	30.03	56.51
Our SM-Bandit†	**18.68**	**22.88**	46.80	**50.02**	**30.15**	56.92
Our HM-Bandit†	18.55	22.82	**46.84**	50.01	30.07	56.78

Table 7.8: Performance of our baselines and multi-armed bandit-based models on question generation task. † denotes that these models use ROUGE-L, QPP, and QAP rewards during the optimization.

7.5.1 Results on Question Generation

Baselines. Table 7.8 presents results on the question generation dataset for our baselines. We use the previous state-of-the-art work (Zhang and Bansal, 2019) as our cross-entropy baseline. Next, we apply policy gradients-based reinforcement learning (RL) approach, and observe that all these models are better than the baseline in all metrics. Next, we will discuss the multi-reward RL models.

Multi-Armed Bandit Approaches. Finally, we evaluate our two bandit approaches: SM-Bandit and HM-Bandit as described in Sec. 7.3.4 and Sec. 7.3.5, respectively. Further, for a fair comparison of our multi-arm bandit-based models, we further implemented multi-reward alternate optimization approach introduced by Pasunuru and Bansal (2018) and considered it as baseline for our multi-reward models.[21,22] This model is slightly better than single reward-based RL baselines. Table 7.8 presents the performance of the proposed two bandit models (SM-Bandit and

[21] We do not compare with fixed weighted combination of metrics during RL optimization, as finding the optimal weighted combination is exponential complexity (searching among 100 values for n metrics needs 100^n tuning experiments), which we want to avoid via our bandit approach.

[22] We also experimented with the random choice of metrics during optimization. The results on the question generation task are very close to the baseline model (Pasunuru and Bansal, 2018): 18.31(BLEU), 22.50 (METEOR), 46.75 (ROUGE-L), 49.65 (Q-BLEU1), 30.04 (QPP), 56.56 (QAP). This is expected as the random choice baseline is same as uniform sampling of metrics, which is closer to alternate optimization.

Models	BLEU (↑)	METEOR (↑)	TER (↓)	ROUGE-L (↑)	Entailment (↑)
			BASELINES		
Cross-Entropy (Zhao et al., 2020)	63.14/22.56	44.85/27.79	33.97/71.02	74.25/49.83	99.27/88.16
ROUGE-RL	63.35/22.96	44.84/28.18	33.85/70.58	74.29/50.07	99.11/88.59
BLEU-RL	63.24/23.06	44.82/28.21	33.94/70.33	74.26/50.00	99.30/87.82
Ent-RL	63.28/22.49	44.96/28.11	34.03/72.29	74.29/49.99	99.84/90.57
			MULTI-REWARD MODELS		
Pasunuru and Bansal (2018)†	63.00/22.58	45.03/28.29	34.22/72.71	74.29/50.05	99.56/91.83
Our SM-Bandit†	**63.46/23.23**	**45.37**/28.68	33.59/**70.21**	74.38/**50.30**	100.13/92.35
Our HM-Bandit†	63.38/23.21	45.34/**28.70**	**33.58**/70.17	**74.39**/50.26	**100.21/92.40**

Table 7.9: Performance of our baselines and multi-arm bandit-based models on the 'seen/unseen' test set of WebNLG data-to-text task. The unseen set has categories that are not seen during the training, hence can be consider as a test-only transfer setup. † denotes that these models use ROUGE-L, BLEU, and Entailment rewards during the optimization. For TER metric, lower (↓) is better. For all other metrics, higher (↑) is better.

HM-Bandit) on various automatic evaluation metrics, and we observe that on average these models perform much better than the cross-entropy and single reward RL baseline models. Further, our bandit models also perform better than the multi-reward approach proposed by Pasunuru and Bansal (2018), suggesting that our bandit-based models are able to dynamically select the reward to optimize for overall improvement in all the metrics that we want to optimize. Also see discussion of significant improvements in human evaluation in Sec 7.5.3.

7.5.2 Results on Data-to-Text Generation

Baselines. Table 7.9 presents our baselines on the WebNLG data-to-text task. Our cross-entropy model is comparable to the very recent state-of-the-art model (Zhao et al., 2020). Further, we present single reward based RL models with ROUGE-L, BLEU, and Entailment score as rewards, which again perform better than our cross-entropy model. Next, we will discuss multi-reward models.

Multi-Armed Bandit Approaches. Table 7.9 also presents our multi-armed bandit models (SM-Bandit and HM-Bandit) which simultaneously use ROUGE-L, BLEU, and Entailment score as rewards. Again, we consider the model proposed by Pasunuru and Bansal (2018) as a baseline for

Model	ROUGE	PB (Pasunuru and Bansal, 2018)	SMB	HMB
QUESTION GENERATION TASK				
Relevance	4.28	4.40	4.56	4.55
Coherence	4.42	4.48	4.49	4.47
WEBNLG DATA-TO-TEXT TASK				
Relevance	4.61	4.68	4.79	4.81
Coherence	4.75	4.79	4.78	4.80

Table 7.10: Human evaluation results on QG and WebNLG tasks. ROUGE: ROUGE-L single-reward RL; PB (Pasunuru and Bansal, 2018): Pasunuru and Bansal (2018). Our SM-Bandit and HM-Bandit are statistically significantly better than ROUGE and PB models (see Sec. 7.5.3).

multi-reward models. On average, our bandit-based models perform better than all our baselines that are discussed in the above paragraph and also the model based on Pasunuru and Bansal (2018).[23] Also see discussion of significant improvements in human evaluation in Sec 7.5.3.

7.5.3 Human Evaluation

It is shown that RL models can game the metric that we use as the objective function (Paulus et al., 2018). This motivated us to optimize the RL models on multiple metrics simultaneously, thus trying to improve all the metrics and making the RL model hard to game any particular metric. In this section, we validate the superiority of our bandit models via human evaluation studies.

We performed anonymous human evaluation studies using Amazon Mechanical Turk (MTurk). We chose human annotators such that they are located in the USA, have at least 10,000 approved HITs, and have an approval rate of greater than 98%. For both question generation and WebNLG data-to-text, we considered 200 samples for each, and compared ROUGE-L RL, Pasunuru and Bansal (2018), SM-Bandit, and HM-Bandit models by asking the annotators to rate the quality of the generated outputs based on relevance and coherence on 5-point Likert scale.[24] Table 7.10

[23]In general, we observe better improvements with our bandit-based models in the unseen-test transfer setup.

[24]For question generation, relevance is defined as how clearly the generated question will be able to point to the right answer, given an input paragraph as context. For WebNLG data-to-text, relevance is defined as how related is the

Model	Pasunuru and Bansal (2018)	HM-Bandit
Relevance	3.49	3.68
Coherence	3.44	3.46

Table 7.11: Human evaluation results on WebNLG 'unseen' test set. Our HM-Bandit is statistically significantly better than Pasunuru and Bansal (2018) on relevance metric.

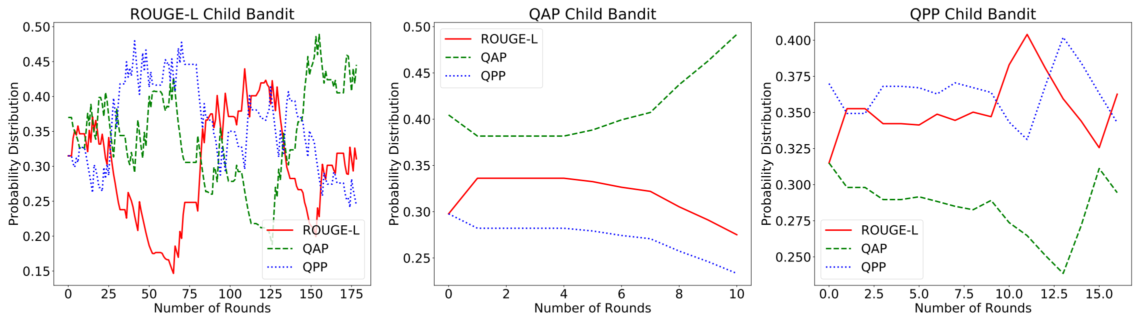


Figure 7.6: Plots showing the probability distribution of each child bandit of the HM-Bandit model on the QG task.

presents these human evaluation studies. In terms of relevance, our SM-Bandit and HM-Bandit models are significantly better than Pasunuru and Bansal (2018) (p<0.01) and ROUGE-L RL models (p<0.01) on question generation, while maintaining coherence.[25] On data-to-text, in terms of relevance, our SM-Bandit and HM-Bandit models are significantly better than Pasunuru and Bansal (2018) with p<0.03 and p<0.02, respectively. Also, both bandit models are significantly better than ROUGE-L RL model with p<0.01. We also performed a similar human evaluation study for the test-only transfer setup on the unseen WebNLG test set, and the results are in Table 7.11. Here also our bandit-based model (HM-Bandit) performed statistically significantly better than Pasunuru and Bansal (2018) on relevance metric with $p < 0.01$, while maintaining coherence.

generated description w.r.t. the given RDF data such as mentioning the facts. For both tasks, coherence is based on the logic, readability, and fluency of the generated question or description.

[25] We use bootstrap test (Efron and Tibshirani, 1994; Noreen, 1989) for calculating the statistical significance score.

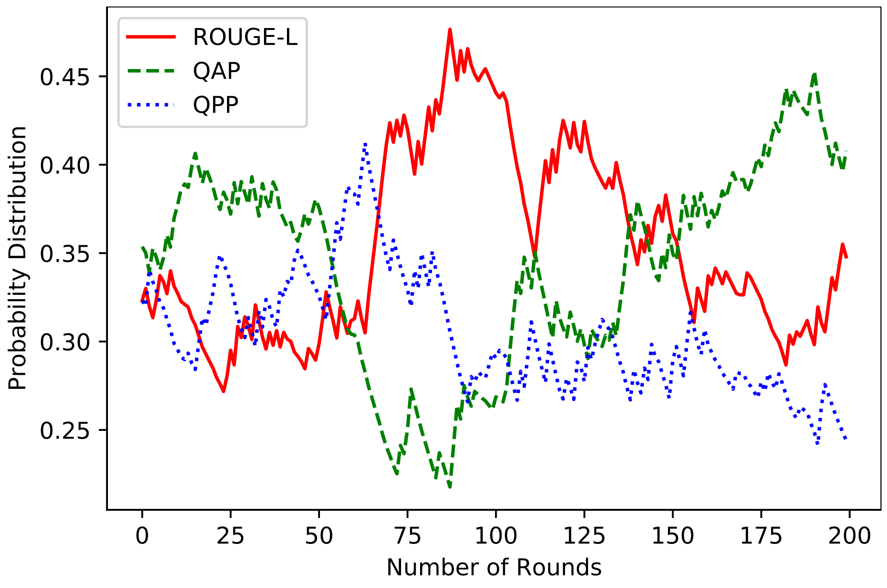


Figure 7.7: Plot showing the probability distribution of each arm of the SM-Bandit on question generation task.

7.5.4 Interpretable Bandit Analysis

Figure 7.7 presents the interpretable visualization of the probability distribution of each arm of the SM-Bandit as the training progresses. We observe that each metric has played an important role (as high probability arm) for at least a few rounds over the training trajectory. Also, there are multiple switchings of these metrics over the training trajectory, suggesting that this kind of automatic dynamic switching is important to improve the overall performance of RL models with multiple rewards.

