INTELLIGENT KNOWLEDGE TRANSFER FOR MULTI-STAGE AND MULTI-TASK LEARNING

A Dissertation

by

YUN HE

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	James Caverlee
Committee Members,	Xia Hu
	Bobak Jack Mortazavi
	Yang Shen
	Ruihong Huang
Head of Department,	Scott Schaefer

August 2022

Major Subject: Computer Science

ABSTRACT

Machine learning plays a significant role in powering artificial intelligence advances in many areas like natural language processing and personalized recommendation, aiming to build models to fit the labeled training data and then predict on the held-out testing data. A key challenge for these machine learning models is the imbalance between scarce labeled data and continuously increased model capacity. On the one hand, the labeled data of many tasks is scarce because human annotations are expensive, which is especially true for some specialized domains like biomedical. On the other hand, the capacity of models are growing continuously in the last decade, with parameters ranging from millions to billions. Without enough labeled data, such large-scale models may overfit on low-resource tasks, resulting in performance deterioration. Recently, many work demonstrate that transferring useful knowledge from pre-training stages or jointly trained related tasks to the target task may alleviate the label scarcity problem and significantly boost the performance of the target task. Despite the prominence achieved in the recent work, there are still many challenges and open problems to be explored for the knowledge transfer. First, transferring domain-specific knowledge from pre-training stages to large-scale language models remains under-explored, which limits the performance of natural language understanding over the corresponding domains. Second, training multiple tasks jointly hinders the performance on individual tasks, which is more serious in transformer-based multi-task co-training because all tasks share a single set of parameters. Third, transferring knowledge from the source might have a negative impact on the target learner, leading to worse results than training the target task alone. To overcome these challenges, three contributions are made in this dissertation:

- To transfer disease knowledge to enhance BERT-like language models over health-related tasks, we propose a new pre-training procedure named disease knowledge infusion, which efficiently exploit the self-supervised learning signals of Wikipedia pages.
- The second contribution is a novel method named HyperPrompt that utilizes HyperNetworks

to generate task-conditioned prompts for multi-task learning, where the task-specific knowledge can be flexibly shared via the HyperNetworks.

- To alleviate the negative transfer problem from the perspective of gradient magnitudes, we propose a novel algorithm named MetaBalance to dynamically and adaptively balance the gradients of auxiliary tasks to better assist the target task.

DEDICATION

To my wife and my parents.

ACKNOWLEDGMENTS

My Ph.D study was from 25 to 30 years old, which is probably the most prime time of my whole life. From a junior researcher who needed lots of help and guidance to an experienced Ph.D candidate who can conduct research independently, from a student who knew little about machine learning and large-scale models to a specialist with the title of research scientist in a top industrial lab, from an international student who could barely talk and listen in English to a researcher who has no problem in communicating with collaborators with various backgrounds, I would like to thank lots of people because it is impossible to obtain these achievements without their support, inspiration and influence.

First of all, I would like to express my sincere gratitude to my advisor Dr. James Caverlee for the best mentorship. When I was a junior student, he patiently gave me lots of guidance how to conduct research step by step. When I gained some experience in the third year and my interest shifted from our familiar recommendation domain to a direction of pre-trained language models, he gave me freedom, support and trust to allow me to explore in the new direction. Except to the research training, I also learned wisdom from him, such as how to make papers and presentations more attractive, being humble but also confident, the fact that connections are very important and how to connect with people, how to manage a project and a team, being honest and never lie about the experiments and how to apply and obtain a tenure-track faulty position (maybe someday I will go back to academia). He is not only an advisor but also a friend and a role model to me. I would also like to thank the rest of my dissertation committee, Dr. Xia Hu, Dr. Bobak Jack Mortazavi, Dr. Ruihong Huang and Dr. Yang Shen for their continuous advice and support during my Ph.D.

Then, I want to thank my labmates, other mentors and all collaborators. I was fortunate to learn and grow together with Ziwei Zhu, Yin Zhang, Jianling Wang, Xing Zhao, Parisa and Majid. They are the best friends, peers and collaborators. Besides, I am grateful to Dr. Xue Feng, Dr. Hanning Zhou, Dr. Xinyi Zhang and Dr. Yunsong Guo, who offered me my first and second industry internships at Facebook AI and gave me a lot of support and inspirations for my research project. I

also would like to thank Dr. Huaixiu Zheng, Dr. Yi Tay and Dr. Heng-Tze Cheng for giving me the internship opportunity to collaborate with them at Google Brain, where I learned a lot from them. In addition, I would like to thank Dr. Qinmin Hu and Dr. Liang He as my enlightening teacher of AI and data mining at ECNU.

Finally, I would like thank my family. My wife Jingwen Han never thought about studying and working in the United States before meeting me. With the love and courage, she came over the pacific with thousands of miles and applied to the same university as I. During the five years, she comforted and encouraged me countless times. I became stronger, more resilient and optimistic because of her. Next, I want to thank my parents for their love and endless support. What I learned from them is diligence, humility and the habit of reading.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor James Caverlee [advisor], Professor Ruihong Huang, Professor Bobak Mortazavi of the Department of Computer Science and Engineering, Professor Yang Shen of the Department of Electrical and Computer Engineering and Professor Xia Hu from Rice University. Chapter III is conducted based on the work majorly done while I was interning at Google. Chapter IV is conducted based on the work majorly done while I was interning at Facebook. All work conducted for the dissertation was completed by the student independently.

Funding Sources

This work is, in part, supported by NSF (#IIS- 1841138).

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGMENTS v

CONTRIBUTORS AND FUNDING SOURCES vii

TABLE OF CONTENTS viii

LIST OF FIGURES x

LIST OF TABLES xiii

1. INTRODUCTION AND LITERATURE REVIEW 1

    1.1 Background 1
    1.2 Knowledge Transfer 1
        1.2.1 Transfer via Multi-Stage Learning 2
        1.2.2 Transfer via Multi-Task Learning 5
    1.3 Challenges to Conduct Knowledge Transfer 6
    1.4 Research Questions and Our Contributions 8

2. INFUSING DISEASE KNOWLEDGE INTO PRE-TRAINED LANGUAGE MODELS FOR BIOMEDICAL LANGUAGE UNDERSTANDING 10

    2.1 Introduction 10
    2.2 Related Work 13
    2.3 Proposed Method: Disease Knowledge Infusion Training 14
        2.3.1 Targeting Diseases and Aspects 15
        2.3.2 Weakly Supervised Knowledge Infusion from Wikipedia 16
        2.3.3 Training Objective and Details 18
    2.4 Experiments 19
        2.4.1 BERT and its Biomedical Variants 20
        2.4.2 Tasks 21
        2.4.3 Results 24
        2.4.4 Ablation Study 25
        2.4.5 Learning Curve 26
    2.5 Summary 27

3. HYPERPROMPT: PROMPT-BASED TASK-CONDITIONING OF TRANSFORMERS .. 28

3.1 Introduction ... 28
3.2 Methods ... 32
3.2.1 Prompt-Based Task-Conditioned Transformer ... 32
3.2.2 HyperPrompt ... 34
3.2.3 HyperPrompt-Global ... 36
3.2.4 Parameter Efficiency of HyperPrompt-Global ... 37
3.3 Experiments ... 38
3.3.1 Experimental Setup ... 38
3.3.2 Experimental Details ... 39
3.3.3 Key Results ... 40
3.3.4 Tuning all vs Task-Conditioned Parameters ... 40
3.3.5 Computational Efficiency ... 42
3.3.6 Ablation Study ... 42
3.3.7 Peeking into Hyper-Prompts ... 45
3.3.8 Impact of Hyper-Prompt Length ... 46
3.3.9 Encoder vs Decoder ... 47
3.4 Related Work ... 49
3.5 Summary ... 50

4. METABALANCE: IMPROVING MULTI-TASK RECOMMENDATIONS VIA ADAPTING GRADIENT MAGNITUDES OF AUXILIARY TASKS ... 51

4.1 Introduction ... 51
4.2 Related Work ... 55
4.3 Problem Statement ... 57
4.4 Proposed Method ... 59
4.4.1 Adapting Auxiliary Gradient Magnitudes ... 60
4.4.2 Adjusting Magnitude Proximity ... 61
4.4.3 Time and Space Complexity Analysis ... 65
4.4.4 Comparison with Previous Methods ... 66
4.4.4.1 Auxiliary Task Adapting Methods ... 66
4.4.4.2 Multi-Task Balancing Methods ... 67
4.5 Experiments ... 69
4.5.1 Experimental Setup ... 69
4.6 Reproducibility of Experiments ... 71
4.6.1 RQ1: Improvement of Target Task via Adapting Auxiliary Gradients ... 72
4.6.2 RQ2: Comparison with Baseline Methods ... 74
4.6.3 RQ3: Collaboration with More Optimizers ... 79
4.6.4 RQ4: Impact of Moving Averages of Gradient Magnitudes ... 80
4.7 Summary ... 81

5. CONCLUSION AND FUTURE RESEARCH OPPORTUNITIES ... 82

REFERENCES ... 85

LIST OF FIGURES

FIGURE Page

1.1 Knowledge transfer via multi-stage and multi-task learning. Blue dataset represents the labeled training set while brown dataset represents the testing (unlabeled) or validation (labeled) set. The blue line with arrow indicates that the dataset is fed into a model. The green line with arrow indicates that the model infers on the testing/validation set. The orange line with arrow indicates the knowledge transfer. (A) Single task with single stage learning: the model is trained on the labeled training set and then infer on the testing/validation set of the same task; (B) Multi-stage learning: (B.1) a PLM is first pre-trained over large-scale general unlabeled datasets (like Wikipedia or crawled web pages) to learn the general-purpose knowledge; (B.2) and then pre-train the same PLM over domain or task-specific unlabeled datasets (like PubMed) to learn the domain or task-specific knowledge; (B.3) finally, the PLM is fine-tuned over the target task as the same procedure as A. The knowledge learned from these pre-training stages can be transferred to improve the downstream task; (C) Multi-task learning – multiple tasks are trained together on the same model, which has two scenarios: (C.1) The tasks are equally important and the high testing performance required for each task. Hence, we are interested in the transferring of task-specific knowledge between the tasks; (C.2) The high testing performance is only required for target task and the role of auxiliary tasks is to assist the target task. Hence, only the knowledge transfer from the auxiliary tasks to the target task is of interest. 3

2.1 Examples of tasks that can benefit from disease knowledge. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 12

2.2 Disease knowledge infusion training: an example with COVID-19. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 16

2.3 Learning curve of disease infusion knowledge. The y-axis is the accuracy of BERT models over MEDIQA-2019. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 27

3.1 HyperPrompt achieves state-of-the-art performance on SuperGLUE for T5 models up to XXL. Prompt-tuning [63] with tuning prompt parameters only achieves competitive performance against multi-task learning (MTL) baseline for the 11B parameter model with a big performance gap for smaller models. HyperPrompt outperforms the strong parameter-efficient adapter variant HyperFormer++ [56], the MTL baseline, and the full fine-tuning of Prompt-Tuning (our implementation) across model sizes with a large margin [e.g. 91.3 vs 90.2 (MTL) for T5 XXL]. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 31

3.2 HyperPrompt framework: (a) in each Transformer block, task-specific hyper-prompts $P_{K,V}$ are prepended to the original key K and value V for the query Q to attend to, (b) in HyperPrompt-Share/Sep, global prompts P is used to generate the hyper-prompts $P_{K,V}$ through local HyperNetworks $h_{k,v}$ at each Transformer layer, which consists of a down-projection matrix $D_{K,V}$, a RELU layer and a up-project matrix $U_{K,V}$, (c) in HyperPrompt-Global, all the local HyperNetworks are generated by global HyperNetworks $H_{k,v}$ using layer-aware task embeddings I as task-specific inputs (see Section 3.2.3 for details). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 33

3.3 Visualization of attention mass and entropy distribution. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 47

3.4 Impact of hyper-prompt length in HyperPrompt-Global (GLUE score on T5 Base). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 48

4.1 Transfer learning from auxiliary tasks to improve the target task on a multi-task network. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022..... 52

4.2 The imbalance of gradient magnitudes in transfer learning from auxiliary tasks for recommendations on Alibaba data. The magnitudes dynamically change throughout the training, with the imbalance varying across different parts of the same multi-task network: in Fig 4.2a, the gradient of auxiliary task click-URL is much larger than the target gradient; in Fig 4.2b, the gradient of auxiliary task Add-to-Favorite is much smaller than the target gradient. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 54

4.3 The impact of relax factor r on magnitude proximity on UserBehavior-2017 dataset. In the legend, "target" represents the target task (i.e., purchase prediction). Y-axis is the average gradient magnitude over all mini-batch iterations in one epoch, where the gradient w.r.t a MLP layer of the multi-task network is taken as the example. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 63

4.4 Multi-task recommendation network in the evaluation. The shared bottom layers is the combination of MLP layer and matrix factorization layer, which is widely adopted for recsys in both academia [44] and industry [23, 84]. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 71

4.5 Impact of relax factor r. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 74

4.6 The training loss on UserBehavior-2017. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 78

4.7 Collaboration with other optimizers. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 80

LIST OF TABLES

TABLE Page

2.1 Disease knowledge of *COVID-19* is presented from three aspects: symptoms, diagnosis and treatment (based on Wikipedia). Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 11

2.2 Eight aspects of knowledge of a disease that are considered in this work. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 15

2.3 Examples of auxiliary sentences. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 18

2.4 Summary of tasks and datasets. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 21

2.5 Experimental results. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 22

2.6 Ablation study on MEDIQA-2019. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020. 26

3.1 Comparison of fine-tuning all vs task-specific parameters on GLUE. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 41

3.2 Comparison of fine-tuning all vs task-specific parameters on SuperGLUE. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 41

3.3 The number of operations for a single forward pass and training time (base model). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 42

3.4 Comparison of HyperPrompt with baselines on GLUE using T5 Base. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 43

3.5 Comparison of HyperPrompt with baselines on SuperGLUE using T5 Base. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 43

3.6 Comparison of HyperPrompt with baselines on GLUE using T5 Large. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 44

3.7 Comparison of HyperPrompt with baselines on SuperGLUE using T5 Large. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 44

3.8 Ablation of inserting hyper-prompts or adapters into Encoder/Decoder/Enc-Dec (base model). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022). 48

4.1 Statistics of preprocessed datasets. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 72

4.2 Strategy selection. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 73

4.3 Experimental results of UserBehavior-2017 dataset. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 75

4.4 Experimental results of IJCAI-2015 dataset. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 76

4.5 MetaBalance plus gradient direction-based methods. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 79

4.6 Ablation study of moving average of magnitude. Reprinted with permission from “MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022. 80

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Background

Machine learning [54] is one of today's most rapidly growing and impactful technical fields. At its core, machine learning typically aims to build a model to fit the training data of a task and then predict on the held-out testing data. This approach has led to wide application in many areas like health care [14], manufacturing [128], education [5], financial modeling [25], and marketing [80].

In the past decade, machine learning has seen rapid growth in both model capacity – e.g., many widely-deployed models have millions or even billions of parameters – and in computing capabilities to support such large models – e.g., the adoption of GPUs and specialized processors like Google's Tensor Processing Unit (TPU). This dramatic progression has opened new opportunities for machine learning to have impact on many areas such as computer vision (CV) [43], natural language processing (NLP) [114] and personalized recommendation [23].

A key challenge for these large machine learning models is the imbalance between scarce labeled data and continuously increased model capacity. On the one side, the labeled data of many tasks is scarce. For example, the paraphrase identification dataset MRPC [31] only contains 5,801 sentence pairs. Moreover, such a scarcity problem becomes more serious in specialized domains like biomedicine because expensive domain experts are required to manually annotate datasets [46, 86]. On the other side, the capacity of models is growing continuously. For example, pre-trained language models have evolved from ELMo [87] with 94M parameters to GPT-3 [17] with over 175B parameters in just two years. Without enough labeled data, such large-scale models may overfit on low-resource tasks, resulting in poor performance.

1.2 Knowledge Transfer

To overcome this challenge, we hypothesize that *transferring useful knowledge from other resources to these large models* may alleviate the label scarcity problem, while delivering improved

performance in the target tasks or domains. Indeed, this direction of knowledge transfer has recently begun attracting more and more attention. Especially, recent models have larger and larger capacity to accumulate knowledge through multiple related stages [39] or simultaneously learn multiple related tasks [139]. Supported by this related work, this dissertation focuses on transferring useful knowledge through *multi-stage learning* and *multi-task learning*.

1.2.1 Transfer via Multi-Stage Learning

Traditionally, a machine learning model's parameters are randomly initialized and then iteratively updated (e.g., through back-propagation). In other words, the model is trained from scratch and the only source of the task knowledge is the labeled training set of the target task, as shown in Figure 1.1.A. However, as introduced before, a low-resource task might not be enough to fully tune the (millions of) parameters of a large-scale model, leading to performance degeneration.

In one exciting direction, the traditional single stage can be extended to a new style of multi-stage learning as shown in Figure 1.1.B, which has shaken natural language understanding (NLU) with dramatic successes in recent years, enabled by Transformer-based pre-trained language models (PLMs) such as (1) BERT [28], ALBERT [61] and Roberta [77] (the encoder of Transformer [114]); (2) GPT2 [90] and GPT3 [17] (the decoder of Transformer); and (3) T5 [91] and BART [65] (the encoder and decoder of Transformer).

Originally, there are two stages [28]: pre-training and fine-tuning. Recent research [39] also shows that an intermediate domain/task adaptive pre-training between the general pre-training and fine-tuning is also helpful.

General Pre-training Stage. In this stage (Figure 1.1.B.1), these models capture general-purpose knowledge of a language [89, 94] in their parameters via self-supervised training tasks over large-scale unannotated data. For example, T5 [91] is pre-trained on a web crawled text collection[1] (about 750 GB). A typical pre-training task is Masked Language Modeling (MLM) [28]: randomly mask some (e.g., 15%) tokens from a sentence and then predict the masked tokens given the unmasked context, which is used in bi-directional attention PLMs like BERT. Via MLM, PLMs

[1] `https://www.tensorflow.org/datasets/catalog/c4`

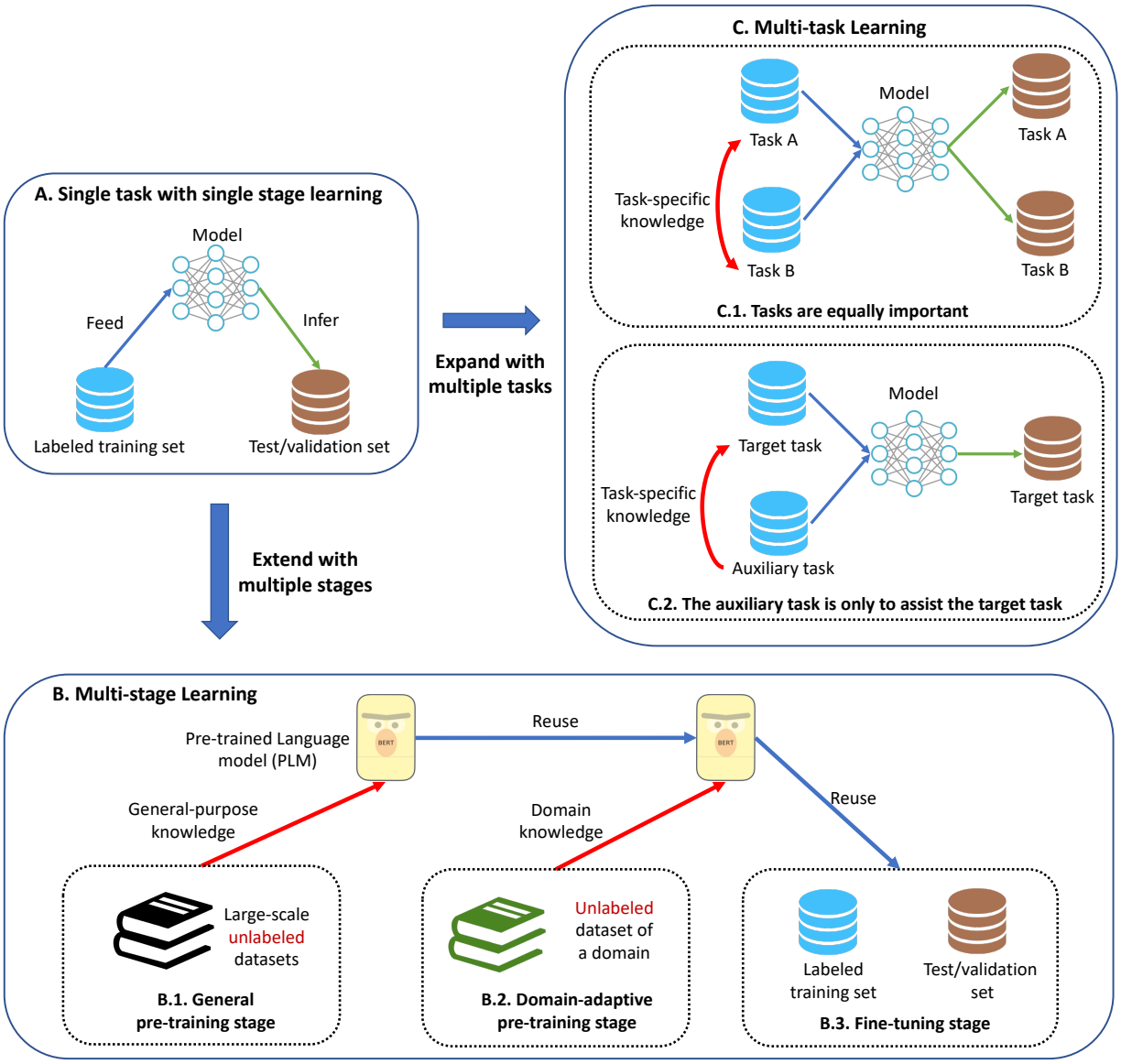


Figure 1.1: Knowledge transfer via multi-stage and multi-task learning. Blue dataset represents the labeled training set while brown dataset represents the testing (unlabeled) or validation (labeled) set. The blue line with arrow indicates that the dataset is fed into a model. The green line with arrow indicates that the model infers on the testing/validation set. The orange line with arrow indicates the knowledge transfer. (A) Single task with single stage learning: the model is trained on the labeled training set and then infer on the testing/validation set of the same task; (B) Multi-stage learning: (B.1) a PLM is first pre-trained over large-scale general unlabeled datasets (like Wikipedia or crawled web pages) to learn the general-purpose knowledge; (B.2) and then pre-train the same PLM over domain or task-specific unlabeled datasets (like PubMed) to learn the domain or task-specific knowledge; (B.3) finally, the PLM is fine-tuned over the target task as the same procedure as A. The knowledge learned from these pre-training stages can be transferred to improve the downstream task; (C) Multi-task learning – multiple tasks are trained together on the same model, which has two scenarios: (C.1) The tasks are equally important and the high testing performance required for each task. Hence, we are interested in the transferring of task-specific knowledge between the tasks; (C.2) The high testing performance is only required for target task and the role of auxiliary tasks is to assist the target task. Hence, only the knowledge transfer from the auxiliary tasks to the target task is of interest.

can learn the basics of language knowledge – e.g., we normally say “a gallon of milk" rather than “a of gallon milk" (syntax error) or “a pair of milk" (semantic error).

Fine-tuning Stage. In this stage (as shown in Figure 1.1.B.3), these models are further trained over the labeled training set of downstream tasks. *With the vast amount of knowledge transferred from the pre-training stage, the label scarcity problem can be significantly alleviated*, while these PLMs achieve the state-of-the-art (SOTA) performance over NLU benchmarks like GLUE [116] and SuperGLUE [117]. Moreover, the extreme cases of the label scarcity problem are few-shot learning [121] and zero-shot learning [119], where large-scale PLMs also obtain promising results. For example, GPT-3 can only see a few examples as the prompts and then achieve results even as good as SOTA models with fitting the full training set.

Domain/Task-Adaptive Pre-training. Recently, as shown in Figure 1.1.B.2, more and more researchers [6, 39, 62, 86] conduct a domain or task-adaptive pre-training after the general pre-training and before the fine-tuning, which aims to to incorporate models with the knowledge of a specific domain or task. Specifically, they continue to train already-pre-trained PLMs over unlabeled datasets of a specific domain or the unlabeled training set for a given task. For example, Gururangan et al. [39] continue training an already-pre-trained Roberta over unlabeled datasets (like articles) of biomedical, computer science and news domain respectively and achieve better performance than the model without such domain-adaptive pre-training on the downstream task of the corresponding domain.

Knowledge Transfer through the Stages. In summary, multi-stage learning is an effective way to alleviate the imbalance between scarce labeled training data and continuously increased model capacity, as introduced in Section 1.1. Rather than randomly initializing the weights of a model as in traditional single stage learning, multi-stage learning enables the model’s parameters to be well-updated before the fine-tuning and hence already imbues the model with vast amount of language, domain and task knowledge. Through the multiple stages, the knowledge learned from the previous stages can be accumulated in the same model and finally be transferred to boost the target task, which outperforms the single stage learning relying only on the scarce training set.

1.2.2 Transfer via Multi-Task Learning

In a complementary direction, single task learning (train only one task on the model) can be expanded to multi-task learning [139], which is to train multiple tasks simultaneously on the same model as shown in Figure 1.1.C and hence the task-specific knowledge can be transferred via the shared model between the tasks to alleviate the label scarcity problem. Compared with multi-stage learning, multi-task learning has been studied over decades but there are new challenges recently, which will be detailed in the next section.

