OPTIMIZING KNOWLEDGE TRANSFER IN CONTINUAL AND MULTI-TASK LEARNING ENVIRONMENTS

SHI GUANGYUAN

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Computing

Optimizing Knowledge Transfer in Continual and Multi-Task Learning Environments

SHI Guangyuan

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
August 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: SHI Guangyuan

Abstract

Optimizing knowledge transfer is a key challenge in machine learning, especially in dynamic environments where tasks and data continually evolve. Conventional machine learning methods, generally rely on the premise that the feature space and data distribution remain consistent between the training and testing phases. In reality, this condition is rarely met, as real-world data often exhibits substantial variability. This limitation reduces the usability and effectiveness of models, particularly when training data is insufficient, tasks have diverse distributions, or environments change, necessitating model retraining. In such settings, models must handle multiple tasks simultaneously while managing diverse and potentially conflicting objectives. Moreover, it is essential for models to learn new tasks while retaining existing knowledge and to swiftly adjust to new situations or tasks with minimal retraining effort. This paper examines strategies for optimizing knowledge transfer in continual learning (CL) and multi-task learning (MTL) to improve their performance in practical applications.

Continual learning (CL) enables models to learn from a series of tasks while retaining knowledge from earlier tasks, thus avoiding catastrophic forgetting, where new learning negatively impacts previously acquired knowledge. We propose a novel approach that guides models to converge to flat local minima during initial training, requiring minimal adjustments when adapting to new tasks. This strategy reduces the likelihood of forgetting and enhances model robustness in dynamic environments, making

it particularly effective for applications requiring continual adaptation to new data and tasks.

In multi-task learning (MTL), the challenge is to transfer knowledge across different tasks without causing negative transfer, where learning one task adversely affects performance on others. Negative transfer often arises from conflicting gradients during model updates, where the update direction is dominated by tasks with larger gradient magnitudes, hindering effective learning of other tasks. To mitigate this, we introduce a method that identifies layers with severe gradient conflicts and switches them from shared to task-specific configurations. This approach prevents gradient conflicts in shared layers, ensuring balanced learning and improving overall model performance and generalization across tasks.

Additionally, considering the increasing size of pretrained base models and the rising costs associated with knowledge transfer, we introduce a parameter-efficient fine-tuning (PEFT) algorithm. This algorithm aims to optimize the adaptability of large language models (LLMs) by selectively fine-tuning only the most critical layers. By learning binary masks for each low-rank weight matrix used in LoRA—determining whether a layer needs a LoRA adapter, where a mask value of 0 indicates that no LoRA adapter is required and thus no change to the model parameters—our approach significantly reduces memory overhead and computational costs while avoiding overfitting. This makes transfer learning more efficient and feasible in resource-constrained environments.

In summary, this thesis explores a set of complementary methods aimed at improving knowledge transfer in machine learning under practical constraints. By addressing critical challenges such as continual adaptation, task interference, and computational efficiency, the proposed approaches contribute to enhancing the robustness and practicality of transfer learning in real-world settings. These methods—focused on reducing forgetting in continual learning, mitigating gradient conflicts in multi-task learning, and improving parameter efficiency in fine-tuning large models—offer

targeted solutions to common limitations in dynamic and resource-constrained environments. Experimental results support their effectiveness, showing improvements in model stability, generalization, and adaptability. Overall, this work offers both practical insights and methodological contributions that can inform future research and applications in scalable, efficient machine learning. The results have been published or submitted in various top AI conferences.

Publications Arising from the Thesis

1. Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Liming Zhan, Xiao-Ming Wu. "Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima", in *Conference on Neural Information Processing Systems (NeurIPS Spotlight) (2021).*

2. Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, Xiao-Ming Wu. "Recon: Reducing conflicting gradients from the root for multi-task learning", in *The Eleventh International Conference on Learning Representations (ICLR) (2023).*

3. Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, Xiao-Ming Wu. "IFILA: Identifying and Fine-tuning the Important Layers for LLM Alignment", submitted to *The Second Conference on Language Modeling (COLM) (2025)*

Acknowledgments

I am profoundly grateful to my supervisor, Prof. Xiao-Ming Wu, for her exceptional guidance and unwavering support throughout my PhD studies. Her deep expertise, thoughtful insights, and commitment to academic excellence have greatly enriched my research and personal development. Her passion for discovery and innovation has been a constant source of inspiration, and I feel privileged to have had the opportunity to learn from such a distinguished mentor and researcher. Her encouragement, patience, and ability to challenge me to think critically have significantly shaped my academic journey.

I also want to extend my sincere thanks to my lab mates, friends, and collaborators who have played a vital role in my academic journey. Their invaluable input, constructive feedback, and unwavering encouragement have been indispensable to my growth as a scholar. A special thank you goes to Wenlong Zhang, Qimai Li, Liming Zhan, Jiaxin Chen, Haode Zhang, Cong Wang, Bo Liu, Lu Fan, Haowen Liang, Zexin Lu, Xiaoyu Dong, Yujie Feng, Xiangyu Zhao, Zaoyu Chen, Yuankai Luo, Nuo Chen, and Yongfeng Zhong for their friendship, stimulating discussions, and the collaborative spirit they brought to our work together. The camaraderie we shared and the intellectual curiosity they sparked have made my time here both productive and enjoyable.

I would also like to acknowledge the invaluable support of my academic community, whose rich dialogues and diverse perspectives have broadened my horizons and en-

hanced my research. Their encouragement and shared passion for learning have made my academic journey truly enriching.

Lastly, I am deeply indebted to my family for their endless love, patience, and support. Their faith in my abilities has been a constant source of strength and has helped me persevere through the challenges of this journey. Their encouragement and sacrifices have been fundamental to my success, and I am forever grateful. Without their belief in me and their unwavering support, this achievement would not have been possible.

Thank you all for being an integral part of this journey and for helping me reach this milestone. Your contributions have been invaluable, and I am sincerely thankful for each of you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Illustration of Knowledge Transfer. Knowledge transfer [301, 183] enhances model performance by leveraging knowledge from a source domain to improve learning in a related target domain, particularly when the target domain suffers from data scarcity or distributional differences. This concept is inspired in part by theories of transfer and generalization in educational psychology, where experience from one context is applied to another. For example, someone familiar with playing the piano may learn to play the electronic keyboard more easily due to shared foundational skills. However, if the source and target domains differ substantially, negative transfer [255] may occur, where the transferred knowledge adversely affects performance. A typical case is learning Spanish and French—despite being related Romance languages, interference between similar grammatical or lexical structures can lead to confusion.

Importance of Knowledge Transfer. Knowledge transfer plays a pivotal role in bridging the gap between traditional machine learning methods and real-world applications. Conventional machine learning assumes that training and test data share the same distribution, an assumption that rarely holds in practice. Labeled data collection is often time-consuming and costly, and data distributions may shift over time, between devices, or across environments. Such changes can significantly

degrade model performance. In these cases, transferring previously acquired knowledge becomes essential, allowing models to adapt more quickly and effectively to new settings.

Applications of Knowledge Transfer. Knowledge transfer has demonstrated remarkable utility across various domains. In natural language processing (NLP), pretrained language models such as BERT [47] significantly improve performance in tasks like sentiment analysis [12] and text generation [293, 132, 212]. Large language models (LLMs), including GPT-4 [1] and Llama 3 [53], trained on vast corpora, achieve state-of-the-art results across a wide range of tasks. These capabilities can be further enhanced via fine-tuning and alignment [238, 118, 106], enabling applications such as code generation [252, 175, 254]. In computer vision, transfer learning is common practice, where CNNs pretrained on large datasets like ImageNet [44] are adapted for tasks such as medical image analysis [139, 219, 235]. Similarly, diffusion models [207, 177, 176, 88], trained on extensive image datasets, can be adapted through knowledge transfer to generate images in domain-specific styles. In summary, knowledge transfer provides a practical and effective approach to overcome limitations in data availability and distributional shifts, with broad applicability in NLP [211], computer vision [300, 283], intelligent transportation [33], and biomedicine [187].

Challenges in Knowledge Transfer. Despite its advantages, knowledge transfer faces three major challenges: **(1) Model Rigidity.** Traditional transfer methods often fail to adapt in dynamic environments where tasks and data evolve over time. In real-world applications, models are frequently exposed to novel tasks or sequential data updates. Continual Learning (CL) [165, 37] addresses this by enabling models to learn from new data while preserving previously acquired knowledge. **(2) Data Heterogeneity.** Tasks often exhibit diverse and non-overlapping data distributions, increasing the risk of negative transfer, where learning one task impairs performance on another. This challenge arises because traditional transfer techniques struggle to reconcile cross-task discrepancies. Multi-task Learning (MTL) [21] mitigates this is-

sue by promoting the sharing of beneficial information across related tasks, thereby reducing interference and enhancing generalization. **(3) Resource Intensiveness.** Fine-tuning large-scale models, such as Llama 3 or GPT-4, demands substantial computational resources and data. This resource burden can hinder the practical deployment of transfer learning. Parameter-efficient fine-tuning (PEFT) offers a solution by modifying only a small subset of parameters or introducing lightweight trainable components, making the transfer process more efficient and scalable.

Motivation and Thesis Scope. While knowledge transfer has achieved impressive results across diverse domains, its practical deployment remains hindered by challenges related to **adaptability**, **data heterogeneity**, and **computational efficiency**. These limitations highlight the need for more flexible, robust, and scalable approaches that can accommodate evolving data landscapes and resource constraints. This thesis aims to address these challenges by exploring novel methodologies that enhance the effectiveness and efficiency of knowledge transfer, particularly in dynamic and low-resource environments. Through a comprehensive investigation of continual learning, multi-task learning, and parameter-efficient fine-tuning, we seek to advance the theoretical understanding and practical utility of knowledge transfer, ultimately contributing to the development of more generalizable and adaptable AI systems.

1.1 Continual Learning

Continual learning (CL) represents a pivotal advancement in enhancing transfer learning (TL) methodologies, particularly when addressing the inherent limitations of adapting models to dynamic environments. Traditional transfer learning is adept at utilizing knowledge from one task to facilitate learning in another; however, it typically assumes static conditions for both the source and target tasks. This assumption rarely holds true in practical scenarios where both conditions and data are subject to continuous change [237]. To overcome these challenges, CL

intervenes to enable models to learn adaptively from new data while retaining previously acquired knowledge, transforming transfer learning models to operate robustly across a range of evolving tasks. This integration significantly elevates their practical utility and effectiveness.

One of the core challenges within continual learning is the phenomenon known as *catastrophic forgetting* [69, 116, 165], which occurs when training on new tasks leads to a substantial deterioration in the model's ability to perform previously learned tasks. This issue primarily arises due to the inability to access data from older tasks while optimizing for new ones, leading models to stray from optimal performance on earlier tasks as more tasks are introduced. Traditional approaches to mitigating this effect include maintaining a subset of data from old tasks [202, 91, 22], applying stringent regularization [164, 116, 286] to restrict model adaptability, or merely fine-tuning parts of the model [81]. These methods, however, often impose restrictions that could hinder model flexibility and adaptation.

In contrast to these existing strategies, our thesis proposes an innovative approach focused on preemptively addressing the problem of forgetting during the initial training phase of the base model. Specifically, we advocate for guiding the model towards converging on a flat local minimum during the learning of existing tasks. This strategy posits that subsequent learning of new tasks should involve only minimal deviations within this flat local minimum area, thereby preserving the model's ability to recall older tasks effectively. This approach not only promises to enhance the robustness of transfer learning models in dynamic settings but also offers a scalable solution to the challenge of learning new tasks with limited data, a common scenario in practical applications known as few-shot continual learning.

This section delineates the significance of integrating continual learning techniques into transfer learning frameworks, focusing on overcoming traditional limitations while innovating to reduce the impacts of catastrophic forgetting. By shifting towards strategies that anticipate and mitigate forgetting at earlier stages of model

training, We create a basis for developing more durable and versatile machine learning applications that can manage the challenges of real-world data environments.

1.2 Multi-task Learning

Multi-task learning (MTL) benefits in transfer learning. Multi-task Learning (MTL) significantly extends the scope and effectiveness of Transfer Learning (TL) by allowing models to concurrently learn from multiple related tasks. This synergistic learning strategy leverages shared common features and representations, which not only improves the model's ability to generalize across tasks but also enhances the overall efficiency of the learning process. Within the transfer learning framework, MTL provides a robust base model that demonstrates inherent versatility and superior performance across diverse tasks. This integrated approach optimizes adaptability and resource utilization, making MTL a pivotal component in advancing the capabilities of transfer learning systems.

A prominent challenge in MTL, similar to transfer learning, is negative transfer [209]. Negative transfer occurs when the learning process for one task inadvertently hampers the performance of another task, despite their relatedness. The root cause of this phenomenon is the presence of conflicting gradients [279, 140]. Conflicting gradients arise when tasks being learned simultaneously influence the model's updates in incompatible ways. Specifically, if tasks have significantly different gradient directions, the task with the larger gradient magnitude can dominate the update process, causing a decline in performance on other tasks. Conflicting gradients are characterized by large angular differences between the gradient vectors of different tasks during the update phase. This misalignment can severely disrupt the learning process, as the dominant task's gradients skew the model's learning trajectory, potentially degrading the outcomes of less dominant tasks.

Previous methodologies to mitigate these challenges have included modifying the loss weights allocated to each task—effectively altering the learning rate for different tasks [35, 114, 146, 142]. Another approach has involved modifying the direction of gradient updates to accommodate multiple tasks more equitably [217, 279, 140, 104]. These methods strive to balance the influence of each task on the model's updates, ensuring that no single task's requirements overshadow the others.

In this thesis, we introduce a novel strategy aimed at fundamentally eliminating conflicting gradients from the onset. Our approach involves identifying the layers within the model where conflicting gradients are most severe—typically layers where the gradient angles are particularly large. We then transform these layers from shared to task-specific configurations, whereby each task is allocated its own independent parameters for these critical layers. This restructuring ensures that the remaining shared layers are free from gradient conflicts, thus preserving the integrity and effectiveness of the multi-task learning process. This targeted adjustment not only addresses the root cause of negative transfer but also enhances the overall performance and adaptability of the model across varied tasks.

This section highlights the transformative impact of integrating Multi-task Learning (MTL) into Transfer Learning (TL) frameworks, emphasizing the resolution of inherent challenges such as negative transfer. By adopting innovative strategies that preemptively address conflicting gradients, our approach reshapes the landscape of multi-task learning. Specifically, by modifying critical model layers to become task-specific, we significantly reduce inter-task interference, enhancing the model's ability to perform optimally across diverse tasks. This targeted intervention not only mitigates the risks associated with negative transfer but also bolsters the overall performance and adaptability of transfer learning systems. Consequently, we establish a robust foundation for developing more versatile and effective machine learning applications, tailored to thrive in the complexities of varied and dynamic data environments.

1.3 Parameter Efficient Fine-Tuning

The rapid advancement of large language models (LLMs) has underscored the significance of Parameter Efficient Fine-Tuning (PEFT) in the realm of Transfer Learning (TL). As these models continue to dominate various AI applications, the traditional approach of extensively fine-tuning entire models becomes increasingly untenable due to the prohibitive costs in computational resources and time. PEFT addresses these challenges by enabling targeted adjustments through methods such as prompt engineering [126] and other parameter-efficient techniques [291, 281]. These strategies allow for significant behavioral modifications of the model with minimal adjustments to the underlying parameters, conserving resources while also expediting the adaptation of LLMs to diverse tasks. This enhancement in adaptability across different domains is achieved without the need for extensive retraining, thereby making transfer learning more accessible and sustainable, and paving the way for rapid deployment and innovation in AI-driven applications.

Currently, the most prominent PEFT algorithms are adapter-based methods [], among which the Low-rank Adaptation (LoRA) [94] stands out. LoRA enhances model flexibility by introducing adapters—composed of two low-rank matrices—at each layer of the model. This setup predicts the change in model parameters, enabling fine-tuning of a minimal number of parameters while maintaining performance on par with full fine-tuning. However, a gap in existing methodologies is the lack of research on determining which specific layers require adapters. Identifying layers that are most relevant to the task could allow for fine-tuning of only those essential layers or adding adapters selectively, which not only reduces the memory overhead during tuning but also helps avoid overfitting, thus potentially enhancing model performance.

To address this gap, we propose a novel algorithm that involves learning a binary mask for each incremental weight matrix used in the LoRA algorithm. This mask serves as an indicator of the significance of each layer relative to specific tasks. If

the mask value is 0, it implies that the layer does not require a LoRA adapter; if the value is 1, it indicates that the layer is crucial and should be fine-tuned. This selective approach not only optimizes the fine-tuning process by focusing on key layers but also ensures that each adaptation precisely targets the task's requirements.

In summary, this section elucidates the critical role of PEFT in enhancing the practicality and efficacy of transfer learning, especially in the context of deploying LLMs. By focusing on adapter-based methods like LoRA and introducing a targeted approach to identify and adapt task-relevant layers, we substantially boost the efficiency and effectiveness of models in handling various tasks. This refined methodology not only mitigates the resource-intensive demands of traditional fine-tuning but also aligns with the evolving needs of AI applications, ensuring that large-scale models remain both versatile and effective in real-world scenarios.

1.4 Contributions

This thesis makes significant contributions to the field of transfer learning by addressing key challenges that limit its practical application and effectiveness across diverse tasks and domains.

Contribution 1: Mitigating Negative Transfer and Enhancing Generalization through Multi-task Learning (MTL). Negative transfer is a major obstacle in transfer learning, where knowledge transfer between tasks can inadvertently degrade performance. We tackle this issue by introducing a novel MTL strategy that reduces negative transfer and enhances the model's generalization ability. By identifying and resolving conflicting gradients during optimization, our approach helps in training a more robust base model capable of handling multiple tasks simultaneously. This improvement not only enhances the overall effectiveness of transfer learning but also ensures that models are better prepared for subsequent transfer tasks.

Contribution 2: Enabling Transfer Learning in Dynamic Environments through Continual Learning (CL). Traditional transfer learning often struggles in dynamic environments where tasks and data evolve over time. To address this, we propose a continual learning methodology that enhances the adaptability of transfer learning models. By guiding the model to converge on a flat local minimum during the initial training, our approach minimizes catastrophic forgetting, allowing the model to retain previously learned knowledge while adapting to new tasks. This contribution significantly improves the practical applicability of transfer learning in real-world, dynamic settings.

Contribution 3: Improving Fine-Tuning Efficiency in Large Models with Parameter Efficient Fine-Tuning (PEFT). As models grow in size, the resource demands for fine-tuning in transfer learning become increasingly prohibitive. We address this by introducing an optimized PEFT approach that uses a binary mask within the LoRA algorithm to selectively fine-tune only the most critical layers of the model. This targeted fine-tuning reduces computational overhead and avoids overfitting, thereby enhancing the efficiency of transfer learning even when dealing with large-scale models like LLMs.

These contributions collectively advance the field of transfer learning by improving model generalization, enabling effective adaptation in dynamic environments, and optimizing fine-tuning processes for large models. These innovations make transfer learning more robust, scalable, and applicable across a wider range of tasks and scenarios.

Thesis organization. In Chapter 2, we provide a comprehensive overview of the foundational concepts relevant to this thesis, including transfer learning, multi-task learning, continual learning, and parameter-efficient fine-tuning. Chapter 3 introduces our method for mitigating negative transfer in multi-task learning by addressing conflicting gradients at the root, thereby tackling the first challenge in transfer learning. In Chapter 4, we address the second challenge by implementing a few-shot continual

learning approach that prevents catastrophic forgetting during the training of the base model. Chapter 5 addresses the third challenge by optimizing transfer learning efficiency through selective fine-tuning of key layers in large language models (LLMs). Finally, in chapter 6, we conclude this thesis and explore several potential directions for future research.

Chapter 2

Background and Related Works

In this chapter, we begin by introducing the concept of knowledge transfer in machine learning, followed by an overview of transfer learning. We then delve into the context of multi-task learning, providing a review of related work and its evolution. Next, we explore the scenario of continual learning, providing a comprehensive overview and pointing out the importance of continual learning. Finally, we discuss the background of parameter-efficient fine-tuning (PEFT). Through a review of relevant studies, we demonstrate how PEFT significantly enhances the efficiency of knowledge transfer while reducing resource consumption.

2.1 Knowledge Transfer

Knowledge transfer in machine learning involves the process of applying the knowledge, patterns, and representations that a model has acquired from one or more tasks—often situated in different domains or contexts—to improve the performance of a new, but related, task. To formally describe transfer learning, we first need to define the concepts of Domain and Task.