Figure 7.6 presents the progress of child bandits of HM-Bandit during the training for question generation. As discussed in Sec. 7.3.5, these child bandits are controlled by a controller that selects the under-performing bandit. We observe that our HM-Bandit mostly used ROUGE-L child bandit for overall improvement in all metrics (as it is the under-performing metric). Further, each child bandit gave more importance to the metric that it wants to improve, e.g., the QAP child bandit gave more importance to the QAP arm. However, there is an exception for the ROUGE-L child bandit, where ROUGE-L arm is not the most important, suggesting that to improve the ROUGE-L metric other RL loss functions (QAP and QPP) are also useful.

7.6 Conclusion

We presented a multi-level, multi-task learning approach to incorporate natural language inference and paraphrasing knowledge into sentence simplification models, via soft sharing at higher-level semantic and lower-level lexico-syntactic levels. To automate MTL, We introduce a multi-armed bandits approach for learning a dynamic mixing ratio of tasks. We demonstrated strong simplification improvements on three standard datasets via automatic and human evaluation, and also discussed several ablation and analysis studies.

To automate multi-reward RL optimization, we presented novel approaches for dynamically optimizing multiple reward metrics simultaneously via multi-armed bandit approach in the context of language generation. We described two such mechanisms, namely single bandit and hierarchical bandit with multiple rewards. We conducted experiments on two challenging language generation tasks: question generation and data-to-text generation, and our method achieved strong improvements based on human evaluation over previous approaches. We further presented interpretable analysis on our bandit methods.

CHAPTER 8: CAS: CONTINUAL ARCHITECTURE SEARCH

8.1 Introduction

Architecture search enables automatic ways of finding the best model architecture and cell structures for the given task or dataset, as opposed to the traditional approach of manually choosing or tuning among different architecture choices, which introduces human inductive bias or is non-scalable. Recently, this idea has been successfully applied to the tasks of language modeling and image classification (Zoph and Le, 2017; Zoph et al., 2018; Cai et al., 2018; Liu et al., 2017a, 2018). The first approach of architecture search involved an RNN controller which samples a model architecture and uses the validation performance of this architecture trained on the given dataset as feedback (or reward) to sample the next architecture. Some recent attempts have made architecture search more computationally feasible (Negrinho and Gordon, 2017; Baker et al., 2017) via tree-structured search space or Q-learning with an ϵ-greedy exploration, and further improvements via a weight-sharing strategy called Efficient Neural Architecture Search (ENAS) (Pham et al., 2018).

In this work, we extend the architecture search approach to an important paradigm of transfer learning across multiple data sources: *continual learning*. The major problem in continual learning is *catastrophic forgetting*. For this, we introduce a novel 'continual architecture search' (CAS) approach, where the model parameters evolves and adapts when trained sequentially on a new task while maintaining the performance on the previously learned tasks. For enabling such continual learning, we formulate a two-step graph-initialization approach with conditions based on block sparsity and orthogonality.

For empirical evaluation of our approach, we choose three domains of natural language inference (NLI) bi-text classification tasks from the GLUE benchmark (Wang et al., 2019a): QNLI, RTE, and WNLI, and three domains of multimodal-generation based video captioning tasks: MSR-VTT (Xu et al., 2016a), MSVD (Chen and Dolan, 2011), and DiDeMo (Hendricks et al., 2017). Note that we are the first ones to use the architecture search approach for text classification tasks as well as multimodal conditioned-generation tasks, which achieves improvements on the strong GLUE and video captioning baselines.

Next, for continual learning, we train the three tasks sequentially for both text classification and video captioning (through our continual architecture search method) and show that this approach tightly maintains the performance on the previously-learned domain (also verified via human evaluation), while also significantly maximizing the performance on the current domain, thus enabling life-long learning (Chen and Liu, 2016). We also present various analyses for the evolution of the learned cell structure in the continual learning approach, which preserves the properties of certain edges while creating new edges for new capabilities.

8.2 Related Work

Neural architecture search (NAS) has been recently introduced for automatic learning of the model structure for the given dataset/task (Zoph and Le, 2017; Zoph et al., 2018), and has shown good improvements on image classification and language modeling. NAS shares some similarity to program synthesis and inductive programming (Summers, 1986; Biermann, 1978), and it has been successfully applied to some simple Q&A tasks (Liang et al., 2010; Neelakantan et al., 2015; Andreas et al., 2016; Lake et al., 2015). NAS was made more computationally feasible via tree-structured search space or Q-learning with ϵ-greedy exploration strategy and experience replay (Negrinho and Gordon, 2017; Baker et al., 2017), or a weight-sharing strategy among search space parameters called Efficient Neural Architecture Search (ENAS) (Pham et al., 2018). We explore architecture search for text classification and video caption generation tasks and their integration to the transfer learning paradigm of continual learning.

The major problem in continual learning is catastrophic forgetting. Some approaches addressed this by adding regularization to penalize functional or shared parameters' change and learning rates (Razavian et al., 2014; Li and Hoiem, 2017; Hinton et al., 2014; Jung et al., 2016; Kirkpatrick et al., 2017; Donahue et al., 2014; Yosinski et al., 2014). Others proposed copying the previous task and augmenting with new task's features (Rusu et al., 2016), intelligent synapses to accumulate task-related information (Zenke et al., 2017), or online variational inference (Nguyen et al., 2017). Also, Yoon et al. (2018) proposed a dynamically expandable network based on incoming new data. In our work, we introduce 'continual architecture search' by extending the NAS paradigm to avoid catastrophic forgetting via block-sparsity and orthogonality constraints, hence enabling a form of life-long learning (Chen and Liu, 2016). To the best of our knowledge, our work is the first to extend architecture search to a continual incoming-data setup. Elsken et al. (2019) and So et al. (2019) proposed evolutionary architecture search algorithms that dynamically allocate more resources for promising architecture candidates, but these works are different from us in that they do not consider the case where we have continual incoming-data from different data sources, but instead focus on the continual evolution of the model search for efficiency purposes.

8.3 Architecture Search for Text Classification and Generation

In this section, we first discuss how we adapt ENAS (Pham et al., 2018) for modeling our bi-text classification and multimodal video captioning tasks. Next, we introduce our continual approach of transfer learning leveraging architecture search.

8.3.1 ENAS Algorithm

Our initial architecture search approach is based on the recent Efficient Neural Architecture Search (ENAS) method of Pham et al. (2018), but modeled for text classification and generation-based video captioning. Fig. 8.1 presents the ENAS controller for sampling an RNN cell structure, which we use to learn the two encoders of our text classification model or encoder-decoder for our video captioning model. The controller is a simple LSTM-RNN and the classifier encoder's or video captioning encoder-decoder's RNN cell structure is based on the combination of N nodes indexed by $h_1^{(t)}, h_2^{(t)}, .., h_N^{(t)}$ (edges between nodes represent weight parameters) and activation functions (ReLU, tanh, sigmoid, identity), where t denotes the time step. For node $h_1^{(t)}$, there are two inputs: $x^{(t)}$ (input signal) and $h_N^{(t-1)}$ (output from previous time-step), and the node computations are:

$$c_1^{(t)} = \text{sigmoid}(x^{(t)} \cdot W^{(x,c)} + h_N^{(t-1)} \cdot W_0^{(c)}) \tag{8.1}$$

$$\begin{aligned} h_1^{(t)} = c_1^{(t)} \odot f_1(x^{(t)} \cdot W^{(x,h)} + h_N^{(t-1)} \cdot W_1^{(h)}) \\ + (1 - c_1^{(t)}) \odot h_N^{(t-1)} \end{aligned} \tag{8.2}$$

where f_1 is the activation function. Node h_l, where $l \in \{2, 3, .., N\}$, receives input from node j_l where $j_l \in \{h_1, h_2, .., h_{l-1}\}$, and the computation is defined as follows:

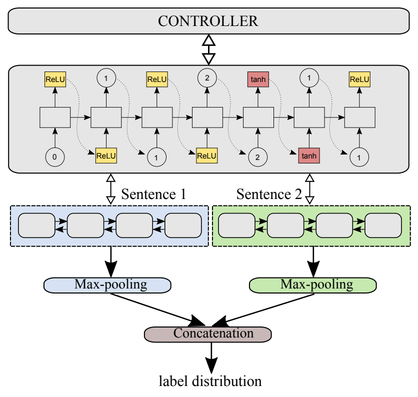
$$c_l^{(t)} = \text{sigmoid}(h_{j_l}^{(t)} \cdot W_{l,j_l}^{(c)}) \tag{8.3}$$

$$h_l^{(t)} = c_l^{(t)} \odot f_l(h_{j_l}^{(t)} \cdot W_{l,j_l}^{(h)}) + (1 - c_l^{(t)}) \odot h_{j_l}^{(t)} \tag{8.4}$$

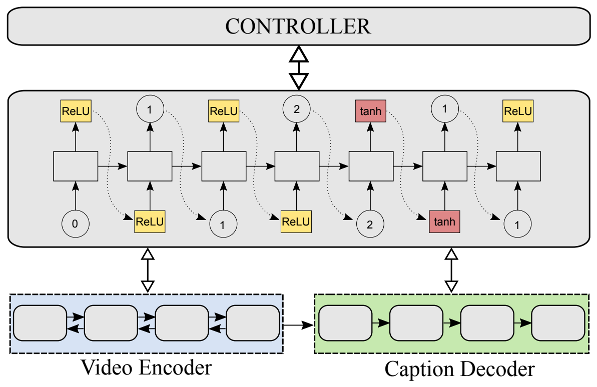
During training, we alternately train the model parameters and controller parameters. First, we sample a Directed Acyclic Graph (DAG) structure from the controller at every mini-batch and use it to update the weight parameters of the task's RNN nodes/parameters. Next, we sample a DAG from the controller and measure the (validation) performance of that structure based on this new updated state of the task model, and use this performance as a reward to allow the controller to update its own parameters. We repeat this alternate training procedure until the model converges. Later, we select the DAG structure with the best performance and use it to retrain the model from scratch.

8.3.2 ENAS for Bi-Text Classification

For our NLI text classification tasks, we are given the sentence pair as input, and we have to classify it as entailment or not. For a strong base model, we follow Conneau et al. (2017) model, and use bidirectional LSTM-RNN encoders to encode both the sentences and then we do max-



(a) Text classification ENAS.



(b) Video captioning ENAS.

Figure 8.1: Architecture search models for bi-text classification and video caption generation tasks.

pooling on the outputs from these encoders. Let v represent the max-pooling output from the first sentence encoder and u represent the max-pooling output from the second sentence encoding. The joint representation h is defined as $h = [u; v; |u - v|; u \odot v]$. The final representation is linearly projected to the label classes, and then fed through softmax to get the final class distribution. Fig. 8.1a presents an overview of our text classification model along with ENAS controller for sampling an RNN cell structure. We sample an RNN cell structure from the ENAS controller and use it in the two recurrent encoders of the bi-text classification model. In the first stage, we learn the best cell structure, by sampling multiple cell structures and giving the corresponding validation accuracy as the feedback reward to the controller. In the second stage, we use the best cell structure from the stage-1 to retrain the text classification model from scratch.