There are two scenarios of multi-task learning: (i) when the tasks are equally important, and (ii) when there is a target task supported by auxiliary tasks.

Tasks are equally important. In the first scenario as shown in Figure 1.1.C.1, all tasks have the same priority and the goal is to improve the performance of all tasks (at least most tasks). Hence, we are interested in the performance over the testing set of each task and the knowledge sharing that benefits all the tasks. An example in computer vision is that a multi-task model can be designed to learn depth regression, semantic segmentation, and instance segmentation to create a complete scene understanding system [57], where the three tasks can reinforce each other. In the area of NLP, HyperGrid [106] jointly trains the tasks of GLUE or SuperGLUE on a Transformer-based language model, leading to the observation that low-resource tasks are improved due to the knowledge transferred from the high-resource tasks. Multi-task learning has been widely applied to model various user behaviors in recommender systems. Lu et al. [78] and Wang et al. [118] design multi-task models to jointly optimize the recommendation task and the corresponding explanation task.

Target task with auxiliary task. As shown in Figure 1.1.C.2, another scenario is to transfer knowledge from auxiliary tasks to improve a primary/target task, which is an example of the *auxiliary learning* paradigm [51, 67]. While multi-task learning aims to improve the performance across all tasks, auxiliary learning differs in that high test accuracy is only required for the target task, and the role of the other tasks is to assist in generalization of the primary task. Auxiliary learning has been widely used in many areas. For example, in social recommendation [38, 79, 120], knowledge

can be transferred from the social network to improve personalized recommendations via training the target task simultaneously with auxiliary tasks like predicting the connections or trust among users. In speech recognition, Toshniwal et al. [109] apply auxiliary supervision from phoneme recognition to improve the performance of conversational speech recognition. In computer vision, Liebel at al. [67] propose auxiliary tasks such as the global description of a scene to boost the performance for single scene depth estimation.

Knowledge Transfer through Shared Parameters. In summary, multi-task learning is another effective way to alleviate the imbalance between scarce labeled training data and continuously increased model capacity, as introduced in Section 1.1. No matter the scenario, the shared parameters of the multi-task model are updated jointly by all tasks and hence each task could receive more supervised signals than training it alone to alleviate the label scarcity problem and achieve potentially higher generalization ability.

1.3 Challenges to Conduct Knowledge Transfer

While these previous work offer promises of multi-stage learning and multi-task learning – and demonstrate the importance of transferring knowledge from the pre-training stages and jointly learned tasks to alleviate the label scarcity problem – there are still key technical challenges that may limit the effectiveness and efficiency of the knowledge transfer. In particular, this dissertation identifies three challenges:

- *Lack of Domain-Specific Knowledge: transferring domain-specific knowledge from pre-training stages to large-scale language models remains under-explored.* Without sufficient knowledge of a specific domain, the performance of fine-tuning PLMs remains limited for NLU tasks on the corresponding domain. For example, we observe that BERT-base only achieves the accuracy of 0.729 on a paraphrase identification dataset[2] of computer science domain [46]. As introduced in Section 1.2.1, when PLMs are fine-tuned on the labeled training set of a task, their knowledge learned from pre-training stages is leveraged to improve their performance on the task. However, the general pre-training stage is designed to

[2] https://github.com/heyunh2015/PARADE_dataset

learn the general-purpose language knowledge, where domain-specific knowledge has not been paid enough attention. The recent developed domain adaptive pre-training (DAPT) [39] stage aims to incorporate domain knowledge via further training the already-pre-trained models over unlabeled domain-specific datasets. For example, BioBERT [62] is obtained via training BERT over biomedical papers. However, DAPT still applies masked language modeling (MLM) as the pre-training task as in the general pre-training stage. In our work [46], we observe that simply using MLM over the domain-specific dataset can only slightly improve the performance. Hence, new pre-training techniques are required for more effectively infusing the domain-specific knowledge into PLMs.

- *Task Interference [55]: training multiple tasks jointly hinders the performance on individual tasks*, which has been studied for decades in general multi-task learning area [113]. Recently, Transformer-based language models like T5 and BART pose new challenges, which are unified text-to-text frameworks where all tasks share the same encoder-decoder architecture. For such universal modules, multi-task learning simply corresponds to mixing task data sets together and there are no task-specific parameters (e.g., prediction layer) for each task. Hence, inevitable task conflicts and difficulty in fitting all models within a single set of hard parameters is a challenging problem for transformer-based multi-task co-training. For example, previous work Raffel et al. [92] shows that co-learning all tasks together on a pre-trained Transformer model is inferior to fine-tuning the model for each task separately.

- *Negative Transfer [138]: transferring knowledge from auxiliary tasks might have a negative impact on the target task*. As introduced in Section 1.2.2, when the target task is trained simultaneously with several auxiliary tasks on the same model, the additional supervised signals can be transferred from the auxiliary tasks to the target task and alleviate the label scarcity problem. However, we observe that the performance of co-training the target task with the auxiliary tasks might be even worse than training it alone [47]. One of the possible reasons for such negative transfer is the divergence between the auxiliary tasks and target

tasks (or domains) [123, 138]. Interestingly, we observe that another key challenge to transfer knowledge from auxiliary tasks is the potential for a significant *imbalance of gradient magnitudes*, which can negatively affect the performance of the target task. Hence, more thinking and studies to uncover the reasons are required to alleviate the negative transfer.

1.4 Research Questions and Our Contributions

To overcome the challenges introduced above, this dissertation makes three unique contributions towards intelligent knowledge transfer for multi-stage and multi-task learning. Our contributions are guided by the following research questions:

- *How to transfer domain-specific knowledge from pre-training stages to large-scale PLMs for downstream domain-specific NLP tasks? (Chapter 2)* Disease is one of the fundamental biological entities in biomedical research and disease knowledge is required in many important health-related NLP tasks such as consumer health question answering, medical language inference and disease name recognition. First, we propose a new *disease knowledge infusion* training procedure to explicitly augment BERT-like PLMs with the disease knowledge. Disease knowledge infusion can be seen as a specialized domain-adaptive pre-training stage, which is between the general pre-training and fine-tuning. Specifically, the structure of Wikipedia pages is exploited as self-supervised learning signals for PLMs to learn the semantic relations between a disease-descriptive text and its corresponding aspect and disease. Extensive experiments show that the integration of PLMs with the disease knowledge can improve the performance on the health-related NLU tasks.

- *How to flexibly share knowledge between multiple tasks for Transformer-based multi-task co-training? (Chapter 3)* Second, we introduce HyperPrompt, a natural but novel extension of Prompt-Tuning [63] to multi-task learning for language understanding. HyperPrompt introduces task-conditioned hyper-prompts that conditions the model on task-specific information for constructing these prompts. We further improve upon this by introducing task-aware and layer-aware HyperNetworks [40] that parameterize and generate weights for the prompt

generation process. The usage of HyperNetwork imbues our model with the necessary flexibility of sharing task-dependent knowledge between the multiple co-trained tasks. HyperPrompt outperforms strong baselines on well-established benchmarks like SuperGLUE.

- *How to intelligently and flexibly transfer the knowledge from auxiliary tasks to improve the target task? (Chapter 4)* We observe that a significant *imbalance of gradient magnitudes* between the target task and the auxiliary tasks might cause the negative transfer and degenerate the performance of the target task. To overcome this challenge, we propose a novel algorithm MetaBalance to *prioritize* the target task via preventing auxiliary tasks from being so strong that they dominate the target task or too weak to help the target task. Specifically, the gradients of auxiliary tasks can be balanced *dynamically* throughout the training process and *adaptively* for different subsets of the shared parameters, which is more flexible than fixed weights for task losses. Extensive experiments over two real-world user behavior datasets show the effectiveness and flexibility of MetaBalance.

2. INFUSING DISEASE KNOWLEDGE INTO PRE-TRAINED LANGUAGE MODELS FOR BIOMEDICAL LANGUAGE UNDERSTANDING[1]

In this chapter, we demonstrate how to transfer specialized domain knowledge from pre-training stages into PLMs for enhancing their performance over downstream natural language understanding (NLU) tasks, which is the first challenge as introduced in Section 1.3. In particular, we focus on infusing disease knowledge into PLMs to improve health-related tasks such as consumer health question answering, medical language inference and disease name recognition. Specifically, we propose a new pre-training method named disease knowledge infusion, where the structure of Wikipedia pages is exploited as self-supervised learning signals for PLMs to learn the disease knowledge.

2.1 Introduction

Human disease is "a disorder of structure or function in a human that produces specific signs or symptoms" [85]. Disease is one of the fundamental biological entities in biomedical research and consequently it is frequently searched for in the scientific literature [50] and on the internet [18]. Knowledge of a disease includes information about various aspects of the disease, like the signs and symptoms, diagnosis, and treatment [34, 97, 111]. As an example, Table 2.1 highlights several aspects for COVID-19. Specialized disease knowledge is critical for many health-related and biomedical NLP tasks, including:

- *Consumer health question answering* [3] - the goal is to rank candidate passages for answering questions like "What is the diagnosis of *COVID-19*?" as shown in Figure 2.1a;
- *Medical language inference* [95] - the goal is to predict if a given hypothesis (description of a patient) can be inferred from a given premise (another description of the patient);

[1]This chapter is reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, 2020. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Copyright 2020 by ACL.

Table 2.1: Disease knowledge of *COVID-19* is presented from three aspects: symptoms, diagnosis and treatment (based on Wikipedia). Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

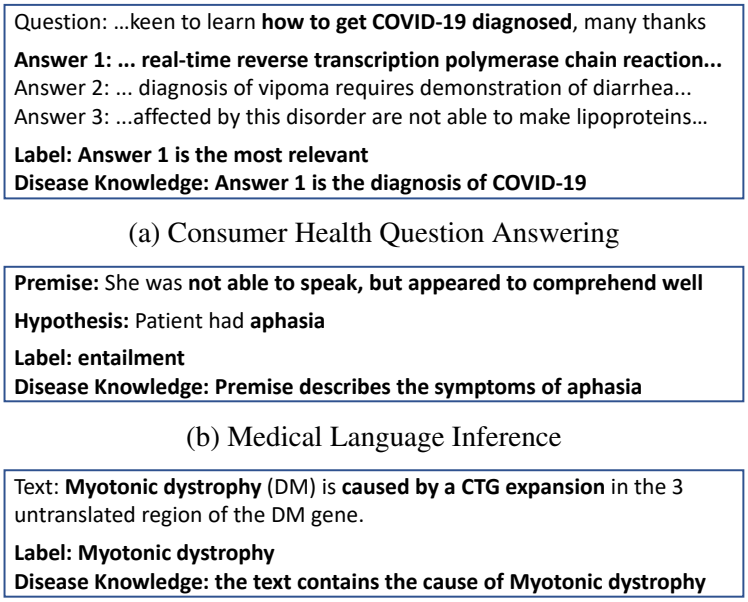
Disease	Aspect	Information
COVID-19	symptoms	Fever is the most common symptom, but highly variable in severity and presentation, with some older...
COVID-19	diagnosis	The standard method of testing is real-time reverse transcription polymerase chain reaction (rRT-PCR)...
COVID-19	treatment	People are managed with supportive care, which may include fluid therapy, oxygen support, and supporting...

- *Disease name recognition* [30] - the goal is to detect disease concepts in text.

For these tasks, it is critical for NLP models to capture disease knowledge, that is the semantic relations between a disease-descriptive text and its corresponding aspect and disease:

- As shown in Figure 2.1a, if models can semantically relate "...real-time reverse transcription polymerase chain reaction..." (disease-descriptive text) to the diagnosis (aspect) of *COVID-19* (disease), it is easier for them to pick up the most relevant answer among the candidates.
- Likewise, as shown in Figure 2.1b, if models know that the premise is the symptoms (aspect) of *Aphasia* (disease) in the hypothesis, they can easily predict that it is entailment not contradiction.
- Another example is shown in Figure 2.1c, if models can semantically relate "CTG expansion' to the cause (aspect) of *Myotonic dystrophy* (disease), it is easier for them to detect this disease.

In a nutshell, NLP models require the disease knowledge for these disease-related tasks.



(a) Consumer Health Question Answering

(b) Medical Language Inference

(c) Disease Name Recognition

Figure 2.1: Examples of tasks that can benefit from disease knowledge. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

In recent years, the family of Transformer-based pre-trained language models (PLMs) such as BERT [27], XLNet [132], RoBERTa [77], ALBERT [60] and T5 [92] has shaken NLP field with dramatic success in many benchmark tasks [36, 116, 117], with the multi-stage learning. As introduced in Section 1.2.1, the general pre-training stage aims to imbue the model with the syntactic and semantic knowledge of a language [89, 94] – we normally say a gallon of milk" rather than a of gallon milk" (syntax error) or a pair of milk" (semantic error). After that, PLMs are fine-tuned over labeled datasets of downstream tasks like text classification and the captured language knowledge is leveraged in this stage. Since the domain-specific knowledge has not been paid enough attention in the general pre-training stage, the performance of PLMs over the corresponding domain's NLP tasks remain limited.

Recently, conducting additional domain-adaptive pre-training (DAPT) [39] as an intermediate stage between the pre-training and fine-tuning has received more and more attention. In biomedical

domain, BioBERT [62], SciBERT [12], ClinicalBERT [6] and BlueBERT [86] report new SOTA performance on several biomedical tasks. Specifically, these Bio-PLMs are obtained via training a general-pre-trained BERT [27] over biomedical corpora like PubMed[1] via predicting randomly masked tokens given their context, which is the masked language model (MLM) [27] task. This MLM strategy is designed to capture the semantic relations between random masked tokens and their context, but not the disease knowledge. Because the corresponding disease and aspect *might not be randomly masked or might not be mentioned at all* in the disease-descriptive text, the semantic relations between them cannot be effectively captured via MLM. Therefore, a new training strategy is required to capture this disease knowledge.

Hence, we propose a new *disease knowledge infusion* training procedure to explicitly augment PLMs with the disease knowledge, which serves as a specialized domain-adaptive pre-training between the general pre-training and fine-tuning. The core idea is to train PLMs to infer the corresponding disease and aspect from a disease-descriptive text, enabled by self-supervised signals from Wikipedia. Given a passage extracted from a section (normally describes an aspect) of a disease's Wikipedia article, PLMs are trained to infer the title of the corresponding section (aspect name) and the title of the corresponding article (disease name). To evaluate the quality of disease knowledge infusion, we conduct experiments on a suite of BERT models – including BERT, BlueBERT, ClinicalBERT, SciBERT, BioBERT, and ALBERT – over consumer health question (CHQ) answering, medical language inference, and disease name recognition. We find that (1) these models can be enhanced in nearly all cases. For example, accuracy of BioBERT on CHQ answering is improved from 68.29% to 72.09%; and (2) our method is superior to MLM for infusing the disease knowledge. Moreover, new SOTA results are observed in two datasets. These results demonstrate the potential of disease knowledge infusion into pre-trained language models like BERT.

2.2 Related Work

Knowledge-Enriched BERT: Incorporating external knowledge into BERT has been shown to be effective. Such external knowledge includes world (factual) knowledge for tasks such as entity

[1] `https://pubmed.ncbi.nlm.nih.gov/`

typing and relation classification [72, 88, 129, 140], sentiment knowledge for sentiment analysis [107, 133], word sense knowledge for word sense disambiguation [64], commonsense knowledge for commonsense reasoning [59] and sarcasm generation [20], legal knowledge for legal element extraction [141], numerical skills for numerical reasoning [37], and coding knowledge for code generation [130].

Biomedical BERT: BERT can also be enriched with biomedical knowledge via pre-training over biomedical corpora like PubMed, as in BioBERT [62], SciBERT [12], ClinicalBERT [6] and BlueBERT [86]. These biomedical BERT models report new SOTA performance on several biomedical tasks. Disease knowledge, of course, is a subset of biomedical knowledge. However, there are two key differences between these biomedical BERT models and our work: (1) Many biomedical BERT models are pre-trained via BERT's default MLM that predicts 15% randomly masked tokens. In contrast, we propose a new training task: disease knowledge infusion, which infers the disease and aspect from the corresponding disease-descriptive text; (2) Biomedical BERT models capture the general syntactic and semantic knowledge of biomedical language, while our work is specifically designed for capturing the semantic relations between a disease-descriptive text and its corresponding aspect and disease. Experiments reported in Section 4.5 show that our proposed method can improve the performance of each of these biomedical BERT models, demonstrating the importance of disease knowledge infusion.

Biomedical Knowledge Integration Methods with UMLS: Previous non-BERT methods connect data of downstream tasks with knowledge bases like UMLS [95, 100]. For example, they map medical concepts and semantic relationships in the data to UMLS. After that, these concepts and relationships are encoded into embeddings and incorporated into models [100]. The advantage is that they can explicitly incorporate knowledge into models. However, these methods have been outperformed by biomedical BERT models such as BioBERT in most cases.

2.3 Proposed Method: Disease Knowledge Infusion Training

In this section, we propose a new training task: Disease Knowledge Infusion Training, serving as a specialized domain-adaptive pre-training between the general pre-training and fine-tuning.

Table 2.2: Eight aspects of knowledge of a disease that are considered in this work. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

Aspect Name	Definition
Information	The general information of a disease.
Causes	The causes of a disease.
Symptoms	The signs and symptoms of a disease.
Diagnosis	How to test and diagnose a disease.
Treatment	How to treat and manage a disease.
Prevention	How to prevent a disease.
Pathophysiology	The physiological processes of a disease.
Transmission	The means by which a disease spread.

Our goal is to integrate BERT-like pre-trained language models with disease knowledge to achieve better performance on a variety of medical domain tasks including answering health questions, medical language inference, and disease name recognition. Our approach is guided by three questions: Which diseases and aspects should we focus on? How do we infuse disease knowledge into BERT-like models? What is the objective function of this training task?

2.3.1 Targeting Diseases and Aspects

First, we seek a disease vocabulary that provides disease terms. Several resources include Medical Subject Headings[2] (MeSH) [69], the National Cancer Institute thesaurus [24], SNOMED CT [32], and Unified Medical Language System (UMLS) [15]. Each has a different scope and design purpose, and it is an open question into which is most appropriate here. As a first step, we select MeSH, which is a comprehensive controlled vocabulary proposed by the National Library of Medicine (NLM) to index journal articles and books in the life sciences, composed of 16 branches like anatomy, organisms, and diseases. We collect all unique disease terms from the Disease (MeSH tree number C01-C26) and Mental Disorder branch (MeSH tree number F01), resulting in 5,853 total disease terms.

Knowledge of a disease involves information about various aspects of the disease [34, 97, 111].

[2] `https://meshb.nlm.nih.gov/treeView`

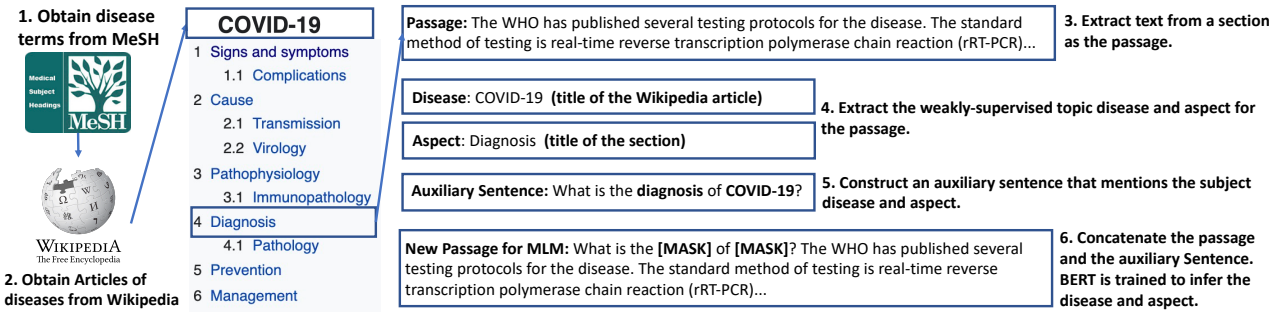


Figure 2.2: Disease knowledge infusion training: an example with COVID-19. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

For each aspect, we focus on text alone (excluding images or other media). Following Abacha and Demner-Fushman [1], we consider eight disease aspects as shown in Table 2.2.

2.3.2 Weakly Supervised Knowledge Infusion from Wikipedia

Given the target set of diseases and aspects, the next challenge is how to infuse knowledge of the aspects of these diseases into BERT-like models. We propose to train BERT to infer the corresponding disease and aspect from a disease-descriptive text. By minimizing the loss between the predicted disease and aspect and the original disease and aspect, the model should memorize the semantic relations between the disease-descriptive text and its corresponding disease and aspect.

A straightforward approach is to mask and predict the disease and aspect in the disease-descriptive text. However, this strategy faces two problems: (1) Given a passage extracted from disease-related papers, clinical notes, or biomedical websites, the ground-truth of its topic (i.e., disease and aspect) is difficult to identify. Medical expert annotation is time-consuming and expensive; while automatic annotation can suffer from large errors. For example, we need to recognize disease names in the passage, which is yet another challenging and still open problem in biomedical text mining [30]; (2) Diseases and aspects mentioned in a passage are not necessarily the topic words. Multiple disease names or aspect names might appear, making it difficult to determine which is the correct topic. For example, in Table 2.1, the symptoms of *COVID-19* also

mentions *fever*[3], while the correct topic is *COVID-19*.

Weakly-Supervised Knowledge Source: Instead of annotating an arbitrary disease-related passage, we exploit the structure of Wikipedia as a weakly-supervised signal. In many cases, each disease's Wikipedia article consists of several sections where each introduces an aspect of the disease (like diagnosis). For example, step 2 in Figure 2.2 shows several aspects on the Wikipedia page for *COVID-19*. By extracting the passage from each section, the title of the section (e.g., diagnosis) is the topic aspect of the passage and the title of the article is the topic disease (e.g., COVID-19). Specifically, we search Wikipedia to obtain the articles for the 5,853 target disease terms from MeSH and apply regular expressions to extract the text of the sections corresponding to the appropriate aspects. In total, we collect a disease knowledge resource consisting of 14,617 passages.[4] In fact, there are other online resources[5] with the similar structure. As a first step, we start with Wikipedia.

Auxiliary Sentences for Disease and Aspect Prediction: The second problem is that the extracted passages do not necessarily mention the corresponding disease and the aspect. For example, in Table 2.1, the disease name "COVID-19" does not appear in the information of its symptoms. In the disease knowledge resource, we find that only 51.4% of passages mention both the corresponding diseases and aspects. Hence, we cannot simply mask-and-predict the disease and aspect because the passage does not mention them at all.

A remedy for this problem is an auxiliary sentence that contains the corresponding disease and aspect for each passage. We use a template of question style: "What is the [*Aspect*] of [*Disease*]?" to automatically generate auxiliary sentences as shown in step 5 in Figure 2.2. Some examples are shown in Table 2.3. The advantage of this question style template is that the cloze statement of the auxiliary sentences for all aspects (except for the "information" aspect) are the same (What is the [MASK] of [MASK]?). Hence, the auxiliary sentences provide no clues (i.e., bias) for predicting the corresponding aspect.

[3]Fever is included in the disease branch of MeSH.

[4]Note that each disease article does not necessarily have all eight target aspects.

[5]`https://medlineplus.gov/skincancer.html`

Table 2.3: Examples of auxiliary sentences. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

Aspect Name	Auxiliary Sentence
Diagnosis	What is the diagnosis of COVID-19?
Treatment	What is the treatment of COVID-19?
Prevention	What is the prevention of COVID-19?
Transmission	What is the transmission of COVID-19?
Cloze Statement	What is the [MASK] of [MASK]?

After that, we replace the corresponding disease and aspect with the special token [MASK] in the auxiliary sentences. Then, we insert the auxiliary sentence at the beginning of its corresponding passage to form a new passage with a question-and-answer style as shown in Figure 2.2, where BERT is trained to predict the original tokens of the masked disease and aspect.

2.3.3 Training Objective and Details

Finally, we show the objective function of disease infusion training. Since most disease names are out of BERT vocabulary, the WordPiece tokenizer [127] will split these terms into sub-word tokens that exist in the vocabulary. For example, “COVID-19" will be split into 4 tokens: “co", “vid", “-" and “19". Formally, let $X = (x_1, ..., x_T)$ denote a sequence of T tokens that are split from a disease name where x_t is the t-th token. The original cross-entropy loss is to get the conditional probability of a masked token as close as possible to the 1-hot vector of the token:

$$\mathcal{L}_{disease} = -\sum_{t=1}^{T} log\ p(x_t|passage) \tag{2.1}$$

where $p(x_t|context)$ is a conditional probability over x_t given the corresponding passage, which can be defined as:

$$p(x_t|passage) = \frac{exp(z_t)}{\sum_{z \in \mathcal{V}} exp(z)} \tag{2.2}$$

where $\mathcal{V}$ is the vocabulary and z_t is the unnormalized log probability of x_t. Let $\mathbf{y}_t$ denote the embedding of token x_t from the output layer of BERT. We can estimate z_t via:

$$z_t = \mathbf{w} \cdot \mathbf{y}_t + b \tag{2.3}$$

where the weight **w** and bias b are learnable vectors.