Definition 2.1 (Domian)**.** *A domain $\mathcal{D}$ is defined by two components:*

$$\mathcal{D} = \{\mathcal{X}, P(X)\}, \tag{2.1}$$

where:

- *$\mathcal{X}$ is the feature space, representing the set of all possible input instances (e.g., vectors, images, text).*
- *$P(X)$ represents the probability distribution over the feature space $\mathcal{X}$, and symbol X denotes an instance set, i.e., $X = \{\mathbf{x}_i \in \mathcal{X}\}_{i=1}^{n}$. A domain characterizes the environment or source from which data is drawn.*

Definition 2.2 (Task)**.** *A task $\mathcal{T}$ is defined by two components:*

$$\mathcal{T} = \{\mathcal{Y}, P(X|Y), f(\cdot)\}, \tag{2.2}$$

where:

- *$\mathcal{Y}$ denotes the label space, which is the set of all possible outcomes or target values. For classification tasks, $\mathcal{Y}$ represents the set of possible classes, while for regression tasks, it represents the range of possible output values.*
- *$P(X|Y)$ is the conditional distribution mapping inputs to outputs.*
- *$f : \mathcal{X} \to \mathcal{Y}$ is an objective predictive function (also called the decision function or hypothesis) that maps any input data point $x \in \mathcal{X}$ to a corresponding label $y \in \mathcal{Y}$. **This function is typically learned from a set of labeled data points** $\{(x_i, y_i)\}_{i=1}^{n}$.*

In summary, a domain $\mathcal{D}$ outlines the type of data (defined by $\mathcal{X}$) and its statistical properties (defined by $P(X)$ relevant to a particular machine learning problem. A task $\mathcal{T}$ describes the target to be predicted (through $\mathcal{Y}$) and the method for making

predictions (through $f(\cdot)$) based on the associated domain $\mathcal{D}$. The task is not merely a set—it aims to learn $f(\cdot)$.

After establishing the concepts of Domain $\mathcal{D}$ and Task $\mathcal{T}$, we can proceed to formally introduce the concept of knowledge transfer. Knowledge transfer is a fundamental technique aimed at improving a model's performance on a target task by drawing on insights learned from a related source task. The formal definition is as follows:

Definition 2.3 (Knowledge Transfer [257])**.** *Given a source domain $\mathcal{D}_S$ with its corresponding task $\mathcal{T}_S = \{\mathcal{Y}_S, f_S(\cdot)\}$ and a target domain $\mathcal{D}_T$ with its corresponding task $\mathcal{T}_T = \{\mathcal{Y}_T, f_T(\cdot)\}$, knowledge transfer aims to improve the learning of the predictive function $f_T(\cdot)$ in the target task $\mathcal{T}_T$ by utilizing the knowledge derived from the source domain $\mathcal{D}_S$ and task $\mathcal{T}_S$.*

Formally, knowledge transfer can be defined as the process of optimizing the function $f_T(\cdot)$ by leveraging:

$$\mathcal{K} = \{\mathcal{D}_S, \mathcal{T}_S, f_S(\cdot)\}, \tag{2.3}$$

where $\mathcal{K}$ represents the knowledge acquired from the source domain and task. The goal is to minimize the prediction error in the target domain $\mathcal{D}_T$ under the task $\mathcal{T}_T$:

$$\min_{f_T} \mathbb{E}_{(x_T, y_T) \sim P_T(X_T, Y_T)} \left[\mathcal{L} \left(f_T(x_T), y_T \right) \right], \tag{2.4}$$

where $\mathcal{L}$ denotes the loss function associated with the task $\mathcal{T}_T$, and $P_T(X_T, Y_T)$ represents the joint distribution of the input-output pairs in the target domain.

This definition formalizes the concept of knowledge transfer, highlighting how information from a source domain and task can be leveraged to improve performance on a related target task. The central objective is to reduce the prediction error in the target domain by leveraging insights drawn from the source domain.

2.2 Continual Learning

Continual learning [165, 37] focuses on enabling models to learn from new data while retaining previously acquired knowledge, effectively mitigating the issue of catastrophic forgetting [69, 116, 165] and the need for adaptability in dynamic environments. The formal definition is given below:

Definition 2.4 (Continual Learning)**.** *Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ denote a sequence of N tasks, where each task $\mathcal{T}_t = (\mathcal{D}_t^{train}, \mathcal{D}_t^{test})$ is associated with a training set $\mathcal{D}_t^{train}$ and a test set $\mathcal{D}_t^{test}$ sampled from a data distribution $\mathcal{P}_t(X, Y)$. A continual learning algorithm aims to learn a single model f_θ parameterized by θ, by processing each task $\mathcal{T}_t$ sequentially, without revisiting the full data from previous tasks.*

The objective of continual learning is to minimize the average risk across all tasks:

$$\min_{\theta} \frac{1}{N} \sum_{t=1}^{N} \mathbb{E}_{(x,y)\sim\mathcal{P}_t} \left[\ell(f_\theta(x), y)\right]$$

under the constraint that each task $\mathcal{T}_t$ is observed only once, and the model must retain knowledge from previous tasks while acquiring new knowledge from the current task.

A continual learning algorithm must therefore address the stability-plasticity trade-off, mitigating catastrophic forgetting of previous tasks while maintaining plasticity to learn new tasks effectively.

To address these challenges, a wide range of methods have been proposed, each employing different mechanisms. These methods can be broadly classified into four categories: regularization-based, optimization-based, representation-based, and architecture-based approaches. Furthermore, applying continual learning to large language models (LLMs) introduces additional complexities and unique challenges. In the following section, we provide an overview of these strategies and discuss their respective contributions to the field.

Regularization-Based Approaches: These methods aim to mitigate catastrophic

forgetting by imposing strong regularization on network parameters [147, 124, 123, 116]. The core idea is to restrict the extent or range of parameter updates so that the model does not deviate significantly from the low-loss regions of previous tasks [205, 216, 286, 3, 26]. A well-known example is Elastic Weight Consolidation (EWC) [116], which utilizes the Fisher information matrix to estimate the importance of each parameter to previous tasks and penalizes changes to these parameters during new task learning. Additionally, there are function regularization approaches that leverage knowledge distillation (KD) [70] to continue reinforcing previously acquired knowledge [133, 20]. These methods often require retaining a subset of exemplars from old tasks, which may include both previously seen [23, 50, 91, 202] and unseen samples [48, 100, 120, 244]. However, the limited number of exemplars can lead to a distribution that diverges from the true data distribution.

Optimization-Based Approaches: Unlike regularization methods that impose constraints on the model through penalties, optimization-based approaches directly modify the gradient updates when learning new tasks [155, 285, 105]. Some techniques [155, 27, 236, 204] store exemplars of old tasks and ensure that the gradient update direction for new tasks remains closely aligned with that of the old tasks, thus preventing forgetting. Other methods do not store any exemplars; for example, OWM [285] and AOP [78] record the input space of old tasks, while OGD [58] preserves the gradient directions of old tasks and ensures that parameter updates are orthogonal to either the input space or the previous gradient directions. Several other methods follow similar principles [214, 136, 112, 141], aiming to limit the direction of gradient updates to avoid conflicts with updates from prior tasks. Additionally, there are meta-learning-based approaches [13, 105, 198, 99] that train the model to learn how to learn, rather than learning task-specific knowledge directly, thereby reducing the risk of forgetting.

Representation-Based Approaches: These approaches primarily focus on two strategies. The first strategy uses self-supervised learning to prevent catastrophic

forgetting [65, 160, 189], as representations learned through self-supervised training tend to be more robust against forgetting. For instance, LUMP [160] improves self-supervised continual learning by interpolating between instances of new and old tasks. The second strategy involves using pre-training to obtain a strong base model [166, 221, 261], which can then be fine-tuned for downstream tasks. Models pre-trained on large datasets not only enhance learning of new tasks in a continual learning setting but also effectively reduce catastrophic forgetting [181, 199, 166]. When models are sufficiently large and trained on ample data, they tend to retain old knowledge more effectively.

Architecture-Based Approaches: Sharing the same network parameters across all tasks in continual learning—i.e., optimizing a single set of model parameters—is a fundamental cause of catastrophic forgetting. Therefore, many studies suggest that continual learning should involve learning separate parameters for different tasks []. Fixed architecture methods [162, 218, 111, 267], for instance, activate different neurons for different tasks to prevent interference between tasks and avoid forgetting. However, fixed network structures can limit scalability, prompting the development of dynamic network architectures [276, 98, 180, 192]. Another approach is to extract task-relevant components from the model [56, 154, 222, 99], resulting in a set of shared parameters and task-specific parameters, similar to multi-task learning architectures. These task-specific parameters are expandable and updated whenever a new task is encountered. Some methods [213, 2, 200, 253], instead of defining explicit task-specific parameters, adopt an incremental approach by adding submodules to the network.

Continual Learning for LLMs: Currently, there are three primary directions in continual learning for large language models (LLMs). The first is continual pre-training, which is considered the most critical. This approach aims to continually update the model with the latest information [103, 102] or adapt it to specific subdomains [79, 268]. The second is continual instruction tuning, which essentially involves

transferring LLMs to downstream tasks in a continual manner, similar to the traditional paradigm of pretrained models followed by continual learning. These downstream tasks may include various tasks [119, 260], domains [38, 251], and tools [193, 8]. The third direction is continual alignment, which addresses the evolving nature of societal values and the diverse preferences of different demographic groups, ensuring that the model continually learns different alignment preferences based on these changes [271, 195].

2.3 Multi-Task Learning

The goal of Multi-task learning (MTL) [21] is to utilize shared knowledge across tasks to improve performance on all tasks simultaneously. The formal definition given as follows:

Definition 2.5 (Multi-Task Learning)**.** *Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ denote a set of N learning tasks. Each task $\mathcal{T}_i$ is associated with its own dataset $\mathcal{D}_i = \{(x_i^{(j)}, y_i^{(j)})\}_{j=1}^{n_i}$, where $x_i^{(j)} \in \mathcal{X}_i$ is the input and $y_i^{(j)} \in \mathcal{Y}_i$ is the corresponding label. Multi-task learning aims to jointly learn a set of functions $\{f_{\theta_i}\}_{i=1}^N$, typically sharing parameters θ_{shared}, such that performance across all tasks is optimized:*

$$\min_{\theta_{sh}, \{\theta_i\}_{i=1}^N} \sum_{i=1}^{N} \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\mathcal{L}_i(f_{\theta_{sh},\theta_i}(x), y)],$$

where $\mathcal{L}_i$ is the loss function specific to task $\mathcal{T}_i$. The goal is to leverage commonalities across tasks to improve generalization performance on each individual task.

A significant challenge in MTL is avoiding negative transfer, where learning multiple tasks together leads to poorer performance than learning them independently. To address this, various methods have been proposed, focusing on network architectures, optimization strategies, and task grouping techniques. In this section, we review related works in these areas, highlighting their contributions and effectiveness in mitigating negative transfer and enhancing multi-task learning.

MTL Architectures: To prevent negative transfer between different tasks, several task-specific network architectures have been proposed for multi-task learning, including those in the fields of computer vision [146, 170, 159] and natural language processing (NLP)[4, 7, 14, 24, 25]. These networks can generally be categorized based on the parameter-sharing strategy. One common approach is hard parameter sharing, where the network uses a shared feature extractor with separate decoders[114, 240, 174, 217, 146] or encoders [264, 247] for each task. Another approach is soft parameter sharing, where each task maintains its own task-specific parameters [170, 210, 66], which make up the majority of the model, and interactions between tasks occur through feature fusion and exchange. Additionally, there is a category known as task routing [229], where different tasks learn distinct model combination paths, or the network learns how to branch dynamically for each task [18].

Optimization Strategies: Optimization-based methods in MTL primarily fall into two categories. The first category involves directly modifying the weights of the loss function based on the signals extracted from each task [35, 114, 76, 142, 146]. The second category focuses on directly adjusting the size and direction of the model's gradient updates to avoid conflicting gradients [217]. For instance, methods like PCGrad [279], CAGrad [140], RotoGrad [104], and FairGrad [161] modify gradients to reduce the impact of conflicting gradients. PCGrad employs gradient projection, CAGrad formulates the problem as an optimization task and solves for the optimal solution, RotoGrad rotates the shared feature space, and FairGrad transforms multi-task optimization into an optimization problem under different fairness constraints. Unlike these methods, which directly modify the magnitude and direction of gradients, GradDrop compares the sign of each gradient component across tasks, and if there is a conflict (one positive and one negative), it drops the conflicting gradient component.

Task Grouping: Unlike network architecture modification and optimization, task grouping [191, 225, 228, 62] aims to find an optimal set of task groups, where the tasks within each group do not overlap and are trained together to mutually benefit

from one another, thus minimizing negative transfer. This strategy seeks to maximize the average performance of each task. Taskonomy [283] provides a detailed grouping of tasks related to segmentation in computer vision, revealing which tasks should be trained together and which should not. TAG [62] uses gradient conflicts to guide efficient task grouping, ensuring that tasks grouped together enhance each other's learning without detrimental interference.

These various approaches highlight the diverse strategies available in multi-task learning to improve model performance while minimizing the risks of negative transfer.

2.4 Parameter Efficient Fine-Tuning For LLMs

Adapter-based Fine-tuning. This approach involves inserting additional adapters either within or alongside the main model layers. An adapter typically comprises three components: a down-projection matrix, an activation function, and an up-projection matrix. The initial designs of adapters were sequential [92, 188], which could hinder the model's inference speed. To address this issue, parallel adapters were subsequently introduced [125, 57]. In addition to parallel adapters, CoDA [125] selects the top-K most important tokens to pass through both the backbone network and the adapters, while the less important tokens are processed solely by the adapters. This strategy further enhances inference efficiency. Additionally, several adapter variants have been proposed to leverage multi-task learning [41, 84], making them more suitable for multi-task scenarios.

Prompt-based Fine-tuning. This method fine-tunes the model by directly modifying its embeddings through the addition of new prompt embeddings. Prefix-tuning [131] and related methods [148, 126] fine-tune models by adding learnable parameters to the key and value embeddings within transformers or initial word embedding. Prompt-based fine-tuning has been widely applied to downstream tasks [259,

230], though the training process can often be unstable. PTP [31] thoroughly investigates this issue, revealing that minor changes in input can significantly impact the loss function. To mitigate this, it introduces regularizers designed to smooth the training process, ensuring that the loss function remains as stable as possible.

Selective Fine-tuning. This category of methods aims to improve fine-tuning efficiency by selectively tuning only the most critical model parameters. For instance, BitFit [282] achieves strong performance by fine-tuning only the bias parameters. PaFi [134] focuses on tuning model parameters with smaller absolute magnitudes. Child-tuning [266] introduces an algorithm that fine-tunes a subset of network parameters in each training iteration, dynamically selecting which parameters to update.

Reparameterized Fine-tuning. LoRA [94] represents the primary method in this category, utilizing two low-rank matrices to represent the change in model parameters and adding this change to each linear layer of the network. During fine-tuning, only these changes are learned, significantly reducing the computational cost. Following LoRA, methods such as DyLoRA [245] and AdaLoRA [291] have been proposed, which dynamically adjust the rank of the LoRA modules for each layer rather than using a uniform rank across all layers. MOELORA [143] considers multi-task learning scenarios and employs a Mixture-of-Experts (MOE) approach to train LoRA modules. LoRA Dropout [138] introduces random noise during the learning of LoRA modules to prevent overfitting.

Chapter 3

Overcoming Catastrophic Forgetting in Few-Shot Continual Learning

3.1 Introduction

Why study few-shot continual learning? Continual learning enables a model to continually learn new concepts from new data without forgetting previously learned knowledge. Rooted from real-world applications, this topic has attracted a significant amount of interest in recent years [26, 121, 133, 202, 113]. Continual learning assumes sufficient training data is provided for new classes, which is impractical in many application scenarios, especially when the new classes are rare categories which are costly or difficult to collect. This motivates the study of few-shot continual learning, a more difficult paradigm that aims to continually learn new tasks with only a few examples.

Challenges. The major challenge for continual learning is *catastrophic forgetting* [69, 116, 165], which refers to the drastic performance drop on previous tasks after learning

new tasks. This phenomenon is caused by the inaccessibility to previous data while learning on new data. Catastrophic forgetting presents a bigger challenge for few-shot continual learning. Due to the small amount of training data in new tasks, the model tends to severely overfit on new classes while quickly forgetting old classes, resulting in catastrophic performance.

Current research. The study of few-shot continual learning has just started [237, 203, 296, 39, 30, 164, 289]. Current works mainly borrow ideas from research in continual learning to overcome the forgetting problem, by enforcing strong constraints on model parameters to penalize the changes of parameters [164, 116, 286], or by saving a small amount of exemplars from old classes and adding constraints on the exemplars to avoid forgetting [202, 91, 22]. However, in our empirical study, we find that an intransigent model that only trains on base classes and does not tune on new tasks consistently outperforms state-of-the-art methods, including a joint-training method [237] that uses all encountered data for training and hence suffers from severe data imbalance. This observation motivates us to address this harsh problem from a different angle.

Our solution. Unlike existing solutions that try to overcome the catastrophic forgetting problem during the process of learning new tasks, we adopt a different approach by considering this issue during the training of base classes. Specifically, we propose to search for flat local minima of the base training objective function. For any parameter vector in the flat region around the minima, the loss is small, and the base classes are supposed to be well separated. The flat local minima can be found by adding random noise to the model parameters for multiple times and jointly optimizing multiple loss functions. During the following few-shot continual learning stage, we fine-tune the model parameters within the flat region, which can be achieved by clamping the parameters after updating them on few-shot tasks. In this way, the model can efficiently learn new classes while preserving the old ones. Our key contributions are summarized as follows:

- We conduct a comprehensive empirical study on existing few-shot continual learning methods and discover that a simple baseline model that only trains on base classes outperforms state-of-the-art methods, which demonstrates the severity of catastrophic forgetting.

- We propose a novel approach for few-shot continual learning by addressing the catastrophic forgetting problem in the primitive stage. Through finding the flat minima region during training on base classes and fine-tuning within the region while learning on new tasks, our model can overcome catastrophic forgetting and avoid overfitting.

- Comprehensive experimental results on CIFAR-100, *mini*ImageNet, and CUB-200-2011 show that our approach outperforms all state-of-the-art methods and achieves performance that is very close to the approximate upper bound.

3.2 Related Work

Few-shot learning aims to learn to generalize to new categories with a few labeled samples in each class. Current few-shot methods mainly include optimization-based methods [63, 101, 149, 201, 232, 231, 280] and metric-based methods [68, 90, 224, 249, 274, 287, 288, 273]. Optimization-based methods can achieve fast adaptation to new tasks with limited samples by learning a specific optimization algorithm. Metric-based approaches exploit different distance metrics such as L2 distance [224], cosine similarity [249], and DeepEMD [287] in the learned metric/embedding space to measure the similarity between samples. Recently, Tian *et al.* [242] find that standard supervised training can learn a good metric space for unseen classes, which echoes with our observation on the proposed baseline model in Sec. 3.3.

Continual learning focuses on the challenging problem of continually learning to recognize new classes in new coming data without forgetting old classes [27, 29, 49,

272]. Previous research mainly includes multi-class continual learning [22, 198, 96, 150, 278, 272] and multi-task continual learning [95, 133, 204]. To overcome the catastrophic forgetting problem, some attempts propose to impose strong constraints on model parameters by penalizing the changes of parameters [116, 3]. Other attempts try to enforce constraints on the exemplars of old classes by restricting the output logits [202] or penalizing the changes of embedding angles [91]. In this work, our empirical study shows that imposing strong constraints on the arriving new classes may not be a promising way to tackle few-shot continual learning, due to the scarcity of training data for new classes.

Few-shot Continual learning [237, 203, 296, 39, 30] aims to incrementally learn from very few samples. TOPCI [237] proposes a neural gas network to learn and preserve the topology of the feature manifold formed by different classes. FSLL [164] only selects few model parameters for continual learning and ensures the parameters are close to the optimal ones. To overcome catastrophic forgetting, IDLVQC [30] imposes constraints on the saved exemplars of each class by restricting the embedding drift, and Zhang *et al.* [289] propose to fix the embedding network for continual learning. Similar to the finding of Zhang *et al.*, we also discover that an intransigent model that simply does not adapt to new tasks can outperform prior state-of-the-art methods.

Robust optimization. It has been found that flat local minima leads to better generalization capabilities than sharp minima in the sense that a flat minimizer is more robust when the test loss is shifted due to random perturbations [89, 87, 109]. A substantial body of methods [11, 190, 55, 82] have been proposed to optimize neural networks towards flat local minima. In this paper, we show that for incremental few-shot learning, finding flat minima in the base session and tuning the model within the flat region on new tasks can significantly mitigate catastrophic forgetting.