8.3.3 ENAS for Conditioned Generation

Next, we go beyond text classification, and look at conditioned text generation with ENAS, where we choose the task of video-conditioned text generation (also known as video captioning) so as to also bring in a multi-modality aspect. For a strong baseline, we use a sequence-to-sequence model with an attention mechanism similar to Pasunuru and Bansal (2017a), where we encode the video frames as a sequence into a bidirectional LSTM-RNN and decode the caption through another LSTM-RNN (see Fig. 8.1b). Our attention mechanism is similar to Bahdanau et al. (2015), where at each time step t of the decoder, the LSTM hidden state s_t is a non-linear function of previous time step's decoder hidden state s_{t-1} and generated word w_{t-1}, and the context vector c_t which is a weighted combination of the encoder hidden states $\{h^i\}$. These weights α_t, are defined as:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{n} \exp(e_{t,k})} \tag{8.5}$$

The attention function $e_{t,i} = w^T \tanh(W_a h_i + U_a s_{t-1} + b_a)$, where w, W_a, U_a, b_a are learned parameters. Fig. 8.1b presents our video captioning model along with ENAS controller. Here, we sample an RNN cell structure from the ENAS controller and use it for both encoder and decoder, and rest of the ENAS procedure is similar to Sec. 8.3.2.

8.4 Continual Architecture Search (CAS)

We introduce a novel continual learning paradigm on top of architecture search, where the RNN cell structure evolves when trained on new incoming data/domains, while maintaining the performance on previously learned data/domains (via our block-sparsity and orthogonality conditions discussed below), thus enabling life-long learning (Chen and Liu, 2016). Let $\theta_{1,k} \in \theta_1$ and $\theta_{2,k} \in \theta_2$ (where k denotes model parameters) be the learned model parameters for task T when independently trained on datasets d_1 and d_2. Then, we can say that $\theta_{2,k} = \theta_{1,k} + \psi_{2,k}$, where, $\psi_{2,k}$ is the change in the model parameters of $\theta_{1,k}$ when trained independently on d_2. There are infinitely many possible local optimal solutions for $\psi_{2,k}$, hence in our continual learning approach, we want to learn the parameters $\psi_{2,k}$ when training on dataset d_2 such that it will not affect the performance of the task w.r.t. dataset d_1. For this, we formulate two important conditions:

Condition 1. *When training the model on dataset d_1, we constrain the model parameters $\theta_{1,k} \in \mathrm{R}^{m\times n}$ to be sparse, specifically, to be block sparse, i.e., minimize $\sum_{i=1}^{m} |(||\theta_{1,k}[i,:]||_2)|_1$.*

Here, $||\cdot||_2$ represents the l_2 norm and $||\cdot||_1$ represents the l_1 norm. l_2 and l_1 norms are efficient in avoiding over-fitting; however, they are not useful for compact representation of the

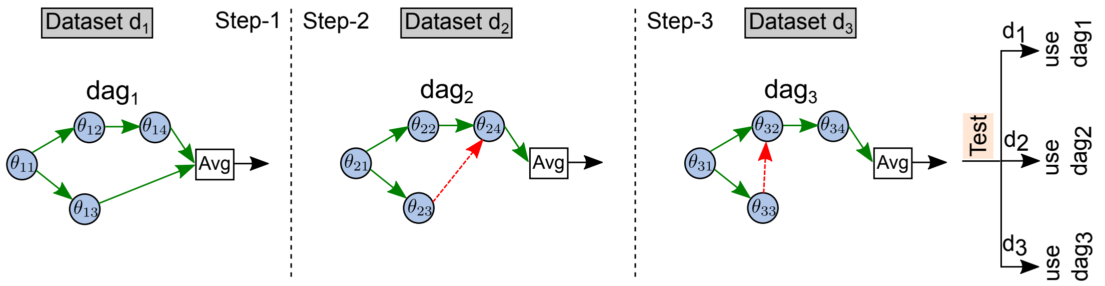


Figure 8.2: Continual architecture search (CAS) approach: green, solid edges (weight parameters) are shared, newly-learned edges are represented with red, dashed edges.

network. Scardapane et al. (2017) proposed group sparsity in the neural networks to completely disconnect some neurons. Our block sparse condition is inspired from their work. This sparsity condition is also useful for our continual learning approach which we discuss in Condition 2.

Condition 2. *When training the model on dataset d_2, we start from $\theta_{1,k}$, keep it constant, and update $\psi_{2,k}$ such that:*

1. *$\psi_{2,k}$ is block sparse, i.e., minimize $\sum_{i=1}^{m} |(||\psi_{2,k}[i,:]||_2)|_1$.*
2. *$\theta_{1,k}$ and $\psi_{2,k}$ are orthogonal.*

It is important in the continual learning paradigm that we do not affect the previously learned knowledge. As stated in Condition 1, we find a block sparse solution $\theta_{1,k}$ such that we find the solution $\theta_{2,k}$ which is close to $\theta_{1,k}$ and the new knowledge is projected in orthogonal direction via $\psi_{2,k}$ so that it will not affect the previously learned knowledge, and thus 'maintain' the performance on previously learned datasets. We constrain the closeness of $\theta_{2,k}$ and $\theta_{1,k}$ by constraining $\psi_{2,k}$ to also be block sparse (Condition 2.1). Also, to avoid affecting previously learned knowledge, we constrain $\theta_{1,k}$ and $\psi_{2,k}$ to be orthogonal (Condition 2.2). However, strictly imposing this condition into the objective function is not feasible (Bousmalis et al., 2016), hence we add a penalizing term into the objective function as an approximation to the orthogonality condition: $L_p(\theta_{2,k}) = ||\theta_{1,k}^T \cdot \psi_{2,k}||_2^2$. Both Condition 2.1 and 2.2 are mutually dependent, because for two matrices' product to be zero, they share basis vectors between them, i.e., for an n-dimensional space, there are n basis vectors and if p of those vectors are assigned to one matrix, then the rest of the

$n - p$ vectors (or subset) should be assigned to the other matrix.[1] If we fill the rest of the rows with zeros, then they are block sparse, which is the reason for using Condition 2.1. Our CAS condition ablation (see Sec. 8.6.1) shows that both these conditions are necessary for continual learning.

Next, we describe the integration of our above continual learning approach with architecture search, where the model continually evolves its cell architecture so as to perform well on the new incoming data, while also tightly maintaining the performance on previously learned data (or domains). Fig. 8.2 presents an overview of our continual learning integration approach into architecture search for sequential training on three datasets. Initially, given the dataset d_1, we train the architecture search model to find the best Directed Acyclic Graph (DAG) structure for RNN cell and model parameters $\theta_{1,k}$ under the block sparse condition described above in Sec. 8.4. We call this step-1, corresponding to dataset d_1. Next, when we have a new dataset d_2 from a different domain, we further continue to find the best DAG and model parameters $\theta_{2,k}$ for best performance on d_2, but initialized the parameters with step-1's parameters $\theta_{1,k}$, and then trained on dataset d_2 following Condition 2 (discussed in Sec. 8.4). We call this step-2, corresponding to dataset d_2. After the end of step-2 training procedure, for re-evaluating the model's performance back on dataset d_1, we still use the final learned model parameters $\theta_{2,k}$, but with the learned DAG from step-1.[2] This is because we cannot use the old step-1 model parameters $\theta_{1,k}$ since we assume that those model parameters are not accessible now (assumption for continual learning with large incoming data streams and memory limit for saving large parameter sets).

[1]Note that it is not necessary for the matrix to contain all of the $n - p$ basis vectors, if the matrix rank is less than n, then it may have less than $n - p$ basis vectors.

[2]For evaluating the model's performance on dataset d_2, we obviously use the final learned model parameters $\theta_{2,k}$, and the learned DAG from step-2.

8.5 Experimental Setup

8.5.1 Text Classification Datasets

We choose the natural inference datasets of QNLI, RTE, and WNLI from the GLUE (Wang et al., 2019a) benchmark to perform experiments for multi-task cell structure and continual architecture search. We use the standard splits provided by (Wang et al., 2019a).

QNLI Dataset: Question-Answering Natural Language Inference (QNLI) is extracted from the Stanford Question Answering Dataset (Rajpurkar et al., 2016), where they created sentence pair classification task by forming a pair between each question and the corresponding sentence containing the answer. Hence the task is to find whether the given sentence context contains the answer for the given question. In this dataset, we use the standard splits, i.e., 108k examples for training, 5.7k for validation, and 5.7k for testing.

RTE Dataset: Recognizing Textual Entailment (RTE) is collected from a series of annual challenges on the task of textual entailment. This dataset spans the news and Wikipedia text. Here, the task is to predict whether the sentence pair is entailment or not. In this dataset, we use the standard splits, i.e., 2.5k examples for training, 276 for validation, and 3k for testing.

WNLI Dataset: Winograd Natural Language Inference (WNLI) is extracted from the dataset of Winograd Schema Challenge for reading comprehension task. Original dataset is converted into a sentence pair classification task by replacing the ambiguous pronoun with each possible referent, where the task is to predict if the sentence with the substituted pronoun is entailed by the original sentence. We use 634 examples for training, 71 for validation, and 146 for testing.

8.5.2 Video Captioning Datasets

For the conditioned-generation paradigm, we use three popular multimodal video captioning datasets: MSR-VTT, MSVD, and DiDeMo to perform experiments for continual architecture search and multi-task architecture search.

MSR-VTT Dataset: MSR-VTT is a collection of $10,000$ short videos clips collected from a commercial search engine covering 41.2 hours of video and annotated through Amazon Mechanical Turk (AMT). Each video clip has 20 human annotated captions. We used the standard splits following previous work, i.e., $6,513$ video clips as training set, 497 as validation set, and $2,990$ as test set.

MSVD Dataset: Microsoft Video Description Corpus (MSVD) is a collection of 1970 short video clips collected in the wild and annotated through Amazon Mechanical Turk (AMT) in different languages. In this work, we use only English language annotations. Each video clip on an average is 10 seconds in length and approximately 40 annotations. We use the standard splits following previous work, i.e., $1,200$ video clips as training set, 100 as validation set, and 670 as test set.

DiDeMo Dataset: Distinct Describable Moments (DiDeMo) is traditionally a video localization task w.r.t. given description query (Hendricks et al., 2017). In this work, we use it as a video description task where given the video as input we have to generate the caption. We use the standard splits as provided by Hendricks et al. (2017).

8.5.3 Evaluation

For GLUE tasks, we use accuracy as an evaluation metric following the previous work (Wang et al., 2019a). For video captioning tasks, we report four diverse automatic evaluation metrics: METEOR (Denkowski and Lavie, 2014a), CIDEr (Vedantam et al., 2015), BLEU-4 (Papineni et al., 2002), and ROUGE-L (Lin, 2004). We use the standard evaluation code (Chen et al., 2015) to obtain these scores for our generated captions w.r.t. the reference captions.