Note that the vocabulary size of BERT is around 30,000 which means masked language modeling task is a 30,000 multi-class problem. The logits (like z_t) after the normalization of softmax (Equation 2.2) will be pretty small (the expectation of mean should be around 1/30,000=3.3*e-5), which might cause some obstacles for the learning. Therefore, we also maximize the raw logits (like z_t) before softmax normalization which might keep more useful information. Empirically, we add the reciprocal of the logits to the cross-entropy loss:

$$\mathcal{L}_{disease} = -\sum_{t=1}^{T} logp(x_t|passage) + \frac{\beta}{\sum_{t=1}^{T} z_t} \tag{2.4}$$

where β balances the two parts of the loss. The final objective function is combined with the loss of the disease and aspect: $\mathcal{L} = \mathcal{L}_{disease} + \mathcal{L}_{aspect}$ where $\mathcal{L}_{aspect} = -log\ p(a|passage)$ and a is the token of the aspect name. By minimizing this loss function, BERT can update its parameters to store the disease knowledge.

2.4 Experiments

In this section, we examine disease knowledge infusion into six BERT variants over three disease-related tasks: health question answering, medical language inference, and disease name recognition.

Reproducibility: *The code and data in this chapter is released.*[6] A model is firstly initialized with the pre-trained parameters from BERT or its variants and then is further trained by disease knowledge infusion to capture the disease knowledge. We use a widely used Pytorch implementation[7] of BERT and Adam as the optimizer. We empirically set learning rate as 1e-5, batch size as

[6] https://github.com/heyunh2015/diseaseBERT

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

16 and β as 10. Because MeSH (5,853 disease terms) is chosen as the disease vocabulary in our experiments, as a smaller vocabulary compared with others like UMLS (540,000 disease terms), we obtain a relatively small dataset of 14,617 passages. Hence, the training of disease knowledge infusion is as fast as fine-tuning BERT over downstream datasets, which takes 2-4 epochs to enhance BERT for a better performance on downstream tasks, which will be discussed in Section 2.4.5. The training is performed on one single NVIDIA V100 GPU and it takes about 10 minutes to complete one training epoch using BERT-base architecture. The reproducibility for fine-tuning over downstream tasks will be detailed in Section 2.4.2.

2.4.1 BERT and its Biomedical Variants

We consider six BERT models: two pre-trained over general language corpora (BERT and ALBERT) and four pre-trained over biomedical corpora (Clinical BERT, BioBERT, BlueBERT and SciBERT).

BERT [28] is a multi-layer bidirectional Transformer encoder. Since the following biomedical versions of BERT are often based on the BERT-base architecture (12 layers and 768 hidden embedding size with 108M parameters), we choose BERT-base here for fair comparison.

ALBERT[8] [61] compresses the architecture of BERT by factorized embedding parameterization and cross-layer parameter sharing. Via this compression, ALBERT can have a substantially higher capacity than BERT, with stronger performance on many tasks. We choose the maximum version ALBERT-xxlarge (12 layers and 4096 hidden embedding size with 235M parameters).

BioBERT[9] [62] is the first BERT pre-trained on biomedical corpora. It is initialized with BERT's pre-trained parameters (108M) and then further trained over PubMed abstracts (4.5B words) and PubMed Central full-text articles (13.5B words). We choose the best version BioBERT v1.1.

ClinicalBERT[10] [6] is a BERT model initialized from BioBERT v1.0 [62] and further pre-trained over approximately 2 million notes in the MIMIC-III v1.4 database of patient notes [53]. We adopt the best performing version of ClinicalBERT (108M parameters) based on discharge summaries of

[8] https://huggingface.co/albert-xxlarge-v2
[9] https://github.com/dmis-lab/biobert
[10] https://huggingface.co/emilyalsentzer

Table 2.4: Summary of tasks and datasets. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

Datasets	Train	Dev	Test
MEDIQA-2019	208 $(1,701)^1$	25 (234)	150 (1,107)
TRECQA-2017	254 (1,969)	25 (234)	104 (839)
MEDNLI	$11,232^2$	1,395	1,422
BC5CDR-disease	$4,182^3$	4,244	4,424
NCBI	5,145	787	960

1, Questions with associated answers; 2, Pairs of premise and hypothesis; 3, Disease name mentions.

clinical notes: Bio-Discharge Summary BERT.

BlueBERT[11] [86] is firstly initialized from BERT (108M parameters) and further pre-trained over a biomedical corpus of PubMed abstracts and clinical notes [53].

SciBERT[12] [12] is a BERT-base (108M parameters) model pre-trained on a random sample of the full text of 1.14M papers from Semantic Scholar [7], with 18% of papers from the computer science domain and 82% from the biomedical domain.

2.4.2 Tasks

We test disease knowledge infusion over three biomedical NLP tasks. The dataset statistics are in Table 2.4. For fine-tuning of BERT and its variants, the batch size is selected from [16, 32] and learning rate is selected from [1e-5, 2e-5, 3e-5, 4e-5, 5e-5].

Task 1: Consumer Health Question Answering. The objective of this task is to rank candidate answers for consumer health questions.

Datasets. We consider two datasets: MEDIQA-2019 [13] and TRECQA-2017 [2].[13] MEDIQA-2019 is based on questions submitted to the consumer health QA system CHiQA[14]. TRECQA-2017 is based on questions submitted to the National Library of Medicine. Medical experts manually re-

[11] https://github.com/ncbi-nlp/bluebert
[12] https://huggingface.co/allenai/scibert_scivocab_uncased
[13] https://sites.google.com/view/mediqa2019
[14] https://chiqa.nlm.nih.gov/

Table 2.5: Experimental results. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

Tasks	Consumer Health Question Answering						NLI	NER	
Datasets	MEDIQA-2019			TRCEQA-2017			MEDNLI	BC5CDR	NCBI
Metrics(%)	Accuracy	MRR	Precision	Accuracy	MRR	Precision	Accuracy	F1	F1
BERT	64.95	82.72	66.49	74.61	56.17	52.55	75.95	83.09	85.14
BERT + disease*	66.40↑	83.33↑	68.94↑	75.33↑	56.41↑	54.01↑	77.29↑	83.47↑	86.81↑
BlueBERT	65.13	81.50	67.35	74.26	48.40	52.55	82.21	85.73	87.78
BlueBERT + disease	68.47↑	81.17	71.57↑	77.59↑	50.96↑	57.62↑	83.90↑	86.30↑	87.79↑
ClinicalBERT	67.30	84.78	70.59	77.00	52.56	56.62	81.50	84.90	87.25
ClinicalBERT + disease	69.02↑	88.94↑	69.84	78.90↑	54.97↑	60.40↑	81.65↑	85.63↑	87.22
SciBERT	68.47	84.47	68.07	77.23	54.57	57.54	80.94	86.16	87.24
SciBERT + disease	73.35↑	85.44↑	76.28↑	79.02↑	56.57↑	59.57↑	82.14↑	86.34↑	88.30↑
BioBERT	68.29	83.61	72.78	77.12	49.84	57.25	81.86	85.99	87.70
BioBERT + disease	72.09↑	87.78↑	74.40↑	78.43↑	54.76↑	58.45↑	82.21↑	86.52↑	87.14
ALBERT	76.54	88.46	81.41	75.09	**58.57**	53.03	85.48	84.28	87.56
ALBERT + disease	**79.49**↑	90.00↑	**84.02**↑	**80.10**↑	57.21	**62.40**↑	**86.15**↑	84.71↑	87.69↑
SOTA*	78.00	**93.67**	81.91	77.23	54.57	57.54	84.00	**87.15**	**89.71**

* SOTA, state-of-the-art as of May 2020, to the best of our knowledge.

* “ + disease" means that we train BERT via disease knowledge infusion training before fine-tuning.

ranked the original retrieved answers and provide *Reference Score* (1 to 11) and *Reference Rank* (4: Excellent, 3: Correct but Incomplete, 2: Related, 1: Incorrect).

Fine-tuning. MEDIQA-2019 and TRECQA-2017 are used as the fine-tuning dataset for each other. MEDIQA-2019 also contains a validation set for tuning hyper-parameters for both datasets. Following Xu et al. [131], the task is cast as a regression problem where the target score is: $score =$ *Reference Score* $- \frac{\textit{Reference Rank}-1}{m}$ where m is the number of candidate answers. Each question-answer pair is packed as a single sequence as the input for BERT. A single linear layer is on top of the output embedding of the special token [CLS] to generate the predicted score. MSE is adopted as the loss and we use Adam as the optimizer. All hyper-parameters are tuned on the validation set in terms of accuracy, where we set the batch size as 16 and learning rate as 1e-5.

SOTA. The state-of-the-art (SOTA) performance on MEDIQA-2019 is achieved by Xu et al. [131], which is an ensemble method. Because TRECQA-2017 is fine-tuned on MEDIQA-2019, which is different from the original settings [2] (BERT had not been proposed at that time), we use the best result of SciBERT among the BERT models as SOTA for TRECQA-2017.

Task 2: Medical Language Inference. The goal of this task is to predict whether a given hypothesis can be inferred from a given premise.

Datasets. MEDNLI [95] is a natural language inference dataset for the clinical domain.[15] For each premise (a description of a patient) selected from clinical notes (MIMIC-III), clinicians generate three hypotheses: entailment (alternate true description of the patient), contradiction (false description of the patient), and neutral (alternate description that might be true).

Fine-tuning. Following Peng et al. [86], we pack the premise and hypothesis together into a single sentence. A linear layer is on top of the output embedding of [CLS] to generate logits. Cross-entropy loss function is adopted, and we use Adam as the optimizer. All hyper-parameters are tuned on the validation set in terms of accuracy, where we set the batch size as 32 and learning rate as 1e-5.

SOTA. To the best of our knowledge, the state-of-the-art on MEDNLI is achieved by BlueBERT,

[15] `https://physionet.org/content/mednli/1.0.0/`

reported in Peng et al. [86].

Task 3: Disease Name Recognition. This task is to detect disease names from free text.

Datasets. BC5CDR[16] [124] and NCBI[17] [30] are collections of PubMed titles and abstracts. Medical experts annotate diseases mentioned in the collection. Since BC5CDR includes both chemicals and diseases, we focus on diseases in this dataset.

Fine-tuning. Following Peng et al. [86], we cast this task as a token-level tagging (classification) problem, where each token is classified into three classes: B (beginning of a disease), I (inside of a disease) or O (out of a disease). Cross-entropy is adopted as the loss function and we use Adam as the optimizer. All hyper-parameters are tuned on the validation set in terms of F1, where we set the batch size as 32 and learning rate as 5e-5.

SOTA. The best performance is achieved by BioBERT v1.1, reported in Lee et al. [62][18].

2.4.3 Results

The experimental results are presented in Table 2.5. We show each original model and its disease knowledge infused variant (e.g,. *BERT* and *BERT + disease*). We have two main findings:

Effectiveness of Disease Infusion. First, by infusing disease knowledge via our new training regimen, we see a significant improvement in nearly all cases. For example, *ALBERT + disease* achieves 80.10% in terms of accuracy which is superior to 75.09% by ALBERT alone on TRECQA-2017. Standing on the shoulders of ALBERT, disease knowledge infusion leads to state-of-the-art results on MEDIQA-2019 and MEDNLI, to the best of our knowledge. Although BERT and ALBERT are pre-trained on all of Wikipedia, including the articles of diseases, they might not pay enough attention to the disease part since Wikipedia is so large. Hence, disease knowledge infusion that leverages the Wikipedia structure to capture the disease knowledge is a complement for BERT and ALBERT. Moreover, it is encouraging to see the improvements of disease knowledge infusion in biomedical BERT models, even though these variants are already pre-trained over

[16] https://github.com/ncbi-nlp/BLUE_Benchmark

[17] https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/

[18] Although SciBERT reports a better result in NCBI, it uses a conditional random field on top of BERT, which is more complicated than the linear layer normally used in fine-tuning for BERT models including BioBERT.

large-scale biomedical corpora like PubMed with access to comprehensive disease information. This improvement demonstrates that the disease knowledge captured by our method – that is, the semantic relations between a disease-descriptive text and its corresponding aspect and disease – is different from the general linguistic knowledge in the biomedical domain captured by the randomly masked tokens prediction strategy of these biomedical BERT models. To sum up, the results show that the proposed disease knowledge infusion method can effectively complement BERT and its biomedical variants and hence improve the performance on health question answering, medical language inference, and disease name recognition.

Effectiveness of Biomedical BERT Models. We also observe that BERT models pre-trained on biomedical corpora outperform the same BERT architecture that is pre-trained on general language corpora. For example, BioBERT achieves 68.29% in terms of accuracy on MEDIQA-2019 while BERT only obtains 64.95%. This demonstrates that with the same model architecture, pre-training on biomedical corpora can capture more biomedical language knowledge that improves BERT for downstream biomedical tasks.[19]

In addition, we find that a high-capacity model like ALBERT can achieve similar performance as biomedical BERT models on TRECQA-2017, BC5CDR and NCBI, and even better performance on MEDIQA-2019 and MEDNLI. This observation might motivate new biomedical pre-trained models based on larger models like ALBERT-xxlarge.

2.4.4 Ablation Study

We present the results of an ablation study on MEDIQA-2019 in Table 2.6. Similar results are observed on other datasets but omitted here due to the space limitation. We first remove "Auxiliary Sentence". That is, we remove the auxiliary question: "What is the [*Aspect*] of [*Disease*]?" and let BERT to predict the corresponding disease and aspect in the original passage if they appear. We observe worse results in terms of accuracy and precision, which shows that the auxiliary sentence is an effective remedy for the problem that some passages do not mention their disease and aspects.

[19]Note that our results for the biomedical BERT models in Table 2.5 are slightly different from the results reported in the original papers that normally only provide a search range for hyper-parameters and not the specific optimal ones.

Table 2.6: Ablation study on MEDIQA-2019. Reprinted with permission from “Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

Variants	Accuracy	MRR	Precision
Default	79.49	90.00	84.02
- Auxiliary Sentence	78.23	90.89	78.10
- Aspect Prediction	78.41	89.06	80.00
- Disease Prediction	72.90	85.72	79.44
15% Randomly Masked Tokens	77.06	87.33	85.18

We also remove aspect prediction or disease prediction in the auxiliary sentence; both lead to worse results but removing disease prediction leads to a much lower performance. This shows that it is more important for BERT to infer the disease than the aspect from the passage. We also pre-train BERT on the same corpus (the disease-related passages) as our method. Following Devlin et al. [28], we randomly mask 15% tokens in each sentence and let BERT to predict them. As shown in “15% Randomly Masked Tokens", we observe that our proposed disease infusion training task outperforms the default masked language model in BERT. This shows that our approach that leverages the structure of Wikipedia article to enhance the disease knowledge infusion works better than simply adding more data to the training process. Specifically, via leveraging the Wikipedia structure, we could effectively mask key words like aspect names and disease names that are related to disease knowledge and hence more effective than randomly masking strategy over the simply added data.

2.4.5 Learning Curve

In this section, we present the learning curve of our proposed disease infusion training task. The x-axis denotes the training epochs and the y-axis denotes the performance of BERT models that are augmented with disease infusion training at that epoch. We take BioBERT and MEDIQA-2019 as examples; similar results are obtained in other models over other tasks. The results in terms of accuracy are presented in Figure 2.3, where we observe that (1) disease knowledge infusion takes

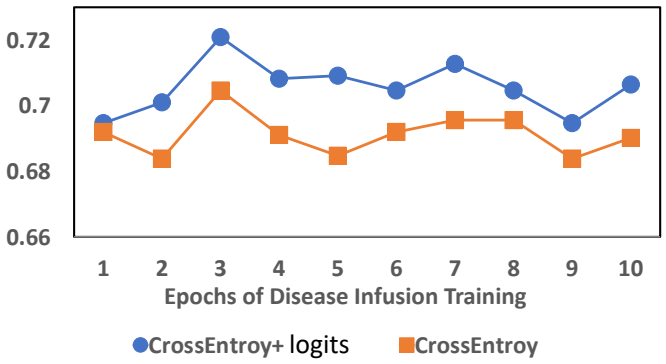


Figure 2.3: Learning curve of disease infusion knowledge. The y-axis is the accuracy of BERT models over MEDIQA-2019. Reprinted with permission from "Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition" by Yun He, Ziwei Zhu, Yin Zhang, Qin Chen and James Caverlee, EMNLP 2020.

only three epochs to achieve the optimal performance on BioBERT over the CHQ answering task. (2) cross-entropy loss used by disease knowledge infusion can be enhanced by adding the term of maximizing the raw logits (Equation 2.4).

2.5 Summary

In this chapter, we propose a new disease infusion training procedure to augment BERT-like pre-trained language models with disease knowledge. We conduct this training procedure on a suite of BERT models and evaluate them over disease-related tasks. Experimental results show that these models can be enhanced by this disease infusion method in nearly all cases. For example, accuracy of BioBERT on CHQ answering is improved from 68.29% to 72.09%. This improvement demonstrates that the disease knowledge captured by our method – that is, the semantic relations between a disease-descriptive text and its corresponding aspect and disease – is different from the general linguistic knowledge in the biomedical domain captured by the MLM strategy of previous bio-PLMs. To sum up, the results show that the proposed disease knowledge infusion method can effectively complement BERT and its biomedical variants and hence improve the performance on health question answering, medical language inference, and disease name recognition. Moreover, new SOTA results are observed in two datasets: MEDIQA-2019 and MEDNLI.

3. HYPERPROMPT: PROMPT-BASED TASK-CONDITIONING OF TRANSFORMERS[1]

The second challenge of effective knowledge transfer that this dissertation identifies is the task interference – that is, training multiple tasks jointly hinders the performance on individual tasks, which is especially true for Transformer-based multi-task learning because there are no task-specific parameters for each task. In this chapter, we explore the use of prompts to alleviate the task interference, inspired by Prompt-Tuning as a new paradigm for fine-tuning pre-trained language model. Specifically, we propose HyperPrompt, a novel architecture for prompt-based task-conditioning of Transformers for multi-task learning. The hyper-prompts are end-to-end learnable via generation by a HyperNetwork, enabling flexible knowledge sharing among tasks. We show that HyperPrompt is competitive against strong multi-task learning baselines with as few as 0.14% of additional task-conditioning parameters, achieving great parameter and computational efficiency. Through extensive empirical experiments, we demonstrate that HyperPrompt can achieve superior performances over strong T5 multi-task learning baselines and parameter-efficient adapter variants including Prompt-Tuning and HyperFormer++ on natural language understanding benchmarks of GLUE and SuperGLUE across many model sizes.

3.1 Introduction

Fine-tuning a pre-trained Transformer model over a specific task is an effective method, achieving dramatic success in the NLP field recently. However, the space complexity of the fine-tuning is $\mathcal{O}(n)$ where n is the number of downstream tasks because an entire new model is required for every task. As an alternative, multi-task learning, where multiple tasks can be jointly trained on the same Transformer model, is more parameter-efficient with the complexity of $\mathcal{O}(1)$.

We consider the general setting of multi-task learning for a set of tasks $\{\mathcal{D}_\tau\}_{\tau=1}^{T}$, where T is the total number of tasks and $\{\mathcal{D}_\tau\} = \{x_\tau^{(n)}, y_\tau^{(n)}\}_{n=1}^{N_\tau}$ indicates the corresponding training set of

[1]This chapter is reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

the τ-th task with N_τ samples. We assume that a pre-trained Transformer model $f_\theta(\cdot)$ (e.g., T5) is given, where the model is parameterized by θ. To tackle such multi-task learning problem with $f_\theta(\cdot)$, we minimize the following objective function:

$$\mathcal{L}(\theta) = \sum_{\tau=1}^{T} \sum_{n=1}^{N_\tau} C(f_\theta(x_\tau^{(n)}), y_\tau^{(n)}) \tag{3.1}$$

where $C(\cdot, \cdot)$ is typically the cross-entropy loss and $f_\theta(x_\tau^{(n)})$ is the output for training sample $x_\tau^{(n)}$.

Transformer-based pre-trained language models such as T5 [91] and BART [65] are unified text-to-text frameworks where all tasks share the same encoder-decoder architecture – $\{\{x_\tau^{(n)}\}_{n=1}^{N_\tau}\}_{\tau=1}^{T}$ are fed into the same encoder and $\{\{\hat{y}_\tau^{(n)}\}_{n=1}^{N_\tau}\}_{\tau=1}^{T}$ are generated by the same decoder. For such universal modules, multi-task learning simply corresponds to mixing task data sets together. Therefore, θ is task-agnostic (i.e., all parameters are shared) and there is no task-specific classification or regression networks for each task as in encoder-only modules Devlin et al. [29], Liu et al. [75].

However, inevitable task interference [55] and conflicts in fitting all task data sets within a single set of parameters is a challenging problem for multi-task co-training. For example, previous work Raffel et al. [91] shows that co-learning all tasks together on a pre-trained Transformer model is inferior to fine-tuning the model for each task separately, which can be especially true for low-resource tasks [91, 106]. To overcome this challenge, a natural way is to introduce a set of task-conditioned parameters $\{\delta_\tau\}_{\tau=1}^{T}$ into $f_\theta(.)$ for Transformers. The objective can be updated as:

$$\mathcal{L}(\theta, \{\delta_\tau\}_{\tau=1}^{T}) = \sum_{\tau=1}^{T} \sum_{n=1}^{N_\tau} C(f_{\theta,\delta_\tau}(x_\tau^{(n)}), y_\tau^{(n)}) \tag{3.2}$$

where δ_τ is the task-specific parameterization for the τ-th task. During training, both θ and $\{\delta_\tau\}_{\tau=1}^{T}$ are updated via back-propagation because we observe a large performance drop in SuperGLUE when backbone model θ is frozen and only task-conditioned parameters are tuned, as done in Karimi Mahabadi et al. [56], which will be detailed in Section 3.3.4.

To this end, our goal is to design task-conditioned parameterization of Transformer models to achieve *greater parameter and computational efficiency as well as Pareto efficiency for multi-*

task learning. More explicitly, we have two goals: (1) improving the fine-tuning performance of most tasks in $\{\mathcal{D}_\tau\}_{\tau=1}^T$ by introducing task-conditioned parameters $\{\delta_\tau\}_{\tau=1}^T$ into $f_\theta(.)$ and (2) under the constraint that $\sum_\tau \|\{\delta_\tau\}_{\tau=1}^T\|_0 \ll \|\theta\|_0$, which means that the model capacity will not be significantly increased. And the computational cost would not increase substantially either.

In this chapter, we introduce HyperPrompt, serving as the task-conditioned parameterization ($\{\delta_\tau\}_{\tau=1}$) of Transformer models, which is a natural but novel extension of Prompt-Tuning [63] to multi-task learning (MTL) for language. HyperPrompt introduces task-conditioned hyper-prompts that conditions the model on task-specific information for constructing these prompts. Hyper-prompts are injected to the keys and values in the self-attention module, reminiscent of memory augmented Transformers [104]. This mitigates the cost of having prompts pass through the standard FFN layers in Transformers and serves as additional task-specific memory tokens for queries to attend to.

We further improve upon this by introducing task-aware and layer-aware HyperNetworks [41] that parameterize and generate weights for the prompt generation process. The usage of HyperNetwork imbues our model with the necessary flexibility and expressiveness, especially when it comes to incorporating task-specific and layer-specific information to the network. Meanwhile, HyperPrompt remains very parameter and computational efficient and friendly to multi-task scaling: the additional parameters scale sub-linearly with, and are independent of the number of tasks in practice. While Hypernetworks have enjoyed some success in learning adapters [56, 105] and/or continual learning [115], we note that this is the first exploration of HyperNetworks as a prompt generator.

Contrary to prior work, we additionally propose to fine-tune the entire network instead of only the hyper-prompts. We make several compelling arguments for this. Firstly, Lester et al. [63] shows that parameter efficient Prompt-Tuning only shines for large (e.g., 11B) models and substantially pales in comparison to fine-tuning when the model is moderately parameterized (e.g., 220M). Secondly, fine-tuning only adaptive parameters (e.g., prompts/adapters) simply presents an illusion of efficiency [26]. In reality, the FLOPs incurred by the model is still identical on

the forward pass, which saves no compute during inference. Parameter counts, especially when including only prompts and adapters, are not the only measurement of computational efficiency. Instead, the FLOPs and training time should be considered together to provide a holistic view.