3.3 Pilot Study: Severity of Catastrophic Forgetting

3.3.1 Problem Statement

Few-shot Continual learning (FCL) aims to continually learn to recognize new classes with only few examples. Similar to continual learning (CL), an FCL model is trained by a sequence of training sessions $\{\mathcal{T}^1, \cdots, \mathcal{T}^t\}$, where $\mathcal{T}^t = \{z_i = (x_i^t, y_i^t)\}_i$ is the training data of session t and x_i^t is an example of class $y_i^t \in \mathcal{C}^t$ (the class set of session t). In FCL, the base session $\mathcal{T}^1$ usually contains a large number of classes with sufficient training data for each class, while the following sessions ($t \geq 2$) only have a small number of classes with few training samples per class, e.g., $\mathcal{T}^t$ is often presented as an N-way K-shot task with small N and K. The key difference between CL and FCL is, for CL, sufficient training data is provided in each session. Similar to CL, in each training session t of FCL, the model has only access to the training data $\mathcal{T}^t$ and possibly a small amount of saved exemplars from previous sessions. When the training of session t is completed, the model is evaluated on test samples from all encountered classes $\mathcal{C} = \bigcup_{i=1}^{t} \mathcal{C}^i$, where it is assumed that there is no overlap between the classes of different sessions, i.e., $\forall i, j$ and $i \neq j$, $\mathcal{C}^i \bigcap \mathcal{C}^j = \emptyset$.

Catastrophic forgetting. FCL is undoubtedly a more challenging problem than CL due to the data scarcity setting. CL suffers from catastrophic forgetting, a well-known phenomenon and long-standing issue, which refers to the drastic drop in test performance on previous (old) classes, caused by the inaccessibility of old data in the current training session. Unfortunately, catastrophic forgetting is an even bigger issue for FCL, because data scarcity makes it difficult to adapt well to new tasks and learn new concepts, while the adaptation process could easily lead to the forgetting of base classes. In the following, we illustrate this point by evaluating a simple baseline model for FCL.

3.3.2 A Simple Baseline Model for FCL

We consider an intransigent model that **simply does not adapt to new tasks**. Particularly, the model only needs to be trained in the base session $\mathcal{T}^1$ and is directly used for inference in all sessions.

Training ($t = 1$). We train a feature extractor f parameterized by ϕ with a fully-connected layer as classifier by minimizing the standard cross-entropy loss using the training examples of $\mathcal{T}^1$. The feature extractor f is *fixed* for the following sessions ($t \geq 2$) without any fine-tuning on new classes.

Inference (test). In each session, the inference is conducted by a simple nearest class mean (NCM) classification algorithm [168]. Specifically, all the training and test samples are mapped to the embedding space of the feature extractor f, and Euclidean distance $d(\cdot, \cdot)$ is used to measure the similarity between them. The classifier is given by

$$c_k^\star = \arg\min_{c \in \mathcal{C}} d(f(x;\phi), p_c), \text{ where } p_c = \frac{1}{N_c} \sum_i \mathbb{1}(y_i = c) f(x_i;\phi), \tag{3.1}$$

where $\mathcal{C}$ denotes all the encountered classes, p_c refers to the prototype of class c (the mean vector of all the training samples of class c in the embedding space), and N_c denotes the number of the training images of class c. **Note that we *save* the prototypes of all classes in $\mathcal{C}^t$ for later evaluation.**

The baseline model outperforms state-of-the-art FCL and CL methods. We compare the above baseline model against state-of-the-art FCL methods including FSLL [164], IDLVQC [30] and TOPIC [237], CL methods including Rebalance [91] and iCarl [202], and a joint-training method that uses all previously seen data including the base and the following few-shot tasks for training, for FCL. The performance is evaluated on miniImageNet, CIFAR-100, and CUB-200. We tune the methods re-implemented by us to the best performance. For the other methods, we use the results reported in the original papers. The experimental details are provided in Sec. 3.5.

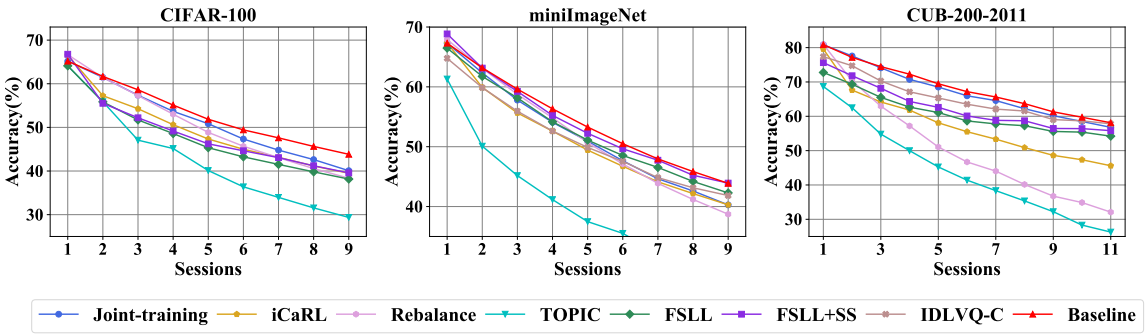


Figure 3.1: Comparison of the proposed baseline model with state-of-the-art FCL and CL methods and the joint-training method.The baseline model outperforms all the other methods.

As shown in Fig. 3.1, the baseline model consistently outperforms all the compared methods including the joint-training method (which suffers from severe data imbalance) on every dataset[1]. Our proposed simple baseline introduces a fixed, pretrained extractor and computes class prototypes as the mean embeddings of training samples. This simple yet effective approach demonstrates strong performance, **particularly in few-shot scenarios**, because **freezing the extractor avoid catastrophic forgetting**. The fact that an intransigent model performs best suggests that

- For FCL, preserving the old (base classes) may be more critical than adapting to the new. Due to data scarcity, the performance gain on new classes is limited and cannot make up for the significant performance drop on base classes.

- Prior works [237, 30, 164, 91, 202] that enforce strong constraints on model parameters or exemplars during fine-tuning on new classes cannot effectively prevent catastrophic forgetting in FCL, indicating that actions may need to be taken in the base training stage.

[1]We notice that a similar observation is made in a newly released paper [289].

3.4 Methodology

The goal of FCL is to preserve the old while adapting to the new efficiently. The results and analysis in Sec. 3.3 suggest that it might be "a bit late" to try to prevent catastrophic forgetting in the few-shot learning sessions ($t \geq 2$), which motivates us to consider this problem in the base training session.

Overview of our approach. To overcome catastrophic forgetting in FCL, we propose to find a b-flat ($b > 0$) local minima $\boldsymbol{\theta}^\star$ of the base training objective function and then fine-tune the model within the flat region in later few-shot learning sessions. Specifically, for any parameter vector $\boldsymbol{\theta}$ in the flat region, i.e., $\boldsymbol{\theta}^\star - b \preceq \boldsymbol{\theta} \preceq \boldsymbol{\theta}^\star + b$, the risk (loss) of the base classes is minimized such that the classes are well separated in the embedding space of $f_{\boldsymbol{\theta}}$. In the later few-shot continual learning sessions ($t \geq 2$), we fine-tune the model parameters within this region to learn new classes, i.e., to find

$$\boldsymbol{\theta}' = \arg\min_{\boldsymbol{\theta}} \sum_{z \in \mathcal{T}^t} \mathcal{L}(z; \boldsymbol{\theta}), \quad \text{s.t.} \quad \boldsymbol{\theta}^\star - b \preceq \boldsymbol{\theta} \preceq \boldsymbol{\theta}^\star + b.$$

As such, the fine-tuned model $\boldsymbol{\theta}'$ can adapt to new classes while preserving the old ones. Also, due to the nature of few-shot learning, to avoid excessive training and overfitting, it suffices to tune the model in a relatively small region. A graphical illustration of our approach and prior arts, as well as the notions of sharp minima and flat minima, are presented in Fig. 3.2.

3.4.1 Searching for Flat Local Minima in the Base Training Stage

A formal definition of b-flat local minima is given as follows.

Definition 3.1 (**b**-flat local minima)**.** *Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be a parameter vector and $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ denote a loss function. We say $\boldsymbol{\theta}$ is a* **b-*flat local minimum*** *if there exists*

a positive vector $\mathbf{b} \in \mathbb{R}^d_+$*, constants* $\epsilon \geq 0$*,* $\gamma_{in} \geq 0$*, and* $\gamma_{out} > \gamma_{in}$*, such that the following conditions hold:*

1. ***Flatness within the*** $\mathbf{b}$***-neighborhood****: For all* $\boldsymbol{\theta}' \in \mathbb{R}^d$ *satisfying* $\boldsymbol{\theta} - \mathbf{b} \preceq \boldsymbol{\theta}' \preceq \boldsymbol{\theta} + \mathbf{b}$*,*

$$|\mathcal{L}(\boldsymbol{\theta}') - \mathcal{L}(\boldsymbol{\theta})| \leq \epsilon \quad \textit{(small loss variation)}, \tag{3.2}$$

$$\|\nabla \mathcal{L}(\boldsymbol{\theta}')\|_2 \leq \gamma_{in} \quad \textit{(weak local gradients)}. \tag{3.3}$$

2. ***Condition 2 (Strict Increase Beyond Neighborhood):*** *There exist* $\mathbf{c}_1 \prec \boldsymbol{\theta}^\star - \mathbf{b}$ *and* $\mathbf{c}_2 \succ \boldsymbol{\theta}^\star + \mathbf{b}$ *such that:*

$$\mathcal{L}(z; \boldsymbol{\theta}) > \mathcal{L}(z; \boldsymbol{\theta}^\star) \quad \textit{for all} \quad \boldsymbol{\theta} \in (\mathbf{c}_1, \boldsymbol{\theta}^\star - \mathbf{b}) \cup (\boldsymbol{\theta}^\star + \mathbf{b}, \mathbf{c}_2)\,.$$

Interpretation*:*

- *The hyperrectangle* $[\boldsymbol{\theta} - \mathbf{b}, \boldsymbol{\theta} + \mathbf{b}]$ *defines a flat region where the loss varies by at most* ϵ*, and gradients are bounded by* γ_{in}*.*
- *Outside the flat local region, the loss increases, which ensures that the flat region corresponds to a local minima.*

Hence, our goal is to find an approximately flat local minima of the base training objective function. To this end, we propose to add some small random noise to the model parameters. The noise can be added for multiple times to obtain similar but different loss functions, which will be optimized together to locate the flat minima region. The intuition is clear – the parameter vectors around the flat local minima also have small function values.

To formally state the idea, we assume that the model is parameterized by $\boldsymbol{\theta} = \{\phi, \psi\}$, where ϕ denotes the parameters of the embedding network and ψ denotes the parameters of the classifier. z denotes a labelled training sample. Denote the loss function

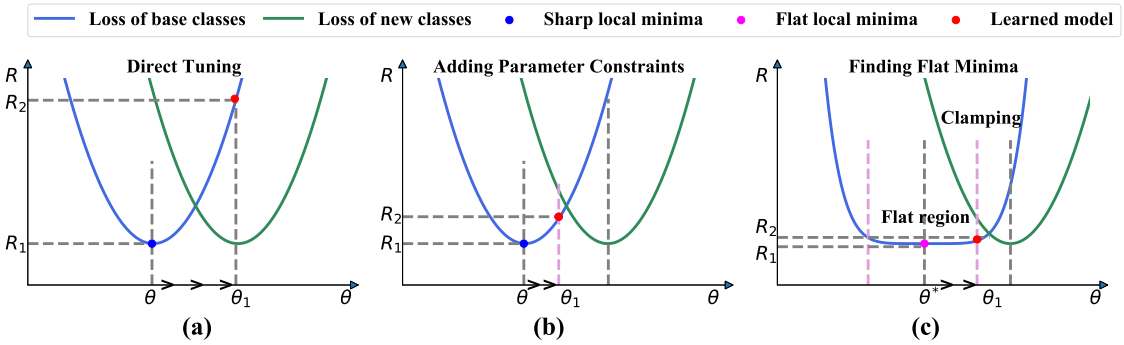


Figure 3.2: Illustration of our approach and existing solutions. $\rightarrow$ indicates the continual learning steps on new classes. R_1 and R_2 respectively denote the loss of base classes before and after minimizing the loss of new classes. (a) SGD finds sharp minima in the base training. Directly tuning the model on new classes will result in a severe performance drop on base classes. (b) Enforcing strong constraints on parameters by penalizing parameter changes [3, 116, 164] may still lead to a significant performance drop on base classes. (c) Finding flat local minima of base classes and clamping the parameters after training on new classes to make them fall within the flat region can effectively mitigate catastrophic forgetting.

by $\mathcal{L}$: $\mathbb{R}^{d_z} \rightarrow \mathbb{R}$. Our target is to minimize the expected loss function R: $\mathbb{R}^d \rightarrow \mathbb{R}$ w.r.t. the joint distribution of data z and noise ϵ, i.e.,

$$R(\boldsymbol{\theta}) = \int_{\mathbb{R}^{d_\epsilon}} \int_{\mathbb{R}^{d_z}} \mathcal{L}(z; \phi + \epsilon, \psi)\, dP(z) dP(\epsilon) = \mathbb{E}[\mathcal{L}(z; \phi + \epsilon, \psi)], \tag{3.4}$$

where $P(z)$ is the data distribution and $P(\epsilon)$ is the noise distribution, and z and ϵ are independent. Since it is impossible to minimize the expected loss, we minimize its estimation, the empirical loss, which is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{j=1}^{M} \mathcal{L}_{\text{base}}(z; \phi + \epsilon_j, \psi), \text{ where} \tag{3.5}$$

$$\mathcal{L}_{\text{base}}(z; \phi + \epsilon_j, \psi) = \frac{1}{|\mathcal{T}^1|} \sum_{z \in \mathcal{T}^1} \mathcal{L}_{ce}(z; \phi + \epsilon_j, \psi) + \lambda \frac{1}{|\mathcal{C}^1|} \sum_{c \in \mathcal{C}^1} \|p_c - p_c^*\|_2^2, \tag{3.6}$$

where ϵ_j is a noise vector sampled from $P(\epsilon)$, M is the sampling times, $\mathcal{L}_{ce}(z; \phi + \epsilon_j, \psi)$ refers to the cross-entropy loss of a training sample z, and p_c and p_c^* are the class

prototypes before and after injecting noise respectively. The first term of $\mathcal{L}_{base}$ is designed to find the flat region where the parameters ϕ of the embedding network can well separate the base classes. The second term enforces the class prototypes fixed within such region, which is designed to solve the prototype drift problem [278, 30] (the class prototypes change after updating the network) in later continual learning sessions such that the saved base class prototypes can be directly used for evaluation in later sessions.

3.4.2 Few-shot Continual Learning within the Flat Region

In the few-shot continual learning sessions ($t \geq 2$), we fine-tune the parameters ϕ of the embedding network *within the flat region* to learn new classes. It is worth noting that while the flat region might be relatively small, it is enough for few-shot continual learning. Because only a few training samples are provided for each new class, to prevent overfitting in few-shot learning, excessive training should be avoided and only a small number of update iterations should be applied.

We employ a metric-based classification algorithm with Euclidean distance to fine-tune the parameters. The loss function is defined as

$$\mathcal{L}_m(z;\phi) = -\sum_{z\in\mathcal{T}}\sum_{c\in\mathcal{C}} \mathbb{1}(y=c)\log(\frac{e^{-d(p_c,f(x;\phi))}}{\sum_{c_k\in\mathcal{C}} e^{-d(p_{c_k},f(x;\phi))}}), z \tag{3.7}$$

where $d(\cdot,\cdot)$ denotes Euclidean distance, p_c is the prototype of class c, $\mathcal{C} = \bigcup_{i=1}^{t}\mathcal{C}^i$ refers to all encountered classes, and $\mathcal{T} = \mathcal{T}^t \bigcup \mathcal{P}$ denotes the union of the current training data $\mathcal{T}^t$ and the exemplar set $\mathcal{P} = \{P_2, ..., P_{t-1}\}$, where $P_{t_e}(2 \leq t_e < t)$ is the set of saved exemplars in session t_e. Note that the prototypes of new classes are computed by Eq. 3.1, and those of base classes are saved in the base session. After updating the embedding network parameters, we clamp them to ensure that they fall within the flat region, i.e. $\phi^\star - b \preceq \phi \preceq \phi^\star + b$, where $\phi^\star$ denotes the optimal parameter vector learned in the base session. After fine-tuning, we evaluate the model using the nearest class mean classifier as in Eq. 3.1, with previously saved prototypes

and newly computed ones. The whole training process is described in Algorithm 1. Note that to calibrate the estimates of the classifier, we normalize all prototypes to make those of base classes and those of new classes have the same norm.

3.4.3 Convergence Analysis

Our aim is to find a flat region within which all parameter vectors work well. We then minimize the expected loss w.r.t. the joint distribution of noise ϵ and data z. To approximate this expected loss, we sample from $P(\epsilon)$ for multiple times in each iteration and optimize the objective function using stochastic gradient descent (SGD). Here, we provide theoretical guarantees for our method. Given the non-convex loss function in Eq. 3.5, we prove the convergence of our proposed method. The proof idea is inspired by the convergence analysis of SGD [15, 115].

Formally, in each batch k, let z_k denote the batch data, $\{\epsilon_j\}_{j=1}^{M}$ be the sampled noises, and α_k be the step size. In the base training session, we update the model parameters as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\alpha_k}{M}\sum_{j=1}^{M}\nabla\mathcal{L}_{\text{base}}(z_k;\phi_k+\epsilon_j,\psi_k) = \boldsymbol{\theta}_k - \frac{\alpha_k}{M}\sum_{j=1}^{M} g(z_k;\phi_k+\epsilon_j,\psi_k), \quad (3.8)$$

where $g(z_k;\phi_k+\epsilon_j,\psi_k) = \nabla\mathcal{L}_{\text{base}}(z_k;\phi_k+\epsilon_j,\psi_k)$ is the gradient. To formally analyze the convergence of our algorithm, we define the following assumptions.

Assumption 3.1 (L-smooth risk function)**.** *The expected loss function* $R:\mathbb{R}^d\to\mathbb{R}$ (Eq. 3.4) *is continuously differentiable and L-smooth with constant* $L>0$ *such that*

$$\|\nabla R(\boldsymbol{\theta}) - \nabla R(\boldsymbol{\theta}')\|_2 \le L\|\boldsymbol{\theta}-\boldsymbol{\theta}'|. \quad (3.9)$$

This assumption is significant for the convergence analysis of gradient-based optimization algorithms, since it limits how fast the gradient of the loss function can change w.r.t. the parameter vector.

Algorithm 1: F2M

Input: the flat region bound b, randomly initialized $\boldsymbol{\theta} = \{\phi, \psi\}$, the step sizes α and β.

// Training over base classes $t = 1$

for epoch k = 1,2,... **do**

  for j = 1,2,..., M **do**

    Sample a noise vector $\epsilon_j \sim P(\epsilon)$, s.t. $-\mathbf{b} \preceq \epsilon_j \preceq \mathbf{b}$;

    Add the noise to the parameters of the embedding network, i.e., $\boldsymbol{\theta} = \{\phi + \epsilon_j, \psi\}$;

    Compute the base loss $\mathcal{L}_{base}$ with Eq. 3.6;

    Reset the parameters, i.e., $\boldsymbol{\theta} = \{\phi, \psi\}$;

  end

  Update $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta})$ with the loss $\mathcal{L}$ defined in Eq. 3.5.

end

Normalize and save the prototype of each base class;

// Incremental learning $t \geq 2$

Combine the training data $\mathcal{D}^t$ and the exemplars saved in previous few-shot sessions $2 \leq t_e < t$;

for epoch k = 1,2,... **do**

  Compute the metric-based classification loss $\mathcal{L}_m$ by Eq. 3.7;

  Update $\phi = \phi - \beta \nabla \mathcal{L}_m(z; \phi)$;

  Clamp the parameters ϕ to ensure they fall in the flat minima region;

end

Randomly select and save a few exemplars from the training data $\mathcal{D}^t$;

Normalize and save the prototype of each new class;

Output: Model parameters $\boldsymbol{\theta} = \{\phi, \psi\}$.