8.5.4 Training Details

In all our experiments, our hyperparameter choices are based on validation set accuracy for GLUE tasks and an average of the four automatic evaluation metrics (METEOR, CIDEr, BLEU-

4, and ROUGE-L) for video captioning tasks. We use same settings for both normal and architecture search models, unless otherwise specified. More details in appendix.

8.6 Experimental Results and Insights

8.6.1 Continual Learning on GLUE Tasks

Baseline Models: We use bidirectional LSTM-RNN encoders with max-pooling (Conneau et al., 2017) as our baseline.[3] Further, we used the ELMo embeddings (Peters et al., 2018) as input to the encoders, where we allowed to train the weights on each layer of ELMo to get a final representation. Table 8.1 shows that our baseline models achieve strong results when compared with GLUE benchmark baselines (Wang et al., 2019a).[4] On top of these strong baselines, we add ENAS approach.

ENAS Models: Next, Table 8.1 shows that our ENAS models (for all three tasks QNLI, RTE, WNLI) perform better or equal than the non-architecture search based models.[5] Note that we only replace the LSTM-RNN cell with our ENAS cell, rest of the model architecture in ENAS model is same as our baseline model.[6]

CAS Models: Next, we apply our continual architecture search (CAS) approach on QNLI, RTE, and WNLI, where we sequentially allow the model to learn QNLI, RTE, and WNLI (in the order of decreasing dataset size, following standard transfer setup practice) and the results are as shown in Table 8.1. We train on QNLI task, RTE task, and WNLI task in step-1, step-2, and step-3, respectively. We observe that even though we learn the models sequentially, we are able to

[3]We also tried various other models e.g., self-attention and cross-attention, but we found that the max-pooling approach performed best on these datasets.

[4]We only report single-task (and not 9-task multi-task) results from the GLUE benchmark for fair comparison to our models.

[5]On validation set, our QNLI ENAS model is statistically significantly better than the corresponding baseline with $p < 0.01$, and statistically equal on RTE and WNLI (where the validations sets are very small), based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples. Since the test set is hidden, we are not able to calculate the statistical significance on it.

[6]Note that ENAS random search baseline vs. optimal search validation performance on QNLI, RTE, and WNLI are 73.3 (vs. 74.8), 58.8 (vs. 60.3), and 54.0 (vs. 55.6), respectively, suggesting that the learned optimal cell structure is better than the random cell structure.

Models	QNLI	RTE	WNLI
PREVIOUS WORK			
BiLSTM+ELMo (Wang et al., 2019a)	69.4	50.1	65.1
BiLSTM+ELMo+Attn (Wang et al., 2019a)	61.1	50.3	65.1
BASELINES			
Baseline (with ELMo)	73.2	52.3	65.1
ENAS (Architecture Search)	74.5	52.9	65.1
CAS RESULTS			
CAS Step-1 (QNLI training)	73.8	N/A	N/A
CAS Step-2 (RTE training)	73.6	54.1	N/A
CAS Step-3 (WNLI training)	73.3	54.0	64.4

Table 8.1: Test results on GLUE tasks for various models: Baseline, ENAS, and CAS (continual architecture search). The CAS results maintain statistical equality across each step.

maintain performance on the previously-learned QNLI task in step-2 (74.1 vs. 74.2 on validation set which is statistically equal, and 73.6 vs. 73.8 on test).[7] Note that if we remove our sparsity and orthogonality conditions (Sec. 8.4), the step-2 QNLI performance drops from 74.1 to 69.1 on validation set, demonstrating the importance of our conditions for CAS (see next paragraph on 'CAS Condition Ablation' for more details). Next, we observe a similar pattern when we extend CAS to the WNLI dataset (see step-3 in Table 8.1), i.e, we are still able to maintain the performance on QNLI (as well as RTE now) from step-2 to step-3 (scores are statistically equal on validation set).[8] Further, if we compare the performance of QNLI from step-1 to step-3, we see that they are also stat. equal on val set (73.9 vs. 74.2). This shows that our CAS method can maintain the performance of a task in a continual learning setting with several steps.

CAS Condition Ablation: We also performed important ablation experiments to understand the importance of our block sparsity and orthogonality conditions in the CAS approach (as discussed in Sec. 8.4). Table 8.2 presents the ablation results of QNLI in step-2 with CAS conditions. Our full model (with both Condition 2.1 and 2.2) achieves a validation performance of 74.1. Next, we separately experimented with each of Condition 2.1 and 2.2 and observe that using only one

[7]Note that there is a small drop in QNLI performance for CAS Step-1 vs. ENAS (74.5 vs. 73.8); however, this is not true across all experiments, e.g., in case of RTE, CAS Step-1 is in fact better than its corresponding ENAS model (ENAS: 52.9 vs. CAS Step-1: 53.8).

[8]On validation set, QNLI step-3 vs. step-2 performance is 73.9 vs. 74.1, which is stat. equal. Similarly, on RTE, step-3 vs. step-2 performance is 61.0 vs. 60.6 on validation set, which is again statistically equal.

Model	Accuracy on QNLI
No Condition with RTE DAG	54.1
No Condition	69.1
Only Condition 2.1	71.5
Only Condition 2.2	69.4
Full Model (Condition 2.1 & 2.2)	74.1

Table 8.2: Ablation (val) results on CAS conditions.

Models	MSR-VTT					MSVD				
	C	B	R	M	AVG	C	B	R	M	AVG
Baseline (Pasunuru and Bansal, 2017b)	48.2	40.8	60.7	28.1	44.5	85.8	52.5	71.2	35.0	61.1
ENAS	48.9	41.3	61.2	28.1	44.9	87.2	52.9	71.7	35.2	61.8
CAS Step-1 (MSR-VTT training)	48.9	41.1	60.5	27.5	44.5	N/A	N/A	N/A	N/A	N/A
CAS Step-2 (MSVD training)	48.4	40.1	59.9	27.1	43.9	88.1	52.4	71.3	35.1	61.7

Table 8.3: Video captioning results with Baseline, ENAS, and CAS models. Baseline is reproduced numbers from github of Pasunuru and Bansal (2017b) which uses advanced latest visual features (ResNet-152 and ResNeXt-101) for video encoder. C, B, R, M: CIDEr, BLEU-4, ROUGE-L, and METEOR metrics.

condition at a time is not able to maintain the performance w.r.t. step-1 QNLI performance (the decrease in score is statistically significant), suggesting that both of these two conditions are important for our CAS approach to work. Further, we remove both conditions and observe that the performance drops to 69.1. Finally, we also replaced the QNLI cell structure with the RTE cell structure along with removing both conditions and the performance further drops to 54.1. This shows that using the cell structure of the actual task is important.

Time Comparison: We compare QNLI training time on a 12GB TITAN-X Nvidia GPU. Our baseline non-ENAS model takes 1.5 hours, while our CAS (and MAS) models take approximately the same training time (4 hours) as the original ENAS setup, and do not add extra time complexity.

8.6.2 Continual Learning on Video Captioning

Baselines Models: Our baseline is a sequence-to-sequence model with attention mechanism as described in Sec. 8.3.3. We achieve comparable results w.r.t. SotA (see Table 8.3), hence serving as a good starting point for the ENAS approach.

ENAS Models: Table 8.3 also shows that our ENAS models (MSR-VTT, MSVD) perform equal/better than non-architecture search based models.[9]

CAS Models: Next, we apply our continual architecture search (CAS) approach on MSR-VTT and MSVD, where we sequentially allow the model to learn MSR-VTT first and then MSVD, and the results are as shown in Table 8.3. We observe that even though we learn the models sequentially, we are able to maintain performance on the previously-learned MSR-VTT task in step-2, while also achieving greater-or-equal performance on the current task of MSVD in comparison with the general ENAS approach.[10]

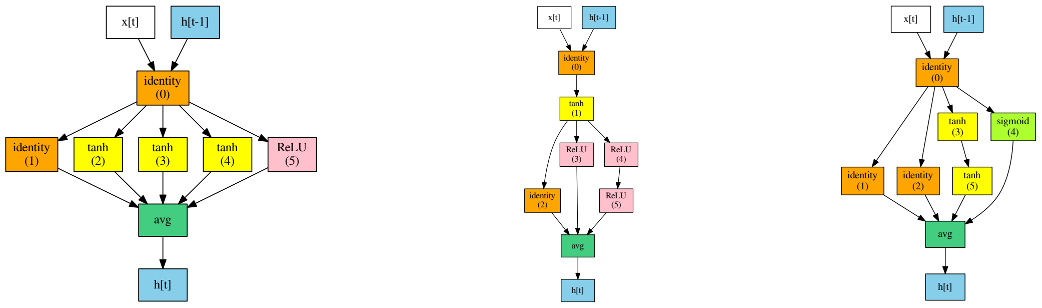
Human Evaluation: We also performed human comparison of our CAS step-1 vs. step-2 via Amazon MTurk (100 anonymized test samples, Likert 1-5 scale). This gave an overall score of 3.62 for CAS step-1 model vs. 3.55 for CAS step-2, which are very close (statistically insignificant with $p = 0.32$), again showing that CAS step-2 is able to maintain performance w.r.t. CAS step-1.

8.6.3 Insights

Evolved Cell Structure with CAS Fig. 8.3 presents the cell structure in each step for the CAS approach, where we sequentially train QNLI, RTE, and WNLI tasks. Overall, we observe that the cell structures in CAS preserve the properties of certain edges while creating new edges for new capabilities. We notice that the cell structure in step-1 and step-2 share some common edges and activation functions (e.g., inputs to node 0) along with some new edge connections in step-2

[9]Note that ENAS random search performance on MSR-VTT test set is C:43.3, B:37.0, R:58.7, M:27.3, AVG: 41.6; and on MSVD test set is C:83.7, B:47.4, R:71.1, M:33.6, AVG: 59.0, suggesting that these are lower than the learned optimal cell structures' performances shown in Table 8.3.

[10]MSR-VTT performance in step-1 and step-2 are stat. equal on CIDEr and ROUGE-L metrics.



(a) Step-1 (b) Step-2 (c) Step-3

Figure 8.3: Learned cell structures for step-1, step-2, and step-3 of continual architecture search for GLUE tasks.

(e.g., node 1 to node 3). Further, we observe that the step-3 cell uses some common edges w.r.t. the step-2 cell, but uses different activation functions, e.g., edge between node 0 and node 1 is the same, but the activation function is different. This shows that those edges are learning weights which are stable w.r.t. change in the activation functions.

8.7 Conclusion

We first presented an architecture search approach for text classification and video caption generation tasks. Next, we introduced a novel paradigm of transfer learning by combining architecture search with continual learning to avoid catastrophic forgetting.

CHAPTER 9: SUMMARY, LIMITATIONS, AND FUTURE WORK

9.1 Summary of Contributions

We presented novel multi-objective learning approaches in the context of multi-task learning, multi-reward reinforcement learning, and continual learning for various multi-modal text generation tasks and text classification tasks. We improved video captioning and textual summarization by jointly training them with their related auxiliary tasks via multi-task learning. This essentially proves that we can share the knowledge of auxiliary tasks to improve a given primary task. Similarly, we improved the video captioning and textual summarization tasks by inducing knowledge from other related tasks in the form of rewards. We further proposed novel and effective ways of inducing multiple skills by 'dynamically' choosing the auxiliary tasks (in MTL) or rewards (in RL) during the training in an automatic way using multi-armed bandits based approaches. In the direction of sequential training of related tasks, we proposed a novel paradigm of transfer learning by combining architecture search with continual learning to avoid catastrophic forgetting. We empirically tested our method on text classification and video caption generation tasks.