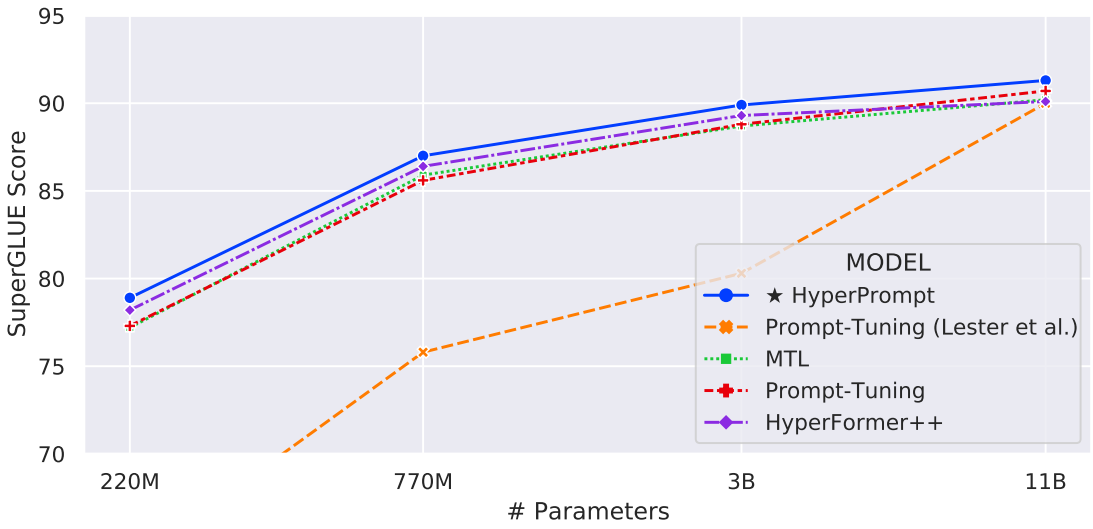


Figure 3.1: HyperPrompt achieves state-of-the-art performance on SuperGLUE for T5 models up to XXL. Prompt-tuning [63] with tuning prompt parameters only achieves competitive performance against multi-task learning (MTL) baseline for the 11B parameter model with a big performance gap for smaller models. HyperPrompt outperforms the strong parameter-efficient adapter variant HyperFormer++ [56], the MTL baseline, and the full fine-tuning of Prompt-Tuning (our implementation) across model sizes with a large margin [e.g. 91.3 vs 90.2 (MTL) for T5 XXL]. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

Our Contributions. Overall, the main contributions include:

- We propose a novel HyperPrompt Transformer architecture with learnable hyper-prompts for multi-task fine-tuning with great parameter and computational efficiency.
- We demonstrate that for difficult tasks, it is crucial to fine-tune the task-specific parameters together with the backbone model to achieve Pareto efficiency on all tasks.

- We explore HyperNetworks as a prompt generator, and inject hyper-prompts into the self-attention module as global task memory tokens.
- HyperPrompt outperforms state-of-the-art parameter-efficient T5 models [91] using Prompt-Tuning or adapters on well-established benchmarks such as SuperGLUE and GLUE, across all explored model sizes (see Figure 3.1).

3.2 Methods

In this section, we introduce HyperPrompt which has three variants: HyperPrompt-Share, HyperPrompt-Sep and HyperPrompt-Global (Figure 3.2). We follow two key design principles to formulate HyperPrompt: (1) injecting task-conditioning into self-attention module for better computational efficiency and more expressive power, and (2) using HyperNetworks to simultaneously improve the parameter efficiency and allow a flexible degree of task sharing for better generalization. The intuition behind the first design principle is based on the fact that self-attention is the key to the power of Transformers via token-level interactions. Injecting task-conditioning into self-attention directly allows the learning of more expressive task-specific representations.

3.2.1 Prompt-Based Task-Conditioned Transformer

Previous adapter-based methods [56, 105] for multi-task learning normally add an adapter (i.e., dense-relu-dense network) for each task after the feed-forward layers at every Transformer block. Instead, the key idea of our approach is to prepend l task-conditioned trainable vectors to the keys and values of the multihead self-attention layer at every Transformer block, where the task-specific attention feature maps are jointly learned with the task-agnostic representation.

The idea of prepending learnable prompts to the network is explored before by Lester et al. [63], Li and Liang [66], Liu et al. [73] for single-task fine-tuning. We first introduce and expand this idea for multi-task learning in this subsection. Specifically, we design a novel method called HyperPrompt following the design principle $\#1$ of injecting hyper-prompts into self-attention and $\#2$ using HyperNetworks as generators for hyper-prompts.

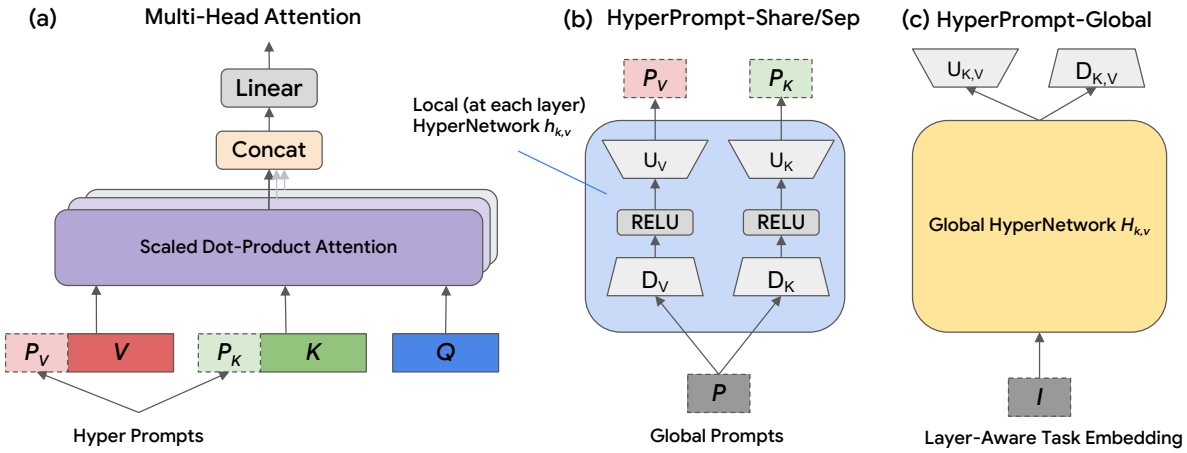


Figure 3.2: HyperPrompt framework: (a) in each Transformer block, task-specific hyper-prompts $P_{K,V}$ are prepended to the original key K and value V for the query Q to attend to, (b) in HyperPrompt-Share/Sep, global prompts P is used to generate the hyper-prompts $P_{K,V}$ through local HyperNetworks $h_{k,v}$ at each Transformer layer, which consists of a down-projection matrix $D_{K,V}$, a RELU layer and a up-project matrix $U_{K,V}$, (c) in HyperPrompt-Global, all the local HyperNetworks are generated by global HyperNetworks $H_{k,v}$ using layer-aware task embeddings I as task-specific inputs (see Section 3.2.3 for details). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

At a multihead self-attention layer, the original key, value and query are calculated as

$$\boldsymbol{K}_\tau = \boldsymbol{X}_\tau \boldsymbol{W}_k,\ \boldsymbol{V}_\tau = \boldsymbol{X}_\tau \boldsymbol{W}_v,\ \boldsymbol{Q}_\tau = \boldsymbol{X}_\tau \boldsymbol{W}_q \tag{3.3}$$

where $\boldsymbol{X}_\tau \in \mathbb{R}^{L\times d}$ is the input sequence of a training sample from the τ-th task, L is the sequence length, d is the model dimension. $\boldsymbol{W}_k \in \mathbb{R}^{d\times h\times d_h}$, $\boldsymbol{W}_v \in \mathbb{R}^{d\times h\times d_h}$ and $\boldsymbol{W}_q \in \mathbb{R}^{d\times h\times d_h}$ project the input into original key $\boldsymbol{K}_\tau \in \mathbb{R}^{L\times h\times d_h}$, value $\boldsymbol{V}_\tau \in \mathbb{R}^{L\times h\times d_h}$ and query $\boldsymbol{Q}_\tau \in \mathbb{R}^{L\times h\times d_h}$, h is the number of heads, d_h is the dimension of each head and typically set to d/h to save parameters.

To learn the task-specific information for the τ-th task, we have l trainable d-dimensional vectors as the hyper-prompts for the key and the value respectively, denoted as $\boldsymbol{P}_{\tau,k} \in \mathbb{R}^{l\times h\times d_h}$ and $\boldsymbol{P}_{\tau,v} \in \mathbb{R}^{l\times h\times d_h}$, as shown in Figure 3.2(a). Then, the hyper-prompts are concatenated with the

original key and value:

$$\boldsymbol{K}'_\tau = \text{concat}(\boldsymbol{P}_{\tau,k},\ \boldsymbol{K}_\tau) \tag{3.4}$$

$$\boldsymbol{V}'_\tau = \text{concat}(\boldsymbol{P}_{\tau,v},\ \boldsymbol{V}_\tau) \tag{3.5}$$

where the new key (value) $\boldsymbol{K}'_\tau$ ($\boldsymbol{V}'_\tau$) $\in \mathbb{R}^{(l+L)\times h\times d_h}$ are used to compute the multihead self-attention.

After that, the multihead self-attention can be operated[2]:

$$\boldsymbol{O}_\tau = \text{Attention}(\boldsymbol{Q}_\tau, \boldsymbol{K}'_\tau, \boldsymbol{V}'_\tau) = \text{softmax}(\boldsymbol{Q}_\tau \boldsymbol{K}'^{\boldsymbol{T}}_\tau)\boldsymbol{V}'_\tau$$

where $\boldsymbol{O}_\tau \in \mathbb{R}^{L\times d}$ is the output of multihead attention.

The hyper-prompts benefit Transformers for multi-task learning in two ways:

- Prompt for key $\boldsymbol{P}_{\tau,k}$ is prepended with the original key and will participate in the calculation of attention feature map: $\text{softmax}(\boldsymbol{Q}_\tau \boldsymbol{K}'^{\boldsymbol{T}}_\tau)$. $\boldsymbol{P}_{\tau,k}$ directly interacts (matrix multiplication) with the original query $\boldsymbol{Q}_\tau$, allowing tokens to acquire task-specific semantics.

- Prompt for value $\boldsymbol{P}_{\tau,v}$ is prepended with the original value and will be absorbed into the self-attention output $\boldsymbol{O}_\tau$, where each position in $\boldsymbol{O}_\tau$ is the weighted-sum of vectors in $\boldsymbol{V}'_\tau$ with weights from the attention scores. This way, $\boldsymbol{P}_{\tau,v}$ can serve as task-specific memories for multihead attention to retrieve information from.

3.2.2 HyperPrompt

How to obtain the prompts for the m-th Transformer block? A straightforward way is to directly initialize $\boldsymbol{P}^m_{\tau,k}$ and $\boldsymbol{P}^m_{\tau,v}$. However, this way is parameter-inefficient, as it scales linearly with both the number of tasks T and the number layers M as $\mathcal{O}(T \times M)$.

Instead, we initialize a global[3] prompt $\boldsymbol{P}_\tau$ for each task and apply local HyperNetworks at every Transformer block to project this prompt into $\{\boldsymbol{P}^m_{\tau,k}\}^M_{m=1}$ and $\{\boldsymbol{P}^m_{\tau,v}\}^M_{m=1}$.

[2]The following equation shows the self-attention on one head. Multihead attention is slightly more involved and we omit it for simplicity.

[3]we term it *global* because it is independent of the layer number as opposed to layer-dependent prompt $\boldsymbol{P}^m_\tau$.

Global Prompts. Specifically, we initialize a set of global prompts $\{\boldsymbol{P}_\tau\}_{\tau=1}^{T}$, where $\boldsymbol{P}_\tau \in \mathbb{R}^{l\times d}$ is a trainable matrix to learn the task-specific information of the τ-th task, d is the model dimension and l is the length of the prompt.

Local HyperNetworks. At the m-th Transformer block, we apply two local HyperNetworks h_k^m and h_v^m to transform the global prompt $\boldsymbol{P}_\tau$ into layer-specific and task-specific prompts as shown in Figure 3.2(b):

$$\boldsymbol{P}_{\tau,k}^m = h_k^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_k^m(\text{Relu}(\boldsymbol{D}_k^m(\boldsymbol{P}_\tau))), \tag{3.6}$$

$$\boldsymbol{P}_{\tau,v}^m = h_v^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_v^m(\text{Relu}(\boldsymbol{D}_v^m(\boldsymbol{P}_\tau))), \tag{3.7}$$

where $\boldsymbol{P}_{\tau,k/v}^m \in \mathbb{R}^{l\times h\times d_h}$. We call these generated prompts *hyper-prompts* to distinguish from global prompts.

In particular, to limit the number of parameters, the local HyperNetworks are designed using a bottleneck architecture: $\boldsymbol{D}_{k/v}^m \in \mathbb{R}^{d\times b}$ and $\boldsymbol{U}_{k/v}^m \in \mathbb{R}^{b\times h\times d_h}$ are down-projection and up-projection matrices, respectively. b is the bottleneck dimension satisfying $b \ll d$.

HyperPrompt-Share. We first have all tasks share the same two local HyperNetworks defined by the down-project matrices $\boldsymbol{D}_k^m$ and $\boldsymbol{D}_v^m$, and the up-project matrices $\boldsymbol{U}_k^m$ and $\boldsymbol{U}_v^m$. We refer to this design choice as HyperPrompt-Share.

Despite the saving of parameters, one drawback of HyperPrompt-Share is that the task conflicts could arise given the limited model capacity [122, 126] of the shared local HyperNetworks.

HyperPrompt-Sep. In the opposite extreme of HyperPrompt-Share, each task can have its own local HyperNetworks $h_{\tau,k}^m(\boldsymbol{P}_\tau)$ and $h_{\tau,v}^m(\boldsymbol{P}_\tau)$ as following:

$$\boldsymbol{P}_{\tau,k}^m = h_{\tau,k}^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_{\tau,k}^m(\text{Relu}(\boldsymbol{D}_{\tau,k}^m(\boldsymbol{P}_\tau))), \tag{3.8}$$

$$\boldsymbol{P}_{\tau,v}^m = h_{\tau,v}^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_{\tau,v}^m(\text{Relu}(\boldsymbol{D}_{\tau,v}^m(\boldsymbol{P}_\tau))), \tag{3.9}$$

where $\boldsymbol{D}_{\tau,k/v}^m$ and $\boldsymbol{U}_{\tau,k/v}^m$ are down-projection and up-projection matrices for the τ task, respectively. In this case, each task hyper-prompt is trained independently and hence there is no informa-

tion sharing.

3.2.3 HyperPrompt-Global

We further propose a novel design of *HyperPrompt-Global* to flexibly share information and knowledge among tasks and layers while maintaining a low parameter cost. As shown in Figure 3.2(c), the key idea of HyperPrompt-Global is to generate the local HyperNetworks using the same shared global HyperNetwork for all tasks.

Layer-Aware Task Embedding. Following the same recipe in Karimi Mahabadi et al. [56], we define a layer-aware task embedding for better generalization. Let $k_\tau \in \mathbb{R}^{t'}$ denote the task embedding for the τ task and t' is the dimension. To capture the layer-specific information, layer embedding $z_m \in \mathbb{R}^{t'}$ is introduced. After that, a task projection network $h_t(\cdot,\cdot)$ is applied to fuse the task embedding and the layer embedding into the final layer-awared task embedding $\boldsymbol{I}_\tau^m = h_t(k_\tau, z_m)$, where $\boldsymbol{I}_\tau^m$ is the input to the shared global HyperNetworks as shown in Figure 3.1(c). h_t is a MLP consisting of two feed-forward layers and a ReLU non-linearity, which takes the concatenation of k_τ and z_m as input.

Global HyperNetworks. $H_k(\cdot)$ generates the weight matrices $(\boldsymbol{U}_{\tau,k}^m, \boldsymbol{D}_{\tau,k}^m)$ in the local HyperNetworks of key hyper-prompts and another global HyperNetwork $H_v(\cdot)$ generates the weight matrices $(\boldsymbol{U}_{\tau,v}^m, \boldsymbol{D}_{\tau,v}^m)$ in the local HyperNetworks of value hyper-prompts:

$$(\boldsymbol{U}_{\tau,k}^m, \boldsymbol{D}_{\tau,k}^m) = H_k(\boldsymbol{I}_\tau^m) = (\boldsymbol{W}^{U_k}, \boldsymbol{W}^{D_k})\boldsymbol{I}_\tau^m, \tag{3.10}$$

$$(\boldsymbol{U}_{\tau,v}^m, \boldsymbol{D}_{\tau,v}^m) = H_v(\boldsymbol{I}_\tau^m) = (\boldsymbol{W}^{U_v}, \boldsymbol{W}^{D_v})\boldsymbol{I}_\tau^m, \tag{3.11}$$

where $\boldsymbol{I}_\tau^m \in \mathbb{R}^t$ is the layer-aware task embedding for the τ task at the m-th block. $\boldsymbol{W}^{D_k} \in \mathbb{R}^{(d\times b)\times t}$, $\boldsymbol{W}^{D_v} \in \mathbb{R}^{(d\times b)\times t}$, $\boldsymbol{W}^{U_k} \in \mathbb{R}^{(b\times h\times d_h)\times t}$ and $\boldsymbol{W}^{U_v} \in \mathbb{R}^{(b\times h\times d_h)\times t}$ are the weight matrices of $H_k(\cdot)$ and $H_v(\cdot)$.

Given that $\boldsymbol{U}_{\tau,k/v}^m$, and $\boldsymbol{D}_{\tau,k/v}^m$ are generated by the global HyperNetworks, we project the global prompts $\boldsymbol{P}_{\tau,k/v}$ into hyper-promtps $\boldsymbol{P}_{\tau,k/v}^m$ following Eqs. 3.8 and 3.9. Finally, the hyper-prompts $\boldsymbol{P}_{\tau,k/v}^m$ are prepended with original key and value at every self-attention layer as shown in Figure

3.1(a) to calculate the task-conditioned attention scores.

Using global HyperNetworks to generate the local HyperNetworks has three benefits:

1. Unlike HyperPrompt-Share where all tasks use the same local HyperNetworks or HyperPrompt-Sep where each task uses different local HyperNetworks, it enables a more flexible way to share information and knowledge across tasks. As shown in Equation 3.10 and 3.11, the weight matrices are decomposed into $H_{k/v}(\cdot)$ that are shared by all tasks and $\boldsymbol{I}_\tau^m$ that represents the specific task. Therefore, the model can adjust the degree of information sharing across tasks through learning the appropriate parameter values in $H_{k/v}(\cdot)$ and $\boldsymbol{I}_\tau^m$ during the end-to-end training.

2. The global HyperNetworks are shared by all layers of Transformer models, where the information and knowledge can be transferred across the different layers.

3. A parameter-efficient task conditioned parameterization is enabled. The number of extra task-conditioned parameters doesn't depend on the number of layers M, and scales sub-linearly with respect to the total number of tasks T. In practice, since task embeddings and task prompts have far fewer parameters than the global HyperNetworks, the additional task-conditioned parameters is almost independent of T.

3.2.4 Parameter Efficiency of HyperPrompt-Global

Since the encoder and the decoder of Transformers have approximately the same capacity, the calculation considers only the decoder-side for simplicity. First, we have global task prompts $\boldsymbol{P}_\tau \in \mathbb{R}^{l\times d}$ for the τ-th task, which contains dlT parameters for T tasks. The global HyperNetworks contain four weight matrices $\boldsymbol{W}^{D_k} \in \mathbb{R}^{(d\times b)\times t}$, $\boldsymbol{W}^{D_v} \in \mathbb{R}^{(d\times b)\times t}$, $\boldsymbol{W}^{U_k} \in \mathbb{R}^{(b\times h\times d_h)\times t}$ and $\boldsymbol{W}^{U_v} \in \mathbb{R}^{(b\times h\times d_h)\times t}$, which result in $4(bdt)$ parameters (we let $d = h \times d_h$). To obtain layer-aware task embedding, HyperPrompt-Global learns task embedding $k_\tau \in \mathbb{R}^{t'}$ for the τ task and layer embedding $z_m \in \mathbb{R}^{t'}$ for the m-th Transformer block, which in total results in $Tt' + Mt'$ parameters. Besides, a task projection network h_t is applied to fuse the task embedding and the layer

embedding into the final layer-aware task embedding $\boldsymbol{I}_\tau^m \in \mathbb{R}^t$. h_t is a two-layer feed-forward networks and contains $(2t' + t)e$ parameters, where e is the hidden dimension for h_t.

To sum up, the total number of additional parameters from HyperPrompt-Global is $dlT + 4(bdt) + Tt' + Mt' + (2t' + t)e$, where d is the model dimension, l is the length of the prompts, T is the total number of tasks, b is the bottleneck dimension of the weight matrices of the local HyperNetworks, d is the model dimension, t'/t is the dimension of the raw/final layer-aware task embedding, and e is the hidden dimension of $h_{k/v}$. Therefore, the space complexity is $\mathcal{O}(d(lT + 4bt))$, given that in practice $M \sim T$, $t' \ll dl$, and $e \ll bd$. This leads to a sub-linear scaling with respect to T.

Furthermore, T is typical $\sim \mathcal{O}(10)$ for multi-task learning. A reasonable $l \sim \mathcal{O}(10)$ is required to achieve the optimal performance, which will be detailed in Section 3.3.8. On the other hand, typical values for $b \sim 24$ and $t \geq 32$, and therefore $4bt \gg lT$ is satisfied in most cases. Hence, the space complexity could be further simplified as $\mathcal{O}(bdt)$. In conclusion, the space complexity of HyperPrompt-Global mainly comes from the global HyperNetworks and is practically independent of the prompt length l, the number of Transformer layers M, and the number of tasks T.

3.3 Experiments

3.3.1 Experimental Setup

Datasets. We evaluate the performance of the models on GLUE [116] and SuperGLUE [117] respectively. Each of them is a collection of text classification tasks to test the general language understanding ability. Specifically, the tasks include: sentence acceptability (CoLA), sentiment analysis (SST-2), paraphrasing/sentence similarity (MRPC, STS-B and QQP), natural language inference (MNLI, QNLI, RTE and CB), coreference resolution (WSC), sentence completion (COPA), word sense disambiguation (WIC) and question answering (MultiRC and ReCoRD, BoolQ).

Transformers. Following previous work Karimi Mahabadi et al. [56] and Tay et al. [105], our models are built on top of the state-of-the-art Transformer model T5 [91], which uses encoder-decoder architecture from Vaswani et al. [114]. We use already pre-trained T5 with sizes from

Base (220M parameters) to XXL (11B).

Evaluation. We save a checkpoint every 2000 steps for all models and follow the same convention as Raffel et al. [91] in selecting the best checkpoint for each task. The emphasis of our evaluation is not to find the best single checkpoint for all tasks but to test the model's ability of knowledge sharing among the co-trained tasks. We first calculate the average of all metrics for each task and then report the average of all tasks for GLUE and SuperGLUE.

Baselines. We compare our proposed HyperPrompt with vanilla T5 models [91] for multi-task learning, which is referred to *MTL*. Another baseline is *Vanilla Adapter* proposed in Houlsby et al. [49] that add adapters modules for each task after each of the the two feed-forward modules in each Transformer block of the T5 model. The state-of-the-art adapter-based method for multi-task learning is *HyperFormer++* proposed in Karimi Mahabadi et al. [56] that use HyperNetworks to generate adapters for each task and add them after the feed-forward modules following Houlsby et al. [49]. In addition, *Prompt-Tuning* [63] is originally for parameter-efficient single-task fine-tuning and only prepends prompts to the input word embeddings in the first layer. We slightly modify it by initializing and prepending prompts for each task respectively so that *Prompt-Tuning* can be applied to multi-task learning.

3.3.2 Experimental Details

Our models were implemented using Mesh Tensorflow[4] [102] with the T5 library[5] [91]. Following Raffel et al. [91], all data are preprocessed as into a "sequence-to-sequence" format. The length of the sequence is 512 at the encoder and 32 at the decoder. For all experiments, we train models 300K steps with a batch size of 128 and each batch is a mixture which samples each task proportionately to the number of examples in the dataset. Learning rate is a constant of 1e-3 with Adafactor optimizer [101].

For hyper-parameters tuning, the length of prompt l is selected from $\{12, 16, 20, 20, 24\}$ at the encoder and $\{2, 4, 6, 8, 10, 12, 14, 16\}$ at the decoder. The bottleneck dimension b in the weight

[4] https://github.com/tensorflow/mesh

[5] https://github.com/google-research/text-to-text-transfer-Transformer

matrices of local HyperNetworks is set to d/r, where d is the model dimension of the T5 models and r is a reduction factor and selected from $\{16, 32, 64\}$. The dimension t of the layer-aware task embedding is selected from $\{32, 64, 128\}$. For a fair comparison, the hyper-parameters of baseline methods are set to have approximately the same numbers of parameters as HyperPrompt with the exception that Prompt-Tuning and HyperPrompt-Share are extremely parameter-efficient with significantly fewer parameters.

3.3.3 Key Results

Figure 3.1 provides an overall summary of the results of HyperPrompt-Global. Previous prompt-tuning [63, 66] methods focus on parameter-efficient single-task fine-tuning and hence freeze the backbone Transformers and only fine-tune the prompts. Their experiments show that the performance of only tuning the prompts can match the full model training with a very large 11B model (Figure 3.1), but substantially pales for moderate model sizes.

Our HyperPrompt-Global architecture when fully fine-tuned achieves state-of-the-art performance on SuperGLUE across four different model sizes. Competitive adapter-tuning variants including Prompt-Tuning and HyperFormer++ can either match or slightly improve upon the multi-task learning (MTL) baseline on the SuperGLUE dataset. In contrast, HyperPrompt-Global outperforms the strong MTL baseline by a large margin on SuperGLUE score (78.9 vs 77.2 for T5 Base). Interestingly, such a performance gain continues all the way to model size as big as XXL (e.g. 91.3 vs 90.2) with only 0.14% additional parameters.

3.3.4 Tuning all vs Task-Conditioned Parameters

Recently, Karimi Mahabadi et al. [56] show that only tuning adapters can be competitive against the full fine-tuning. However, the evaluation is conducted only on the GLUE with smaller models including T5 Small and Base.