Assumption 3.2. *The expected loss function satisfies the following conditions:*

- *Condition 1: R is bounded below by a scalar $R^\star$, given the sequence of parameters $\{\boldsymbol{\theta}_k\}$.*
- *Condition 2: For all $k \in \mathbb{N}$ and $j \in [1, M]$,*

$$\mathbb{E}_{z_k,\epsilon_j}[g(z_k; \phi_k + \epsilon_j, \psi_k)] = \nabla R(\boldsymbol{\theta}_k). \tag{3.10}$$

- *Condition 3: There exist scalars $m_1 \geq 0$ and $m_2 \geq 0$, for all $k \in \mathbb{N}$ and $j \in [1, M]$,*

$$\mathbb{V}_{z_k,\epsilon_j}[g(z_k; \phi_k + \epsilon_j, \psi_k)] \leq m_1 + m_2 \|\nabla R(\boldsymbol{\theta}_k)\|_2^2. \tag{3.11}$$

$\mathbb{E}_{z_k,\epsilon_j}[\cdot]$ denotes the expectation w.r.t. the joint distribution of random variables z_k and ϵ_j, and $\mathbb{V}_{z_k,\epsilon_j}[\cdot]$ denotes the variance. Condition 1 ensures that the expected loss R is bounded by a minimum value $R^\star$ during the updates, which is a natural and practical assumption. Condition 2 assumes that the gradient $g(z_k; \phi_k + \epsilon_j, \psi_k)$ is an unbiased estimate of $\nabla R(\boldsymbol{\theta}_k)$. This is a strict assumption made to simplify the proof, but it can be easily relaxed to a general and easily-met condition that there exist $\mu_1 \geq \mu_2 > 0$ satisfying $\|\mathbb{E}_{z_k,\epsilon_j}[g(z_k; \phi_k + \epsilon_j, \psi_k)]\|_2 \leq \mu_1 \|\nabla R(\boldsymbol{\theta}_k)\|_2$ and $\nabla R(\boldsymbol{\theta}_k)^T \mathbb{E}_{z_k,\epsilon_j}[g(z_k; \phi_k + \epsilon_j, \psi_k)] \geq \mu_2 \|\nabla R(\boldsymbol{\theta}_k)\|_2^2$. Condition 3 assumes the variance of the gradient $g(z_k; \phi_k + \epsilon_j, \psi_k)$ cannot be arbitrarily large, which is also reasonable in practice. It is worth noting that Bottou et al. also adopts the same assumption in their work proving the convergence of SGD, as it facilitates obtaining the desired bound more easily. The steps in the proofs where assumptions are used have been bolded to improve readability.

To facilitate later analysis, similar to [206], we restrict the step sizes as follows.

Assumption 3.3. *The learning rates satisfy:*

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \ \sum_{k=1}^{\infty} \alpha_k^2 < \infty. \tag{3.12}$$

This assumption can be easily met, since in practice the learning rate α_k is usually far less than 1 and decreases w.r.t. k. Based on the above assumptions, we can derive the following theorem.

Theorem 3.1. *Under assumptions 3.1, 3.2 and 3.3, we further assume that the risk function R is twice differentiable, and that $\|\nabla R(\boldsymbol{\theta})\|_2^2$ is L_2-smooth with constant $L_2 > 0$, then we have*

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla R(\boldsymbol{\theta}_k)\|_2^2] = 0. \tag{3.13}$$

This theorem establishes the convergence of our algorithm. The proof is presented as follows:

Lemma 3.1. *By Assumption 3.1 and 3.2, we have*

$$\mathbb{E}_{z_k,\epsilon_j}[R(\theta_{k+1})] - R(\theta_k) \leq -\alpha_k \frac{2M - \alpha_k L(m_2 + M)}{2M} \|\nabla R(\theta_k)\|_2^2 + \frac{\alpha_k^2 L m_1}{2M}. \tag{3.14}$$

Proof. By **Assumption 3.1**, an important consequence is that for all $\{\theta, \theta'\} \subset \mathbb{R}^d$, it satisfies that

$$R(\theta) \leq R(\theta') + \nabla R(\theta')^T(\theta - \theta') + \frac{1}{2} L \|\theta - \theta'\|_2^2. \tag{3.15}$$

Taken together, the above inequality and the parameter update equation (Eq. 3.8), it yields

$$R(\theta_{k+1}) - R(\theta_k) \leq \nabla R(\theta_k)^T(\theta_{k+1} - \theta_k) + \frac{1}{2} L \|\theta_{k+1} - \theta_k\|_2^2 \leq -\alpha_k \nabla R(\theta_k)^T \overline{g} + \frac{\alpha_k^2 L}{2} \|\overline{g}\|_2^2, \tag{3.16}$$

where $\overline{g} = \frac{1}{M} \sum_{j=1}^{M} g(z_k; \phi_k + \epsilon_j, \psi_k)$. Taking expectation on both sides of Eq. 3.16, it yields

$$\mathbb{E}_{z_k,\epsilon_j}[R(\theta_{k+1})] - R(\theta_k) \leq -\alpha_k \nabla R(\theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[\overline{g}] + \frac{\alpha_k^2 L}{2} \mathbb{E}_{z_k,\epsilon_j}[\|\overline{g}\|_2^2]. \tag{3.17}$$

$\mathbb{E}_{z_k,\epsilon_j}[\cdot]$ denotes the expectation w.r.t. the joint distribution of random variables z_k and ϵ_j given θ_k. Note that θ_{k+1} (not θ_k) depends on z_k and ϵ_j. **Under Condition 2**

of Assumption 3.2, the expectation of $\overline{g}$ satisfies that

$$\mathbb{E}_{z_k,\epsilon_j}[\overline{g}] = \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{z_k,\epsilon_j}[g(z_k;\phi_k+\epsilon_j,\psi_k)] = \nabla R(\theta_k). \tag{3.18}$$

Assume that we sample the noise vector ϵ_j from $P(\epsilon)$ *without replacement*. **Under Condition 3 of Assumption 3.2**, we have

$$\mathbb{V}_{z_k,\epsilon_j}[\overline{g}] \leq \frac{\mathbb{V}_{z_k,\epsilon_j}[g(z_k;\phi_k+\epsilon_j,\psi_k)]}{M} \leq \frac{m_1}{M} + \frac{m_2}{M}\|\nabla R(\theta_k)\|_2^2. \tag{3.19}$$

Taken together, Eq. 3.18 and Eq. 3.19, one obtains

$$\mathbb{E}_{z_k,\epsilon_j}[\|\overline{g}\|_2^2] = \mathbb{V}_{z_k,\epsilon_j}[\overline{g}] + \|\mathbb{E}_{z_k,\epsilon_j}[\overline{g}]\|_2^2 \leq \frac{m_1}{M} + \frac{m_2+M}{M}\|\nabla R(\theta_k)\|_2^2. \tag{3.20}$$

Therefore, by Eq. 3.17, 3.18 and 3.20, it yields

$$\mathbb{E}_{z_k,\epsilon_j}[R(\theta_{k+1})] - R(\theta_k) \leq -\alpha_k\frac{2M-\alpha_k L(m_2+M)}{2M}\|\nabla R(\theta_k)\|_2^2 + \frac{\alpha_k^2 L m_1}{2M}. \tag{3.21}$$

□

Lemma 3.2. *By Assumption 3.1, 3.2 and 3.3, we have*

$$\liminf_{k\to\infty}\mathbb{E}[\|\nabla R(\theta_k)\|_2^2] = 0. \tag{3.22}$$

Proof. The first condition in Assumption 3.3 ensures that $\lim_{k\to\infty}\alpha_k = 0$. Without loss of generality, we assume that for any $k \in \mathbb{N}$, $\alpha_k L(m_2+M) \leq M$. Denote by $\mathbb{E}[\cdot]$ the *total expectation* w.r.t. all involved random variables. For example, θ_k is determined by the set of random variables $\{z_0, z_1, ..., z_{k-1}, \epsilon_0, \epsilon_1, ..., \epsilon_{k-1}\}$, and therefore the *total expectation* of $R(\theta_k)$ is given by

$$\mathbb{E}[R(\theta_k)] = \mathbb{E}_{z_0,\epsilon_0}\mathbb{E}_{z_1,\epsilon_1}...\mathbb{E}_{z_{k-1},\epsilon_{k-1}}[R(\theta_k)]. \tag{3.23}$$

Taking total expectation on both sides of Eq.3.14, we have

$$\mathbb{E}[R(\theta_k+1)] - \mathbb{E}[R(\theta_k)] \leq -\frac{\alpha_k}{2}\mathbb{E}[\|\nabla R(\theta_k)\|_2^2] + \frac{\alpha_k^2 L m_1}{2M}. \tag{3.24}$$

For $k = 0, 1, 2, ..., K$, summing both sides of this inequality yields

$$R^\star - \mathbb{E}[R(\theta_1)] \leq \mathbb{E}[R(\theta_{K+1})] - \mathbb{E}[R(\theta_0)] \leq -\frac{1}{2}\sum_{k=0}^{K}\alpha_k \mathbb{E}[\|\nabla R(\theta_k)\|_2^2] + \frac{Lm_1}{2M}\sum_{k=0}^{K}\alpha_k^2, \tag{3.25}$$

where $R^\star$ is the **lower bound in Condition 1 of Assumption 3.2**. Rearranging the term gives

$$\sum_{k=0}^{K}\alpha_k \mathbb{E}[\|\nabla R(\theta_k)\|_2^2] \leq 2(\mathbb{E}[R(\theta_1)] - R^\star) + \frac{Lm_1}{M}\sum_{k=0}^{K}\alpha_k^2. \tag{3.26}$$

By **the second condition of Assumption 3.3**, we have

$$\lim_{K\to\infty} \mathbb{E}[\sum_{k=0}^{K}\alpha_k\|\nabla R(\theta_k)\|_2^2] \leq 2(\mathbb{E}[R(\theta_0)] - R^\star) + \lim_{K\to\infty}\frac{Lm_1}{M}\sum_{k=0}^{K}\alpha_k^2 < \infty. \tag{3.27}$$

Dividing both sides of Eq. 3.27 by $\sum_{k=1}^{K}\alpha_k$ and **by the first condition of Assumption 3.3**, we have

$$\lim_{K\to\infty} \mathbb{E}[\frac{\sum_{k=1}^{K}\alpha_k\|\nabla R(\theta_k)\|_2^2}{\sum_{k=1}^{K}\alpha_k}] = 0. \tag{3.28}$$

The left-hand term of this equation is the weighed average of $\|\nabla R(\theta_k)\|_2^2$, and $\{\alpha_k\}$ are the weights. Hence, a direct consequence of this equation is that $\|\nabla R(\theta_k)\|_2^2$ cannot asymptotically stay far from zero, i.e.

$$\liminf_{k\to\infty} \mathbb{E}[\|\nabla R(\theta_k)\|_2^2] = 0. \tag{3.29}$$

□

We now prove Theorem 3.1, which is a stronger consequence than Lemma 3.2.

Theorem 4.1. *Under assumptions 3.1, 3.2 and 3.3, we further assume that the risk function R is twice differentiable, and that $\|\nabla R(\theta)\|_2^2$ is L_2-smooth with constant $L_2 > 0$, then we have*

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla R(\theta_k)\|_2^2] = 0. \tag{3.30}$$

Proof. Define $F(\theta) \coloneqq \|R(\theta)\|_2^2$, then we have

$$\begin{aligned}
\mathbb{E}_{z_k,\epsilon_j}[F(\theta_{k+1})] - F(\theta_k) &\leq \nabla F(\theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[(\theta_{k+1} - \theta_k)] + \frac{1}{2} L_2 \mathbb{E}_{z_k,\epsilon_j}[\|\theta_{k+1} - \theta_k\|_2^2] \\
&\leq -\alpha_k \nabla F(\theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[\overline{g}] + \frac{\alpha_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_j}[\|\overline{g}\|_2^2] \\
&\leq -2\alpha_k \nabla R(\theta_k)^T \nabla^2 R(\theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[\overline{g}] + \frac{\alpha_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_j}[\|\overline{g}\|_2^2] \\
&\leq 2\alpha_k \|\nabla R(\theta_k)\|_2^2 \|\nabla^2 R(\theta_k)\|_2 \|\mathbb{E}_{z_k,\epsilon_j}[\overline{g}]\|_2 + \frac{\alpha_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_j}[\|\overline{g}\|_2^2] \\
&\leq 2\alpha_k L \|\nabla R(\theta_k)\|_2^2 + \frac{\alpha_k^2 L_2}{2} (\frac{m_1}{M} + \frac{m_2 + M}{M} \|\nabla R(\theta_k)\|_2^2).
\end{aligned} \tag{3.31}$$

Taking total expectation of both sides of Eq. 3.31 yields

$$\mathbb{E}[F(\theta_{k+1})] - \mathbb{E}[F(\theta_k)] \leq 2\alpha_k L \mathbb{E}[\|\nabla R(\theta_k)\|_2^2] + \frac{\alpha_k^2 L_2}{2} (\frac{m_1}{M} + \frac{m_2 + M}{M} \mathbb{E}[\|\nabla R(\theta_k)\|_2^2]). \tag{3.32}$$

Eq. 3.27 implies that $2\alpha_k L \mathbb{E}[\|\nabla R(\theta_k)\|_2^2]$ is the term of a convergent sum. Besides, $\frac{\alpha_k^2 L_2}{2} (\frac{m_1}{M} + \frac{m_2+M}{M} \mathbb{E}[\|\nabla R(\theta_k)\|_2^2])$ is also the term of a convergent sum, because $\sum_{k=1}^{\infty} \alpha_k^2$ converges. Hence, the bound (Eq. 3.32) is also the term of a convergent sum. Now, let us define

$$A_K^+ = \sum_{k=0}^{K-1} \max(0, \mathbb{E}[F(\theta_{k+1})] - \mathbb{E}[F(\theta_k)]), \tag{3.33}$$

$$\text{and} \quad A_K^- = \sum_{k=0}^{K-1} \max(0, \mathbb{E}[F(\theta_k)] - \mathbb{E}[F(\theta_{k+1})]). \tag{3.34}$$

Because the bound of $\mathbb{E}[F(\theta_{k+1})] - \mathbb{E}[F(\theta_k)]$ is positive and is the term a of convergent sum, and the sequence A_K^+ is upper bounded by the sum of the bound of $\mathbb{E}[F(\theta_{k+1})] - \mathbb{E}[F(\theta_k)]$, A_K^+ converges. Similarly, A_K^- also converges. Since for any $K \in \mathbb{N}$, $F(\theta_K) = F(\theta_0) + A_K^+ - A_K^-$, we can obtain that $F(\theta_k)$ converges. By Lemma 3.2 and the fact that $F(\theta_k)$ converges, we have

$$\lim_{k \to \infty} \mathbb{E}[\|R(\theta_k)\|_2^2] = 0. \tag{3.35}$$

□

3.5 Experiments

In this section, we comprehensively evaluate our proposed method for few-shot continual learning and demonstrate its effectiveness through detailed comparisons with state-of-the-art methods.

3.5.1 Experimental Setup

Table 3.1: Classification accuracy on CIFAR-100 for 5-way 5-shot incremental learning. * indicates our re-implementation.

Method	**sessions**									**The gap with cRT**
	1	2	3	4	5	6	7	8	9	
cRT [110]*	65.18	63.89	60.20	57.23	53.71	50.39	48.77	47.29	45.28	-
Joint-training*	65.18	61.45	57.36	53.68	50.84	47.33	44.79	42.62	40.08	-5.20
Baseline	65.18	61.67	58.61	55.11	51.86	49.43	47.60	45.64	43.83	-1.45
iCaRL [202]*	66.52	57.26	54.27	50.62	47.33	44.99	43.14	41.16	39.49	-5.79
Rebalance [91]*	**66.66**	61.42	57.29	53.02	48.85	45.68	43.06	40.56	38.35	-6.93
FSLL [164]*	65.18	56.24	54.55	51.61	49.11	47.27	45.35	43.95	42.22	-3.08
iCaRL [202]	64.10	53.28	41.69	34.13	27.93	25.06	20.41	15.48	13.73	-31.55
Rebalance [91]	64.10	53.05	43.96	36.97	31.61	26.73	21.23	16.78	13.54	-31.74
TOPIC [237]	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	-15.91
FSLL [164]	64.10	55.85	51.71	48.59	45.34	43.25	41.52	39.81	38.16	-7.12
FSLL+SS [164]	66.76	55.52	52.20	49.17	46.23	44.64	43.07	41.20	39.57	-5.71
F2M	64.71	**62.05**	**59.01**	**55.58**	**52.55**	**49.96**	**48.08**	**46.28**	**44.67**	**-0.61**

Datasets. We utilize CIFAR-100, *mini*ImageNet, and CUB-200-2011 for our evaluations. For CIFAR-100 and *mini*ImageNet, we randomly select 60 classes as base and the remaining 40 as new classes. Each class in CIFAR-100 comprises 500 training images and 100 test images, sized 32×32. In *mini*ImageNet, each class includes 500 training images and 100 test images, each sized 84×84. For the CUB-200-2011 dataset, which contains 5994 training and 5794 test images, we resize and crop each image to 224×224. This dataset is split into 100 base and 100 new classes, where we test 10-way 5-shot tasks.

Table 3.2: Classification accuracy on *mini*ImageNet for 5-way 5-shot incremental learning. * indicates our re-implementation.

Method	**sessions**									**The gap with cRT**
	1	2	3	4	5	6	7	8	9	
cRT [110]*	67.30	64.15	60.59	57.32	54.22	51.43	48.92	46.78	44.85	-
Joint-training*	67.30	62.34	57.79	54.08	50.93	47.65	44.64	42.61	40.29	-4.56
Baseline	67.30	63.18	59.62	56.33	53.28	50.50	47.96	45.85	43.88	-0.97
iCaRL [202]*	67.35	59.91	55.64	52.60	49.43	46.73	44.13	42.17	40.29	-4.56
Rebalance [91]*	67.91	63.11	58.75	54.83	50.68	47.11	43.88	41.19	38.72	-6.13
FSLL [164]*	67.30	59.81	57.26	54.57	52.05	49.42	46.95	44.94	42.87	-1.11
iCaRL [202]	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	-27.64
Rebalance [91]	61.31	47.80	39.31	31.91	25.68	21.35	18.67	17.24	14.17	-30.68
TOPIC [237]	61.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	-20.43
FSLL [164]	66.48	61.75	58.16	54.16	51.10	48.53	46.54	44.20	42.28	-2.57
FSLL+SS [164]	**68.85**	63.14	59.24	55.23	52.24	49.65	47.74	45.23	43.92	-0.93
IDLVQ-C [30]	64.77	59.87	55.93	52.62	49.88	47.55	44.83	43.14	41.84	-3.01
F2M	67.28	**63.80**	**60.38**	**57.06**	**54.08**	**51.39**	**48.82**	**46.58**	**44.65**	**-0.20**

Baselines. We compare our method F2M with 8 methods: the Baseline proposed in Sec. 3.3, a joint-training method that uses all previously seen data including the base and the following few-shot tasks for training, the classifier re-training method (cRT) [110] for long-tailed classification trained with all encountered data, iCaRL [202], Rebalance [91], TOPIC [237], FSLL [164], and IDLVQ-C [30]. For a fair comparison, we re-implement cRT [110], iCaRL [202], Rebalance [91], FSLL [164], and the joint-training method and tune them to their best performance. We also provide the results reported in the original papers for comparison. The results of TOPIC [237] and IDLVQ-C [30] are copied from the original papers. Note that for CL, joint-training is naturally the upper bound of continual learning algorithms, however, for FCL, joint-training is not a good approximation of the upper bound because data imbalance makes the model perform significantly poorer on new classes (long-tailed classes). To address the data imbalance issue, we re-implement the cRT method as the *approximate upper bound*.

Table 3.3: Classification accuracy on CUB-200-2011 for 10-way 5-shot incremental learning.* indicates our re-implementation.

Method	sessions											The gap with cRT
	1	2	3	4	5	6	7	8	9	10	11	
cRT [110]*	80.83	78.51	76.12	73.93	71.46	68.96	67.73	66.75	64.22	62.53	61.08	-
Joint-training*	80.83	77.57	74.11	70.75	68.52	65.97	64.58	62.22	60.18	58.49	56.78	-4.30
Baseline	80.87	77.15	74.46	72.26	69.47	67.18	65.62	63.68	61.30	59.72	58.12	-2.96
iCaRL [202]*	79.58	67.63	64.17	61.80	58.10	55.51	53.34	50.89	48.62	47.34	45.60	-15.48
Rebalance [91]*	80.94	70.32	62.96	57.19	51.06	46.70	44.03	40.15	36.75	34.88	32.09	-28.99
FSLL [164]*	80.83	77.38	72.37	71.84	67.51	65.30	63.75	61.16	59.05	58.03	55.82	-5.26
iCaRL [202]	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	-39.92
Rebalance [91]	68.68	57.12	44.21	28.78	26.71	25.66	24.62	21.52	20.12	20.06	19.87	-41.21
TOPIC [237]	68.68	62.49	54.81	49.99	45.25	41.40	38.35	35.36	32.22	28.31	26.28	-34.80
FSLL [164]	72.77	69.33	65.51	62.66	61.10	58.65	57.78	57.26	55.59	55.39	54.21	-6.87
FSLL+SS [164]	75.63	71.81	68.16	64.32	62.61	60.10	58.82	58.70	56.45	56.41	55.82	-5.26
IDLVQ-C [30]	77.37	74.72	70.28	67.13	65.34	63.52	62.10	61.54	59.04	58.68	57.81	-3.27
F2M	**81.07**	**78.16**	**75.57**	**72.89**	**70.86**	**68.17**	**67.01**	**65.26**	**63.36**	**61.76**	**60.26**	**-0.82**

Experimental details. The experiments are conducted with NVIDIA GPU RTX3090 on CUDA 11.0. We randomly split each dataset into multiple tasks (sessions). For each dataset (with a fixed split), we run each algorithm for 10 times and report the mean accuracy. We adopt ResNet18 [83] as the backbone network. For data augmentation, we use standard random crop and horizontal flip. In the base training stage, we select the last 4 or 8 convolution layers to inject noise, because these layers output higher-level feature representations. The flat region bound b is set as 0.01. We set the number of times for noise sampling as $M = 2 \sim 4$, since a larger M will increase the training time. In each few-shot continual learning session, the total number of training epochs is 6, and the learning rate is 0.02.