9.2 Limitations and Future Work

Multi-Task Learning. In our multi-task learning approaches for text generation tasks, we have intuitively figured out what auxiliary tasks make sense for a given primary task. This scenario is not always realistic, especially when the number of choices of auxiliary tasks increases. Addressing this issue, we recently proposed a method that automatically select the most useful auxiliary tasks for a given primary task via a Beta-Bernoulli multi-armed bandit with Thompson Sampling (Guo et al., 2019a). However, the limitation of the current MTL methods is about what parameters to share across the related tasks. Our work has explored intuitive ways of what parameters to share, but this is not realistic with the increase in the number of related tasks or the complexity of the models. In future work, we would like to explore this direction.

Further, it would also be interesting to build a large-scale multi-task model that jointly trains multiple NLP tasks (more than 10 tasks) together creating a unified model that can do inference on multiple tasks. This could greatly reduce the computational cost of using multiple models for various tasks/scenarios. Another interesting direction of our MTL work is to extend it to the recent large-scale pre-trained language models (Devlin et al., 2019; Liu et al., 2019b), where we can jointly fine-tune a pre-trained language model along with training various NLP tasks to leverage the pre-existing knowledge of these pre-trained language models.

Multi-Reward Reinforcement Learning. In our multi-reward RL approaches, we proposed to dynamically optimize multiple reward metrics simultaneously using multi-armed bandits. However, there are few limitations to this approach, e.g., RL methods have high variance and our bandit approach doesn't consider the non-stationary aspect of the model during training, further leading to more variance during the RL training. In future work, we would like to explore other variants of bandits that also consider the non-stationary aspect. Further, our work has only shown our method to apply on a maximum of three or four rewards. It would be an interesting future direction to use multiple rewards at scale (more than 10 rewards) in the hopes that these multiple rewards can approximate human-level feedback for RL training.

Continual Learning. In our continual architecture search (CAS), we have successfully shown that we can retrain the performance of previously trained tasks while continually training new tasks in a sequential manner. We have shown it for three NLI tasks. We observed that the old tasks are slowly losing some performance when we add new tasks. If we sequentially train more tasks, there will be a point where old tasks start showing a significant drop in their performances. This is a limitation of our CAS approach. Note that scaling the continual learning to a lot of tasks is one of the biggest challenges of AI.

My dissertation looked at the continual learning of various tasks in a sequential fashion. One could also do continual learning within the same task where the new examples try to teach the model about new linguistic phenomena or correct a class of errors. The examples that provide supervision to correct mistakes or learn a phenomenon are often hard or impossible to acquire (e.g., due to privacy or ethics issues) (Wang et al., 2020). Hence, it is important to effectively learn to correct mistakes using few extra training examples. Recent work has shown the generalization capability of large pre-trained models to handle multiple tasks with zero to few training examples (Schick and Schütze, 2021; Brown et al., 2020; Yin et al., 2020). For example, Yin et al. (2020) has shown that system trained for NLI can be used to perform new tasks zero-shot, i.e., without any task-specific training data. We believe that similar models can be used to rapidly learn to correct a phenomenon within the same task from a few (e.g., 10 or 15) training examples. Towards addressing this problem, curating few-shot datasets which can try to correct a class of errors or teach a new linguistic phenomenon could be an interesting future direction.

APPENDIX A: EXPERIMENTAL SETUP AND TRAINING DETAILS

A.1 Full Training Details for MTL Video Captioning.

In all of our experiments, we tune all the model hyperparameters on validation (development) set of the corresponding dataset. We consider the following short hyperparameters ranges and tune lightly on: LSTM-RNN hidden state size - $\{256, 512, 1024\}$; learning rate in the range $[10^{-5}, 10^{-2}]$ with uniform intervals on a log-scale; weight initializations in the range $[-0.1, 0.1]$ and mixing ratios in the range 1:$[0.01, 3]$ with uniform intervals on a log-scale. We use the following settings in all of our models (unless otherwise specified in a subsection below): we unroll video encoder/decoder LSTM-RNNs to 50 time steps and language encoder/decoder LSTM-RNNs to 30 time steps. We use a 1024-dimension LSTM-RNN hidden state size. We use 512-dimension vectors to embed frame level visual features and word vectors. These embedding weights are learned during the training. We use the Adam optimizer (Kingma and Ba, 2015) with default coefficients and a batch size of 32. We apply a dropout with probability 0.5 to the vertical connections of LSTM (Zaremba et al., 2014) to reduce overfitting.

A.1.1 Video Captioning on YouTube2Text

Baseline and Attention Models Our primary baseline model (Inception-v4, attention, ensemble) uses a learning rate of 0.0001 and initializes all its weights with a uniform distribution in the range $[-0.05, 0.05]$.

Multi-Task with Video Prediction (1-to-M) In this model, the video captioning and unsupervised video prediction tasks share their encoder LSTM-RNN weights and image embeddings in a one-to-many multi-task setting. We again use a learning rate of 0.0001 and initialize all the learnable weights with a uniform distribution in the range $[-0.05, 0.05]$. Two important hyperparameters tuned (on the validation set of captioning datasets) are the ratio of encoder vs decoder frames for video prediction on UCF-101 (where we found that 80% of frames as input and 20%

for prediction performs best); and the mini-batch mixing ratio between the captioning and video prediction tasks (where we found $100 : 200$ works well).

Multi-Task with Entailment Generation (M-to-1) In this model, the video captioning and entailment generation tasks share their language decoder LSTM-RNN weights and word embeddings in a many-to-one multi-task setting. We again use a learning rate of 0.0001. All the trainable weights are initialized with a uniform distribution in the range $[-0.08, 0.08]$. We observe that a mixing ratio of $100 : 50$ (between the captioning and entailment generation tasks) alternating mini-batches works well here.

Multi-Task with Video and Entailment Generation (M-to-M) In this many-to-many, three-task model, the video encoder is shared between the video captioning and unsupervised video prediction tasks, and the language decoder is shared between the video captioning and entailment generation tasks. We again use a learning rate of 0.0001. All the trainable weights are initialized with a uniform distribution in the range $[-0.08, 0.08]$. We found that a mixing ratio of $100 : 100 : 50$ alternative mini-batches of video captioning, unsupervised video prediction, and entailment generation works best.

A.1.2 Video Captioning on MSR-VTT

We also evaluate our many-to-many multi-task model on other video captioning datasets. For MSR-VTT, we train the model again using a learning rate of 0.0001. All the trainable weights are initialized with a uniform distribution in the range $[-0.05, 0.05]$. We found that a mixing ratio of $100 : 20 : 20$ alternative mini-batches of video captioning, unsupervised video prediction, and entailment generation works best.

A.1.3 Video Captioning on M-VAD

For the M-VAD dataset, we use 512 dimension hidden vectors for the LSTMs to reduce overfitting. We initialize the LSTM weights with a uniform distribution in the range $[-0.1, 0.1]$ and all other weights with a uniform distribution in the range $[-0.05, 0.05]$. We use a learning rate of 0.001. We found a mixing ratio of $100 : 5 : 5$ alternative mini-batches of video captioning, unsupervised video prediction, and entailment generation works best.

A.1.4 Entailment Generation

Here, we use video captioning to in turn help improve entailment generation results. We use the same hyperparameters for both the baseline and the multi-task model (Sec. 5.3 and Table 4). We use a learning rate of 0.001. All the trainable weights are initialized with a uniform distribution in the range $[-0.08, 0.08]$. We found a mixing ratio of $100 : 20$ alternate mini-batches training of entailment generation and video captioning to perform best.

A.2 Datasets and Training Details for MTL Summarization

A.2.1 Dataset Details

CNN/DailyMail Dataset CNN/DailyMail dataset (Hermann et al., 2015; Nallapati et al., 2016) is a large collection of online news articles and their multi-sentence summaries. We use the original, non-anonymized version of the dataset provided by See et al. (2017). Overall, the dataset has $287,226$ training pairs, $13,368$ validation pairs and, $11,490$ test pairs. On an average, a source document has 781 tokens and a target summary has 56 tokens.

Gigaword Corpus Gigaword is based on a large collection of news articles, where the article's first sentence is considered as the input document and the headline of the article as output summary. We use the annotated corpus provided by Rush et al. (2015). It has around 3.8 million training samples. For validation, we use $2,000$ samples and for test evaluation we use the standard test set provided by Rush et al. (2015). Following previous work, we keep our vocabulary size to $50,000$ frequent words.

DUC Corpus We use the DUC-2002[1] document summarization dataset for checking our model's generalizability capabilities. DUC-2002 corpus consists of 567 documents with one or two human annotated reference summaries. We also tried beam retuning using DUC-2003[2] as a validation set, which consists of 624 documents with single human annotated reference summaries.

SNLI corpus We use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) for our entailment generation task. Following Pasunuru and Bansal (2017a), we use the same re-splits provided by them to ensure a zero train-test overlap and multi-reference setup. This dataset has a total of $145,822$ unique premise pairs out of $190,113$ pairs, which are used for training, and the rest of them are divided equally into validation and test sets.

SQuAD Dataset We use Stanford Question Answering Dataset (SQuAD) for our question generation task (Rajpurkar et al., 2016). In SQuAD dataset, given the comprehension and question, the task is to predict the answer span in the comprehension. However, in our question generation task, we extract the sentence from the comprehension containing the answer span and create a sentence-question pair similar to Du et al. (2017). The dataset has around 100K sentence-question pairs from 536 articles.

A.2.2 Training Details

The following training details are common across all models and datasets. We use LSTM-RNN in our sequence models with hidden state size of 256 dimension. We use 128 dimension word embedding representations. We do not use dropout or any other regularization techniques, but we clip the gradient to allow a maximum gradient norm value of 2.0. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. Also, we share the word embeddings representation of both encoder and decoder in our models. All our tuning decisions (including soft/hard and layer-specific sharing decisions) were made on the appropriate validation/development set.

[1]https://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

[2]https://www-nlpir.nist.gov/projects/duc/guidelines/2003.html

CNN/DailyMail: For all the models involving CNN/DailyMail dataset, we use a maximum encoder RNN step size of 400 and a maximum decoder RNN step size of 100. We use a mini-batch size of 16. We initialize the LSTM-RNNs with uniform random initialization in the range $[-0.02, 0.02]$. We set λ to 1.0 in the joint cross-entropy and coverage loss. Also, we only add coverage to the converged model with attention and pointer mechanism, and make the learning rate from 0.001 to 0.0001. During multi-task learning, we use coverage mechanism for primary (CNN/DailyMail summarization) task but not for auxiliary tasks (because they do not have traditional redundancy issues). The penalty coefficient γ for soft-sharing is set to 5×10^{-5} and 1×10^{-5} for 2-way and 3-way multi-task models respectively (the range of the penalty value is intuitively chosen such that we balance the cross-entropy and regularization losses). In inference time, we use a beam search size of 4, following previous work (See et al., 2017).
Gigaword: For all the models involving Gigaword dataset, we use a maximum encoder RNN step size of 50 and a maximum decoder RNN step size of 20. We use a mini-batch size of 256. We initialize the LSTM-RNNs with uniform random initialization in the range $[-0.01, 0.01]$. We do not use coverage mechanism to our Gigaword models. Also, we set our beam search size to 5, following previous work (Nallapati et al., 2016).
DUC: For the CNN/DM to DUC domain-transfer experiments where we allow the beam sizes of all models to be individually re-tuned on DUC-2003, the chosen tuned beam values are $10, 4, 3$ for the multi-task model, baseline, and See et al. (2017), respectively.