In the experiments, we first compare tuning the full model (the backbone model plus task-conditioned parameters) vs. only task-conditioned parameters. Table 3.1 and 3.2 shows the comparison on the GLUE and SuperGLUE average scores using T5 large. For GLUE, the observation

Tunable Parameters	Model	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
All	MTL	59.4	96.6	93.3/90.7	90.6/90.4	89.8/92.3	90.8/90.8	95.2	90.8	88.3
All	HyperFormer++-T5.1.1$_{\text{LARGE}}$	63.3	96.6	93.2/90.7	92.1/91.9	89.7/92.3	90.5/90.7	95.1	89.9	88.8
All	HyperPrompt-Global-T5.1.1$_{\text{LARGE}}$	64.6	96.7	94.0/91.8	91.3/91.4	90.0/92.4	90.8/91.0	95.4	91.9	89.4
Task-Specific	HyperFormer++-T5.1.1$_{\text{LARGE}}$	58.9	95.7	92.7/90.0	91.6/91.5	87.7/90.7	89.8/90.0	94.5	87.0	87.3
Task-Specific	HyperPrompt-Global-T5.1.1$_{\text{LARGE}}$	57.5	96.7	93.6/91.2	91.9/92.0	87.0/90.1	90.3/90.6	95.0	87.7	87.5

Table 3.1: Comparison of fine-tuning all vs task-specific parameters on GLUE. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

Tunable Parameters	Model	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AVG
All	MTL	88.5	95.8/98.2	87.0	85.5/56.3	89.2/88.6	91.7	74.0	89.4	85.9
All	HyperFormer++-T5.1.1$_{\text{LARGE}}$	88.9	98.7/98.2	86.7	85.4/56.7	89.4/88.8	92.1	74.5	90.7	86.4
All	HyperPrompt-Global-T5.1.1$_{\text{LARGE}}$	88.7	99.1/98.8	91.0	85.0/55.6	89.8/89.1	91.3	74.2	92.0	87.0
Task-Specific	HyperFormer++-T5.1.1$_{\text{LARGE}}$	85.2	90.9/94.6	76.7	81.5/48.8	87.2/86.4	87.7	67.8	82.1	80.5
Task-Specific	HyperPrompt-Global-T5.1.1$_{\text{LARGE}}$	85.2	95.2/95.5	75.5	82.9/52.9	89.1/88.3	85.7	71.1	82.2	81.5

Table 3.2: Comparison of fine-tuning all vs task-specific parameters on SuperGLUE. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

is consistent with [56], where task-specific only fine-tuning of HyperFormer++ and HyperPrompt-Global is comparable to the MTL baseline. However, on SuperGLUE, we observe a large performance drop: the average score drops by 5.5 and 5.9 for HyperPrompt-Global and HyperFormer++, respectively.

Therefore, these experiments show that only tuning the task-conditioned parameters is not enough to achieve competitive results as full model training for multi-task learning on high-difficulty tasks such as SuperGLUE. This is consistent with the results of Prompt-Tuning [63]. Hence, the rest of the experiments are conducted with tuning all model parameters.

Model	# Ops	Training Time
Vanilla Adapter	1.01×10^{13}	8.4h
HyperFormer++	3.14×10^{13}	10.3h
Prompt-Tuning	1.16×10^{13}	11.1h
HyperPrompt-Sep	1.01×10^{13}	8.9h
HyperPrompt-Share	$\mathbf{9.8 \times 10^{12}}$	**8.0h**
HyperPrompt-Global	$\mathbf{9.8 \times 10^{12}}$	8.7h

Table 3.3: The number of operations for a single forward pass and training time (base model). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

3.3.5 Computational Efficiency

Table 3.3 presents the computational efficiency analysis of the Adapter/Prompt models. HyperPrompt-Share has the lowest # Ops since hyper-prompts are injected into self-attention and skip the standard FFN layers. In contrast, HyperFormer++ has $\sim$ 3x # Ops compared to other variants. Regarding training time, HyperPrompt-Share is fastest given that the local HyperNetworks are shared across tasks. Vanilla Adapter and HyperPrompt-Global are comparable while HyperFormer++ and Prompt-Tuning take significant longer to do the full fine-tuning. This shows the computational efficiency of HyperPrompt for both training and inference, in addition to the parameter efficiency.

3.3.6 Ablation Study

Table 3.4 and 3.5 presents the results on T5 Base and Table 3.6 and 3.7 presents the results on T5 Large. HyperPrompt outperforms all baselines in terms of the average score of GLUE and SuperGLUE.

HyperPrompt-Global vs. Prompt-Tuning. The original Prompt-Tuning [63] is for single-task fine-tuning. To be parameter-efficient, it only trains the prompts but freezes the backbone T5. To make a fair comparison, we modify Prompt-Tuning by (1) training both prompts and backbone, and (2) adding prompt to each task and co-train all tasks together. As shown in Table 3.4, 3.5, 3.6

and 3.7, HyperPrompt-Global outperforms Prompt-Tuning by 2.0 (0.6) and 1.6 (1.4) on GLUE and SuperGLUE using T5 Base (Large), respectively. HyperPrompt-Global improves upon Prompt-Tuning in two places: (1) Prompt-Tuning only adds prompts to the word embedding layer while HyperPrompt adds hyper-prompts at every Transformer layer and hence is more expressive; and (2) Prompts of tasks are trained independently in Prompt-Tuning while HyperPrompt enables a flexible knowledge sharing via the HyperNetworks.

Model	#Params	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
MTL	1.0x	49.8	94.6	92.5/89.8	90.7/90.5	89.2/91.9	**88.8/88.5**	93.3	85.0	85.5
Vanilla Adapter	1.06x	**60.0**	**95.4**	92.7/89.8	90.2/90.2	**89.3**/91.9	88.5/88.1	**93.5**	84.4	86.7
HyperFormer++	1.04x	56.9	94.8	92.9/90.1	**91.1/90.9**	88.9/91.7	88.7/88.3	93.4	85.6	86.5
Prompt-Tuning	1.0003x	48.0	95.0	92.2/89.0	90.3/90.2	89.0/91.7	**88.8/88.5**	93.2	82.9	84.8
HyperPrompt-Share (ours)	1.008x	56.2	94.7	93.0/90.4	90.6/90.4	89.2/91.9	88.7/88.4	93.4	85.2	86.4
HyperPrompt-Sep (ours)	1.06x	57.2	94.6	**93.8/91.4**	91.0/90.8	89.2/91.9	88.5/88.4	93.4	86.6	**86.8**
HyperPrompt-Global (ours)	1.04x	57.0	95.2	93.4/90.9	90.4/90.2	89.2/**92.0**	88.7/**88.5**	93.4	**87.1**	**86.8**

Table 3.4: Comparison of HyperPrompt with baselines on GLUE using T5 Base. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

Model	#Params	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WIC	WSC	AVG
MTL	1.0x	82.6	93.4/93.5	65.7	76.7/39.7	80.9/80.2	85.6	70.5	81.4	77.2
Vanilla Adapter	1.03x	**83.5**	93.4/94.6	65.3	77.6/**42.7**	81.0/80.2	**88.2**	71.0	76.9	77.5
HyperFormer++	1.02x	**83.5**	96.2/97.0	66.3	**77.8**/41.9	81.2/80.4	87.4	71.0	80.1	78.2
Prompt-Tuning	1.0003x	82.5	94.0/95.8	68.0	76.9/40.2	80.9/80.2	84.1	69.3	80.8	77.3
HyperPrompt-Share (ours)	1.004x	83.1	95.7/95.2	67.7	77.3/41.3	**81.9**/81.0	87.4	70.4	80.8	78.2
HyperPrompt-Sep (ours)	1.03x	83.3	**97.8/97.0**	61.7	77.6/42.3	81.5/80.6	86.8	**71.4**	78.2	77.5
HyperPrompt-Global (ours)	1.02x	83.3	96.6/96.4	**69.7**	77.5/41.0	**81.7/80.9**	86.8	70.5	**83.7**	**78.9**

Table 3.5: Comparison of HyperPrompt with baselines on SuperGLUE using T5 Base. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

Model	#Params	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
MTL	1.0x	59.4	96.6	93.3/90.7	90.6/90.4	89.8/92.3	90.8/90.8	95.2	90.8	88.3
Vanilla Adapter	1.06x	63.8	96.5	93.7/91.3	92.0/**91.9**	**90.0/92.5**	90.6/90.5	94.9	88.7	88.8
HyperFormer++	1.02x	63.3	96.6	93.2/90.7	**92.1/91.9**	89.7/92.3	90.5/90.7	95.1	89.9	88.8
Prompt-Tuning	1.0001x	62.5	**96.7**	93.4/91.0	91.3/91.0	**90.0**/92.4	**90.9**/91.0	**95.4**	89.9	88.8
HyperPrompt-Share (ours)	1.008x	**65.0**	**96.7**	93.8/91.6	91.1/90.8	**90.0**/92.4	90.8/**91.1**	95.3	91.3	89.3
HyperPrompt-Sep (ours)	1.06x	63.9	96.6	**94.6/92.6**	92.0/91.7	**90.0**/92.4	**90.9**/91.0	95.2	91.6	**89.4**
HyperPrompt-Global (ours)	1.02x	64.6	**96.7**	94.0/91.8	91.3/91.4	**90.0**/92.4	90.8/91.0	**95.4**	**91.9**	**89.4**

Table 3.6: Comparison of HyperPrompt with baselines on GLUE using T5 Large. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

Model	#Params	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WIC	WSC	AVG
MTL	1.0x	88.5	95.8/98.2	87.0	**85.5**/56.3	89.2/88.6	91.7	74.0	89.4	85.9
Vanilla Adapter	1.03x	88.8	98.3/**98.8**	86.0	85.3/56.0	89.3/88.7	91.2	73.6	91.3	86.1
HyperFormer++	1.01x	**88.9**	98.7/98.2	86.7	85.4/**56.7**	89.4/88.8	**92.1**	**74.5**	90.7	86.4
Prompt-Tuning	1.0001x	88.5	97.6/**98.8**	85.0	84.9/55.2	89.0/88.4	91.5	72.8	90.1	85.6
HyperPrompt-Share (ours)	1.004x	88.5	98.7/98.2	88.0	85.2/55.8	89.7/89.1	91.8	74.1	**93.9**	86.8
HyperPrompt-Sep (ours)	1.03x	88.6	97.6/**98.8**	87.7	85.2/56.4	89.7/89.1	91.6	73.5	89.4	86.1
HyperPrompt-Global (ours)	1.01x	88.7	**99.1/98.8**	**91.0**	85.0/55.6	**89.8/89.1**	91.3	74.2	92.0	**87.0**

Table 3.7: Comparison of HyperPrompt with baselines on SuperGLUE using T5 Large. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

HyperPrompt-Global vs. HyperFormer++. HyperPrompt-Global is superior to the state-of-the-art baseline HyperFormer++ in the average score of GLUE and SuperGLUE for both Base and Large T5 model. For example, HyperPrompt-Global of T5 large achieves 87.0 on the SuperGLUE compared to 86.4 by HyperFormer++ (Table 3.7). Note that the main difference between the two methods is that HyperPrompt-Global inserts the task-conditioned parameters as prompts into self-attention layers while HyperFormer++ insert adapters after each block. We believe task-conditioning in self-attention gives more expressive power than in the feed-forward network as

done in adapters. Hyper-prompts that are prepended with the key and value participate in the attention interactions between different token positions, which helps the model to better capture the task-dependent semantics.

HyperPrompt-Global vs. MTL. Next, we observe that using HyperPrompt-Global can greatly improve the performance upon the vanilla Transformer model (referred to MTL): 1.7 (1.1) gain on SuperGLUE score for T5 Base (Large) with 4% (2%) additional parameters. In conclusion, the experiments show that HyperPrompt-Global is a parameter-efficient and effective task-conditioned parameterization of Transformers for multi-task learning.

In addition, taking Table 3.5 as an example, we observe that COPA (400 training samples) and WSC (554 training samples) as the two small datasets are improved the most (65.7 to 69.7 and 81.4 to 83.7 respectively). Hence, the experiments show that HyperPrompt-Global can enhance the knowledge transfer from high-resource tasks to low-resource tasks, alleviating the task inference problem.

HyperPrompt-Global vs. HyperPrompt-Share/Sep. Interestingly, HyperPrompt-Share is better than HyperPrompt-Sep on the SuperGLUE on both Base and Large models while the opposite is true for GLUE. Notice that all tasks share the same two local HyperNetworks in HyperPrompt-Share while each task has its own local HyperNetworks in HyperPrompt-Sep. More importantly, we observe that HyperPrompt-Global, where the local HyperNetworks are generated by the global HyperNetworks, always achieves the best performance on both GLUE and SuperGLUE. Hence, the experiments show that HyperPrompt-Global can adjust the degree of knowledge sharing for better multi-task generalization, compared to HyperPrompt-Share/Sep in the extremes.

3.3.7 Peeking into Hyper-Prompts

To shed light on how hyper-prompts help improve the multi-task generalization via the task-conditioning, we peek into HyperPrompt-Global models by looking at the distribution of attention scores. We choose the GLUE task MRPC as an example. Specifically, to calculate the attention mass over hyper-prompts per layer as visualized in Figure 3.3 (top), we averaged the hyper-prompt attention softmax scores across 100 validation examples and each attention head in a layer, and

summed across each query attending to the hyper-prompts. In other words, we aggregated the amount of attention given to hyper-prompts by queries.

To calculate the attention entropy over tokens (other than hyper-prompts) as visualized in Figure 3.3 (bottom), we calculated the entropy of the attention distributions (averaged across attention heads) for 100 validation examples. This results in $\sum_{n=1}^{100} \sum_{L=1}^{12} |X_n|$ entropies calculated. For the HyperPrompt-Global model, this involved re-normalizing the softmax distribution after removing hyper-prompts, as we wanted to understand how the original tokens are attended to.

First, we compute the attention mass on hyper-prompts for each encoder layer. Figure 3.3 (top) shows that the network has lower attention mass on hyper-prompts in the lower layers and gradually increases attention mass for higher layers. This phenomenon indicates that higher-levels of Transformer becomes more task-specialized while it is beneficial for the lower-levels to learn task-agnostic representation [134] by casting lower attention mass on hyper-prompts. Furthermore, we calculate the entropy of the attention scores on the tokens. For HyperPrompt-Global, we remove the hyper-prompts from the calculation and re-normalize the attention scores on the tokens to make a fair comparison with the MTL baseline. Figure 3.3 (bottom) shows a shift of entropy distribution towards higher values for HyperPrompt-Global. This signifies that injecting hyper-prompts encourages a more diverse attention distribution, which seems to be beneficial to model generalization.

3.3.8 Impact of Hyper-Prompt Length

HyperPrompt-Global prepends l trainable hyper-prompts to the keys and values of self-attention layer at every Transformer layer. In Figure 3.4, we present the results of tuning the prompt length l on GLUE using T5 Base as the example (similar patterns are observed on T5 Large and SuperGLUE). We first add hyper-prompts on the decoder and search the best l and then search the best l for the encoder with the fixed best decoder hyper-prompt length. As shown in Figure 3.4a, $l = 6$ is the best for the decoder. As shown in Figure 3.4b, HyperPrompt achieves the best result of 86.8 when $l = 16$ on the encoder with $l = 6$ fixed for the decoder. The experiments show that hyper-prompts with length $l \sim \mathcal{O}(10)$ are good enough to achieve superior performance. Note that the

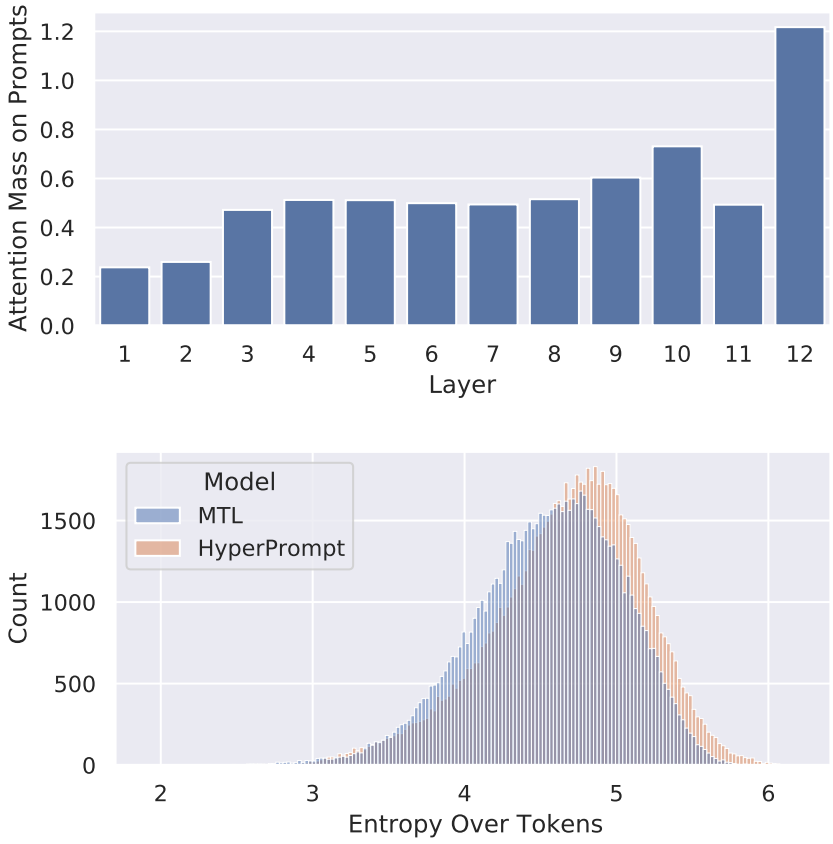
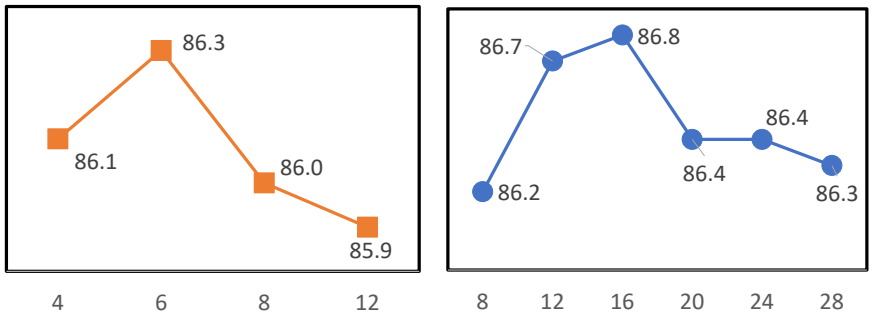


Figure 3.3: Visualization of attention mass and entropy distribution. Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

original sequence length is 512 on the encoder and 32 on the decoder. Therefore, HyperPrompt does not substantially increase the time complexity of self-attention layers in practice.

3.3.9 Encoder vs Decoder

To understand the effect of adding task-conditioned parameters to different parts of the network, we present the results of HyperPrompt-Global and HyperFormer++ with adding hyper-prompts/adapters to: (1) encoder-only, (2) decoder-only, and (3) both encoder-decoder. As shown in Table 3.8, we observe adding task-conditioned parameters to encoder (encoder-only) performs better than decoder-only on GLUE. However, the opposite is true for SuperGLUE, where encoder-



(a) Decoder (b) Encoder

Figure 3.4: Impact of hyper-prompt length in HyperPrompt-Global (GLUE score on T5 Base). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

only is substantially worse than decoder-only. This potentially could be a trainability issue when prompts are inserted into encoders, i.e. a different learning rate might be required to learn the prompt parameters from scratch. We leave this investigation as a future work. Based on this experiment, we add task-conditioned parameters to the decoder for SuperGLUE in our experiments.

Model	#Params	GLUE	SuperGLUE
MTL	1.0x	85.5	77.2
HyperFormer++-Encoder	1.02x	85.9	74.4
HyperFormer++-Decoder	1.02x	85.7	78.2
HyperFormer++-Enc-Dec	1.04x	86.5	74.8
HyperPrompt-Global-Encoder	1.02x	86.6	76.5
HyperPrompt-Global-Decoder	1.02x	86.3	**78.9**
HyperPrompt-Global-Enc-Dec	1.04x	**86.8**	78.7

Table 3.8: Ablation of inserting hyper-prompts or adapters into Encoder/Decoder/Enc-Dec (base model). Reprinted with permission from "HyperPrompt: Prompt-based Task-Conditioning of Transformers." by Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng and Ed H. Chi, 2022. ArXiv preprint arXiv:2203.00759 (2022).

3.4 Related Work

Prompt-Tuning. Prompt tuning is becoming a new paradigm for adapting pre-trained general-purpose language models to downstream tasks, as a lightweight alternative to the popular fine-tuning approach. Here, we use the term Prompt-Tuning to cover a general family of methods following the prompting idea in GPT-3 [16]. To avoid manually design the prompts, recent efforts have focused on search for discrete prommpting words automatically [103]. On the other hand, soft prompts [42, 63, 66, 73] in the form of continuous vectors are introduced to simplify the process and have shown competitive results in both natural language understanding [63, 73] and generation tasks [66]. In particular, Lester et al. [63] show that soft prompts can become competitive against full fine-tuning for a 11B parameters model, but with a big performance gap when the model size is moderate. In our work, we close this gap in the full fine-tuning setting and demonstrated that HyperPrompt can outperform strong multi-task baselines across all model sizes studied.

Adapter-Tuning. Adapter tuning [48, 49, 56] is an alternative approach for parameter-efficient lightweight tuning of pre-trained langauge models for downstream tasks. Task-specific adapter layers [48] are inserted into the Transformer block for fine-tuning while the rest of the backbone model is frozen. By adding only a few percent of additional parameters, Karimi Mahabadi et al. [56] show that competitive performance can be obtained on NLU benchmarks such as GLUE [116]. However, one limitation from the existing work is the evaluation of NLU on GLUE dataset, which is known to be no longer suitable for measuring the progress of language understanding [117]. In our work, we evaluate HyperPrompt on SuperGLUE in addition to GLUE dataset, and show that indeed higher-difficulty tasks such as SuperGLUE requires full-tuning of the model beyond adapter tuning, to be competitive against state-of-the-art multi-task baselines. We also demonstrate that it is advantageous to inject prompts into self-attention than adding adapters.

Multi-task Natural Language Understanding. Multi-task learning is an important and challenge research direction in both full fine-tuning and prompt-tuning paradigms because of the competing needs of training and serving a single model while achieving Pareto efficiency in all tasks.

The T5 model [91] renders all NLP tasks as a Text-to-Text problem. However, the best results

are obtained by task-specific fine-tuning. MTDNN (multi-task deep neural network) [74] shares parameters between several NLP tasks, and achieves strong performance on the GLUE benchmark. Aghajanyan et al. [4] use around 50 tasks to boost the multi-task learning performance. Aribandi et al. [8] builds an extremely diverse set of 107 NLP tasks for extreme multi-task scaling and demonstrate superior performances on a wide range of benchmarks. Recently, Sanh et al. [98], Wei et al. [125] also illustrated how a multi-task learning stage can greatly improve the zero-shot prompting performance of large language models.

3.5 Summary

We propose a novel architecture for prompt-based task-conditioning of self-attention in Transformers. The hyper-prompts are generated by a HyperNetwork to enable flexible information and knowledge sharing among tasks while remain efficient in parameters and computation. HyperPrompt allows the network to learn task-specific feature maps where the hyper-prompts serve as task global memories, encouraging a more diverse distribution of attention. Extensive experiments show that HyperPrompt can achieve superior performances over strong T5 multi-task learning baselines and parameter-efficient models including Prompt-Tuning and HyperFormer++ on GLUE and SuperGLUE benchmarks. For example, HyperPrompt outperforms Prompt-Tuning by 2.0 (0.6) and 1.6 (1.4) on GLUE and SuperGLUE using T5 Base (Large), respectively.

4. METABALANCE: IMPROVING MULTI-TASK RECOMMENDATIONS VIA ADAPTING GRADIENT MAGNITUDES OF AUXILIARY TASKS[1]

The third challenge of intelligent knowledge transfer that this dissertation identifies is the negative transfer – that is, transferring knowledge from auxiliary tasks might have a negative impact on the target task. This chapter aims to alleviate the negative transfer for personalized recommendations from the perspective of the imbalance between the target task and auxiliary tasks. On the one hand, one or more auxiliary tasks might have a larger influence than the target task and even dominate the network weights, resulting in worse recommendation accuracy for the target task. On the other hand, the influence of one or more auxiliary tasks might be too weak to assist the target task. More challenging is that this imbalance dynamically changes throughout the training process and varies across the parts of the same network. We propose a new method: MetaBalance to balance auxiliary losses via directly manipulating their gradients w.r.t the shared parameters in the multi-task network. Specifically, in each training iteration and adaptively for each part of the network, the gradient of an auxiliary loss is carefully reduced or enlarged to have a closer magnitude to the gradient of the target loss, preventing auxiliary tasks from being so strong that dominate the target task or too weak to help the target task. Moreover, the proximity between the gradient magnitudes can be flexibly adjusted to adapt MetaBalance to different scenarios. The code of our approach can be found at here.[2]

4.1 Introduction

The accuracy of personalized recommendations can often be improved by transferring knowledge from related auxiliary information. For example, a primary task on e-commerce platforms like Amazon and eBay is to predict if a user will purchase an item. This purchase prediction task can benefit from transferring knowledge about the user's preference from auxiliary information

[1]This chapter is reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, 2022. Proceedings of the ACM Web Conference 2022. Copyright 2022 held by the authors.