Exemplar Management. Since there lacks a unified standard in storing/saving exemplars for few-shot continual learning, we choose the setting that we consider most reasonable and practical. In real-world applications, normally there exists a large number of base classes with sufficient training data (e.g., the base dataset is ImageNet-1K [44]), whereas the number of unseen novel classes that lack training

Table 3.4: Our re-implementation results of FSLL are very close to those reported in [3] on CIFAR-100 for 5-way 5-shot incremental learning. * indicates our re-implementation. The results are obtained without saving any exemplars.

Method	**sessions**								
	1	2	3	4	5	6	7	8	9
FSLL [3]*	65.18	56.37	52.59	48.39	47.46	43.44	41.37	40.17	38.56
FSLL [3]	64.10	55.85	51.71	48.59	45.34	43.25	41.52	39.81	38.16

data is relatively small. Therefore, for computational efficiency and efficient use of storage, it is desirable NOT saving any exemplars for base classes but store some exemplars for new classes. In our experiments, we do not store any exemplar for base classes, but save 5 exemplars for each new class. This will hardly cost any storage space or slow down computation considerably due to the small number of new classes.

To ensure a fair comparison, for ICaRL [202] and Rebalance [91], we store 2 exemplars per class (for both base classes and new classes). As a result, in each session, they store more examplars than our method. For our re-implementation of FSLL [164], we store the same number of exemplars for each new class as in our method. For other approaches, since the code is not available or the method is too complex to re-implement, we directly use the results reported in their paper, which are substantially lower than the Baseline.

Correctness of our implementation. To verify the correctness of our implementation of ICaRL* [202] and Rebalance* [91], we conduct experiments on CIFAR-100 for *incremental learning*. We adopt 32-layer ResNet as backbone and store 20 exemplars per class as in Rebalance [91]. The comparative results are presented in Fig 3.3. It can be seen that our re-implementation results of ICaRL and Rebalance are very close to those reported in [91].

To verify the correctness of our implementation of FSLL [164], we compare the results

of our implementation and those reported in [164] in Table 3.4. It can be seen that our implementation achieves similar and slightly higher results than those reported in the original paper [164]. Here, the experiments are conducted following the settings in [164] without saving any exemplars for new classes.

3.5.2 Comparison with the State-of-the-Art

F2M outperforms the state-of-the-art methods. The main results on CIFAR-100, miniImageNet and CUB-200-2011 are presented in Table 3.1, Table 3.2 and Table 3.3 respectively. Based on the experiment results, we have the following observations: **1)** The Baseline introduced in Sec. 3.3 outperforms the state-of-the-art approaches on all continual sessions. **2)** As expected, cRT consistently outperforms the Baseline up to 1% to 3% by considering the data imbalance problem and applying proper techniques to tackle the long-tailed classification problem to improve performance. Hence, it is reasonable to use cRT as the approximate upper bound of FCL. **3)** Our F2M outperforms the state-of-the-art methods and the Baseline. Moreover, the performance of F2M is very close to the approximate upper bound, i.e., the gap with cRT is only 0.2% in the last session on *mini*ImageNet. The results show that even with strong constraints [91, 202, 164] and saved examplars of base classes [91, 202, 30], current methods cannot effectively address the catastrophic forgetting problem. In contrast, finding flat minima seems a promising approach to overcome this harsh problem.

3.5.3 Ablation Study and Analysis

Analysis on the flatness of local minima. Here, we verify that our method can find a more flat local minima than the Baseline. For a found local minima $\theta^\star$, we measure its flatness as follows. We sample the noise for 1000 times. For each time, we inject the sampled noise to $\theta^\star$ and calculate the loss $\mathcal{L}_i$. Then, we adopt the indicator

Table 3.5: Comparison of the flatness of the local minima found by the Baseline and our F2M.

Method	Indicator I		Variance σ^2	
	Training Set	Testing Set	Training Set	Testing Set
Baseline	0.2993	0.4582	0.1451	0.2395
F2M	**0.0506**	**0.0800**	**0.0296**	**0.0334**

$I = \frac{1}{1000}\sum_{i=1}^{1000}(\mathcal{L}_i - \mathcal{L}^*)^2$ and variance $\sigma^2 = \frac{1}{1000}\sum_{i=1}^{1000}(\mathcal{L}_i - \overline{\mathcal{L}})^2$ to measure the flatness. $\mathcal{L}^*$ denotes the loss of $\boldsymbol{\theta}^\star$, and $\overline{\mathcal{L}}$ denotes the average loss of $\{\mathcal{L}_i\}_{i=1}^{1000}$. The values of the indicator and variance of F2M and the Baseline are presented in Table 3.5, which clearly demonstrate that our method can find a more flat local minima.

Table 3.6: Ablation study of our F2M on CIFAR-100. PD refers to the performance dropping rate.

FM	**PF**	**PC**	**PN**	**sessions**									**PD** ↓
				1	2	3	4	5	6	7	8	9	
				65.18	60.83	53.13	43.57	23.75	10.76	08.26	07.24	06.45	58.73
		✓		**65.18**	59.48	56.77	52.99	50.09	47.80	45.92	44.20	42.55	22.63
✓	✓		✓	64.71	59.54	53.03	45.09	41.68	39.04	38.64	37.19	36.01	28.70
✓		✓	✓	64.55	61.27	58.33	54.82	51.60	49.22	47.48	45.78	44.08	20.47
✓	✓	✓		64.71	61.75	58.80	55.33	52.27	49.75	47.72	46.01	44.43	20.28
✓	✓	✓	✓	64.71	**61.99**	**58.99**	**55.58**	**52.55**	**49.96**	**48.08**	**46.28**	**44.67**	**20.04**

Ablation study on the designs of our method. Here, we study the effectiveness of each design of our method, including adding noise to the model parameters for finding b-flat local minima (FM) during the base training session, the prototype fixing term (PF) used in the base training objective (Eq. 3.6), parameter clamping (PC) during continual learning, and prototype normalization (PN). We conduct an ablation study by removing each component in turn and report the experimental results in Table 3.6.

Finding b-flat local minima. Standard supervised training with SGD as the optimizer tends to converge to a sharp local minima. It leads to a significant drop in performance because the loss changes quickly in the neighborhood of the sharp local minima. As shown in Table 3.6, even with parameter clamping during continual learning, the performance still drops significantly. In contrast, restricting the parameters in a small flat region can mitigate the forgetting problem.

Prototype fixing. Without fixing the prototypes after injecting noise to selected layers during the process of finding local minima, i.e. removing the second term of Eq. 3.6, it is still possible to tune the model within the flat region to well separate base classes. However, the saved prototypes of base classes will become less accurate because the embeddings of the base samples suffer from semantic drift [278]. As shown in Table 3.6, it results in a performance drop of nearly 0.6%.

Parameter clamping. Parameter clamping restricts the model parameters to the b-flat region after few-shot continual learning. Outside the b-flat region, the performance drops quickly. It can be seen from Table 3.6 that removing parameter clamping leads to a significant drop in performance.

Prototype normalization. In our experiments, we observe that after training on base classes with balanced data, the norms of the class prototypes of base classes tend to be similar. However, after fine-tuning with very few data on unseen new classes, the norms of the new class prototypes are noticeably smaller than those of the base classes. In Table 3.7, we show the average norms of the prototypes of base classes and new classes after few-shot continual learning on CIFAR-100, where we randomly select 60 classes as base classes and the remaining 40 classes as new classes. To calibrate the estimates of the classifier, we normalize the class prototypes to calibrate the estimates of the class mean classifier. The results in Table 3.6 show the effectiveness of normalization, which helps to further improve the performance.

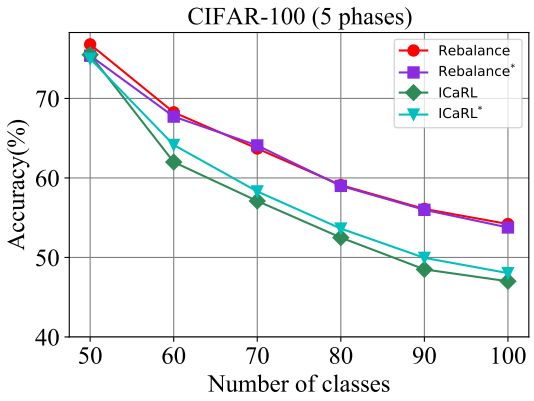


Figure 3.3: Our re-implementation results of Rebalance and ICaRL are very close to those reported in [202]. * indicates our re-implementation.

Table 3.7: The average norm of the class prototypes of new classes is significantly smaller than that of old classes. The experiment is conducted on CIFAR-100 with 60 base classes and 40 new classes.

	Mean	Standard Deviation
Base classes	7.97	0.63
New classes	7.48	0.71

Study of the flat region bound b. We study the effect of the flat region bound b for 5-way 5-shot continual learning on CIFAR-100. We report the test accuracy in session 1 (base session) and session 9 (last session) w.r.t. different b in Table 3.8. It can be seen that the best results are achieved for $b \in [0.005, 0.02]$. A larger b (e.g., 0.04 or 0.08) leads to a significant performance drop on base classes, even for those in session 1, indicating that there may not exist a large flat region around a good local minima. Meanwhile, a smaller b (e.g., 0.0025) results in a performance decline on new classes, due to the overly small capacity of the flat region. This illustrates the trade-off effect of b.

F2M exhibits minimal performance degradation on base classes. As shown in Table 3.8, even after 9 sessions of incremental learning and across different values of b, the accuracy on base classes remains close to 60%. This indicates that the performance on base classes is largely preserved. Considering that the classification task expands from 60 classes to 100 classes, a slight drop in accuracy is expected due to increased difficulty. This further demonstrates the effectiveness of our proposed approach in identifying flat local minima. Specifically, it confirms that we indeed

Table 3.8: Study of the flat region bound b for 5-way 5-shot incremental learning on CIFAR-100. The top 3 results in each row are in boldface.

Session	The hyperparameter b					
	0.0025	0.005	0.01	0.02	0.04	0.08
Session 1 (60 bases classes)	**64.85**	64.67	**64.81**	**64.71**	63.30	62.25
Session 9 (All 100 classes)	44.16	**44.54**	**44.58**	**44.67**	43.75	43.04
Session 9 (60 base classes)	**59.58**	**59.69**	**59.73**	59.44	58.38	57.21
Session 9 (40 new classes)	21.03	**21.81**	**21.86**	**22.52**	21.80	21.77

locate flat minima, and fine-tuning within such regions can preserve the performance on base classes with minimal degradation.

Results with the same class splits as in TOPIC [237]. The experimental results of our F2M and some other methods (our re-implementations) presented in Table 3.1, Table 3.2, and Table 3.3 are on random class splits with random seed 1997. Here, we conduct experiments using the same class split as in TOPIC [237]. The experimental results on CIFAR-100, *mini*ImageNet, and CUB-200-2011 are presented in Table 3.9, Table 3.10, and Table 3.11 respectively. The results show that the Baseline and our F2M still consistently outperform other methods. Note that on CUB-200-2011, joint-training outperforms the Baseline and our F2M. The reasons may include: 1) The data imbalance issue is not very significant since the average number of images per class of this dataset is relatively small (about 30); and 2) During the base training stage, we use a smaller learning rate (e.g., 0.001) for the embedding network (pretrained on ImageNet) and a higher learning rate (e.g., 0.01) for the classifier.

3.6 Chapter Review

We propose an innovative approach to address the challenge of catastrophic forgetting in few-shot continual learning. Specifically, during the training phase of the

Table 3.9: Classification accuracy on CIFAR-100 for 5-way 5-shot incremental learning with the same class split as in TOPIC [6]. * indicates our re-implementation.

Method	**sessions**									**The gap with cRT**
	1	2	3	4	5	6	7	8	9	
cRT [8]*	72.28	69.58	65.16	61.41	58.83	55.87	53.28	51.38	49.51	-
Joint-training*	72.28	68.40	63.31	59.16	55.73	52.81	49.01	46.74	44.34	-5.17
Baseline	72.28	68.01	64.18	60.56	57.44	54.69	52.98	50.80	48.70	-0.81
iCaRL [4]*	72.05	65.35	61.55	57.83	54.61	51.74	49.71	47.49	45.03	-4.48
Rebalance [2]*	**74.45**	67.74	62.72	57.14	52.78	48.62	45.56	42.43	39.22	-10.29
FSLL [3]*	72.28	63.84	59.64	55.49	53.21	51.77	50.93	48.94	46.96	-2.55
iCaRL [4]	64.10	53.28	41.69	34.13	27.93	25.06	20.41	15.48	13.73	-35.78
Rebalance [2]	64.10	53.05	43.96	36.97	31.61	26.73	21.23	16.78	13.54	-35.97
TOPIC [6]	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	-20.14
FSLL [3]	64.10	55.85	51.71	48.59	45.34	43.25	41.52	39.81	38.16	-11.35
FSLL+SS [3]	66.76	55.52	52.20	49.17	46.23	44.64	43.07	41.20	39.57	-9.94
F2M	71.45	**68.10**	**64.43**	**60.80**	**57.76**	**55.26**	**53.53**	**51.57**	**49.35**	**-0.16**

Table 3.10: Classification accuracy on *mini*ImageNet for 5-way 5-shot incremental learning with the same class split as in TOPIC [6]. * indicates our re-implementation.

Method	**sessions**									**The gap with cRT**
	1	2	3	4	5	6	7	8	9	
cRT [8]*	72.08	68.15	63.06	61.12	56.57	54.47	51.81	49.86	48.31	-
Joint-training*	72.08	67.31	62.04	58.51	54.41	51.53	48.70	45.49	43.88	-4.43
Baseline	72.08	66.29	61.99	58.71	55.73	53.04	50.40	48.59	47.31	-1.0
iCaRL [4]*	71.77	61.85	58.12	54.60	51.49	48.47	45.90	44.19	42.71	-5.6
Rebalance [2]*	**72.30**	66.37	61.00	56.93	53.31	49.93	46.47	44.13	42.19	-6.12
FSLL [3]*	72.08	59.04	53.75	51.17	49.11	47.21	45.35	44.06	43.65	-4.66
iCaRL [4]	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	-31.10
Rebalance [2]	61.31	47.80	39.31	31.91	25.68	21.35	18.67	17.24	14.17	-34.14
TOPIC [6]	61.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	-23.89
FSLL [3]	66.48	61.75	58.16	54.16	51.10	48.53	46.54	44.20	42.28	-6.03
FSLL+SS [3]	68.85	63.14	59.24	55.23	52.24	49.65	47.74	45.23	43.92	-4.39
F2M	72.05	**67.47**	**63.16**	**59.70**	**56.71**	**53.77**	**51.11**	**49.21**	**47.84**	**-0.43**

Table 3.11: Classification accuracy on CUB-200-2011 for 10-way 5-shot incremental learning with the same class split as in TOPIC [6]. * indicates our re-implementation.

Method	**sessions**											**The gap with cRT**
	1	2	3	4	5	6	7	8	9	10	11	
cRT [8]*	77.16	74.41	71.31	68.08	65.57	63.08	62.44	61.29	60.12	59.85	59.30	-
Joint-training*	77.16	74.39	69.83	67.17	64.72	62.25	59.77	59.05	57.99	57.81	56.82	-2.48
Baseline	77.16	**74.00**	70.21	66.07	63.90	61.35	60.01	58.66	56.33	56.12	55.07	-4.23
iCaRL [4]*	75.95	60.90	57.65	54.51	50.83	48.21	46.95	45.74	43.21	43.01	41.27	-18.03
Rebalance [2]*	**77.44**	58.10	50.15	44.80	39.12	34.44	31.73	29.75	27.56	26.93	25.30	-34.00
FSLL [3]*	77.16	71.85	66.53	59.95	58.01	57.00	56.06	54.78	52.24	52.01	51.47	-7.83
iCaRL [4]	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	-39.92
Rebalance [2]	68.68	57.12	44.21	28.78	26.71	25.66	24.62	21.52	20.12	20.06	19.87	-41.21
TOPIC [6]	68.68	62.49	54.81	49.99	45.25	41.40	38.35	35.36	32.22	28.31	26.28	-34.80
FSLL [3]	72.77	69.33	65.51	62.66	61.10	58.65	57.78	57.26	55.59	55.39	54.21	-6.87
FSLL+SS [3]	75.63	71.81	68.16	64.32	62.61	60.10	58.82	58.70	56.45	56.41	55.82	-5.26
F2M	77.13	73.92	**70.27**	**66.37**	**64.34**	**61.69**	**60.52**	**59.38**	**57.15**	**56.94**	**55.89**	**-3.41**

base model, we identify flat local minima of the objective function. When new tasks are introduced, we fine-tune the model within these flat regions to mitigate catastrophic forgetting. In Section 3.4.3, we provide a theoretical proof that our algorithm converges to a flat local minimum. We further demonstrate the effectiveness of our method through extensive experiments on various datasets. However, F2M has the following limitations.

Limitations in scenarios with a large number of new samples. Our algorithm may struggle when tasks introduce a large number of new samples. This is because the flat region identified during base model training may be relatively narrow, limiting the model's capacity to accommodate substantial updates. Nevertheless, the core idea of F2M provides valuable insights for the continual learning community. Future work may explore identifying broader—albeit less flat—local minima during initial training, enabling more flexible adaptation during incremental learning. Additionally, regularization techniques such as Elastic Weight Consolidation (EWC) could

be integrated to constrain parameter updates in this wider region.

Need for improved methods to search for flat local minima. F2M ensures flatness by adding random noise to the model parameters and requiring the perturbed models to maintain comparable performance to the original parameters θ. While this strategy can identify flat regions, it imposes a strong constraint—demanding an excessively flat landscape around the solution—which may hinder convergence to a good local optimum. Consequently, the base model trained under this constraint may experience slight performance degradation compared to standard training. In contrast, optimization-based methods like Sharpness-Aware Minimization (SAM) [64] offer a more flexible approach to locating flat local minima and can mitigate such issues more effectively, making them a promising alternative.

Chapter 4

Reducing Conflicting Gradient For Multi-Task Learning

4.1 Introduction

Multi-task learning (MTL) is a learning paradigm in which multiple different but correlated tasks are jointly trained with a shared model [21], in the hope of achieving better performance with an overall smaller model size than learning each task independently. By discovering shared structures across tasks and leveraging domain-specific training signals of related tasks, MTL can achieve efficiency and effectiveness. Indeed, MTL has been successfully applied in many domains including natural language processing [80], reinforcement learning [186, 45] and computer vision [248].

A major challenge for multi-task learning is *negative transfer* [209], which refers to the performance drop on a task caused by the learning of other tasks, resulting in worse overall performance than learning them separately. This is caused by *task conflicts*, i.e., tasks compete with each other and unrelated information of individual tasks may impede the learning of common structures. From the optimization point of view, a cause of negative transfer is *conflicting gradients* [279], which refers to two

task gradients pointing away from each other and the update of one task will have a negative effect on the other. Conflicting gradients make it difficult to optimize the multi-task objective, since task gradients with larger magnitude may dominate the update vector, making the optimizer prioritize some tasks over others and struggle to converge to a desirable solution.

Prior works address task/gradient conflicts mainly by balancing the tasks via task reweighting or gradient manipulation. Task reweighting methods adaptively re-weight the loss functions by homoscedastic uncertainty [114], balancing the pace at which tasks are learned [35, 146], or learning a loss weight parameter [142]. Gradient manipulation methods reduce the influence of conflicting gradients by directly altering the gradients based on different criteria [217, 279, 36, 140] or rotating the shared features [104]. While these methods have demonstrated effectiveness in different scenarios, in our empirical study, we find that they cannot reduce the occurrence of conflicting gradients (see Sec. 4.3.3 for more discussion).