Multi-Task Learning with Question Generation Two important hyperparameters tuned are the mixing ratio between summarization and entailment generation, as well as the soft-sharing coefficient. Here, we choose the mixing ratios 3:2 between CNN/DailyMail and SQuAD, 100:1 between Gigaword and SQuAD. Intuitively, these mixing ratios are close to the ratio of their dataset sizes. We set the soft-sharing coefficient γ to 5×10^{-5} and 1×10^{-5} for CNN/DailyMail and Gigaword, resp.

Multi-Task Learning with Entailment Generation Here, we choose the mixing ratios $3{:}2$ between CNN/DailyMail and SNLI, $20{:}1$ between Gigaword and SNLI. We again set the soft-sharing coefficient γ to 5×10^{-5} and 1×10^{-5} for CNN/DailyMail and Gigaword, resp.

Multi-Task Learning with Question and Entailment Generation Here, we choose the mixing ratios and soft-sharing coefficients to be $4{:}3{:}3$ and 5×10^{-5} for CNN/DailyMail, and $100{:}1{:}5$ and 1.5×10^{-6} for Gigaword respectively.

A.3 Experimental Setup for RL Video Captioning

A.3.1 Datasets

MSR-VTT Dataset. MSR-VTT is a diverse collection of $10,000$ video clips (41.2 hours of duration) from a commercial video search engine. Each video has 20 human annotated reference captions collected through Amazon Mechanical Turk (AMT). We use the standard split as provided in (Xu et al., 2016a), i.e., 6513 for training, 497 for testing , and remaining for testing. For each video, we sample at $3fps$ and we extract Inception-v4 (Szegedy et al., 2016) features from these sampled frames and we also remove all the punctuations from the text data.

YouTube2Text Dataset. We also evaluate our models on YouTube2Text dataset (Chen and Dolan, 2011). This dataset has 1970 video clips and each clip is annotated with an average of 40 captions by humans. We use the standard split as given in (Venugopalan et al., 2015a), i.e., 1200 clips for training, 100 for validation and 670 for testing. We do similar pre-processing as the MSR-VTT dataset.

A.3.2 Training Details

All the hyperparameters are tuned on the validation set. For each of our main models (baseline, CIDEr and CIDEnt), we report the results on a 5-avg-ensemble, where we run the model 5 times with different initialization random seeds and take the average probabilities at each time step of the decoder during inference time. We use a fixed size step LSTM-RNN encoder-decoder, with encoder step size of 50 and decoder step size of 16. Each LSTM has a hidden size of 1024. We use Inception-v4 features as video frame-level features. We use word embedding size of 512. Also, we project down the 1536-dim image features (Inception-v4) to 512-dim.

We apply dropout to vertical connections as proposed in Zaremba et al. (2014), with a value 0.5 and a gradient clip size of 10. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0001 for baseline cross-entropy loss. All the trainable weights are initialized with a uniform distribution in the range $[-0.08, 0.08]$. During the test time inference, we use beam search of size 5. All our reward-based models use mixed loss optimization (Paulus et al., 2018; Wu et al., 2016), where we train the model based on weighted (γ) combination of cross-entropy loss and reinforcement loss. For MSR-VTT dataset, we use $\gamma = 0.9995$ for our CIDEr-RL model and $\gamma = 0.9990$ for our CIDEnt-RL model. For YouTube2Text/MSVD dataset, we use $\gamma = 0.9985$ for our CIDEr-RL model and $\gamma = 0.9990$ and for our CIDEnt-RL model. The learning rate for the mixed-loss optimization is 1×10^{-5} for MSR-VTT, and 1×10^{-6} for YouTube2Text/MSVD. The λ hyperparameter in our CIDEnt reward formulation is roughly equal to the baseline cross-entropy model's score on that metric, i.e., $\lambda = 0.45$ for MSR-VTT CIDEnt-RL model and $\lambda = 0.75$ for YouTube2Text/MSVD CIDEnt-RL model.

A.4 Additional Saliency Reward Details for RL Summarization

Here, we describe the ROUGE-L formulation at summary-level and later describe how we incorporate saliency information into it. Given a reference summary of u sentences containing a total of m tokens ($\{w_{r,k}\}_{k=1}^{m}$) and a generated summary of v sentences with a total of n tokens ($\{w_{c,k}\}_{k=1}^{n}$), let r_i be the reference summary sentence and c_j be the generated summary sentence. Then, the precision (P_{lcs}), recall (R_{lcs}), and F-score (F_{lcs}) for ROUGE-L are defined as follows:

$$P_{lcs} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{n} \tag{A.1}$$

$$R_{lcs} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{m} \tag{A.2}$$

$$F_{lcs} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{A.3}$$

where $LCS_{\cup}$ takes the *union* Longest Common Subsequence (LCS) between a reference summary sentence r_i and every generated summary sentence c_j ($c_j \in C$), and β is defined in Lin (2004). In the above ROUGE-L scores, we assume that every token has equal weight, i.e, 1. However, every summary has salient tokens which should be rewarded with more weight. Hence, we use the weights obtained from our novel saliency predictor to modify the ROUGE-L scores with salient information as follows:

$$P^s_{lcs} = \frac{\sum_{i=1}^{u} LCS^*_{\cup}(r_i, C)}{\sum_{k=1}^{n} \eta(w_{c,k})} \tag{A.4}$$

$$R^s_{lcs} = \frac{\sum_{i=1}^{u} LCS^*_{\cup}(r_i, C)}{\sum_{k=1}^{m} \eta(w_{r,k})} \tag{A.5}$$

$$F^s_{lcs} = \frac{(1+\beta^2) R^s_{lcs} P^s_{lcs}}{R^s_{lcs} + \beta^2 P^s_{lcs}} \tag{A.6}$$

where $\eta(w)$ is the weight assigned by the saliency predictor for token w, and β is defined in Lin (2004).[3] Let $\{w_k\}_{k=1}^{p}$ be the union LCS set, then $LCS^*_{\cup}(r_i, C)$ is defined as follows:

$$LCS^*_{\cup}(r_i, C) = \sum_{k=1}^{p} \eta(w_k) \tag{A.7}$$

A.5 Training Details for Dynamic MTL

All LSTMs use hidden state size of 256. We train word vectors with embedding size of 128 with random initialization. We use gradient clipped norm of 2.0. Our model selection (tuning) criteria is based on the average of our 3 metrics (SARI, BLEU, 1/FKGL) on the validation set. The mixing ratios are α_{ss}:α_{eg}:α_{pp} $=$ 6:1:3 for Newsela, 6:1:3 for WikiSmall, and 7:2:1 for WikiLarge. The soft-sharing coefficient λ is set such that we balance the cross-entropy and regularization

[3]If a token is repeated at multiple times in the input sentence, we average the probabilities of those instances.

losses (at convergence), which is 5×10^{-6} for Newsela, 1×10^{-6} WikiSmall, and 1×10^{-5} for WikiLarge. We train models from scratch for Newsela and WikiSmall (using Adam (Kingma and Ba, 2015) optimizer with learning rate of 0.002 and 0.0015, respectively). However, because of the large size and computation overhead for WikiLarge, we first pre-train both main and auxiliary models on their own domain until they reach 90% convergence, and use these models to initialize the multi-task models, and set the learning rate to $1/10$ of its original default value (0.001). We set the decay rate α in the bandit controller to be 0.3. We use the negative validation loss as the reward at each sampling step to the bandit algorithm. The validation loss is divided by two as a smoothing technique.[4] All our soft/hard and layer-specific sharing decisions (Sec. 7.2.7) were made on the validation/dev set. We follow previous work (Zhang and Lapata, 2017b) in their pre-processing and post-processing of named entities. We capped vocabulary size to be 50K and replaced less frequent words with UNK token.[5] Unlike previous work (Zhang and Lapata, 2017b), we do not use UNK-replacement at test time, but instead rely on our pointer-copy mechanism. We use beam search with beam size of 5. All other details provided in our released code.

A.6 Training Details for DORB models

All the hyperparameters are tuned on the validation set for both question generation and data-to-text tasks. We use TITAN X and GeForce GTX 1080 GPUs for all our experiments, where all our RL models roughly take 1 day to train on a single GPU.

For the question generation task, we use two layers for both bi-directional encoder and uni-directional decoder. We set the hidden size of LSTM-RNN to 600 and use BERT-based contex-

[4]This constant serves the same purpose as the temperature variable in the softmax function.

[5]We measured the vocabulary overlap between the main and auxiliary tasks, and found that "word-form-overlap" (percentage of unique word types in auxiliary task that also appear in the main task) to be 40.7% (entailment) and 41.0% (paraphrase), and "word-count-overlap" (percentage of words in auxiliary task that also appear in the main task, based on token frequency counts) to be 95.2% (entailment) and 94.9% (paraphrase). Hence, this suggests that only rare words (which make up for very few counts) aren't considered in training process, and our pointer mechanism handles these extra UNK words by copying the actual word-form from the source to the output.

tual embeddings as input instead of word embeddings. The number of parameters in our model is 33.3 million. We use a batch size of 32, encoder maximum length of 512 and decoder maximum length of 50, and maximum gradient clipping of 5. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-3 and 1e-6 for cross-entropy model and RL models, respectively. We use a dropout of 0.3 for the cross-entropy model and no dropout for RL models. For multi-reward bandit models, we set the bandit coefficient (γ) to 0.1, and each round of the bandit consists of optimization of 100 mini-batches of training data. For HM-Bandit, we set the controller round size to 300 mini-batches. We consider the following short hyperparameters ranges and manually tune on: learning rate in the range [1e-5, 1e-7]; bandit coefficient in the range [0.01, 0.5]; bandit round - $\{10, 100\}$; and controller round size - $\{30, 300\}$.

For WebNLG data-to-text task, we first serialize and reorder the RDF data as an intermediate planning setup, and then feed the plan into an encoder-attention-decoder style architecture with copy mechanism, to generate the text describing the RDF data. We use same hyperparameters as discussed in Zhao et al. (2020) for the cross-entropy model, e.g., we use Adam with a batch size of 64, initial learning rate of 0.001, and a dropout of 0.3. All RL models are initialized with the best cross-entropy model checkpoint, and use Adam with a learning rate of 1e-6. We do not use dropout for RL models. The number of parameters in our model is 5.9 million. For multi-reward bandit models, we set the bandit coefficient (γ) to 0.15, and each round of the bandit consists of optimization of 10 mini-batches of training data. For HM-Bandit, we set the controller round size to 30 mini-batches. We consider the following short hyperparameters ranges and manually tune on: learning rate in the range [1e-5, 1e-7]; bandit coefficient in the range [0.01, 0.5]; bandit round - $\{10, 100\}$; and controller round size - $\{30, 300\}$.