[2]`https://github.com/facebookresearch/MetaBalance`

like which item URLs the user has clicked and which items the user has put into the shopping cart. A common way to enable such transfer learning is to formulate this auxiliary information as auxiliary tasks (e.g., predict if a user will click a URL) and optimize them jointly with the target task (e.g., purchase prediction) on a multi-task network. In this way, knowledge can be transferred from the auxiliary tasks to the target task via the shared bottom layer of the multi-task network as shown in Figure 4.1. Enhanced with auxiliary information, the target task can obtain better performance than training the target task in isolation. Since the motivation of introducing those auxiliary tasks is often to purely assist the target task, in this chapter, we focus on scenarios where only the performance of the target task is of interest.

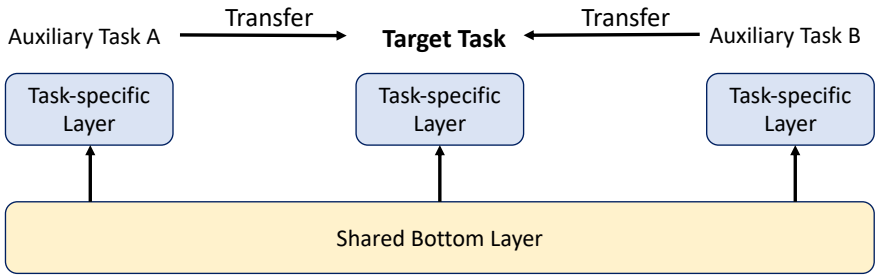


Figure 4.1: Transfer learning from auxiliary tasks to improve the target task on a multi-task network. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

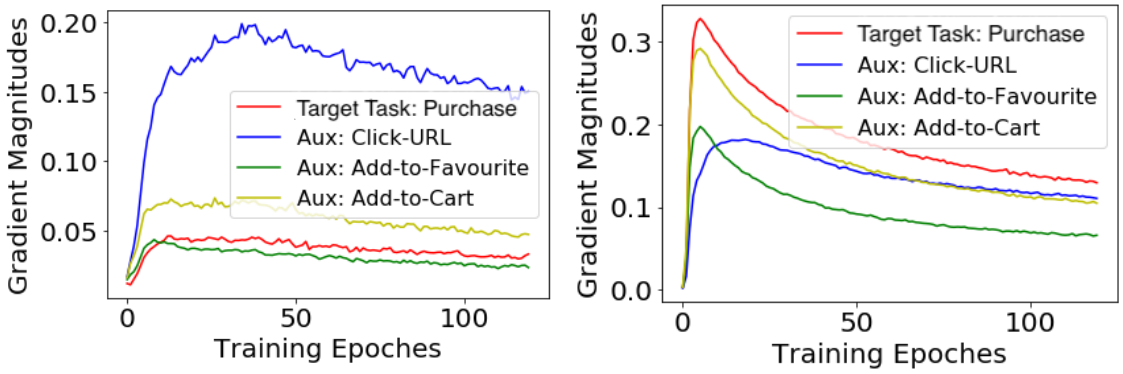
Beyond purchase prediction, many other recommendation scenarios [10, 19, 38, 45, 76, 79, 81, 120, 137] can also benefit from such transfer learning from auxiliary tasks. In social recommendation [38, 79, 120], knowledge can be transferred from the social network to improve personalized recommendations via training the target task simultaneously with auxiliary tasks like predicting the connections or trust among users. To better estimate post-click conversion rate (CVR) in online advertising, related information like post-view click-through rate (CTR) and post-view click-through & conversion rate (CTCVR) can be introduced as auxiliary tasks [81]. Another example is that learning user and item embeddings from product review text can be designed as auxiliary tasks to

improve the target goal of predicting ratings on e-commerce platforms [137].

However, a key challenge to knowledge transfer from auxiliary tasks in personalized recommendation is the potential for a significant *imbalance of gradient magnitudes*, which can negatively affect the performance of the target task. As mentioned before, such knowledge transfer is often conducted on a multi-task network, which is commonly composed of a bottom layer with shared parameters and several task-specific layers. In training, each task has a corresponding loss and each loss has a corresponding gradient with respect to the shared parameters of that multi-task network. The sum of these gradients (for the target task and the auxiliary tasks) impacts how the shared parameters are updated. Hence, the larger the gradient is, the greater the impact this gradient has on the shared parameters. As a result, if the gradient of an auxiliary loss is much larger than the gradient of the target loss, the shared parameters will be most impacted by this auxiliary task rather than the target task. Consequently, the target task could be swamped by the auxiliary tasks, resulting in worse performance. On the other hand, if an auxiliary gradient is much smaller than the target gradient, the influence of this auxiliary task might be too weak to assist the target task. This *imbalance of gradient magnitudes* is common in industrial recommender systems: Figure 4.2a and 4.2b highlight two examples from Alibaba, which respectively demonstrate how the target task gradient can be dominated by an auxiliary task, and how some auxiliary tasks have gradients so small that they may only weakly inform the target task.

So how can we overcome this gradient imbalance? A simple and often used approach is to tune the weights of task losses (or gradients) through a grid or random search. However, such fixed task weights are not optimal because the gradient magnitudes change dynamically throughout the training and the imbalance might vary across the different subsets of the shared parameters as shown in Figure 4.2. Besides, it is time-consuming to tune the weights for multiple auxiliary tasks.

In this chapter, we propose MetaBalance as a novel algorithm and flexible framework that adapts auxiliary tasks to better assist the target task from the perspective of gradient magnitudes. Specifically, MetaBalance has three strategies: (A) Strengthening the dominance of the target task – auxiliary gradients with larger magnitudes than the target gradient will be carefully reduced in



(a) Imbalance on one part of the shared parameters (e.g., MLP layer)

(b) example: Imbalance on the user embedding layer

Figure 4.2: The imbalance of gradient magnitudes in transfer learning from auxiliary tasks for recommendations on Alibaba data. The magnitudes dynamically change throughout the training, with the imbalance varying across different parts of the same multi-task network: in Fig 4.2a, the gradient of auxiliary task click-URL is much larger than the target gradient; in Fig 4.2b, the gradient of auxiliary task Add-to-Favorite is much smaller than the target gradient. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

each training iteration; (B) Enhancing the knowledge transferring from weak auxiliary tasks – auxiliary gradients with smaller magnitudes than the target gradient will be carefully enlarged; and (C) MetaBalance adopts both (A) and (B) in the same iteration. In the absence of sufficient prior knowledge, which strategy to apply is treated as a data-driven problem, where the best strategy can be empirically selected based on the performance over the validation set of the target task.

Moreover, MetaBalance has three key characteristics:

1. Auxiliary gradients can be balanced *dynamically* throughout the training process and *adaptively* for different subsets of the shared parameters, which is more flexible than fixed weights for task losses;

2. MetaBalance *prioritizes* the target task via preventing auxiliary tasks from being so strong that they dominate the target task or too weak to help the target task, which can be easily monitored by choosing one of the three strategies;

3. The next important question is *how much should the auxiliary gradient magnitudes be reduced or enlarged?* We design a relax factor to control this to flexibly adapt MetaBalance to different scenarios. The relax factor can also be empirically selected based on the performance over the validation dataset of the target task.

In sum, MetaBalance provides a flexible framework for adapting auxiliary gradients to better improve the target task from the perspective of gradient magnitudes. Extensive experiments over two real-world user behavior datasets from Alibaba show the effectiveness and flexibility of MetaBalance. In particular, we have four target observations:

- With the best strategy and relax factor selected from the validation set, MetaBalance can significantly boost the test accuracy of the target task, which shows that auxiliary knowledge can be better transferred to the target task via MetaBalance.
- MetaBalance can significantly outperform previous methods for adapting auxiliary tasks to improve the target task. For example, we observe a significant improvement of 8.34% upon the strongest baselines in terms of NDCG@10.
- Only one hyper-parameter in MetaBalance (the relax factor) needs to be tuned, irrespective of the number of tasks. Hence, MetaBalance requires only a few training runs, which is more efficient than tuning the weights of task losses, which can be computationally intensive as the number of tasks increases.
- MetaBalance can collaborate well with several popular optimizers including Adam, Adagrad and RMSProp, which shows the potential that MetaBalance can be widely applied in many scenarios.

4.2 Related Work

Recommendations with Auxiliary Tasks. In many personalized recommendation scenarios, the test accuracy of the target task can be improved via joint learning with auxiliary tasks. In social recommendation [38, 79, 120], the knowledge about the user preference can be transferred

from social network to the improve recommendations while the target task like rating prediction jointly train with auxiliary tasks like predicting the connections and trust among users. To improve post-click conversion rate (CVR) prediction, Ma et al. [81] consider the sequential pattern of user actions and introduce post-view click-through rate (CTR) and post-view click-through&conversion rate (CTCVR) as auxiliary tasks. To enhance music playlists or booklists recommendations, predicting if a user will like an individual song or book can also be used as auxiliary tasks and jointly learn with the list-based recommendations. Besides, Bansal et al. [10] design auxiliary tasks of predicting item meta-data (e.g., tags, genres) to improve the rating prediction as the target task. To improve the target goal of predicting ratings, learning user and item embeddings from product review text can also be designed as auxiliary tasks [137].

Auxiliary Learning. In this chapter, we focus on transferring knowledge from auxiliary tasks to improve the target recommendation task, which is an example of *auxiliary learning* paradigm. While multi-task learning aims to improve the performance across all tasks, auxiliary learning differs in that high test accuracy is only required for a primary task, and the role of the other tasks is to assist in generalization of the primary task. Auxiliary learning has been widely used in many areas. In speech recognition, Toshniwal et al. [109] apply auxiliary supervision from phoneme recognition to improve the performance of conversational speech recognition. In computer vision, Liebel at al. [67] propose auxiliary tasks such as the global description of a scene to boost the performance for single scene depth estimation. Mordan et al. [83] observe that object detection can be enhanced if it jointly learns with depth prediction and surface normal prediction as auxiliary tasks. Liu et al. [70] propose a Meta AuXiliary Learning (MAXL) framework that automatically learns appropriate labels for auxiliary tasks. In NLP, Trinh et al. [110] show that unsupervised auxiliary losses significantly improve optimization and generalization of LSTMs. Auxiliary learning has also been applied to improve reinforcement learning [51, 68].

Gradient Direction-based Methods for Adapting Auxiliary Tasks. In auxiliary learning, several methods [33, 68, 135] have been proposed to adapt auxiliary tasks to avoid the situation where they dominate or compete with the target task, where an auxiliary gradient will be down-weighted or

masked out if its direction is conflicting with the direction of the target gradient. We will introduce these methods in detail in Section 4.4.4.1 and compare them with the proposed MetaBalance. In particular, MetaBalance does not punish auxiliary gradients with conflict directions but strengths the dominance of the target task from the perspective of gradient magnitudes. In the experiments, MetaBalance shows better generalization than these gradient direction-based methods.

Multi-Task Learning. Multi-task learning [96, 113] is used to improve the learning efficiency and prediction accuracy of multiple tasks via training them jointly. Shared-bottom model [113] is a commonly used structure where task-specific tower networks receive the same representations that come from a shared bottom network.

Multi-Task Balancing Methods. In multi-task learning, methods have been proposed to balance the joint learning of tasks to avoid the situation where one or more tasks have a dominant influence on the network weight [21, 57, 71, 82, 99]. Although these methods have no special preference to the target task (as in our focus in this chapter), we do discuss their connection to MetaBalance in Section 4.4.4.2 and experimentally compare with them in Section 4.5.

4.3 Problem Statement

Our goal is to improve the test accuracy of a target task via training auxiliary tasks alongside this target task on a multi-task network, where useful knowledge from auxiliary tasks can be transferred so that the shared parameters of the network converge to more robust features for the target task. In the context of personalized recommendation, the target task is normally to predict if a user will interact (e.g., purchase or click) with an item, which can be formulated as a binary classification problem. The test accuracy is measured over the top-K items ranked by their probabilities of being interacted with by the user against the ground-truth set of items that the user actually interacted with.

Let θ denote a subset of the shared parameters. For example, θ could be the weight matrix or the bias vector of a multi-layer perceptron in the shared bottom network. θ is learned by jointly

minimizing the target task loss $\mathcal{L}_{tar}$ with auxiliary task losses $\mathcal{L}_{aux,i}, i = 1, ..., K$:

$$\mathcal{L}_{total} = \mathcal{L}_{tar} + \sum_{i=1}^{K} \mathcal{L}_{aux,i} \tag{4.1}$$

We assume that we update θ^t via gradient descent with learning rate α:

$$\theta^{t+1} = \theta^t - \alpha * \mathbf{G}^t_{total} \tag{4.2}$$

where t means the t-th training iteration over the mini-batches ($t = 1, 2...T$) and $\mathbf{G}^t_{total}$ is the gradient of $\mathcal{L}^t_{total}$ w.r.t θ:

$$\mathbf{G}^t_{total} = \nabla_\theta \mathcal{L}^t_{total} = \nabla_\theta \mathcal{L}^t_{tar} + \sum_{i=1}^{K} \nabla_\theta \mathcal{L}^t_{aux,i} \tag{4.3}$$

where $\mathbf{G}_{total}$ is equivalent to adding up each gradient of the target and auxiliary losses. To simplify the notations, we have:

- $\mathbf{G}_{tar}$ (i.e., $\nabla_\theta \mathcal{L}_{tar}$): the gradient of the target task loss $\mathcal{L}_{tar}$ with respect to θ.
- $\mathbf{G}_{aux,i}$ (i.e., $\nabla_\theta \mathcal{L}_{aux,i}$): the gradient of the i-th auxiliary task loss $\mathcal{L}_{aux,i}$ with respect to θ, where $i = 1, 2...K$.
- $\|\mathbf{G}\|$: the magnitude (L2 Norm) of the corresponding gradient.

As shown in Eq 4.3 and 4.2, the larger the magnitude of a gradient is, the greater the influence this gradient has in updating θ.

Algorithm 1 The Basic Version of MetaBalance

Require: θ^1, $\mathcal{L}_{tar}$, $\mathcal{L}_{aux,1}, ..., \mathcal{L}_{aux,K}$, ***Strategy*** that is selected from $\{\|\mathbf{G}_{aux,i}\| > \|\mathbf{G}_{tar}\|$, $\|\mathbf{G}_{aux,i}\| < \|\mathbf{G}_{tar}\|$, $(\|\mathbf{G}_{aux,i}\| > \|\mathbf{G}_{tar}\|)$ or $(\|\mathbf{G}_{aux,i}\| < \|\mathbf{G}_{tar}\|)\}$

Ensure: θ^T

for t = 1 to T **do**
  $\mathbf{G}^t_{tar} = \nabla_\theta \mathcal{L}^t_{tar}$
  for i = 1 to K **do**
    $\mathbf{G}^t_{aux,i} = \nabla_\theta \mathcal{L}^t_{aux,i}$
    if (*Strategy*) **then**
      $\mathbf{G}^t_{aux,i} \leftarrow \mathbf{G}^t_{aux,i} * \frac{\|\mathbf{G}^t_{tar}\|}{\|\mathbf{G}^t_{aux,i}\|}$
    end if
  end for
  $\mathbf{G}^t_{total} = \mathbf{G}^t_{tar} + \mathbf{G}^t_{aux,1} + \ldots \mathbf{G}^t_{aux,K}$ (element-wise addition)
  Update θ using $\mathbf{G}^t_{total}$ (e.g., Gradient Descent: $\theta^{t+1} = \theta^t - \alpha * \mathbf{G}^t_{total}$)
end for

4.4 Proposed Method

The imbalance of gradient magnitudes may negatively affect the target task optimization. On the one hand, if $\|\mathbf{G}_{aux,i}\|$ $(\exists i \in \{1, 2...K\})$ is much larger than $\|\mathbf{G}_{tar}\|$, the target task will lose its dominance of updating θ and get lower performance. On the other hand, if $\|\mathbf{G}_{aux,i}\|$ $(\exists i \in \{1, 2...K\})$ is much smaller than $\|\mathbf{G}_{tar}\|$, the corresponding auxiliary task might become less influential to assist the target task. As illustrated in Figure 4.2, many personalized recommendations may suffer from this imbalance. Hence, we are motivated to propose a new algorithm that adapts auxiliary tasks from the perspective of gradient magnitudes.

4.4.1 Adapting Auxiliary Gradient Magnitudes

As discussed above, the magnitude imbalance between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}$, ..., $\mathbf{G}_{aux,K}$ may negatively affect the target task optimization. To alleviate this imbalance, MetaBalance is proposed to dynamically and adaptively balance the magnitudes of auxiliary gradients with three strategies and a relax factor (will be detailed in the next subsection).

The basic version of MetaBalance is presented in Algorithm 1, including four steps:

1. **Calculating the Gradients.** In each training iteration, we firstly calculate $\mathbf{G}^t_{tar}$ and $\mathbf{G}^t_{aux,i}$ respectively (line 2 and 4).

2. **Applying the Strategy.** In line 5, we can choose either reducing auxiliary gradients with larger magnitudes than the target gradient, or enlarging auxiliary gradients with smaller magnitudes, or applying the two strategies together. The strategy can be selected based on the validation performance of the target task.

3. **Balancing the Gradients.** Next, $\mathbf{G}^t_{aux,i}$ is normalized to be a unit vector by dividing by $\|\mathbf{G}^t_{aux,i}\|$ and then rescaled to have the same magnitude as $\mathbf{G}^t_{tar}$ by multiplying $\|\mathbf{G}^t_{tar}\|$ (line 6).

4. **Updating the Parameters.** After that, $\mathbf{G}^t_{total}$ (line 9) is obtained by summing $\mathbf{G}^t_{tar}$ and balanced $\mathbf{G}^t_{aux,1}, \dots \mathbf{G}^t_{aux,K}$ together. Then, $\mathbf{G}^t_{total}$ is used to update θ following an optimizer's rule such as gradient descent (line 10). Since step (3) and (4) are completely decoupled, MetaBalance has the potential to collaborate with most commonly used optimizers like Adam and Adagrad [35].

MetaBalance benefits auxiliary learning from six aspects:

1. $\mathbf{G}^t_{aux,i}$ with much larger magnitude than $\mathbf{G}^t_{tar}$ could be automatically reduced, which prevents the dominance of one or more auxiliary tasks for the target task. (Strategy A)

2. $\mathbf{G}^t_{aux,i}$ with much smaller magnitude than $\mathbf{G}^t_{tar}$ could be automatically enlarged, which enhances the knowledge transference from the corresponding auxiliary task. (Strategy B)

3. The (1) and (2) could be done together if necessary. (Strategy C)

4. The strategy is selected based on the target task's performance over validation dataset, which is the empirically best strategy for a specific task and dataset.

5. Because $\frac{\|\mathbf{G}_{tar}^t\|}{\|\mathbf{G}_{aux,i}^t\|}$ can be regarded as a dynamic weight for $\mathbf{G}_{aux,i}^t$ in line 6, MetaBalance can balance $\mathbf{G}_{aux,i}^t$ dynamically throughout the training process.

6. As shown in Figure 4.2, the imbalance of gradient magnitudes varies across the different parts of the same network (e.g., the auxiliary gradients might be much larger than the target gradient in an MLP but much smaller in an embedding layer). Because MetaBalance can be easily applied to each part of the shared parameters separately (θ is an input of Algorithm 1), the training of the different parts can be balanced respectively and adaptively. (5) and (6) makes MetaBalance more flexible than using fixed weights for task losses.

However, the drawback of this basic version in Algorithm 1 is also obvious: forcing auxiliary gradients to have exactly the same magnitude as the target gradient might not be optimal for the target task. To overcome this inflexibility of the magnitude scaling, we design a relax factor to control the closeness of $\|\mathbf{G}_{aux,i}^t\|$ to $\|\mathbf{G}_{tar}^t\|$ in the following subsection.

4.4.2 Adjusting Magnitude Proximity

The next question is how to flexibly adjust the magnitude proximity between $\mathbf{G}_{aux,i}$ and $\mathbf{G}_{tar}$ to adapt to different scenarios? We design a relax factor r to control this magnitude proximity, which is used in line 6 of Algorithm 1:

$$\mathbf{G}_{aux,i}^t \leftarrow (\mathbf{G}_{aux,i}^t * \frac{\|\mathbf{G}_{tar}^t\|}{\|\mathbf{G}_{aux,i}^t\|}) * r + \mathbf{G}_{aux,i}^t * (1 - r)$$

where, if $r = 1$, then $\mathbf{G}_{aux,i}^t$ has exactly the same magnitude as $\mathbf{G}_{tar}^t$. If $r = 0$, then $\mathbf{G}_{aux,i}^t$ keeps its original magnitude. The larger r is, the closer $\|\mathbf{G}_{aux,i}^t\|$ gets to $\|\mathbf{G}_{tar}^t\|$. Hence, r balances the magnitude information between each auxiliary gradient and the target gradient.

The impact of r on the magnitude proximity is illustrated in Figure 4.3. We observe that the target gradient is dominated by an auxiliary gradient with its much larger magnitude when $r = 0$ in Figure 4.2a. In contrast, $r = 1$ lets all gradients have the same but very small magnitude as the target gradient in Figure 4.3d. Between the two extremes, Figure 4.3b ($r = 0.2$) and Figure 4.3c ($r = 0.7$) balance the gradient magnitudes in a more moderate way, which pushes $\|\mathbf{G}^t_{aux,i}\|$ closer to $\|\mathbf{G}^t_{tar}\|$ but not exactly the same – the original magnitude can be partially kept.

More than that, *r can actually affect the weight for each auxiliary task.* We can further reformulate line 6 in Algorithm 1 as:

$$\mathbf{G}^t_{aux,i} \leftarrow \mathbf{G}^t_{aux,i} * w^t_{aux,i}$$

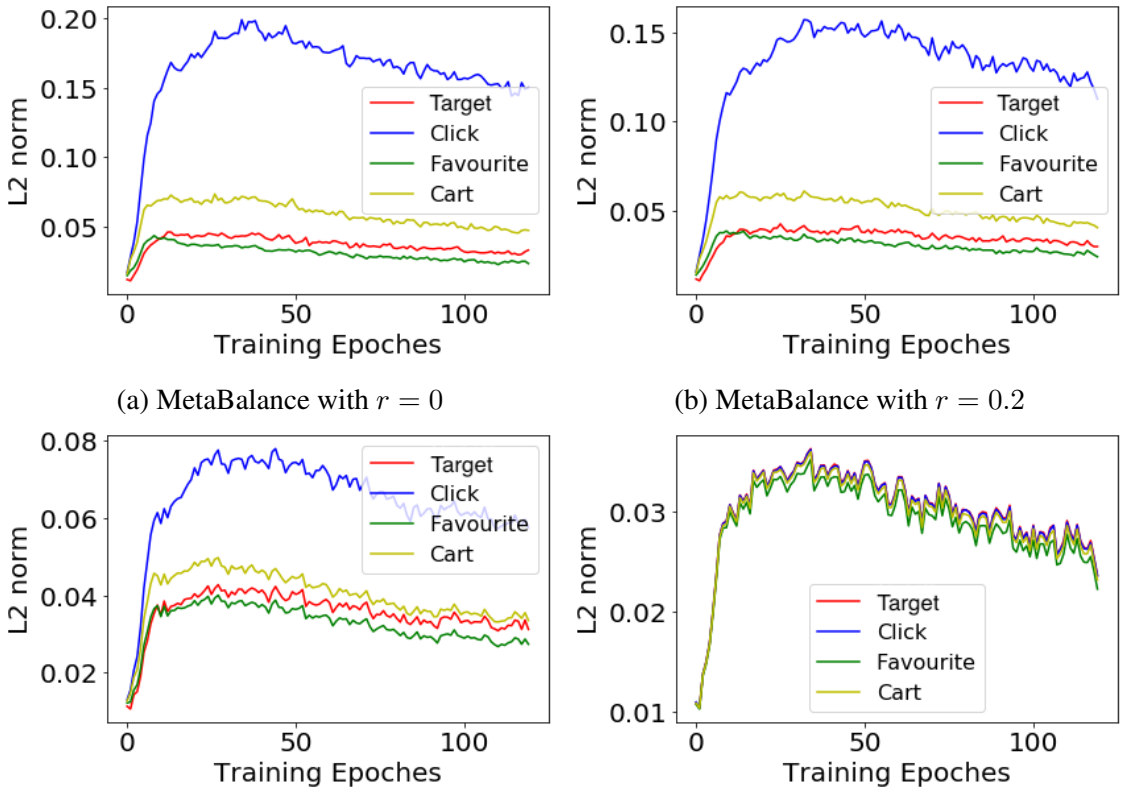
where $w^t_{aux,i}$ is the weight for $\mathbf{G}^t_{aux,i}$ and we have:

$$w^t_{aux,i} = (\frac{\|\mathbf{G}^t_{tar}\|}{\|\mathbf{G}^t_{aux,i}\|} - 1) * r + 1 \tag{4.4}$$

where, if $\|\mathbf{G}^t_{tar}\| > \|\mathbf{G}^t_{aux,i}\|$, the higher r is, the higher $w^t_{aux,i}$ will be; however, if $\|\mathbf{G}^t_{tar}\| < \|\mathbf{G}^t_{aux,i}\|$, the higher r is, the lower $w^t_{aux,i}$ will be.