We propose a different approach to reduce conflicting gradients for MTL. Specifically, we investigate layer-wise conflicting gradients, i.e., the task gradients w.r.t. each shared network layer. We first train the network with a regular MTL algorithm (e.g., joint-training) for a number of iterations, compute the conflict scores for all shared layers, and select those with highest conflict scores (indicating severe conflicts). We then set the selected shared layers task-specific and train the modified network from scratch. As demonstrated by comprehensive experiments and analysis, our simple approach Recon has the following key advantages: **(1)** Recon can greatly reduce conflicting gradients with only a slight increase in model parameters (less than 1% in some cases) and lead to significantly better performance. **(2)** Recon can be easily applied to improve various gradient manipulation methods and branched architecture search methods. Given a network architecture, it only needs to search for the conflict layers once, and the network can be modified to be used with different methods and even on different datasets to gain performance improvement. **(3)** Recon can achieve

better performance than branched architecture search methods with a much smaller model.

4.2 Related Works

In this section, we briefly review related works in multi-task learning in four categories: tasks clustering, architecture design, architecture search, and task balancing. For a comprehensive review, please refer to [295, 248]. Tasks clustering methods mainly focus on identifying which tasks should be learned together [241, 283, 227, 220, 61].

Architecture design methods include *hard parameter sharing* methods [117, 153, 16], which learn a common feature extractor and task-specific decoders, and *soft parameters sharing* methods [170, 210, 66, 67], in which each task has a portion of parameters to do cross-task talk through some sharing mechanism. Differently, MTAN [146] adopts task-specific attention to extract shared information for each task. Compared with soft parameters sharing methods, our approach Recon has much better scalability when dealing with a large number of tasks.

Instead of designing a fixed network structure, some methods [208, 169, 270] propose to dynamically self-organize the network for different tasks. Among them, **branched architecture search** [77, 18] methods are more related to our work. They proposes an automated architecture search algorithm to build a tree-structured network by learning the branch position of the network. In contrast, our method Recon decides which layers to share across tasks by considering the severity of layer-wise conflicting gradients, which leads to a better and more compact architecture with lower time cost.

Another line of works is **task balancing** methods. To address task/gradient conflicts, some methods attempt to re-weight the multi-task loss function using homoscedastic uncertainty [114], task prioritization [76], or similar learning pace [146, 142]. Grad-

Norm [35] learns a weight parameter to ensure the similar learning paces across tasks. MGDA [217] adopts Frank-Wolfe algorithm to find the weights such that the weighted sum of task gradients has a minimum norm. To reduce the influence of conflicting gradients, PCGrad [279] projects each gradient onto the normal plane of another gradient and uses the average projected gradient as the update vector. Graddrop [36] randomly drops the elements of gradients based on element-wise conflict. CAGrad [140] ensures convergence to a minimum of the average loss by gradient manipulation. RotoGrad [104] reduces the influence of conflicting gradients by rotating the shared feature space. Instead of manipulating gradients, our method Recon leverages gradient information to modify network structure to mitigate task conflicts from the root.

4.3 Pilot Study: Task Conflicts in Multi-Task Learning

4.3.1 Multi-task Learning: Problem Definition

Multi-task learning aims to jointly learn a set of functions $\{f_{\theta_i}\}_{i=1}^{N}$, typically sharing parameters θ_{shared}

Multi-task learning (MTL) aims to learn a set of correlated tasks $\{\mathcal{T}_i\}_{i=1}^{T}$ simultaneously. Each task $\mathcal{T}_i$ is associated with its own dataset $\mathcal{D}_i = \{(x_i^{(j)}, y_i^{(j)})\}_{j=1}^{n_i}$, where $x_i^{(j)} \in \mathcal{X}_i$ is the input and $y_i^{(j)} \in \mathcal{Y}_i$ is the corresponding label. For each task $\mathcal{T}_i$, the empirical loss function is defined as $\mathcal{L}_i(f_{\boldsymbol{\theta}_{\text{sh}},\boldsymbol{\theta}_i}(x_i), y_i)$, where $\boldsymbol{\theta}_{\text{sh}}$ denotes the parameters shared across all tasks, and $\boldsymbol{\theta}_i$ represents the task-specific parameters. For simplicity, we denote the loss function as $\mathcal{L}_i(\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_i)$. The goal is to find optimal parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_T\}$ to achieve high performance across all tasks.

Formally, it aims to minimize a multi-task objective:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i}^{T} w_i \mathcal{L}_i(\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_i), \tag{4.1}$$

where w_i are pre-defined or dynamically computed weights for different tasks. A popular choice is to use the average loss (i.e., equal weights). However, optimizing the multi-task objective is difficult, and a known cause is conflicting gradients.

4.3.2 Conflicting Gradients

Let $\mathbf{g}_i = \nabla_{\boldsymbol{\theta}_{\text{sh}}} \mathcal{L}_i(\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_i)$ denote the gradient of task $\mathcal{T}_i$ w.r.t. the shared parameters $\boldsymbol{\theta}_{\text{sh}}$ (i.e., a vector of the partial derivatives of $\mathcal{L}_i$ w.r.t. $\boldsymbol{\theta}_{\text{sh}}$) and $g_i^{\text{ts}} = \nabla_{\boldsymbol{\theta}_i} \mathcal{L}_i(\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_i)$ denote the gradient w.r.t. the task-specific parameters $\boldsymbol{\theta}_i$. A small change of $\boldsymbol{\theta}_{\text{sh}}$ in the direction of negative $\mathbf{g}_i$ is $\boldsymbol{\theta}_{\text{sh}} \leftarrow \boldsymbol{\theta}_{\text{sh}} - \alpha \mathbf{g}_i$, with a sufficiently small step size α. The effect of this change on the performance of another task $\mathcal{T}_j$ is measured by:

$$\Delta\mathcal{L}_j = \mathcal{L}_j(\boldsymbol{\theta}_{\text{sh}} - \alpha \mathbf{g}_i, \boldsymbol{\theta}_j) - \mathcal{L}_j(\boldsymbol{\theta}_{\text{sh}}, \boldsymbol{\theta}_j) = -\alpha \mathbf{g}_i \cdot \mathbf{g}_j + o(\alpha), \tag{4.2}$$

where the second equality is obtained by first order Taylor approximation. Likewise, the effect of a small update of $\boldsymbol{\theta}_{\text{sh}}$ in the direction of the negative gradient of task $\mathcal{T}_j$ (i.e., $-\mathbf{g}_j$) on the performance of task $\mathcal{T}_i$ is $\Delta\mathcal{L}_i = -\alpha \mathbf{g}_i \cdot \mathbf{g}_j + o(\alpha)$. Notably, the model update for task $\mathcal{T}_i$ is considered to have a negative effect on task $\mathcal{T}_j$ when $\mathbf{g}_i \cdot \mathbf{g}_j < 0$, since it increases the loss of task $\mathcal{T}_j$, and vice versa. A formal definition of conflicting gradients is given as follows [279].

Definition 4.1 (Conflicting Gradients)**.** *The gradients* $\mathbf{g}_i$ *and* $\mathbf{g}_j (i \neq j)$ *are said to be conflicting with each other if* $\cos\phi_{ij} < 0$*, where* ϕ_{ij} *is the angle between* $\mathbf{g}_i$ *and* $\mathbf{g}_j$*.*

As shown in [279], conflicts in gradient pose serious challenges for optimizing the multi-task objective (Eq. 4.1). Using the average gradient (i.e., $\frac{1}{T}\sum_{i=1}^{T} \mathbf{g}_i$) for gradient decent may hurt the performance of individual tasks, especially when there is a large difference in gradient magnitudes, which will make the optimizer struggle to converge to a desirable solution.

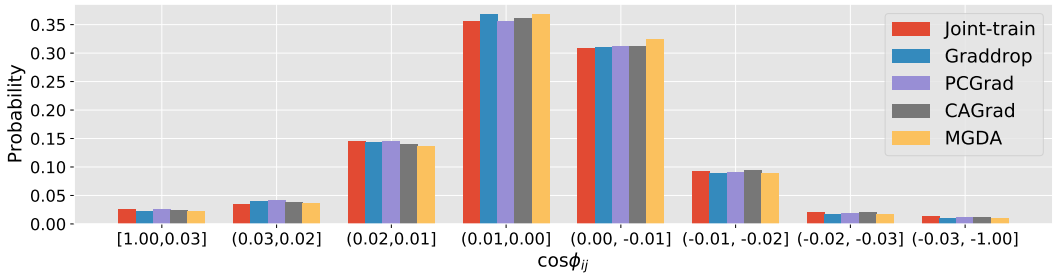


Figure 4.1: The distributions of gradient conflicts (in terms of $\cos\phi_{ij}$) of the joint-training baseline and state-of-the-art gradient manipulation methods on Multi-Fashion+MNIST benchmark.

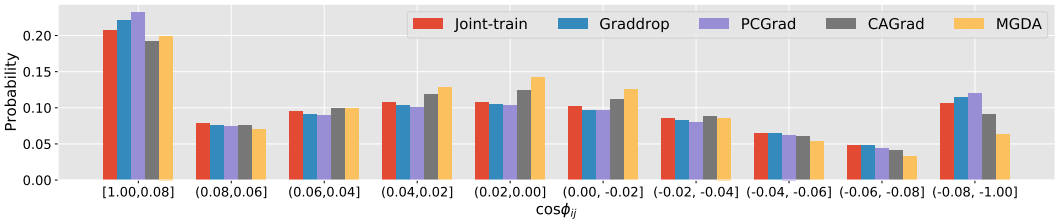


Figure 4.2: The distributions of gradient conflicts (in terms of $\cos\phi_{ij}$) of the joint-training baseline and state-of-the-art gradient manipulation methods on CityScapes dataset.

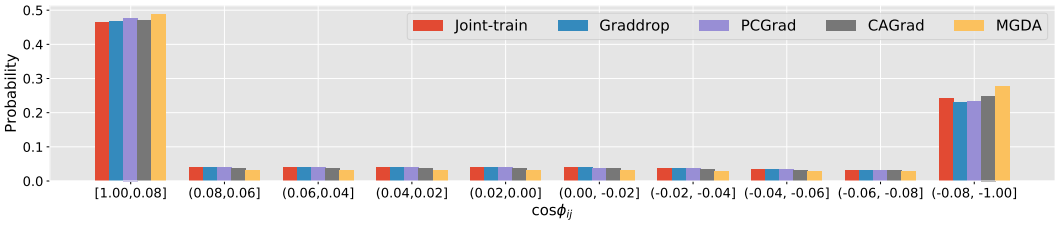


Figure 4.3: The distributions of gradient conflicts (in terms of $\cos\phi_{ij}$) of the joint-training baseline and state-of-the-art gradient manipulation methods on NYUv2 dataset.

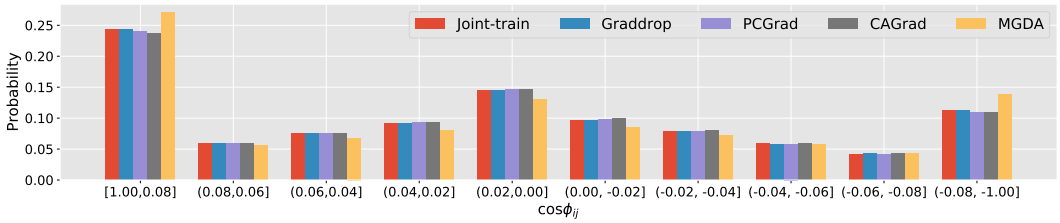


Figure 4.4: The distributions of gradient conflicts (in terms of $\cos\phi_{ij}$) of the joint-training baseline and state-of-the-art gradient manipulation methods on PASCAL-Context dataset.

4.3.3 Gradient Surgery Cannot Effectively Reduce Conflicting Gradients

To mitigate the influence of conflicting gradients, several methods [279, 36, 140] have been proposed to perform "gradient surgery". Instead of following the average gradient direction, they alter conflicting gradients based on some criteria and use the modified gradients for model update. We conduct a pilot study to investigate whether gradient manipulation can effectively reduce the occurrence of conflicting gradients. For each training iteration, we first calculate the task gradients of all tasks w.r.t. the shared parameters (i.e., $\mathbf{g}_i$ for any task i) and compute the conflict angle between any two task gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ in terms of $cos\phi_{ij}$. We then count and draw the distribution of $cos\phi_{ij}$ in all training iterations. We provide the statistics of the joint-training baseline (i.e., training all tasks jointly with equal loss weights and all parameters shared) and several state-of-the-art gradient manipulation methods including GradDrop [36], PCGrad [279], CAGrad [140], and MGDA [217] on Multi-Fashion+MNIST [137], CityScapes, NYUv2, and PASCAL-Context datasets. The results are provided in Fig. 4.1, Fig. 4.2, Fig. 4.3, Fig. 4.4, Table 4.11, and Tables 4.12-4.14. It can be seen that gradient manipulation methods can only slightly reduce the occurrence of conflicting gradients (compared to joint-training) in some cases, and in some other cases they even increase it.

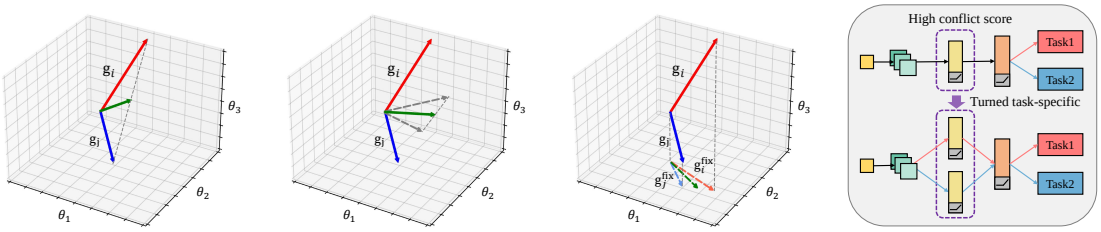
Why can't gradient surgery effectively reduce conflicting gradients? Gradient manipulation methods aim to solve an optimization problem by ensuring that gradient updates are not biased toward a specific task, thereby promoting balanced learning across all tasks. However, a fundamental issue remains: due to the inherent conflicts among tasks, such methods cannot completely eliminate gradient conflicts. Even if they manage to reduce the negative impact of such conflicts, they fail to prevent them entirely. As a result, the angles between task gradients may still remain large, which hinders effective learning for each individual task.

4.4 Methodology

Our pilot study shows that adjusting gradients for model update cannot effectively prevent the occurrence of conflicting gradients in MTL, which suggests that the root causes of this phenomenon may be closely related to the nature of different tasks and the way how model parameters are shared among them. Therefore, to mitigate task conflicts for MTL, in this paper, we take a different approach to reduce the occurrence of conflicting gradients from the root.

4.4.1 Recon: Removing Layer-wise Conflicting Gradients

Our approach is extremely simple and intuitive. We first identify the shared network layers where conflicts occur most frequently and then turn them into task-specific parameters, as shown in Fig. 4.5. Suppose the shared model parameters $\boldsymbol{\theta}_{\text{sh}}$ are composed of n layers, i.e., $\boldsymbol{\theta}_{\text{sh}} = \{\boldsymbol{\theta}_{\text{sh}}^{(k)}\}_{k=1}^{n}$, where $\boldsymbol{\theta}_{\text{sh}}^{(k)}$ is the k^{th} shared layer. Let $\mathbf{g}_i^{(k)}$ denote the gradient of task $\mathcal{T}_i$ w.r.t. the k^{th} shared layer $\boldsymbol{\theta}_{\text{sh}}^{(k)}$, i.e., $\mathbf{g}_i^{(k)}$ is a vector of the partial derivatives of $\mathcal{L}_i$ w.r.t. the parameters of $\boldsymbol{\theta}_{\text{sh}}^{(k)}$. Let $\phi_{ij}^{(k)}$ denote the angle between $\mathbf{g}_i^{(k)}$ and $\mathbf{g}_j^{(k)}$. We define layer-wise conflicting gradients and S-conflict score as follows.



(a) Joint-train (b) PCGrad (c) Recon (d) Recon

Figure 4.5: Illustration of the differences between joint-training, gradient manipulation, and our approach. (a) In joint-training, the update vector (in green) is the average gradient $\frac{1}{2}(\mathbf{g}_i + \mathbf{g}_j)$. Due to the conflict between $\mathbf{g}_i$ and $\mathbf{g}_j$, the update vector is dominated by $\mathbf{g}_i$ (in red). (b) PCGrad [279] projects each gradient onto the normal plane of the other one and uses the average of the projected gradients (indicated by dashed grey arrows) as the update vector (in green). As such, the update vector is less dominated by $\mathbf{g}_i$. (c) Our approach Recon finds the parameters contributing most (e.g., θ_3) to gradient conflicts and turns them into task specific ones. In effect, it performs an orthographic/coordinate projection of conflicting gradients to the space of the rest parameters (e.g., θ_1 and θ_2) such that the projected gradients $\mathbf{g}_i^{\text{fix}}$ and $\mathbf{g}_j^{\text{fix}}$ are better aligned. (d) Illustration of Recon turning a shared layer with high conflict score to task-specific layers.

Definition 4.2 (Layer-wise Conflicting Gradients)**.** *The gradients* $\mathbf{g}_i^{(k)}$ *and* $\mathbf{g}_j^{(k)}$ *(*$i \neq j$*) are said to be conflicting with each other if* $\cos\phi_{ij}^{(k)} < 0$.

Definition 4.3 (S-Conflict Score)**.** *For any* $-1 < S \leq 0$*, the* S*-conflict score for the* k^{th} *shared layer is the number of different pairs* $(i, j)(i \neq j)$ *s.t.* $\cos\phi_{ij}^{(k)} < S$*, denoted as* $s^{(k)}$.

S represents the severity level of gradient conflicts; a smaller S value indicates a focus on more severe conflict cases. The S-conflict score $s^{(k)}$ quantifies how often conflicting gradients occur at severity level S for the k^{th} shared layer. If $s^{(k)} = \binom{T}{2}$, this implies that every pair of tasks exhibits a conflict in their gradients with respect

Algorithm 2: Recon: Removing Layer-wise Conflicting Gradients

Input: Model parameters θ, learning rate α, a set of tasks $\{\mathcal{T}_i\}_{i=1}^{T}$, number of iterations I for computing conflict scores, conflict severity level S, number of selected layers K.

// Train the network and compute conflict scores for all layers

for iteration $i = 1, 2, \ldots, I$ **do**

  for $i = 1, 2, \ldots, T$ **do**

    Compute the gradients of task $\mathcal{T}_i$ w.r.t. all shared layers, i.e., $\{\mathbf{g}_i^{(k)}\}_{k=1}^{n}$;

  end

  Calculate the S-conflict scores for all shared layers in the current iteration, i.e., $\{s_i^{(k)}\}_{k=1}^{n}$;

  Update θ with joint-training or any gradient manipulation method ;

end

// Set layers with top conflict scores task-specific

For each layer k, calculate the sum of S-conflict scores in all iterations, i.e., $s^{(k)} = \sum_{i=1}^{I} s_i^{(k)}$;

Select the top K layers with highest $s^{(k)}$ and set them task-specific;

// Train the modified network from scratch

for iteration $i = 1, 2, \ldots$ **do**

  Update θ with joint-training or any gradient manipulation method;

end

Output: Model parameters θ.

to the k^{th} layer. By computing S-conflict scores, we can identify the shared layers where conflicts are most frequent.

We present our method, Recon, in Algorithm 2. We begin by training the network for I iterations and compute the S-conflict score $s_i^{(k)}$ for each shared layer $\boldsymbol{\theta}^{(k)}$ at every iteration i. The overall conflict score for each layer is then obtained by summing across all iterations: $s^{(k)} = \sum_{i=1}^{I} s_i^{(k)}$. We identify the layers with the highest $s^{(k)}$ values and designate them as task-specific. The modified network is then retrained from scratch.

Why count the number of conflicting task pairs instead of directly using $\phi_{ij}^{(k)}$ as the conflict score? Using the sum of $\phi_{ij}^{(k)}$ to quantify conflict severity often leads to unstable rankings of conflicting layers across different random seeds. In contrast, our approach focuses on the frequency of conflict occurrences, resulting in more consistent and robust layer rankings. This design also aligns with intuition: detecting the presence of a conflict is simpler and more stable than measuring its exact severity, which can be sensitive to randomness during training.

We validate the effectiveness of Recon through both theoretical analysis (Section 4.4.2) and extensive experiments (Section 4.5). Results show that Recon significantly reduces gradient conflicts in the remaining shared layers and achieves substantial improvements over state-of-the-art methods.