A.7 Training Details for CAS

We use Adam optimizer (Kingma and Ba, 2015) and a mini-batch size of 64. We set the dropout to 0.5. In all of our architecture search models, we use 6 nodes. For the controller's optimization, we again use Adam optimizer with a learning rate of 0.00035.

For GLUE tasks, we use 256 dimensions for the hidden states of the RNNs, and for word embeddings we use ELMo representations (Peters et al., 2018), where we down project the 1024 dimensions ELMo embeddings to 256. We use a learning rate of 0.001, and both encoder RNNs are unrolled to 50 steps. For CAS conditions, we set the coefficients for block-sparsity and orthogonality conditions to 0.001 and 0.001, respectively.

For video captioning tasks, we use hidden state size of 1024 and word embedding size of 512. For visual features, we use a concatenation of both ResNet-152 (He et al., 2016) and ResNeXt-101 (Xie et al., 2017) image features. We use a learning rate of 0.0001, and we unroll the video encoder and caption decoder to 50 and 20 steps, respectively. For CAS conditions, we set both the coefficients of block-sparsity and orthogonality conditions to 0.0001.

REFERENCES

Amancio, M. and Specia, L. (2014). An analysis of crowdsourced text simplifications. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130.

Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). SPICE: Semantic propositional image caption evaluation. In *ECCV*, pages 382–398.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Learning to compose neural networks for question answering. In *NAACL*.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In *NeurIPS*.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2017). Designing neural network architectures using reinforcement learning. In *ICLR*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. (2017). What do neural machine translation models learn about morphology? In *ACL*.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer.

Biermann, A. W. (1978). The inference of regular lisp programs from examples. *IEEE transactions on Systems, Man, and Cybernetics*, 8(8):585–600.

Bingel, J. and Søgaard, A. (2017). Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain separation networks. In *NeurIPS*, pages 343–351.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *EMNLP*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *NeurIPS*.

Bubeck, S., Cesa-Bianchi, N., et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

Burtini, G., Loeppky, J., and Lawrence, R. (2015). A survey of online experiment design with the stochastic multi-armed bandit. *arXiv preprint arXiv:1510.00757*.

Cai, H., Chen, T., Zhang, W., Yu, Y., and Wang, J. (2018). Efficient architecture search by network transformation. In *AAAI*.

Carroll, J. A., Minnen, G., Pearce, D., Canning, Y., Devlin, S., and Tait, J. (1999). Simplifying text for language-impaired readers. In *EACL*, pages 269–270.

Caruana, R. (1998). Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675.

Chandrasekar, R., Doran, C., and Srinivas, B. (1996). Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics.

Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Chen, D. L. and Dolan, W. B. (2011). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics.

Chen, D. L. and Mooney, R. J. (2008). Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016). Distraction-based neural networks for modeling documents. In *IJCAI*.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017). Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Chen, X., Kim, S., Lin, Q., Carbonell, J. G., and Xing, E. P. (2010). Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *arXiv preprint arXiv:1005.3579*.

Chen, Y., Wang, S., Zhang, W., and Huang, Q. (2018). Less is more: Picking informative frames for video captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 358–373.

Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 675–686.

Chen, Z. and Liu, B. (2016). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145.

Cheung, J. C. K. and Penn, G. (2014). Unsupervised sentence enhancement for automatic summarization. In *EMNLP*, pages 775–786.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*.

Clark, K., Luong, M.-T., Khandelwal, U., Manning, C. D., and Le, Q. (2019). Bam! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937.

Clarke, J. and Lapata, M. (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 377–384. Association for Computational Linguistics.

Clarke, J. and Lapata, M. (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Coster, W. and Kauchak, D. (2011). Learning to simplify sentences using wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9. Association for Computational Linguistics.

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. *arXiv preprint arXiv:0907.0783*.

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. IEEE.

Deng, L., Hinton, G., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE.

Denkowski, M. and Lavie, A. (2014a). Meteor universal: Language specific translation evaluation for any target language. In *EACL*.

Denkowski, M. and Lavie, A. (2014b). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Devlin, S. L. (1999). *Simplifying natural language for aphasic readers.* PhD thesis, University of Sunderland.

Distiawan, B., Qi, J., Zhang, R., and Wang, W. (2018). Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637.

Dohare, S. and Karnick, H. (2017). Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655.

Du, X. and Cardie, C. (2018). Harvesting paragraph-level question-answer pairs from wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917.

Du, X., Shao, J., and Cardie, C. (2017). Learning to ask: Neural question generation for reading comprehension. In *ACL*.

Duong, L., Cohn, T., Bird, S., and Cook, P. (2015). Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850.

Dušek, O. and Jurcicek, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 45–51.

Dušek, O., Novikova, J., and Rieser, V. (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Efficient multi-objective neural architecture search via lamarckian evolution. In *ICLR*.

Evans, R., Orasan, C., and Dornescu, I. (2014). An evaluation of syntactic simplification rules for people with autism. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 131–140.

Evgeniou, T., Micchelli, C. A., Pontil, M., and Shawe-Taylor, J. (2005). Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(4).

Falke, T. and Gurevych, I. (2017). Bringing structure into summaries: Crowdsourcing a benchmark corpus of concept maps. In *EMNLP*.

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, L., and Vinyals, O. (2015). Sentence compression by deletion with lstms. In *EMNLP*, pages 360–368.

Filippova, K. and Strube, M. (2008). Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32. Association for Computational Linguistics.

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

Ganesan, K., Zhai, C., and Han, J. (2010). Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. ACL.

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017). The WebNLG challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Gehrmann, S., Dai, F., Elder, H., and Rush, A. M. (2018a). End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56.

Gehrmann, S., Deng, Y., and Rush, A. M. (2018b). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Gerani, S., Mehdad, Y., Carenini, G., Ng, R. T., and Nejat, B. (2014). Abstractive summarization of product reviews using discourse structure. In *EMNLP*, volume 14, pages 1602–1613.

Giannakopoulos, G. (2009). Automatic summarization from multiple documents. *Ph. D. dissertation*.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Glavaš, G. and Štajner, S. (2015). Simplifying lexical simplification: Do we need simplified corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 63–68.

Gong, H. (2018). Technical report for e2e nlg challenge. *E2E NLG Challenge System Descriptions*.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772.

Gu, J., Cho, K., and Li, V. O. (2017). Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., and Saenko, K. (2013). Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *CVPR*, pages 2712–2719.

Guo, H., Pasunuru, R., and Bansal, M. (2017). Interactive-length multi-task video captioning with cooperative feedback. In *NeurIPS Demo Track*.

Guo, H., Pasunuru, R., and Bansal, M. (2018). Dynamic multi-level multi-task learning for sentence simplification. In *COLING*.

Guo, H., Pasunuru, R., and Bansal, M. (2019a). AutoSeM: Automatic task selection and mixing in multi-task learning. In *NAACL*, pages 3520–3531.

Guo, Z., Zhang, Y., Teng, Z., and Lu, W. (2019b). Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Hal Daumé, I., Langford, J., and Marcu, D. (2009). Search-based structured prediction as classification. *Machine Learning Journal*.

Hashimoto, K., Xiong, C., Tsuruoka, Y., and Socher, R. (2017). A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Hendricks, L. A., Wang, O., Shechtman, E., Sivic, J., Darrell, T., and Russell, B. (2017). Localizing moments in video with natural language. In *ICCV*, pages 5803–5812.

Henß, S., Mieskes, M., and Gurevych, I. (2015). A reinforcement learning approach for adaptive single-and multi-document summarization. In *GSCL*, pages 3–12.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *NeurIPS*, pages 1693–1701.

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

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoffman, M. D., Brochu, E., and de Freitas, N. (2011). Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336. Citeseer.

Horn, C., Manduca, C., and Kauchak, D. (2014). Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

Huang, H., Lu, Y., Zhang, F., and Sun, S. (2013). A multi-modal clustering method for web videos. In *International Conference on Trustworthy Computing and Services*, pages 163–169.

Hwang, W., Hajishirzi, H., Ostendorf, M., and Wu, W. (2015). Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL-HLT*, pages 211–217.

Inui, K., Fujita, A., Takahashi, T., Iida, R., and Iwakura, T. (2003). Text simplification for reading assistance: a project note. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 9–16. Association for Computational Linguistics.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jalali, A., Sanghavi, S., Ruan, C., and Ravikumar, P. (2010). A dirty model for multi-task learning. *Advances in neural information processing systems*, 23:964–972.

Jing, H. and McKeown, K. R. (2000). Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 178–185. Association for Computational Linguistics.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., et al. (2016). Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Jung, H., Ju, J., Jung, M., and Kim, J. (2016). Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J. (2017). One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Kaji, N., Kawahara, D., Kurohash, S., and Sato, S. (2002). Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 215–222. Association for Computational Linguistics.

Kauchak, D. (2013). Improving text simplification language modeling using unsimplified text data. In *ACL (1)*, pages 1537–1546.

Kedzie, C., McKeown, K., and Diaz, F. (2015). Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1608–1617.

Kim, Y., Lee, H., Shin, J., and Jung, K. (2019). Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6602–6609.

Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., and Chissom, B. S. (1975). Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

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.

Klebanov, B. B., Knight, K., and Marcu, D. (2004). Text simplification for information-seeking applications. *Lecture Notes in Computer Science*, pages 735–747.

Knight, K. and Marcu, D. (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Kumar, A. and Daumé III, H. (2012). Learning task grouping and overlap in multi-task learning. In *ICML*.

Kumar, V., Ramakrishnan, G., and Li, Y.-F. (2019). Putting the horse before the cart: A generator-evaluator framework for question generation from text. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 812–821.

Kveton, B., Szepesvari, C., Vaswani, S., Wen, Z., Lattimore, T., and Ghavamzadeh, M. (2019). Garbage in, reward out: Bootstrapping exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 3601–3610.

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

Lampouras, G. and Vlachos, A. (2016). Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112.

Lattimore, T. and Szepesvári, C. (2019). *Bandit Algorithms*. Cambridge University Press (preprint).

Lawrence, N. D. and Platt, J. C. (2004). Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, page 65.

Lebret, R., Grangier, D., and Auli, M. (2016). Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Lee, S. (2017). Toward continual learning for conversational agents. *arXiv preprint arXiv:1712.09943*.

Levesque, H., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. (2016). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liang, P., Jordan, M. I., and Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Liang, P., Jordan, M. I., and Klein, D. (2010). Learning programs: A hierarchical bayesian approach. In *ICML*, pages 639–646.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*, volume 8.