The next key question is how to choose this r? As presented in Equation 4.4, r affects the weight for each auxiliary task. Without the prior knowledge of the importance of each auxiliary task to the target task, we treat the setting of r as a data-driven problem and believe that r should be carefully adjusted to adapt to different scenarios. Since r is only used in the backward propagation and hence has no gradient from any loss, r is not a learnable parameter inherently. Hence, we treat r as a hyper-parameter, which is tuned over validation datasets. Note that the same r for all auxiliary tasks does not mean that they will have the same weight or gradient magnitude because $w^t_{aux,i}$ is not only decided by r but also affected by $\|\mathbf{G}^t_{tar}\|$ and $\|\mathbf{G}^t_{aux,i}\|$ (see Equation 4.4).

Therefore, there is only one hyper-parameter r in MetaBalance that needs to be tuned, which is irrespective of the number of tasks. In contrast, the computational complexity of tuning weights of task losses increases exponentially for each task added. Moreover, we also observe that Meta-



(a) MetaBalance with $r = 0$

(b) MetaBalance with $r = 0.2$

(c) MetaBalance with $r = 0.7$

(d) MetaBalance with $r = 1.0$

Figure 4.3: The impact of relax factor r on magnitude proximity on UserBehavior-2017 dataset. In the legend, "target" represents the target task (i.e., purchase prediction). Y-axis is the average gradient magnitude over all mini-batch iterations in one epoch, where the gradient w.r.t a MLP layer of the multi-task network is taken as the example. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

Balance achieves higher test accuracy than tuning the task weights in our experiments.

Finally, instead of using current magnitudes $\|\mathbf{G}^t_{tar}\|$ and $\|\mathbf{G}^t_{aux,i}\|$ in Algorithm 1, following [82], we apply the moving average of magnitude of the corresponding gradient to take into account the variance among all gradient magnitudes over the training iterations:

$$m^t_{tar} = \beta * m^{t-1}_{tar} + (1-\beta) * \|\mathbf{G}^t_{tar}\| \tag{4.5}$$

$$m^t_{aux,i} = \beta * m^{t-1}_{aux,i} + (1-\beta) * \|\mathbf{G}^t_{aux,i}\|, \forall i = 1, ..., K \tag{4.6}$$

where $m^0_{tar} = m^0_{aux,i} = 0$ and β is to control the exponential decay rates of the moving averages, which could be empirically set as 0.9. The moving averages make the training more stable and will be discussed in the experiments. Finally, the complete version of MetaBalance is shown in Algorithm 2.

Algorithm 2 The Complete Version of MetaBalance

Require: θ^1, $\mathcal{L}_{tar}$, $\mathcal{L}_{aux,1}$, ..., $\mathcal{L}_{aux,K}$, relax factor r, β in moving averages, ***Strategy*** that is selected from $\{m_{aux,i} > m_{tar}, m_{aux,i} < m_{tar}, (m_{aux,i} > m_{tar})$ or $(m_{aux,i} < m_{tar})\}$

Ensure: θ^T

Initialize $m^0_{tar} = m^0_{aux,i} = 0$
for t = 1 to T **do**
  $\mathbf{G}^t_{tar} = \nabla_\theta \mathcal{L}^t_{tar}$
  $m^t_{tar} = \beta * m^{t-1}_{tar} + (1-\beta) * \|\mathbf{G}^t_{tar}\|$
  for i = 1 to K **do**
    $\mathbf{G}^t_{aux,i} = \nabla_\theta \mathcal{L}^t_{aux,i}$
    $m^t_{aux,i} = \beta * m^{t-1}_{aux,i} + (1-\beta) * \|\mathbf{G}^t_{aux,i}\|$
    if (*Strategy*) **then**
      $\mathbf{G}^t_{aux,i} \leftarrow (\mathbf{G}^t_{aux,i} * \frac{m^t_{tar}}{m^t_{aux,i}}) * r + \mathbf{G}^t_{aux,i} * (1-r)$
    end if
  end for
  $\mathbf{G}^t_{total} = \mathbf{G}^t_{tar} + \mathbf{G}^t_{aux,1} + \ldots \mathbf{G}^t_{aux,K}$ (element-wise addition)
  Update θ using $\mathbf{G}^t_{total}$ (e.g., $\theta^{t+1} = \theta^t - \alpha * \mathbf{G}^t_{total}$)
end for

4.4.3 Time and Space Complexity Analysis

In this section, we show that MetaBalance does **not** significantly increase the time and space complexity of training multi-task networks. Assume that addition, subtraction, multiplication, division and square root take "one unit" of time. The time complexity of training a multi-task network depends on the network's structure. For simplicity, assume that an MLP is the shared layer of a multi-task network and θ is a weight matrix of a single layer in the MLP, where θ has input dimension n and output dimension m. The time complexity of updating θ is $\mathcal{O}(T(1+K)nmd)$ [11], where T is the number of training iterations over the mini-batches, $(1+K)$ is the count

of the target task plus K auxiliary tasks and d is the size of the mini-batch. For MetaBalance in Algorithm 2, in each training iteration and for each task, r is a hyper-parameter, calculating $m^t_{aux,i}$ or m^t_{tar} takes $\mathcal{O}(nmd)$, and the time complexity of updating the magnitude of $\mathbf{G}^t_{aux,i}$ (line 9) is also $\mathcal{O}(nmd)$. To sum up, the time complexity of MetaBalance is still $\mathcal{O}(T(1+K)nmd)$. Therefore, MetaBalance will not significantly slow down the training of multi-task networks.

Except for the space of training a multi-task network, MetaBalance only requires extra space for m_{tar}, r, β and $m_{aux,i}$,..., $m_{aux,K}$, where the space complexity is $\mathcal{O}(3+K) = \mathcal{O}(1)$ (K is normally a small number). Hence, MetaBalance does not significantly increase the space complexity of multi-task networks training either.

4.4.4 Comparison with Previous Methods

In this section, we compare MetaBalance with previous methods.

4.4.4.1 Auxiliary Task Adapting Methods

These methods are specifically designed for auxiliary learning such that auxiliary tasks are adapted to better improve the target task. The common idea is that if $\mathbf{G}_{aux,i}$ is conflicting with $\mathbf{G}_{tar}$ from the perspective of direction, $\mathbf{G}_{aux,i}$ will be down-weighted or masked out. Compared with them, MetaBalance is the first method that adapts auxiliary task to assist the target task from the perspective of gradient magnitudes rather than punishing $\mathbf{G}_{aux,i}$ due to conflicting directions. The experimental results show that keeping inner-competition between the target gradient and conflicting auxiliary gradients as MetaBalance does improves the generalization ability of the model.

GradSimilarity [33] adapts auxiliary tasks via the gradient similarity between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}$. Specifically, if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative, then $\mathbf{G}_{aux,i}$ will not be added to $\mathbf{G}_{total}$ and hence will be ignored in updating the shared layers.

GradSurgery[3] [135] replaces $\mathbf{G}_{aux,i}$ by its projection onto the normal plane of $\mathbf{G}_{tar}$ if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative, unlike [33] where $\mathbf{G}_{aux,i}$ is just ignored. Formally, if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative,

[3]GradSurgery is originally for balancing multi-task learning. We can easily apply GradSurgery for auxiliary learning by specifying a task as the target task and the others as the auxiliary tasks

they let:

$$\mathbf{G}^t_{aux,i} = \mathbf{G}^t_{aux,i} - \frac{\mathbf{G}^t_{aux,i} \cdot \mathbf{G}^t_{tar}}{\|\mathbf{G}^t_{tar}\|^2} \cdot \mathbf{G}^t_{tar} \tag{4.7}$$

In this way, the conflict between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}$ can be alleviated.

OL-AUX (Oline learning for Auxiliary Losses) [68] defines the total loss as $\mathcal{L}^t = \mathcal{L}^t_{tar} + \sum_{i=0}^{K} w^t_{aux,i} \cdot \mathcal{L}^t_{aux,i}$ where $w_{aux,i}$ is the weight of $\mathcal{L}_{aux,i}$ and $\mathcal{V}^t(\mathbf{w})$ as the speed at which the target task loss decreases at the t-th iteration and $\mathbf{w} = [w_1, ..., w_K]^T$. OL-AUX seek to optimize the N-step decrease of the target task w.r.t $\mathbf{w}$:

$$\mathcal{V}^{t,t+N}(\mathbf{w}) = \mathcal{L}^{t+N}_{tar} - \mathcal{L}^t_{tar} \tag{4.8}$$

With some approximations, they find that $\forall i = 1, ..., K$:

$$\nabla_{w_{aux,i}} \mathcal{V}^{t,t+N}(w_{aux,i}) = -\sum_{j=0}^{N-1} (\mathbf{G}^{t+j}_{tar})^T \mathbf{G}^{t+j}_{aux,i} \tag{4.9}$$

Then, $\mathbf{w} \leftarrow \mathbf{w} - \beta \cdot \nabla_{\mathbf{w}} \mathcal{V}^{t,t+N}(\mathbf{w})$ such that the speed at which $\mathcal{L}_{tar}$ decreases could be maximized.

4.4.4.2 Multi-Task Balancing Methods

In contrast to the auxiliary learning-specific methods, these multi-task balancing methods are for general learning where all tasks are treated equally important. Although they have no preference to the target task, they are valid baselines because MetaBalance is specifically for auxiliary learning and is supposed to outperform them. For convenience, we let j denotes the index of any task in this subsection, where $j = 0, 1, ..., K$. Let w_j be the weight of the j-th task loss $\mathcal{L}_j$.

Uncertainty [57] assumes that the higher the uncertainty of task data is, the lower the weight of this task loss should be assigned. They design a learnable parameter σ_j to model the uncertainty for each task. Specifically, they optimize the model parameters and σ_j to minimize the following objective:

$$\mathcal{L} = \sum_{j=0}^{K} \frac{1}{\sigma_j^2} \mathcal{L}_j + \sum_{j=0}^{K} log \sigma_j \tag{4.10}$$

Minimizing the loss $\mathcal{L}$ w.r.t. σ_j can automatically balance $\mathcal{L}_j$ during training, where increasing σ_j

reduces the weight for task loss $\mathcal{L}_j$.

GradNorm [21] encourages $\|\mathbf{G}_j^t\|$ to be the mean of all $\|\mathbf{G}_j^t\|,\ j = 0, ..., K$. In this way, all tasks could have a similar impact on the updating of shared-parameters. In particular, they minimize the following two objectives:

$$\mathcal{L}^t = \sum_{j=0}^{K} w_j^t \cdot \mathcal{L}_j^t \tag{4.11}$$

$$\mathcal{L}_{normLoss}^t = \sum_{j=0}^{K} L1Norm(\|w_j^t \cdot \mathbf{G}_j^t\| - \|\overline{\mathbf{G}^t}\| \cdot [r_j^t]^\alpha) \tag{4.12}$$

In each iteration, $\mathcal{L}^t$ is firstly optimized w.r.t model parameters θ (not including w_j^t) to obtain $\mathbf{G}_j^t$ and the $\mathcal{L}_{normLoss}^t$ is optimized w.r.t w_j^t. In the next iteration, updated w_j^t can balance $\mathcal{L}_j^t$. Moreover, r_j^t is to model the pace at which different tasks are learned, where $r_j^t = p_j^t / E[p_j^t]$ and $p_j^t = \mathcal{L}_j^t / \mathcal{L}_j^0$. And α is a hyper-parameter which sets the strength of forcing tasks back to a common training rate.

DWA (Dynamic Weight Averaging) [71] balances the pace at which tasks are learned. In DWA, w_j^t is set as:

$$w_j^t = \frac{N \cdot exp(p_j^{t-1}/T)}{\sum_n exp(p_n^{t-1}/T)},\ \ p_j^{t-1} = \frac{\mathcal{L}_j^{t-1}}{\mathcal{L}_j^{t-2}} \tag{4.13}$$

where N is the number of tasks and temperature T controls the softness of the task weighting in the softmax function. p_j estimates the relative descending rate of $\mathcal{L}_j$. When $\mathcal{L}_j$ decreases at a slower rate compared with other task losses, w_j will be increased.

MGDA (Multiple-Gradient Descent Algorithm) [99] treats multi-task learning as multi-objective optimization problem and finds solutions that satisfies Pareto optimality – as long as there is a common direction along which losses can be decreased, we have not reached a Pareto optimal point yet. Since the shared parameters are only updated along **common** directions of the task-specific gradients, MGDA has no preference on a particular task.

MTAdam [82] is an Adam-based optimizer that balances gradient magnitudes and then update parameters according to the rule of Adam [58]. Following MTAdam, we also directly manipulate

the gradient magnitudes, instead of weighting task losses like Uncertainty [57], GradNorm [21] and DWA [71]. MetaBalance differs from MTAdam in the following aspects:

- MTAdam lets all gradient magnitudes be similar to that of the first loss (not necessarily the target loss) while MetaBalance has three strategies that can flexibly encourage auxiliary gradients to better help the target task optimization.
- $\|\mathbf{G}_{aux,i}^t\|$ can only be very similar to $\|\mathbf{G}_{tar}^t\|$ in MTAdam while MetaBalance can adjust the proximity of $\|\mathbf{G}_{aux,i}^t\|$ to $\|\mathbf{G}_{tar}^t\|$ via the relax factor, which is vital for adapting MetaBalance to different scenarios.
- MetaBalance is an auxiliary task adapting algorithm that can collaborate with most optimizers like Adagrad or RMSprop to update parameters, whereas MTAdam is specially designed for Adam-based optimizers only.

4.5 Experiments

In this section, we present our results and discussion toward answering the following experimental research questions:

- **RQ1:** How well does MetaBalance improve the target task via adapting the magnitudes of auxiliary gradients?
- **RQ2:** How well does MetaBalance perform compared to previous auxiliary task adapting and multi-task balancing methods?
- **RQ3:** How well does MetaBalance collaborate with commonly used optimizers such as Adam and Adagrad?
- **RQ4:** What is the impact of moving averages of gradient magnitudes in MetaBalance?

4.5.1 Experimental Setup

Following the auxiliary learning setting [70, 112], high test accuracy is only required for a target task while the role of auxiliary tasks is to assist the target task to achieve better test accuracy.

Datasets. IJCAI-2015[4] is a public dataset from IJCAI-15 contest, which contains millions of anonymized users' shopping logs in the past 6 months. UserBehavior-2017[5] is a public dataset of anonymized user behaviors from Alibaba. The two datasets both contain users' behaviors including click, add-to-cart, purchase and add-to-favorite. The statistics of preprocessed datasets are summarized in Table 4.1. We treat purchase prediction as the target task and the prediction of other behaviors as auxiliary tasks. We formulate the prediction of each behavior like purchase as a binary classification problem and negative samples are randomly selected.

Evaluation and metrics. In the evaluation, **all items** are ranked according to the probability of being purchased by the user and the top-K items are returned and measured against the ground-truth items set of what users actually purchased, where we adopt three metrics: Normalized Discounted Cumulative Gain (NDCG) [52] at 10 and 20 (N@10 and N@20), precision at 10 and 20 (P@10 and P@20), and recall at 10 and 20 (R@10 and R@20).

Multi-task network. Because how to design a better multi-task network is not the emphasis of this chapter, we directly adopt the combination of MLP layer and matrix factorization layer as the shared bottom network, which is widely adopted for recommendations in both academia [44] and industry like Google [23] and Facebook [84]. We build MLP layer as the task-specific tower for each task. The multi-task network is shown in Figure 4.4.

Baselines. We compare MetaBalance with 10 baseline methods. Gradient direction-based methods that are designed for adapting auxiliary tasks to improve the target task, which will be detailed in Section 4.4.4.1, including: **GradSimilarity** [33], **GradSurgery** [135], **OL-AUX** [68]. Multi-Task balancing methods that treat all tasks equally, which will be detailed in Section 4.4.4.2, including: **Uncertainty** [57], **GradNorm** [21], **DWA** [71], **MTAdam** [82] and **MGDA** [99]. And three simple baselines. **Single-Loss**: we mask out the loss terms of auxiliary tasks and only use target task loss to calculate gradients and update parameters in the model. **Vanilla-Multi**: multiple loss terms are not balanced where the weights for all loss terms are 1. **Weights-Tuning**: weights of

[4] https://tianchi.aliyun.com/dataset/dataDetail?dataId=47&userId=1
[5] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

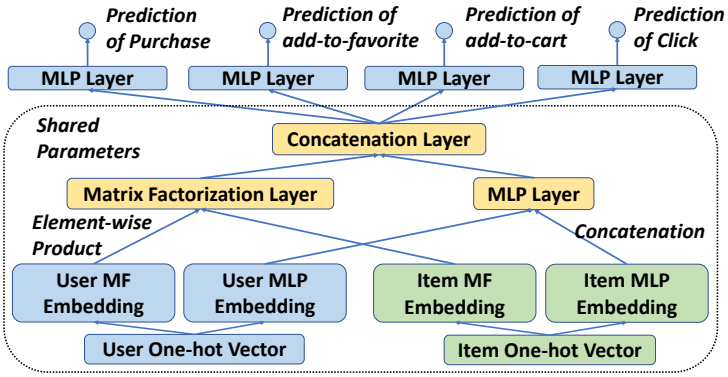


Figure 4.4: Multi-task recommendation network in the evaluation. The shared bottom layers is the combination of MLP layer and matrix factorization layer, which is widely adopted for recsys in both academia [44] and industry [23, 84]. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

loss terms are obtained by random search.

4.6 Reproducibility of Experiments

The code of our approach can be found at here.[6]

Dataset Preprocessed and Split. IJCAI-2015 is preprocessed by filtering out users who purchase fewer than 20 unique items and items which are purchased by fewer than 10 unique users. We omit add-to-cart as an auxiliary task in IJCAI-2015 because this behavior only has 1,693 feedbacks. For UserBehavior-2017, we filter out users who purchase fewer than 10 unique items and items which are purchased by fewer than 10 unique users. The datasets are summarized in Table 4.1. We randomly split purchase interactions into a training set (70%), validation set (10%) and testing set (20%). For the interactions of auxiliary tasks like add-to-cart, we merge them into the training set. Since auxiliary interactions like add-to-cart are highly related to purchase interaction, to prevent possible information leakage, we remove user-item pairs from the auxiliary interactions if these pairs appear in the validation set and testing set of the purchase interactions.

Implementation and Training Details. We implement MetaBalance, Uncertainty, DWA, Grad-

[6] https://github.com/facebookresearch/MetaBalance

Table 4.1: Statistics of preprocessed datasets. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

			Target Task		Auxiliary Tasks		
Dataset	#User	#Item	#Buy	Density of Buy	#Add-to-Cart	#Click	#Add-to-Favorite
IJCAI-2015	19,839	50,973	390,600	0.039%	1,693	2,025,910	224,279
UserBehavior-2017	16,089	25,813	89,404	0.022%	53,245	394,246	19,585

Similarity, GradSurgery and OL-AUX via Pytorch. The code of GradNorm is from this repo[7] and the code of MTAdam is from the authors.[8] All experiments are conducted on an Nvidia GeForce GTX Titan X GPU with 12 GB memory. Cross-entropy loss is adopted for each task and Adam [58] is the optimizer with batch size of 256 and learning rate of 0.001.

Hyper-parameters. All hyper-parameters are carefully tuned in the validation set, where early stopping strategy is applied such that we terminate training if validation performance does not improve over 20 epochs. In the multi-task recommendation network, the size of user and item embeddings is 64, the size of the shared MLP layers is {32, 16, 8} and the size of the task-specific MLP layers is {64, 32}. To prevent overfitting, dropout with rate of 0.5 is applied for each layer and we also use weight decay with rate of e-7. For MetaBalance, r is selected from 0.1, 0.2, ...0.9 and 0.7 is the best for UserBehavior-2017 and 0.9 is the best for IJCAI-2015. For MTAdam, β_1, β_2, β_3 are respectively set as 0.9, 0.999 and 0.9. For DWA, T is set as 2 and we calculate the mean of losses in very 5 iterations on IJCAI-2015 and in very 10 iterations on UserBehavior-2017. For GradNorm, α is set as 0.75 on IJCAI-2015 and 0 on UserBehavior-2017. For OL-AUX, β is set as 0.1 on IJCAI-2015 and 1 on UserBehavior-2017.

4.6.1 RQ1: Improvement of Target Task via Adapting Auxiliary Gradients

In this subsection, we discuss the impact of adapting auxiliary gradient magnitudes on the target task's performance.

[7] https://github.com/hosseinshn/GradNorm
[8] https://github.com/ItzikMalkiel/MTAdam

Table 4.2: Strategy selection. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

Datasets	UserBehavior-2017			IJCAI-2015		
Metrics (%)	N@10	R@10	P@10	N@10	R@10	P@10
Vanilla-Multi	0.820	1.284	0.291	0.844	0.965	0.437
Strategy A	0.948	1.487	0.316	0.858	0.963	0.424
Strategy B	0.904	1.384	0.301	0.818	0.950	0.425
Strategy C	0.990	1.550	0.339	0.974	1.164	0.509

Strategy A: strengthening the dominance of the target task;
Strategy B: enhancing the knowledge transferring from weak auxiliary tasks;
Strategy C: Adopting Strategy A and Strategy B together.

Impact of Strategy Selection. We firstly study which strategy is optimal for the two recommendation datasets. Note that we firstly compare the three strategies over the validation dataset to choose the best one and apply it on the test dataset. To be consistent with other experimental results, we present the results of the three strategies over the test dataset in Table 4.2, which reflects the same pattern as the validation dataset. First of all, all three strategies significantly outperform vanilla multi-task learning baseline ("Vanilla-Multi") in UserBehavior-2017 and Strategy C significantly outperforms the baseline in IJCAI-2015, which shows the effectiveness and robustness of MetaBalance. We observe the pattern "Strategy C > Strategy A > Strategy B" across the two datasets, which shows that strengthening the dominance of the target task (Strategy A) is more important than enhancing the knowledge transferring from weak auxiliary tasks (Strategy B) and combining the two strategies together can achieve further improvements for the two datasets. Therefore, we apply Strategy C in the rest of the experiments.

Impact of Relax Factor. Based on Strategy C, we further study the impact of the relax factor. Figure 4.5 presents the varying of NDCG@10 and Recall@10 as r changes in UserBehavior-2017 dataset (the similar observation is obtained in IJCAI-2015 dataset).

The worst NDCG@10 and Recall@10 are achieved when $r = 0$ (Vanilla-Multi), where auxiliary gradients ($\|\mathbf{G}^t_{aux,i}\|$) keep their original magnitudes (i.e., not balanced as in Vanilla-Multi).

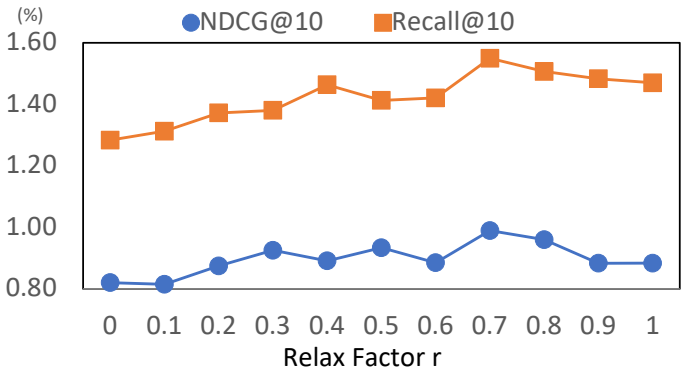


Figure 4.5: Impact of relax factor r. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

In Figure 4.2a, we observe that click task's gradient magnitude (blue curve) is much larger than $\|\mathbf{G}^t_{tar}\|$ (red curve). Hence, the target task gradient is probably dominated by the click task gradient, which explains the low target task performance of Vanilla-Multi (see Table 4.3).

In contrast, $r = 1$ in MetaBalance means that $\|\mathbf{G}^t_{aux,i}\|$ becomes very close to $\|\mathbf{G}^t_{tar}\|$ as shown in Figure 4.3d, where the four curves are twisted together. However, $r = 1$ achieves suboptimal results as shown in Figure 4.5, which demonstrates that the target task performance might be negatively impacted by a large r. A possible reason is that most auxiliary gradients are reduced to be very similar to the target gradient and hence the update of the shared parameters becomes so small that it negatively affects the optimization.

Between the two extremes, Figure 4.3b ($r = 0.2$) and Figure 4.3c ($r = 0.7$) balance the gradient magnitudes in a more moderate way – getting $\|\mathbf{G}^t_{aux,i}\|$ closer to $\|\mathbf{G}^t_{tar}\|$ but not exactly the same, where $r = 0.7$ achieves the best performance as shown in Figure 4.5.

4.6.2 RQ2: Comparison with Baseline Methods

Table 4.3 and Table 4.4 present the experimental results and the improvement of MetaBalance upon the strongest baseline in terms of each metric, where MetaBalance significantly outperforms all baselines over most of metrics on the two datasets.