4.4.2 Theoretical Analysis

Here, we provide a theoretical analysis of Recon. Let $\boldsymbol{\theta}_{\text{sh}} = \{\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}, \boldsymbol{\theta}_{\text{sh}}^{\text{cf}}\}$, where $\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}$ are the remaining shared parameters, and $\boldsymbol{\theta}_{\text{sh}}^{\text{cf}}$ are those that will be turned to task-specific parameters $\boldsymbol{\theta}_1^{\text{cf}}, \boldsymbol{\theta}_2^{\text{cf}}, \cdots, \boldsymbol{\theta}_T^{\text{cf}}$. Notice that $\boldsymbol{\theta}_1^{\text{cf}}, \boldsymbol{\theta}_2^{\text{cf}}, \cdots, \boldsymbol{\theta}_T^{\text{cf}}$ will all be initialized with $\boldsymbol{\theta}_{\text{sh}}^{\text{cf}}$. Therefore, after applying Recon, the model parameters are $\boldsymbol{\theta}_r = \{\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}, \boldsymbol{\theta}_1^{\text{cf}}, \ldots, \boldsymbol{\theta}_T^{\text{cf}}, \boldsymbol{\theta}_1^{\text{ts}}, \ldots, \boldsymbol{\theta}_T^{\text{ts}}\}$. An one-step gradient update of $\boldsymbol{\theta}_r$ is:

$$\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}} = \boldsymbol{\theta}_{\text{sh}}^{\text{fix}} - \alpha \sum_{i=1}^{T} w_i \mathbf{g}_i^{\text{fix}}, \quad \hat{\boldsymbol{\theta}}_i^{\text{cf}} = \boldsymbol{\theta}_i^{\text{cf}} - \alpha \mathbf{g}_i^{\text{cf}}, \quad \hat{\boldsymbol{\theta}}_i^{\text{ts}} = \boldsymbol{\theta}_i^{\text{ts}} - \alpha \mathbf{g}_i^{\text{ts}}, \quad i = 1, \ldots, T, \tag{4.3}$$

where w_i are weight parameters, $\mathbf{g}_i^{\text{ts}} = \nabla_{\boldsymbol{\theta}_i^{\text{ts}}} \mathcal{L}_i$, $\mathbf{g}_i^{\text{cf}} = \nabla_{\boldsymbol{\theta}_{\text{sh}}^{\text{cf}}} \mathcal{L}_i$ and $\mathbf{g}_i^{\text{fix}} = \nabla_{\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}} \mathcal{L}_i$. Notice that different methods such as joint-training, MGDA [217], PCGrad [279], and CAGrad [140] choose different w_i dynamically.

Without applying Recon, the model parameters are $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}, \boldsymbol{\theta}_{\text{sh}}^{\text{cf}}, \boldsymbol{\theta}_1^{\text{ts}}, \ldots, \boldsymbol{\theta}_T^{\text{ts}}\}$. An one-step gradient update of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}} = \boldsymbol{\theta}_{\text{sh}}^{\text{fix}} - \alpha \sum_{i=1}^{T} w_i \mathbf{g}_i^{\text{fix}}, \quad \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}} = \boldsymbol{\theta}_{\text{sh}}^{\text{cf}} - \alpha \sum_{i=1}^{T} w_i \mathbf{g}_i^{\text{cf}}, \quad \hat{\boldsymbol{\theta}}_i^{\text{ts}} = \boldsymbol{\theta}_i^{\text{ts}} - \alpha \mathbf{g}_i^{\text{ts}}, \quad i = 1, \ldots, T. \tag{4.4}$$

After the one-step updates, the loss functions with the updated parameters $\hat{\boldsymbol{\theta}}_r$ and $\hat{\boldsymbol{\theta}}$ respectively are:

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_r) = \sum_{i=1}^{T} \mathcal{L}_i \left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_i^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}} \right), \text{ and, } \mathcal{L}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{T} \mathcal{L}_i \left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}} \right), \tag{4.5}$$

where $\mathcal{L}_i$ is the loss function of task $\mathcal{T}_i$. Denote the set of indices of the layers turned task-specific by $\mathbb{P}$, then $\boldsymbol{\theta}_{\text{sh}}^{\text{cf}} = \{\boldsymbol{\theta}_{\text{sh}}^{(k)}\}, k \in \mathbb{P}$. Assume that $\sum_{i=1}^{T} w_i = 1$, then we have the following theorem.

Theorem 4.1. *Assume that $\mathcal{L}$ is differentiable and for any two different tasks $\mathcal{T}_i$ and $\mathcal{T}_j$, it satisfies*

$$\cos \phi_{ij}^{(k)} \|\mathbf{g}_i^{(k)}\| < \|\mathbf{g}_j^{(k)}\|, \quad \forall k \in \mathbb{P} \tag{4.6}$$

then for any sufficiently small learning rate $\alpha > 0$,

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_r) < \mathcal{L}(\hat{\boldsymbol{\theta}}). \tag{4.7}$$

The theorem indicates that a single gradient update on the model parameters of Recon achieves lower loss than that on the original model parameters. The proof is provided as follows:

Proof. We consider the first order Taylor approximation of $\mathcal{L}_i$. For normal update, we have

$$\begin{aligned} \mathcal{L}_i \left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}} \right) =& \mathcal{L}_i \left(\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}, \boldsymbol{\theta}_{\text{sh}}^{\text{cf}}, \boldsymbol{\theta}_i^{\text{ts}} \right) + (\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}} - \boldsymbol{\theta}_{\text{sh}}^{\text{fix}})^\top \mathbf{g}_i^{\text{fix}} && (4.8) \\ &+ (\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}} - \boldsymbol{\theta}_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + (\hat{\boldsymbol{\theta}}_i^{\text{ts}} - \boldsymbol{\theta}_i^{\text{ts}})^\top \mathbf{g}_i^{\text{ts}} + o(\alpha). && (4.9) \end{aligned}$$

For Recon update, we have

$$\begin{aligned} \mathcal{L}_i \left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_i^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}} \right) =& \mathcal{L}_i \left(\boldsymbol{\theta}_{\text{sh}}^{\text{fix}}, \boldsymbol{\theta}_{\text{sh}}^{\text{cf}}, \boldsymbol{\theta}_i^{\text{ts}} \right) + (\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}} - \boldsymbol{\theta}_{\text{sh}}^{\text{fix}})^\top \mathbf{g}_i^{\text{ts}} && (4.10) \\ &+ (\hat{\boldsymbol{\theta}}_i^{\text{cf}} - \boldsymbol{\theta}_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + (\hat{\boldsymbol{\theta}}_i^{\text{ts}} - \boldsymbol{\theta}_i^{\text{ts}})^\top \mathbf{g}_i^{\text{ts}} + o(\alpha). && (4.11) \end{aligned}$$

The difference between the two loss functions after the update is

$$\mathcal{L}_i\left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_i^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}}\right) - \mathcal{L}_i\left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}}\right) = (\hat{\boldsymbol{\theta}}_i^{\text{cf}} - \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) \tag{4.12}$$

$$= -\alpha \left(\mathbf{g}_i^{\text{cf}} - \sum_{j=1}^{T} w_j \mathbf{g}_j^{\text{cf}}\right)^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) \tag{4.13}$$

$$= -\alpha \sum_{j=1}^{T} w_j \left(\mathbf{g}_i^{\text{cf}} - \mathbf{g}_j^{\text{cf}}\right)^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) \tag{4.14}$$

$$= -\alpha \sum_{j=1}^{T} w_j \left(\|\mathbf{g}_i^{\text{cf}}\|^2 - \mathbf{g}_j^{\text{cf}\top} \mathbf{g}_i^{\text{cf}}\right) + o(\alpha). \tag{4.15}$$

Assume, without loss of generality, that $\|\mathbf{g}_i^{cf}\| \neq 0$, then

$$\left\|\mathbf{g}_i^{\text{cf}}\right\|^2 - \mathbf{g}_j^{\text{cf}\top} \mathbf{g}_i^{\text{cf}} = \sum_{k \in \mathbb{P}} \left(\left\|\mathbf{g}_i^{(k)}\right\|^2 - \mathbf{g}_i^{(k)\top} \mathbf{g}_j^{(k)}\right) \tag{4.16}$$

$$= \sum_{k \in \mathbb{P}} \left\|\mathbf{g}_i^{(k)}\right\| \left(\left\|\mathbf{g}_i^{(k)}\right\| - \cos\phi_{ij}^{(k)} \left\|\mathbf{g}_j^{(k)}\right\|\right) \tag{4.17}$$

$$> 0. \tag{4.18}$$

Hence, the above difference is negative, if α is sufficiently small. As such, the difference between the multi-task loss functions is also negative, if α is sufficiently small.

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_r) - \mathcal{L}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{T} \mathcal{L}_i\left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_i^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}}\right) - \sum_{i=1}^{T} \mathcal{L}_i\left(\hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{fix}}, \hat{\boldsymbol{\theta}}_{\text{sh}}^{\text{cf}}, \hat{\boldsymbol{\theta}}_i^{\text{ts}}\right) < 0 \tag{4.19}$$

□

4.5 Experiments

In this section, we conduct extensive experiments to evaluate our approach Recon for multi-task learning and demonstrate its effectiveness, efficiency and generality.

4.5.1 Experimental Setup

Datasets. We evaluate Recon on 4 multi-task datasets, namely **Multi-Fashion** plus **MNIST** [137], **CityScapes** [42], **NYUv2** [43], **PASCAL-Context** [171], and

CelebA [152]. The tasks of each dataset are described as follows. 1) Multi-Fashion plus MNIST contains two image classification tasks. Each image consists of an item from FashionMNIST and an item from MNIST. 2) CityScapes contains 2 vision tasks: 7-class semantic segmentation and depth estimation. 3) NYUv2 contains 3 tasks: 13-class semantic segmentation, depth estimation and normal prediction. 4) PASCAL-Context consists of 5 tasks: semantic segmentation, human parts segmentation and saliency estimation, surface normal estimation, and edge detection. 5) CelebA contains 40 binary classification tasks.

Baselines. The baselines include 1) single-task learning (single-task): training all tasks independently; 2) joint-training (joint-train): training all tasks together with equal loss weights and all parameters shared; 3) gradient manipulation methods: MGDA [217], PCGrad [279], GradDrop [36], CAGrad [140], RotoGrad [104]; 4) branched architecture search methods: BMTAS [18]; 5) Architecture design methods: Cross-Stitch [170], MMoE [158]. Following [140], we implement Cross-Stitch based on SegNet [9]. For a fair comparison, all methods use same configurations and random seeds. We run all experiments 3 times with different random seeds.

Table 4.1: Multi-task learning results on Multi-Fashion+MNIST dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Single-task	RotoGrad	BMTAS	Joint-train	w/ Recon	MGDA	w/ Recon	PCGrad	w/ Recon	GradDrop	w/ Recon	CAGrad	w/ Recon	MMoE	w/ Recon
T1 Acc↑	98.37	98.10	98.20	97.42	98.13	95.19	**98.33**	97.37	98.30	97.38	98.25	97.47	98.28	98.27	98.25
T2 Acc↑	89.63	88.25	89.71	88.82	89.26	89.46	89.28	88.68	**89.77**	88.57	89.51	88.85	89.65	89.51	89.67
$\Delta m\%\uparrow$	-	-0.91	-0.04	-0.94	-0.33	-1.71	-0.22	-1.04	**0.04**	-1.10	-0.13	-0.90	-0.04	-0.12	-0.04
#P.	85.62	42.81	85.61	42.81	43.43	42.81	43.43	42.81	43.43	42.81	43.43	42.81	43.43	85.62	105.70

Relative task improvement. Following [163], we compute the relative task improvement with respect to the single-task baseline for each task. Given a task $\mathcal{T}_j$, the

Table 4.2: Multi-task learning results on CelebA dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Single-task	Joint-train	w/ Recon	CAGrad	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon
Average Error	8.38	8.33	8.22	8.31	8.23	8.33	**8.20**	8.64	8.36
$\Delta m\% \uparrow$	-	0.55	1.92	0.79	1.74	0.23	**2.13**	-3.14	0.24
#P.	1706.03	43.26	68.03	43.26	68.03	43.26	68.03	43.26	68.03

relative task improvement is $\Delta m_{\mathcal{T}_j} = \frac{1}{K}\sum_{i=1}^{K}(-1)^{l_i}(M_i - S_i)/S_i$, where M_i, S_i refer to metrics for the i^{th} criterion obtained by objective model and single-task model respectively, $l_i = 1$ if a lower value for the criterion is better and 0 otherwise. The average relative task improvement is $\Delta m = \frac{1}{T}\sum_{j=1}^{T}\Delta m_{\mathcal{T}_j}$.

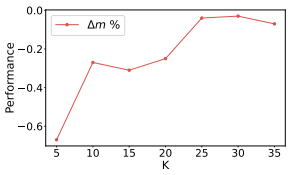
4.5.2 Comparison with the State-of-the-Art

Recon improves the performance of all base models. The main results on Multi-Fashion+MNIST, and CelebA, CityScapes, PASCAL-Context, and NYUv2, are presented in Table 4.1, Table 4.2, Table 4.3, Table 4.4, and Table 4.5 respectively. **(1)** Compared to gradient manipulation methods, Recon consistently improves their performance in most evaluation metrics, and achieve comparable performance on the rest of evaluation metrics. **(2)** Compared with branched architecture search methods and architecture design methods, Recon can further improve the performance of BMTAS and MMoE. Besides, Recon combined with other gradient manipulation methods with small model size can achieve better results than branched architecture search methods with much bigger models.

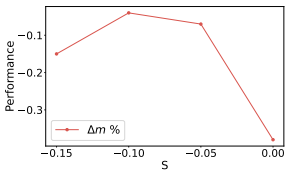
Small increases in model parameters can lead to good performance gains.

Table 4.3: Multi-task learning results on CityScapes dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Segmentation (Higher Better)		Depth (Lower Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err		
Single-task	74.36	93.22	0.0128	29.98		190.59
Cross-Stitch	74.05	93.17	0.0162	116.66	-79.04	190.59
RotoGrad	73.38	92.97	0.0147	82.31	-47.81	103.43
Joint-train	74.13	93.13	0.0166	116.00	-79.32	95.43
w/ Recon	74.17	**93.21**	0.0136	43.18	-12.63	108.44
MGDA	70.74	92.19	0.0130	47.09	-16.22	95.43
w/ Recon	71.01	92.17	**0.0129**	**33.41**	**-4.46**	108.44
Graddrop	74.08	93.08	0.0173	115.79	-80.48	95.43
w/ Recon	74.17	93.11	0.0134	41.37	-10.69	108.44
PCGrad	73.98	93.08	0.02	114.50	-78.39	95.43
w/ Recon	74.18	93.14	0.0136	46.02	-14.92	108.44
CAGrad	73.81	93.02	0.0153	88.29	-53.81	95.43
w/ Recon	74.22	93.10	0.0130	38.27	-7.38	108.44



(a)



(b)

Figure 4.6: The performance of CAGrad combined with Recon on the Multi-Fashion+MNIST benchmark with (a) different number of selected layers K (b) different severity value S for computing conflict scores.

Note that Recon only changes a small portion of shared parameters to task-specific. As shown in Table 4.1-4.5, Recon increases the model size by 0.52% to 57.25%. Recon turns 1.42%, 1.46%, 12.77%, 0.26%, 9.80% shared parameters to task-specific on Multi-Fashion+MNIST, CelebA, CityScapes, NYUv2 and PASCAL-Context respectively. The results suggest that the gradient conflicts in a small portion (less than 13%) of shared parameters impede the training of the model for multi-task learning.

Recon is compatible with various neural network architectures. We use ResNet18 on Multi-Fashion+MNIST, SegNet [9] on CityScapes, MTAN [146] on

Table 4.4: Multi-task learning results on PASCAL-Context dataset with 4-task setting. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates Recon improves the result of the base model. The best average result is marked in bold.

Method	SemSeg (Higher Better)		PartSeg (Lower Better)		saliency (Higher Better)	Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	mIoU	Pix Acc	mIoU	Mean	Median	11.25	22.5		
Single-task	65.00	90.53	59.59	92.61	65.61	14.55	12.36	46.51	81.29		30.09
Joint-train	64.06	90.45	57.91	92.17	62.71	16.40	14.23	39.38	75.93	-4.82	8.04
w/ Recon	64.73	**90.50**	59.00	92.44	66.17	14.99	12.68	44.82	80.11	-0.66	10.20
MGDA	46.05	86.62	54.82	91.39	64.76	15.77	13.54	41.98	77.82	-7.67	8.04
w/ Recon	55.82	87.73	56.31	91.67	64.91	15.12	12.88	44.36	79.81	-4.14	10.20
PCGrad	63.91	90.45	58.01	92.19	63.09	16.34	14.19	39.62	76.06	-4.59	8.04
w/ Recon	**65.02**	90.45	59.22	92.46	66.14	14.95	**12.73**	44.96	80.22	-0.55	10.20
Graddrop	64.14	90.34	57.62	92.12	62.64	16.46	14.28	39.29	75.71	-5.00	8.04
w/ Recon	64.48	90.45	59.08	92.46	**66.23**	14.94	12.72	45.03	80.25	-0.63	10.20
CAGrad	63.37	90.17	57.49	92.07	64.16	16.30	14.12	39.80	76.23	-4.37	8.04
w/ Recon	64.60	90.40	59.27	92.47	65.67	14.92	12.71	45.10	80.33	-0.76	10.20
BMTAS	64.89	90.44	58.87	92.36	63.42	15.66	13.44	42.29	78.14	-2.89	15.18
w/ Recon	64.78	90.46	**59.96**	**92.58**	65.96	**14.74**	**12.57**	**45.62**	**80.84**	**-0.19**	16.83

NYUv2, and MobileNetV2 [215] on PASCAL-Context. Recon improves the performance of baselines with different neural network architectures, including the architecture search method BMTAS [18] which finds a tree-like structure for multi-task learning.

Only one search of conflict layers is needed for the same network architecture. An interesting observation from our experiments is that network architecture seems to be the deciding factor for the conflict layers found by Recon. With the same network architecture (e.g., ResNet18), the found conflict layers are quite consistent w.r.t. (1) different training stages (e.g., the first 25% iterations, or the middle or last

Table 4.5: Multi-task learning results on NYUv2 dataset with MTAN as backbone. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Segmentation (Higher Better)		Depth (Lower Better)		Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)			$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30		
Single-task	38.67	64.27	0.6881	0.2788	24.87	18.99	30.43	57.81	69.70		285.88
Cross-Stitch	**40.45**	66.15	**0.5051**	**0.2134**	27.58	23.00	24.69	49.47	62.36	4.16	285.88
Joint-train	39.48	65.23	0.5491	0.2235	27.87	23.76	22.68	47.91	61.58	0.75	168.72
w/ Recon	39.54	65.20	0.5312	0.2234	26.55	21.40	26.53	52.60	65.31	4.14	169.59
MGDA	29.28	60.30	0.6027	0.2515	**24.89**	**19.32**	**29.85**	**57.18**	**69.38**	-2.26	168.72
w/ Recon	32.82	61.26	0.5884	0.2295	25.17	19.72	28.18	56.49	68.96	0.53	169.59
Graddrop	38.70	64.97	0.5565	0.2333	27.41	23.00	23.79	49.45	62.87	0.49	168.72
w/ Recon	40.14	66.08	0.5265	0.2241	26.51	21.45	26.51	52.48	65.26	4.67	169.59
PCGrad	38.55	65.07	0.54	0.23	26.90	22.05	24.98	51.36	64.41	2.02	168.72
w/ Recon	38.61	65.48	0.5350	0.2271	26.31	21.11	26.90	53.21	65.95	3.87	169.59
CAGrad	39.89	**66.47**	0.5496	0.2281	26.36	21.47	25.50	52.68	65.90	3.74	168.72
w/ Recon	39.92	66.07	0.5320	0.2200	25.80	20.59	27.60	54.31	67.05	**5.80**	169.59

ones) (see Table 4.6 and Table 4.7. (2) different MTL methods (e.g., joint-training or gradient manipulation methods) (see Table 4.8), and (3) different datasets (see Table 4.9 and Table 4.10). Hence, in our experiments, we only search for the conflict layers *once* with the joint-training baseline in the first 25% training iterations and modify the network to improve various methods on the same dataset. We also find that the conflict layers found on one dataset can be used to modify the network to be directly applied on another dataset to gain performance improvement.

Recon finds similar layers in different training stages. Recon ranks the network layers according to the computed S-conflict scores. The ranking result can be

Table 4.6: The distance between the layer permutations (rankings) obtained in different training stages on Multi-Fashion+MNIST dataset. "Iter." denotes iterations.