Liu, B., Zhao, M., Niu, D., Lai, K., He, Y., Wei, H., and Xu, Y. (2019a). Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*, pages 1106–1118.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2017a). Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016a). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Liu, F., Flanigan, J., Thomson, S., Sadeh, N., and Smith, N. A. (2015). Toward abstractive summarization using semantic representations. In *NAACL: HLT*, pages 1077–1086.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018). Hierarchical representations for efficient architecture search. In *CVPR*.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016b). Improved image captioning via policy gradient optimization of SPIDEr. *arXiv preprint arXiv:1612.00370*.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2017b). Improved image captioning via policy gradient optimization of spider. In *Proceedings of the IEEE international conference on computer vision*, pages 873–881.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In *EMNLP*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Long, M. and Wang, J. (2015). Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117*, 2(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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016). Multi-task sequence to sequence learning. In *ICLR*.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Macready, W. G. and Wolpert, D. H. (1998). Bandit problems and the exploration/exploitation tradeoff. *IEEE Transactions on evolutionary computation*, 2(1):2–22.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marcheggiani, D. and Perez-Beltrachini, L. (2018). Deep graph convolutional encoders for structured data to text generation. In *INLG*.

Mei, H., UChicago, T., Bansal, M., and Walter, M. R. (2016). What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

Merentitis, A., Rasul, K., Vollgraf, R., Sheikh, A.-S., and Bergmann, U. (2018). A bandit framework for optimal selection of reinforcement learning agents. In *NeurIPS 2018 Workshop on Deep Reinforcement Learning*.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Nallapati, R., Zhou, B., dos santos, C. N., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Narayan, S. and Gardent, C. (2014). Hybrid simplification using deep semantics and machine translation. In *ACL*.

Neelakantan, A., Le, Q. V., and Sutskever, I. (2015). Neural programmer: Inducing latent programs with gradient descent. In *ICLR*.

Negahban, S. and Wainwright, M. J. (2008). Joint support recovery under high-dimensional scaling: Benefits and perils of $\ell_{1,\infty}$-regularization. *Advances in Neural Information Processing Systems*, 21:1161–1168.

Negrinho, R. and Gordon, G. (2017). Deeparchitect: Automatically designing and training deep architectures. In *CVPR*.

Nema, P. and Khapra, M. M. (2018). Towards a better metric for evaluating question generation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3950–3959.

Neubig, G. and Watanabe, T. (2016). Optimization for statistical machine translation: A survey. *Computational Linguistics*, 42(1):1–54.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2017). Variational continual learning. *arXiv preprint arXiv:1710.10628*.

Noreen, E. W. (1989). *Computer-intensive methods for testing hypotheses*. Wiley New York.

Pan, P., Xu, Z., Yang, Y., Wu, F., and Zhuang, Y. (2016a). Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, pages 1029–1038.

Pan, Y., Mei, T., Yao, T., Li, H., and Rui, Y. (2016b). Jointly modeling embedding and translation to bridge video and language. In *CVPR*, pages 4594–4602.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *EMNLP*.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Pasunuru, R. and Bansal, M. (2017a). Multi-task video captioning with video and entailment generation. In *ACL*.

Pasunuru, R. and Bansal, M. (2017b). Reinforced video captioning with entailment rewards. In *EMNLP*.

Pasunuru, R. and Bansal, M. (2018). Multi-reward reinforced summarization with saliency and entailment. In *NAACL*.

Pasunuru, R. and Bansal, M. (2019). Continual and multi-task architecture search. In *ACL*.

Pasunuru, R., Guo, H., and Bansal, M. (2017). Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization, EMNLP*.

Pasunuru, R., Guo, H., and Bansal, M. (2018). Soft layer-specific multi-task summarization with entailment and question generation. In *ACL*.

Pasunuru, R., Guo, H., and Bansal, M. (2020). Dorb: Dynamically optimizing multiple rewards with bandits. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7766–7780.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017). Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Perez-Martin, J., Bustos, B., and Perez, J. (2021). Improving video captioning with temporal composition of a visual-syntactic embedding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3039–3049.

Peters, M. t. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL*.

Petersen, S. E. and Ostendorf, M. (2007). Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. (2018). Efficient neural architecture search via parameter sharing. In *ICML*.

Raj, V. and Kalyani, S. (2017). Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *ICLR*.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE.

Reiter, E. (1995). NLG vs. templates. In *Proc of the Fifth European Workshop on Natural-Language Generation*.

Reiter, E. (2007). An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 97–104. Association for Computational Linguistics.

Reiter, E. (2017). You need to understand your corpora-the weathergov example. *Blogpost-https://ehudreiter. com/2017/05/09/weathergov*.

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Reiter, E., Sripada, S., Hunter, J., Yu, J., and Davy, I. (2005). Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Rello, L., Baeza-Yates, R., and Saggion, H. (2013). The impact of lexical simplification by verbal paraphrases for people with and without dyslexia. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 501–512. Springer.

Ren, Z., Wang, X., Zhang, N., Lv, X., and Li, L.-J. (2017). Deep reinforcement learning-based image captioning with embedding reward. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 290–298.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195. IEEE.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *EMNLP*.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al. (2018). A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Sarwar, S. S., Ankit, A., and Roy, K. (2019). Incremental learning in deep convolutional neural networks using partial network sharing. *IEEE Access*, 8:4615–4628.

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. (2017). Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89.

Schick, T. and Schütze, H. (2021). Exploiting cloze questions for few-shot text classification and natural language inference. In *EACL*.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *ACL*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Sharaf, A. and Daumé III, H. (2019). Meta-learning for contextual bandit exploration. *arXiv preprint arXiv:1901.08159*.

Shardlow, M. (2014). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Sharma, S. and Ravindran, B. (2017). Online multi-task learning using active sampling. In *ICLR 2017 Workshop*.

She, L., Cheng, Y., Chai, J. Y., Jia, Y., Yang, S., and Xi, N. (2014). Teaching robots new actions through natural language instructions. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873. IEEE.

Shi, Z., Chen, X., Qiu, X., and Huang, X. (2018). Toward diverse text generation with inverse reinforcement learning. In *IJCAI*.

Siddharthan, A. (2006). Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Siddharthan, A. (2014). A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*.

So, D. R., Liang, C., and Le, Q. V. (2019). The evolved transformer. *arXiv preprint arXiv:1901.11117*.

Song, L., Wang, Z., and Hamza, W. (2018a). A unified query-based generative model for question generation and question answering. In *NAACL*.

Song, L., Wang, Z., Hamza, W., Zhang, Y., and Gildea, D. (2018b). Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574.

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Specia, L. (2010). Translating from complex to simplified sentences. *Computational Processing of the Portuguese Language*, pages 30–39.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852.

Srivastava, S., Labutov, I., and Mitchell, T. (2019). Learning to ask for conversational machine learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4155–4165.

Štajner, S., Béchara, H., and Saggion, H. (2015). A deeper exploration of the standard pb-smt approach to text simplification and its evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Summers, P. D. (1986). A methodology for lisp program construction from examples. In *Readings in artificial intelligence and software engineering*, pages 309–316. Elsevier.

Sun, X., Liu, J., Lyu, Y., He, W., Ma, Y., and Wang, S. (2018). Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Suzuki, J. and Nagata, M. (2016). Rnn-based encoder-decoder approach with word frequency estimation. In *EACL*.

Szegedy, C., Ioffe, S., and Vanhoucke, V. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. In *CoRR*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR*, pages 1–9.

Tan, J., Wan, X., and Xiao, J. (2017). Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Thomason, J., Venugopalan, S., Guadarrama, S., Saenko, K., and Mooney, R. J. (2014). Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*.

Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pages 489–497.

Torabi, A., Pal, C., Larochelle, H., and Courville, A. (2015). Using descriptive video services to create a large data source for video annotation research. *arXiv preprint arXiv:1503.01070*.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Venugopalan, S., Hendricks, L. A., Mooney, R., and Saenko, K. (2016). Improving lstm-based video description with linguistic knowledge mined from text. In *EMNLP*.

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015a). Sequence to sequence-video to text. In *CVPR*, pages 4534–4542.

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., and Saenko, K. (2015b). Translating videos to natural language using deep recurrent neural networks. In *NAACL HLT*.

Vickrey, D. and Koller, D. (2008). Sentence simplification for semantic role labeling. In *ACL*, pages 344–352.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *NeurIPS*, pages 2692–2700.

Vinyals, O. and Le, Q. (2015). A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Wang, B., Ma, L., Zhang, W., Jiang, W., Wang, J., and Liu, W. (2019b). Controllable video captioning with pos sequence guidance based on gated fusion network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2641–2650.

Wang, L., Raghavan, H., Castelli, V., Florian, R., and Cardie, C. (2013). A sentence compression based framework to query-focused multi-document summarization. In *ACL*.

Wang, W., Yang, N., Wei, F., Chang, B., and Zhou, M. (2017). Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198.

Wang, X., Chen, W., Wu, J., Wang, Y.-F., and Yang Wang, W. (2018). Video captioning via hierarchical reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4213–4222.

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.

Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

White, D. A. and Sofge, D. A. (1992). *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*. Van Nostrand Reinhold Company.

Wieting, J. and Gimpel, K. (2017a). Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *CoRR*, abs/1711.05732.

Wieting, J. and Gimpel, K. (2017b). Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Williams, A., Nangia, N., and Bowman, S. (2018a). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Williams, A., Nangia, N., and Bowman, S. R. (2018b). A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wiseman, S., Shieber, S., and Rush, A. (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Woodsend, K. and Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Woodsend, K. and Lapata, M. (2014). Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, 51:133–164.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wubben, S., van den Bosch, A., and Krahmer, E. (2012). Sentence simplification by monolingual machine translation. In *ACL*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995. IEEE.

Xu, J., Mei, T., Yao, T., and Rui, Y. (2016a). MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*, pages 5288–5296.

Xu, J. and Zhu, Z. (2018). Reinforced continual learning. In *Advances in Neural Information Processing Systems*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015a). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Xu, W., Callison-Burch, C., and Napoles, C. (2015b). Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

Xu, W., Napoles, C., Pavlick, E., Chen, Q., and Callison-Burch, C. (2016b). Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Yang, Y. and Hospedales, T. M. (2017). Trace norm regularised deep multi-task learning. In *ICLR Workshop Track*.

Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. (2015). Describing videos by exploiting temporal structure. In *CVPR*, pages 4507–4515.

Yin, W., Rajani, N. F., Radev, D., Socher, R., and Xiong, C. (2020). Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *EMNLP*.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *NeurIPS*, pages 3320–3328.

Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. (2016). Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Yuan, X., Wang, T., Gulcehre, C., Sordoni, A., Bachman, P., Zhang, S., Subramanian, S., and Trischler, A. (2017). Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25.

Zaremba, W. and Sutskever, I. (2015). Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995.

Zhang, C.-H., Huang, J., et al. (2008). The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, 36(4):1567–1594.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020a). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Zhang, S. and Bansal, M. (2019). Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509.

Zhang, X. and Lapata, M. (2017a). Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Zhang, X. and Lapata, M. (2017b). Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer.

Zhang, Z., Shi, Y., Yuan, C., Li, B., Wang, P., Hu, W., and Zha, Z.-J. (2020b). Object relational graph with teacher-recommended learning for video captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13278–13288.

Zhao, C., Walker, M., and Chaturvedi, S. (2020). Bridging the structural gap between encoding and decoding for data-to-text generation. In *ACL*.

Zhao, Y., Ni, X., Ding, Y., and Ke, Q. (2018). Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., and Zhou, M. (2017). Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *ICLR*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *CVPR*.