MetaBalance vs. gradient direction-based methods. First, we observe that MetaBalance

Table 4.3: Experimental results of UserBehavior-2017 dataset. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

	UserBehavior-2017					
Metric(%)	N@10	R@10	P@10	N@20	R@20	P@20
Single-Loss	0.817	1.265	0.275	0.994	1.825	0.208
Vanilla-Multi	0.820	1.284	0.291	1.074	2.107	0.237
Weights-Tuning	0.909	1.378	<u>0.326</u>	1.165	2.195	0.263
Uncertainty	0.724	1.158	0.266	0.903	1.739	0.201
GradNorm	0.913	1.292	0.297	1.147	2.044	0.237
DWA	0.915	1.419	0.309	1.165	2.232	0.248
MGDA	0.845	1.328	0.292	1.075	2.058	0.237
MTAdam	0.869	1.382	0.305	1.112	2.153	0.247
GradSimilarity	0.923	1.444	0.308	1.186	2.270	0.255
GradSurgery	<u>0.936</u>	<u>1.471</u>	0.319	<u>1.213</u>	<u>2.371</u>	<u>0.263</u>
OL-AUX	0.931	<u>1.471</u>	0.311	1.162	2.224	0.243
MetaBalance	**0.990***	**1.550***	**0.339***	**1.258***	**2.421***	**0.269***
Improvement	5.77%	5.32%	3.96%	3.66%	2.09%	2.08%

$*$ We conduct a two-sided significant test between MetaBalance and the strongest baseline (highlighted by underscore), where * means the p-value is smaller than 0.05.

Table 4.4: Experimental results of IJCAI-2015 dataset. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

	IJCAI-2015					
Metric(%)	N@10	R@10	P@10	N@20	R@20	P@20
Single-Loss	0.883	0.935	0.431	1.022	1.314	0.298
Vanilla-Multi	0.844	0.965	0.437	0.992	1.353	0.311
Weights-Tuning	0.866	1.013	0.445	1.037	1.448	0.330
Uncertainty	0.695	0.818	0.365	0.834	1.186	0.266
GradNorm	0.878	0.953	0.430	1.035	1.375	0.307
DWA	<u>0.899</u>	1.005	0.442	1.040	1.372	0.312
MGDA	0.809	1.104	0.439	<u>1.104</u>	<u>1.673</u>	<u>0.350</u>
MTAdam	0.880	<u>1.015</u>	<u>0.463</u>	1.071	1.525	0.348
GradSimilarity	0.817	0.977	0.427	1.025	1.529	0.336
GradSurgery	0.876	0.998	0.445	1.042	1.434	0.327
OL-AUX	0.931	0.921	0.413	0.950	1.312	0.295
MetaBalance	**0.974***	**1.164***	**0.509***	**1.134***	**1.588**	**0.353**
Improvement	8.34%	14.68%	10.01%	2.72%	–	0.86%

$*$ We conduct a two-sided significant test between MetaBalance and the strongest baseline (highlighted by underscore), where * means the p-value is smaller than 0.05.

outperforms GradSimilarity, OL-AUX and GradSurgery, which are designed to boost the target task via adapting auxiliary tasks. Remember that the same idea behind these methods is that the less similar the direction of target gradient and one auxiliary gradient is, the lower weight will be assigned to that auxiliary task. While these gradient direction-based methods have worse performance than MetaBalance over the testing dataset, interestingly, they actually achieve better **training** loss than MetaBalance, where an example is shown in Figure 4.6, which demonstrates they are more prone to overfitting than MetaBalance. Hence, this observation reveals that auxiliary gradients that have dissimilar directions with the target gradient might be sometimes helpful to improve the generalization ability of the model, which is consistent with the observations in the literature [22, 113]. For example, they might help the target task to correct its direction of the optimization to achieve a better generalization ability. MetaBalance keeps the direction conflicts between the target gradient and the auxiliary gradients but reduces the auxiliary gradients whose magnitudes are much larger than the target gradient, which prevents the dominance of auxiliary tasks and shows more robust performance for personalized recommendations.

In addition, we are curious if MetaBalance can be enhanced when it also considers using the direction similarity to adapt auxiliary gradients. Specifically, in each training iteration, we first enlarge or reduce auxiliary gradients via MetaBalance and then enlarge or reduce them again according to one of the gradient direction-based methods. The results in Table 4.5 show that the performance of MetaBalance mostly drops after including the gradient direction-based methods, which demonstrates that naively combining both magnitude and direction-based approaches can interfere with each another. We leave how to better consider both gradient magnitudes and directions for adapting auxiliary tasks to help the target task in the future work.

MetaBalance vs. multi-task balancing methods. Second, it is understandable that Uncertainty, GradNorm, DWA are inferior to MetaBalance because they have no special preference to the target task. In DWA, the lower the loss decreases, the higher the weight is assigned to that loss. In GradNorm, the target task gradient magnitude is regularized to be similar to the average of all gradient magnitudes, which might not be the optimal magnitude for the target task optimization.

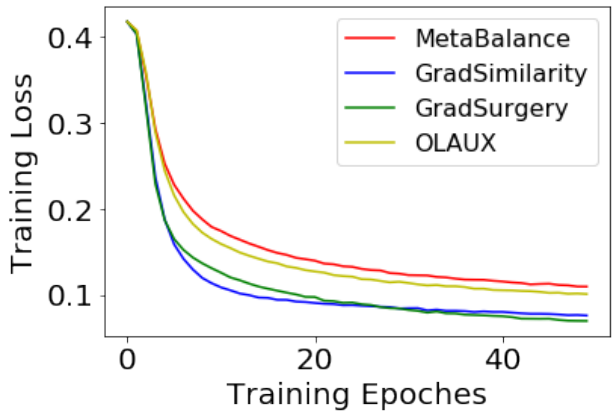


Figure 4.6: The training loss on UserBehavior-2017. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

In Uncertainty, the higher the uncertainty of the task dataset, the higher weight is assigned to that task loss. We also compare MGDA [99] as one of the most representative Pareto methods with MetaBalance. MGDA treats multi-task learning as multi-objective optimization problem and finds solutions that satisfy Pareto optimality. In MGDA, the shared parameters are only updated along common directions of the gradients for all tasks, which might not be the best optimization direction for the target task. Consequently, the target task is not guaranteed to be improved the most among all tasks in Pareto optimal solutions like MGDA. In contrast, MetaBalance is a specialized method designed for boosting the target task. As Table 4.3 shows, MetaBalance significantly outperforms MGDA over most of the metrics. Although MTAdam is not originally designed for auxiliary learning, we let the target task serve as the anchor task in MTAdam. In this way, MetaBalance and MTAdam share the same core idea that the auxiliary gradient magnitudes become closer to the target gradient. However, Table 4.3 shows that MetaBalance significantly outperforms MTAdam. The possible reason might be the relax factor in MetaBalance that can control the magnitude proximity, which makes MetaBalance more flexible than MTAdam.

In addition, Vanilla-Multi is even inferior to Single-loss over most of metrics on both datasets. This demonstrates that transfer learning from auxiliary tasks is a non-trivial task – that might hurt the performance of the target task rather than boosting it. After that, Table 4.3 shows that Weights-

Tuning, where the target task loss normally has a higher weight assigned than the auxiliary tasks, outperforms Vanilla-Multi over all metrics on both datasets. However, the performance of Weights-Tuning is significantly inferior to MetaBalance. A possible reason is that the tuned weights are fixed during the training and hence behave sub-optimally in adapting auxiliary tasks.

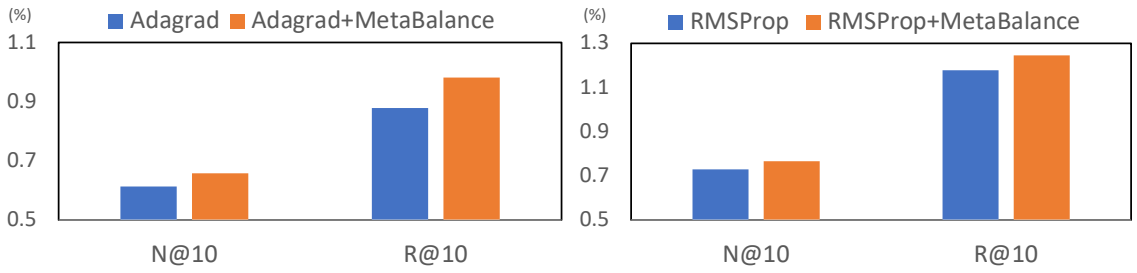
To sum up, the results demonstrate that the three strategies and the relax factor make MetaBalance a flexible and effective framework to adapt auxiliary tasks from the perspective of gradient magnitudes, which significantly improves the target task's performance and outperforms baselines.

Table 4.5: MetaBalance plus gradient direction-based methods. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

Metric(%)	N@10	R@10	P@10	N@20	R@20	P@20
MetaBalance (MB)	0.990	1.550	0.339	1.258	2.421	0.269
MB+GradientSimilarity	0.937	1.398	0.311	1.190	2.210	0.250
MB+GradientSurgery	0.925	1.585	0.329	1.167	2.351	0.258
MB+OL-AUX	0.898	1.374	0.308	1.158	2.224	0.248

4.6.3 RQ3: Collaboration with More Optimizers

As shown in Algorithm 1 and 2, MetaBalance balances the gradient magnitudes and these balanced gradients are used to update shared parameters following the rules of optimizers. Results in Table 4.3 have shown that MetaBalance can collaborate with Adam well. We are also curious if MetaBalance can collaborate with other popular optimizers – achieving higher performance for the target task compared to the multi-task network that is trained without MetaBalance. In Figure 4.7, we observe that two other widely used optimizers – Adagrad [35] and RMSProp [108] – can also achieve better performance via using the balanced gradients from MetaBalance. This result demonstrates that MetaBalance can flexibly collaborate with commonly-used optimizers.



(a) Adagrad (b) RMSProp

Figure 4.7: Collaboration with other optimizers. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

4.6.4 RQ4: Impact of Moving Averages of Gradient Magnitudes

In Table 4.6, we compare the performance of MetaBalance with its variant ("−Moving Average") where the moving averages of magnitude m^t_{tar} and $m^t_{aux,i}$ (in Equation 4.5 and 4.6) are replaced with the current magnitudes $\mathbf{G}^t_{tar}$ and $\mathbf{G}^t_{aux,i}$ at each iteration. We observe that the performance drops slightly on UserBehavior-2017 and drastically on IJCAI-2015 dataset. This result demonstrates the moving averages of magnitudes benefits the optimization, which takes into account the variance among all gradient magnitudes over the training iterations.

Table 4.6: Ablation study of moving average of magnitude. Reprinted with permission from "MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks" by Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo and James Caverlee, Web Conference 2022.

Datasets	UserBehavior-2017			IJCAI-2015		
Metrics (%)	N@10	R@10	P@10	N@10	R@10	P@10
MetaBalance	0.990	1.550	0.339	0.974	1.164	0.509
−Moving Average	0.983	1.513	0.325	0.835	0.956	0.426

4.7 Summary

In many personalized recommendation scenarios, the target task can be improved via training auxiliary tasks alongside this target task on a multi-task network. In this chapter, we propose MetaBalance to adapt auxiliary tasks to better assist the target task from the perspective of gradient magnitude. Specifically, MetaBalance has three adapting strategies, such that it not only protects the target task from the dominance of auxiliary tasks but also avoids that one or more auxiliary tasks are ignored. Moreover, auxiliary gradients are balanced dynamically throughout the training and adaptively for each part of the network. Our experiments show that MetaBalance can be flexibly adapted to different scenarios and significantly outperforms previous methods. For example, our proposed method achieves a significant improvement of 8.34% in terms of NDCG@10 upon the strongest baseline on the two real-world datasets.

5. CONCLUSION AND FUTURE RESEARCH OPPORTUNITIES

The past decade has seen rapid growth in model capacity, which paves the way for machine learning algorithms to achieve dramatic success on many important domains like computer vision, natural language processing and personalized recommendations. However, human-labeled datasets are still scarce and hence these large models may overfit on low-resource tasks, resulting in poor performance. Recently, transferring useful knowledge from previous pre-training stages or related tasks has received more and more attention, which may alleviate the label scarcity problem. In this dissertation, we made a series of contributions to enable more effective and efficient knowledge transfer to boost the target task.

In particular, this dissertation include three contributions which towards tackling the three corresponding challenges of conducting knowledge transfer. First, transferring domain-specific knowledge from pre-training stages to large-scale language models remains under-explored, resulting in limited performance over many specialized domains like biomedical. To tackle this challenge, we propose a new pre-training procedure named disease knowledge infusion, which explicitly augment BERT-like language models with the disease knowledge to enhance health-related tasks such consumer health question answering, medical language inference and disease name recognition. Second, to alleviate the problem of task interference – training multiple tasks jointly on transformer-based models hinders the performance on individual tasks, we propose HyperPrompt to flexibly share knowledge between the co-trained tasks. HyperPrompt generates task-conditioned hyper-prompts that enables the model to better learn task-specific information. Moreover, the usage of HyperNetworks imbues the model with the flexibility of task-specific knowledge sharing among the co-trained tasks. Third, to overcome the challenge of negative transfer – transferring knowledge from the source can have a negative impact on the target learner, we introduce MetaBalance as a novel algorithm to intelligently and flexibly transfer the knowledge from auxiliary tasks to improve the target task. MetaBalance prioritizes the target task via carefully reducing the gradient of auxiliary tasks to prevent they from being so strong that they dominate the target

task or carefully enlarging the auxiliary gradients to enhance the knowledge transfer from them.

Our research would facilitate the exploration of pre-training and multi-task learning, as one step towards a more intelligent and flexible knowledge transfer to better assist the target task. Despite the efforts made in the thesis, there are still many challenges and open problems to be explored. With respect to future work, we are interested in investigating the following directions: o

- **Transfer domain-specific knowledge for language generation.** Previous work focus on infusing domain-specific knowledge into language models for natural language understanding such as sentence-level classification (e.g., medical language inference) and token-level classification (e.g., disease name recognition), while transferring domain-specific knowledge into language models for better specialized language generation is under-explored. The domain knowledge is vital to many important applications such as medical dialogue response generation [136] and task-oriented dialogue [93], which aims to assist the user in completing certain tasks in a specific domain such as restaurant booking and weather query.

- **Alleviate the catastrophic forgetting of knowledge.** The self-supervised pre-training over large-scale unlabeled corpus imbues language models the general language knowledge. However, when these models are further fine-tuned over downstream labeled datasets, the catastrophic forgetting might happen – parameters shift towards capturing the current task and lose the general-purpose knowledge learned from the pre-training [9], resulting in performance deterioration. In our work of HyperPrompt, we observe that injecting task-conditioned parameters (prompts or adapters) into the decoder of transformers achieves better performance than injecting them into the encoder. A possible reason is that decoder learns the task-specific knowledge while the encoder mainly learns the general language knowledge, which should not be interfered by the prompts. We could conduct more fine-grained study on knowledge retention through the multi-stage learning.

- **Study the theoretical foundation of negative transfer.** MetaBalance balances the gradient magnitudes of auxiliary tasks to alleviate the negative transfer. The extensive experiments

over real-world datasets empirically demonstrate the effectiveness of MetaBalance. In the future, more study can be conducted on investigating how the gradient magnitudes impact the knowledge transfer, which might give a theoretical explanation on why MetaBalance can significantly outperform the gradient-similarity based methods like GradSimilarity.

REFERENCES

[1] Asma Ben Abacha and Dina Demner-Fushman. A question-entailment approach to question answering. *BMC bioinformatics*, 20(1):511, 2019.

[2] Asma Ben Abacha, Eugene Agichtein, Yuval Pinter, and Dina Demner-Fushman. Overview of the medical question answering task at trec 2017 liveqa. In *TREC*, 2017.

[3] Asma Ben Abacha, Chaitanya Shivade, and Dina Demner-Fushman. Overview of the mediqa 2019 shared task on textual inference, question entailment and question answering. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 370–379, 2019.

[4] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.468. URL https://aclanthology.org/2021.emnlp-main.468.

[5] Hadeel S Alenezi and Maha H Faisal. Utilizing crowdsourcing and machine learning in education: Literature review. *Education and Information Technologies*, 25(4):2971–2986, 2020.

[6] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[7] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 3 (Industry Papers), pages 84–91, New Orleans - Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-3011. URL https://www.aclweb.org/anthology/N18-3011.

[8] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. 2021.

[9] Kristjan Arumae, Qing Sun, and Parminder Bhatia. An empirical investigation towards efficient multi-domain language model pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4854–4864, 2020.

[10] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114, 2016.

[11] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.

[12] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. 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 3606–3611, 2019.

[13] Asma Ben Abacha, Chaitanya Shivade, and Dina Demner-Fushman. Overview of the MEDIQA 2019 shared task on textual inference, question entailment and question answering. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 370–379, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5039. URL https://www.aclweb.org/anthology/W19-5039.

[14] Rohan Bhardwaj, Ankita R Nambiar, and Debojyoti Dutta. A study of machine learning in healthcare. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 236–241. IEEE, 2017.

[15] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[18] John S Brownstein, Clark C Freifeld, and Lawrence C Madoff. Digital disease detection-harnessing the web for public health surveillance. *The New England journal of medicine*, 360(21):2153, 2009.

[19] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*, 2017.

[20] Tuhin Chakrabarty, Debanjan Ghosh, Smaranda Muresan, and Nanyun Peng. $\mathbf{R^3}$: Reverse, retrieve, and rank for sarcasm generation with commonsense knowledge. *arXiv preprint arXiv:2004.13248*, 2020.

[21] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International*

Conference on Machine Learning, pages 794–803. PMLR, 2018.

[22] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *arXiv preprint arXiv:2010.06808*, 2020.

[23] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[24] Sherri De Coronado, Margaret W Haber, Nicholas Sioutos, Mark S Tuttle, Lawrence W Wright, et al. Nci thesaurus: using science-based terminology to integrate cancer research results. In *Medinfo*, pages 33–37, 2004.

[25] Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.

[26] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguis-*

tics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[30] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

[31] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[32] Kevin Donnelly. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006.

[33] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

[34] In Du Jeong, Moo In Park, Sung Eun Kim, Beom Jin Kim, Sang Wook Kim, Jie-Hyun Kim, Hye Young Sung, Tae-Hoon Oh, Yeon Soo Kim, The Korean Society of Neurogastroenterology, et al. The degree of disease knowledge in patients with gastroesophageal reflux disease: A multi-center prospective study in korea. *Journal of neurogastroenterology and motility*, 23(3):385, 2017.

[35] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[36] Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*, 2021.

[37] Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*, 2020.

[38] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both

the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[39] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740.

[40] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[41] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

[42] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.381. URL https://aclanthology.org/2021.acl-long.381.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[44] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[45] Yun He, Jianling Wang, Wei Niu, and James Caverlee. A hierarchical self-attentive model for recommending user-generated item lists. In *Proceedings of the 28th ACM International*

Conference on Information and Knowledge Management, pages 1481–1490. ACM, 2019.

[46] Yun He, Zhuoer Wang, Yin Zhang, Ruihong Huang, and James Caverlee. Parade: A new dataset for paraphrase identification requiring computer science domain knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7572–7582, 2020.

[47] Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. Metabalance: Improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. *arXiv preprint arXiv:2203.06801*, 2022.

[48] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/houlsby19a.html`.

[49] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[50] Rezarta Islamaj Dogan, G Craig Murray, Aurélie Névéol, and Zhiyong Lu. Understanding pubmed® user search behavior through log analysis. *Database*, 2009, 2009.

[51] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[52] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *TOIS*, 2002.

[53] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark.

Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[54] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[55] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. Reparameterizing convolutions for incremental multi-task learning without task interference. In *European Conference on Computer Vision*, pages 689–707. Springer, 2020.

[56] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, August 2021.

[57] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[58] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[59] Tassilo Klein and Moin Nabi. Contrastive self-supervised learning for commonsense reasoning. *arXiv preprint arXiv:2005.00669*, 2020.

[60] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

[61] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1eA7AEtvS`.

[62] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So,

and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[63] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

[64] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. Sensebert: Driving some sense into bert. *arXiv preprint arXiv:1908.05646*, 2019.

[65] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[66] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[67] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018.

[68] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4772–4783, 2019.

[69] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[70] Shikun Liu, Andrew Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. In *Advances in Neural Information Processing Systems*, pages 1679–1689, 2019.

[71] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recog-*

nition, pages 1871–1880, 2019.

[72] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. *arXiv preprint arXiv:1909.07606*, 2019.

[73] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[74] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL `https://aclanthology.org/P19-1441`.

[75] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, 2019.

[76] Yidan Liu, Min Xie, and Laks VS Lakshmanan. Recommending user generated item lists. In *Recsys*, 2014.

[77] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[78] Yichao Lu, Ruihai Dong, and Barry Smyth. Why i like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 4–12, 2018.

[79] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940, 2008.

[80] Liye Ma and Baohong Sun. Machine learning and ai in marketing–connecting computing power to human insights. *International Journal of Research in Marketing*, 37(3):481–504,

2020.

[81] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1137–1140, 2018.

[82] Itzik Malkiel and Lior Wolf. Mtadam: Automatic balancing of multiple training loss terms. *arXiv preprint arXiv:2006.14683*, 2020.

[83] Taylor Mordan, Nicolas Thome, Gilles Henaff, and Matthieu Cord. Revisiting multi-task learning with rock: a deep residual auxiliary block for visual detection. *Advances in neural information processing systems*, 31:1310–1322, 2018.

[84] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[85] Oxford-English-Dictionary. Definition of disease in englis, 2020. URL `https://www.lexico.com/en/definition/disease`.

[86] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5006. URL `https://www.aclweb.org/anthology/W19-5006`.

[87] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

[88] Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. 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 43–54, 2019.

[89] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*, 2020.

[90] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[91] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[92] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[93] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

[94] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*, 2020.

[95] Alexey Romanov and Chaitanya Shivade. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[96] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv*, pages arXiv–1706, 2017.

[97] Fahad Saleem, Mohamed Azmi Hassali, Asrul Akmal Shafie, Muhammad Atif, Noman ul Haq, and Hisham Aljadhey. Disease related knowledge and quality of life: a descriptive study focusing on hypertensive population in pakistan. *Southern med review*, 5(1):47, 2012.

[98] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. 2021.

[99] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

[100] Soumya Sharma, Bishal Santra, Abhik Jana, TYSS Santosh, Niloy Ganguly, and Pawan Goyal. Incorporating domain knowledge into medical nli using knowledge graphs. *arXiv preprint arXiv:1909.00160*, 2019.

[101] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[102] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. Mesh-tensorflow: Deep learning for supercomputers. *arXiv preprint arXiv:1811.02084*, 2018.

[103] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh.

AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346.

[104] Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

[105] Yi Tay, Zhe Zhao, Dara Bahri, Donald Metzler, and Da-Cheng Juan. Hypergrid: Efficient multi-task transformers with grid-wise decomposable hyper projections. *arXiv preprint arXiv:2007.05891*, 2020.

[106] Yi Tay, Zhe Zhao, Dara Bahri, Donald Metzler, and Da-Cheng Juan. Hypergrid transformers: Towards a single model for multiple tasks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=hiq1rHO8pNT.

[107] Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. Skep: Sentiment knowledge enhanced pre-training for sentiment analysis. *arXiv preprint arXiv:2005.05635*, 2020.

[108] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. 2012.

[109] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *Proc. Interspeech 2017*, pages 3532–3536, 2017.

[110] Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pages 4965–4974, 2018.

[111] Jorgen Urnes, Hermod Petersen, and Per G Farup. Disease knowledge after an educational

program in patients with gerd–a randomized controlled trial. *BMC health services research*, 8(1):236, 2008.

[112] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018.

[113] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *arXiv preprint arXiv:2004.13379*, 2020.

[114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[115] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[116] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[117] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[118] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 165–174, 2018.

[119] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[120] Xin Wang, Wenwu Zhu, and Chenghao Liu. Social recommendation with optimal limited at-

tention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1518–1527, 2019.

[121] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[122] Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, and Ed H Chi. Small towers make big differences. *arXiv preprint arXiv:2008.05808*, 2020.

[123] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[124] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database*, 2016, 2016.

[125] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. 2021.

[126] Sen Wu, Hongyang R. Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SylzhkBtDB`.

[127] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[128] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[129] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*, 2019.

[130] Frank F Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. Incorporating external knowledge through pre-training for natural language to code generation. *arXiv preprint arXiv:2004.09015*, 2020.

[131] Yichong Xu, Xiaodong Liu, Chunyuan Li, Hoifung Poon, and Jianfeng Gao. Doubletransfer at mediqa 2019: Multi-source transfer learning for natural language understanding in the medical domain. *arXiv preprint arXiv:1906.04382*, 2019.

[132] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[133] Da Yin, Tao Meng, and Kai-Wei Chang. Sentibert: A transferable transformer-based architecture for compositional sentiment semantics. *arXiv preprint arXiv:2005.04114*, 2020.

[134] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[135] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[136] Guangtao Zeng, Wenmian Yang, Zeqian Ju, Yue Yang, Sicheng Wang, Ruisi Zhang, Meng Zhou, Jiaqi Zeng, Xiangyu Dong, Ruoyu Zhang, et al. Meddialog: A large-scale medical dialogue dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9241–9250, 2020.

[137] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. Collaborative multi-level embedding learning from reviews for rating prediction. In *IJCAI*, volume 16, pages 2986–2992, 2016.

[138] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer.

arXiv preprint arXiv:2009.00909, 2020.

[139] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[140] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, 2019.

[141] Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. How does nlp benefit legal system: A summary of legal artificial intelligence. *arXiv preprint arXiv:2004.12158*, 2020.