Training Stage	1st 25% Iter.	2nd 25% Iter.	3rd 25% Iter.	4th 25% Iter.	All Iter.
1st 25% Iter.	0	-	-	-	-
2nd 25% Iter.	**2.39**	0	-	-	-
3rd 25% Iter.	1.85	2.14	0	-	-
4th 25% Iter.	1.95	2.24	0.68	0	-
All Iter.	1.36	1.95	0.82	0.97	0

represented as a layer permutation, denoted as π, and $\pi(l)$ is the position of layer l. The similarity between two rankings π_i and π_j can be measured as:

$$d(\pi_i, \pi_j) = \frac{1}{|\mathbb{L}|} \sum_{l \in \mathbb{L}} |\pi_i(l) - \pi_j(l)|, \tag{4.20}$$

where $\mathbb{L}$ denotes the set of neural network layers. In Table 4.6, we measure the differences in rankings obtained in different training stages (e.g., in the first 25% iterations or the second 25% iterations) on Multi-Fashion+MNIST by Eq. 4.20. The small distances (less than 2.4) indicate that the layers found in different training stages are quite similar. In Table 4.7, we compare the performance of the networks modified by Recon with conflict layers found in different training stages on CityScapes. It can be seen that the results of the last three rows are the same, which is because the layers found in the 3rd 25% iterations, 4th 25% iterations, and all iterations are *exactly the same* (the rankings may be slightly different though). The layers found in the later stages lead to slightly better performance than those found in the early stages (i.e., 1st 25% iterations and 2nd 25% iterations), indicating the conflict scores in early iterations might be a little noisy. However, since the performance gaps are acceptably small, to save time, we use the initial 25% training iterations to find conflict layers.

Recon finds similar layers with different MTL methods. In Table 4.8, we

Table 4.7: Performance of the networks modified by Recon with conflict layers found in different training stages of joint-training on CityScapes dataset. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The best result is marked in bold.

Model	Segmentation (Higher Better)		Depth (Lower Better)		$\Delta m\%$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err		
Single-task	74.36	93.22	0.0128	29.98		190.59
1st 25% Iterations	74.17	93.21	0.0136	43.18	-12.63	108.439
2nd 25% Iterations	74.20	93.19	0.0135	42.45	-11.83	108.440
3rd 25% Iterations	**74.80**	93.19	**0.0136**	**41.34**	-10.90	109.567
4th 25% Iterations	**74.80**	93.19	**0.0136**	**41.34**	-10.90	109.567
All Iterations	**74.80**	93.19	**0.0136**	**41.34**	-10.90	109.567

Table 4.8: The distance between the layer permutations (rankings) obtained by Recon with different methods on Multi-Fashion+MNIST dataset.

Method	Joint-train	CAGrad	PCGrad	Gradrop	MGDA
Joint-train	0	-	-	-	-
CAGrad	1.07	0	-	-	-
PCGrad	0.78	1.17	0	-	-
Gradrop	0.59	0.83	0.68	0	-
MGDA	1.71	1.32	**1.90**	1.56	0

measure the differences in layer permutations (rankings) obtained by Recon with different methods (e.g., CAGrad and PCGrad) on Multi-Fashion+MNIST by Eq. 4.20. The small distances (less than 1.9) indicate that the layers found by Recon with different methods are quite similar. Therefore, in our experiments, we only use joint-training to search for the conflict layers once, and directly apply the modified network to improve different gradient manipulation methods as shown in Tables 4.1-4.5.

Table 4.9: Multi-task learning results on NYUv2 dataset with SegNet as backbone. Recon* denotes setting the layers found on CityScapes to task-specific. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon or Recon* improves the result of the base model.

Method	Segmentation (Higher Better)		Depth (Lower Better)		Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)			$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30		
Single-task	38.67	64.27	0.6881	0.2788	24.8683	18.9919	30.43	57.81	69.7		285.88
Joint-train	38.62	65.36	0.5378	0.2273	29.92	25.82	20.79	44.29	57.36	-1.62	95.58
w/ Recon	40.68	66.12	0.5786	0.2558	26.72	21.41	26.58	52.58	65.20	2.15	139.59
w/ Recon*	38.81	63.69	0.5637	0.2413	26.75	21.73	26.16	51.80	64.64	1.59	121.59
MGDA	25.71	57.72	0.6033	0.2358	24.53	18.65	31.22	58.46	70.21	-2.15	95.58
w/ Recon	36.64	62.36	0.5613	0.2255	24.66	18.66	31.30	58.47	70.16	5.37	139.59
w/ Recon*	36.85	63.51	0.5760	0.2362	24.89	18.96	30.53	57.94	69.82	4.34	121.59
Graddrop	39.01	66.13	0.5462	0.2296	29.72	25.51	19.87	44.68	58.12	-1.52	95.58
w/ Recon	39.78	65.63	0.5460	0.2280	26.42	21.16	26.89	53.16	65.84	4.45	139.59
w/ Recon*	39.97	65.71	0.5544	0.2261	26.52	21.37	26.65	52.65	65.46	4.21	121.59
PCGrad	40.01	65.77	0.5349	0.2227	28.53	24.08	22.33	47.42	60.69	1.43	95.58
w/ Recon	40.03	65.92	0.5523	0.2384	26.24	20.89	27.30	53.66	66.25	4.19	139.59
w/ Recon*	39.93	65.46	0.5494	0.2315	26.82	21.70	26.34	52.04	64.74	3.53	121.59
CAGrad	38.87	66.54	0.5331	0.2289	25.85	20.60	27.50	54.41	67.10	5.60	95.58
w/ Recon	40.68	66.12	0.5372	0.2266	25.44	19.87	28.96	56.00	68.28	6.99	139.59
w/ Recon*	39.97	65.92	0.5298	0.2273	25.56	20.11	28.69	55.37	67.75	6.47	121.59

The conflict layers found by Recon with the same architecture are transferable between different datasets. We conduct experiments with three different architectures: ResNet18, SegNet, and MTAN. **(1)** For Resnet18, we find that the layers found by Recon on CelebA and those found on Multi-Fashion+MNIST are *exactly the same*. **(2)** For SegNet, we find that 95% layers (38 out of 40) found on NYUv2 are identical to those found on CityScapes. On NYUv2, we compare the performance of using conflict layers found on NYUv2 (baselines w/ Recon) to that of using conflict

Table 4.10: Multi-task learning results on CityScapes dataset with MTAN as backbone. Recon* denotes setting the layers found on NYUv2 to task-specific. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon or Recon* improves the result of the base model.

Method	Segmentation (Higher Better)		Depth (Lower Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err		
Single-task	73.74	93.05	0.0129	27.71		190.58
Joint-train	75.35	93.55	0.0169	45.64	-23.26	157.19
w/ Recon	75.72	93.74	0.0130	40.90	-11.36	196.32
w/ Recon*	76.32	93.76	0.0132	46.40	-16.44	159.19
MGDA	70.46	91.75	0.0224	34.33	-26.02	157.19
w/ Recon	72.23	92.60	**0.0122**	26.93	**1.37**	196.32
w/ Recon*	70.83	92.14	0.0125	**25.69**	1.31	159.19
Graddrop	75.19	93.53	0.0168	46.35	-23.90	157.19
w/ Recon	75.60	93.72	0.0127	38.55	-8.71	196.32
w/ Recon*	**76.49**	**93.82**	0.0129	47.54	-16.81	159.19
PCGrad	75.64	93.54	0.02	43.53	-23.60	157.19
w/ Recon	75.89	93.71	0.0129	40.05	-10.35	196.32
w/ Recon*	76.24	93.69	0.0128	45.24	-14.66	159.19
CAGrad	75.26	93.50	0.0176	44.23	-23.40	157.19
w/ Recon	75.65	93.71	0.0125	36.23	-6.15	196.32
w/ Recon*	76.25	93.74	0.0123	40.05	-8.99	159.19

layers found on CityScapes (i.e., baselines w/ Recon*), as shown in Table 4.9. **(3)** For MTAN (SegNet with attention), we find that 68% layers (17 out of 25) found on CityScapes are identical to those found on NYUv2. On CityScapes, we compare the performance of using conflict layers found on CityScapes (baselines w/ Recon) to that of using conflict layers found on NYUv2 (i.e., baselines w/ Recon*), as shown in Table 4.10. The results show that the conflict layers found on one dataset can be used to modify the network to be directly used on another dataset to consistently improve the performance of various baselines, while searching for the conflict layers again on the new dataset may lead to better performance.

Table 4.11: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) w.r.t. the shared parameters on Multi-Fashion+MNIST dataset. "Reduction" means the percentage of conflicting gradients in the interval of $(-0.01, -1.0]$ reduced by the model compared with joint-training. The grey cell color indicates Recon greatly reduces the conflicting gradients (more than 50%). In contrast, gradient manipulation methods only slightly decrease their occurrence, and some method even increases it.

$\cos\phi_{ij}$	Joint-train	w/ RSL	w/ RSP	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1.0, 0)	56.56	53.44	58.15	58.53	56.06	56.50	57.26	57.61	56.72	57.75	56.18	59.06
(0, -0.01]	31.25	27.35	34.33	37.67	32.36	40.93	31.06	38.28	31.19	38.76	31.25	37.84
(-0.01, -0.02]	9.26	13.45	6.38	3.04	8.87	2.12	8.93	3.32	9.09	2.87	9.37	2.44
(-0.02, -0.03]	2.05	4.18	0.8	0.5	1.71	0.26	1.72	0.54	1.90	0.42	2.00	0.41
(-0.03, -1.0]	1.25	1.58	0.34	0.25	1.0	0.18	1.03	0.26	1.10	0.2	1.20	0.25
Reduction (%)	-	-52.94	40.13	69.82	7.80	79.62	7.01	67.20	3.74	72.21	-0.08	75.32

4.5.3 Ablation Study and Analysis

Recon greatly reduces the occurrence of conflicting gradients. In Fig. 4.7 and Table 4.11, we compare the distribution of $\cos\phi_{ij}$ before and after applying Recon on Multi-Fashion+MNIST. It can be seen that Recon greatly reduces the numbers of gradient pairs with severe conflicts ($\cos\phi_{ij} \in (-0.01, -1]$) by at least 67% and up to 79% when compared with joint-training, while gradient manipulation methods only slightly reduce the percentage and some even increases it. Similar observations can be made from Fig. 4.8-Fig. 4.10 and Tables 4.12-4.14.

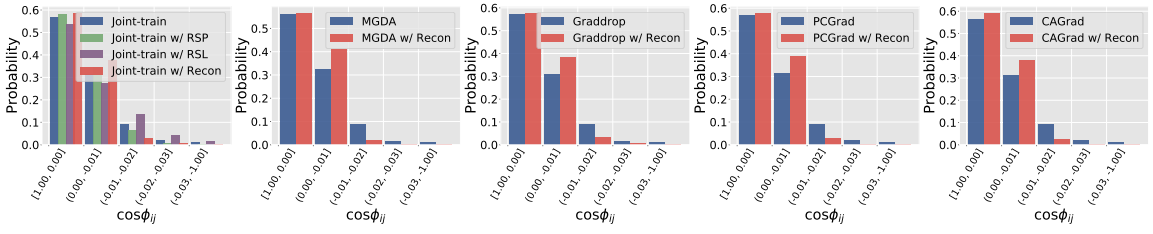


Figure 4.7: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) of baselines and baselines with Recon on Multi-Fashion+MNIST dataset.

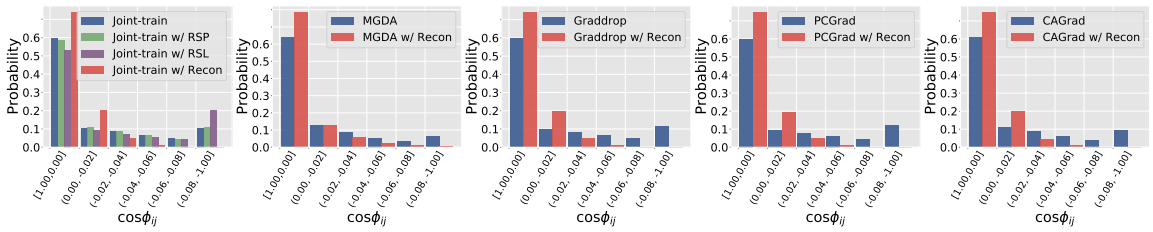


Figure 4.8: The distribution of gradient conflicts (in terms of $\cos \phi_{ij}$) w.r.t. the shared parameters on CityScapes. RSL: randomly selecting same number of layers as Recon and set them task-specific. RSP: randomly selecting similar amount of parameters as Recon and set them task-specific.

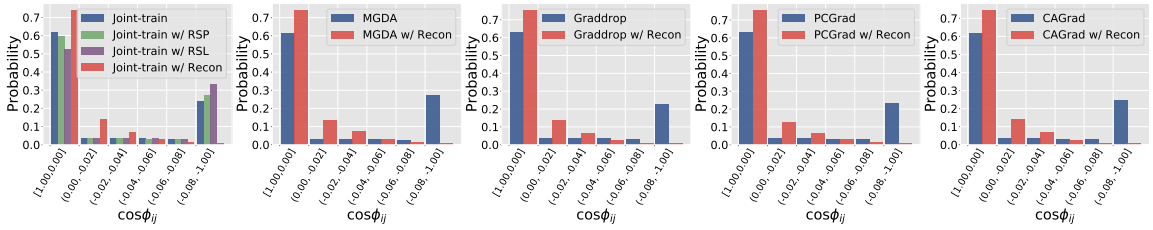


Figure 4.9: The distribution of gradient conflicts (in terms of $\cos \phi_{ij}$) of baselines and baselines with Recon on NYUv2. RSL: randomly selecting same number of layers as Recon and set them task-specific. RSP: randomly selecting similar amount of parameters as Recon and set them task-specific.

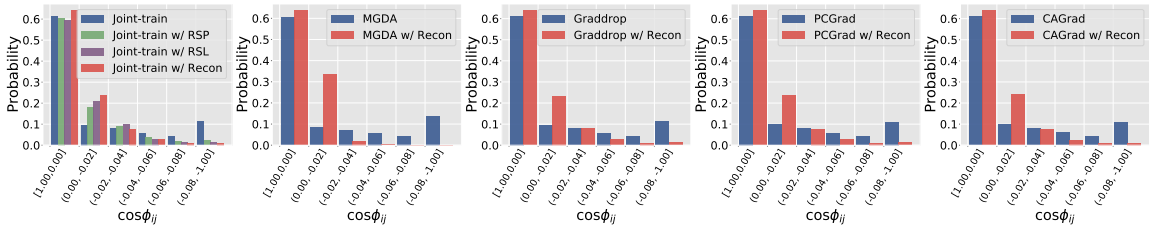


Figure 4.10: The distribution of gradient conflicts (in terms of $\cos \phi_{ij}$) of baselines and baselines with Recon on PASCAL-Context. RSL: randomly selecting same number of layers as Recon and set them task-specific. RSP: randomly selecting similar amount of parameters as Recon and set them task-specific.

Table 4.12: The distribution of gradient conflicts (in terms of $\cos \phi_{ij}$) w.r.t. the shared parameters on CityScapes dataset. "Reduction" means the percentage of conflicting gradients in the interval of $(-0.02, -1.0]$ reduced by the model compared with joint-training. The grey cell color indicates Recon greatly reduces the conflicting gradients (more than 50%). In contrast, gradient manipulation methods only moderately decrease their occurrence (MGDA deceases it by 22%), and some methods even increase it.

$\cos \phi_{ij}$	Joint-train	w/ RSL	w/ RSP	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1.0, 0)	59.55	53.16	58.29	73.62	63.9	78.27	59.56	73.82	59.85	74.52	60.79	74.54
(0, -0.02]	10.14	9.01	10.77	20.13	12.51	12.54	9.61	19.75	9.58	19.43	11.13	19.77
(-0.02, -0.04]	8.52	7.34	8.72	5.13	8.59	5.54	8.19	5.17	7.94	4.89	8.83	4.62
(-0.04, -0.06]	6.45	5.69	6.48	0.94	5.39	2.23	6.49	1.05	6.24	0.96	6.05	0.89
(-0.06, -0.08]	4.79	4.53	4.61	0.14	3.29	0.85	4.76	0.16	4.41	0.15	4.06	0.13
(-0.08, -1.0]	10.54	20.26	11.13	0.03	6.33	0.56	11.38	0.05	11.98	0.06	9.13	0.04
Reduction (%)	-	-24.82	-2.11	79.41	22.11	69.70	-1.72	78.78	-0.89	80.03	7.36	81.22

Table 4.13: The distribution of gradient conflicts (in terms of $\cos \phi_{ij}$) w.r.t. the shared parameters on NYUv2 dataset. "Reduction" means the percentage of conflicting gradients in the interval of $(-0.04, -1.0]$ reduced by the model compared with joint-training. The grey cell color indicates Recon greatly reduces the conflicting gradients (more than 50%). In contrast, gradient manipulation methods only slightly decrease their occurrence, and some methods even increase it.

$\cos \phi_{ij}$	Joint-train	w/ RSL	w/ RSP	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1.0, 0)	61.96	52.61	59.70	73.99	61.28	74.08	62.93	75.35	63.25	75.54	61.95	74.49
(0, -0.02]	3.85	3.75	3.47	14.17	2.97	13.38	3.83	13.50	3.61	12.66	3.53	14.20
(-0.02, -0.04]	3.63	3.60	3.41	7.07	2.77	7.21	3.70	6.71	3.62	6.66	3.39	6.96
(-0.04, -0.06]	3.39	3.43	3.11	2.89	2.81	3.19	3.45	2.71	3.26	2.98	3.21	2.71
(-0.06, -0.08]	3.11	3.30	2.94	1.13	2.64	1.28	3.16	1.03	3.06	1.25	3.05	1.01
(-0.08, -1.0]	24.05	33.31	27.37	0.76	27.53	0.87	22.92	0.70	23.20	0.90	24.88	0.63
Reduction (%)	-	-31.06	-9.39	84.35	-7.95	82.52	3.34	85.47	3.37	83.21	-1.93	85.76

Table 4.14: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) w.r.t. the shared parameters on PASCAL-Context dataset. "Reduction" means the percentage of conflicting gradients in the interval of $(-0.02, -1.0]$ reduced by the model compared with joint-training. The grey cell color indicates Recon greatly reduces the conflicting gradients (more than 50%). In contrast, gradient manipulation methods only slightly decrease their occurrence, and some methods even increase it.

$\cos\phi_{ij}$	Joint-train	w/ RSL	w/ RSP	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1.0, 0)	61.26	59.20	60.47	63.99	60.40	63.61	61.18	63.76	61.35	63.83	60.99	63.78
(0, -0.02]	9.66	21.01	18.25	23.57	8.51	33.53	9.66	23.41	9.83	23.61	9.95	24.04
(-0.02, -0.04]	7.90	9.91	9.10	7.65	7.27	2.04	7.89	7.83	7.90	7.65	8.03	7.53
(-0.04, -0.06]	5.85	3.05	3.88	2.59	5.68	0.45	5.80	2.71	5.82	2.66	5.91	2.51
(-0.06, -0.08]	4.16	1.32	1.79	1.07	4.35	0.17	4.21	1.12	4.13	1.10	4.23	1.04
(-0.08, -1.0]	11.16	1.30	2.29	1.13	13.80	0.20	11.24	1.16	10.97	1.16	10.88	1.08
Reduction (%)	-	46.41	41.31	57.21	-6.98	90.16	-0.24	55.90	0.86	56.76	0.07	58.20

Randomly selecting conflict layers does not work. To show that the performance gain of Recon comes from selecting the layers with most severe conflicts instead of merely increasing model parameters, we further compare Recon with the following two baselines. RSL: randomly selecting same number of layers as Recon and set them task-specific. RSP: randomly selecting similar amount of parameters as Recon and set them task-specific. The results in Table 4.15 show that both RSL and RSP lead to significant performance drops, which verifies the effectiveness of the selection strategy of Recon.

Selecting the first K layers and the last K Layers as conflict layers does not work. To further support the conclusion that the selection of parameters with higher probability of conflicting gradients contributes most to the performance gain rather than the increase in model capacity. We compare Recon with two baselines: (1) Select the first K neural network layers and turn them into task-specific layers. (2) Select the last K neural network layers and turn them into task-specific layers. The multi-task learning results on the Multi-Fashion+MNIST benchmark are presented

Table 4.15: Comparison of Recon with RSL and RSP. PD: performance drop compared to Recon.

Seed	w/ RSL	w/ RSP	w/ Recon	CAGrad					PCGrad				
				Task 1		Task2		#P.	Task 1		Task2		#P.
				Acc↑	PD	Acc↑	PD		Acc↑	PD	Acc↑	PD	
0	✓			97.60	**0.68**	64.39	**25.26**	73.02	97.43	**0.87**	65.57	**24.21**	73.02
1	✓			97.11	**1.18**	87.61	**2.04**	83.63	94.92	**3.39**	87.31	**2.46**	83.63
2	✓			94.62	**3.66**	87.68	**1.96**	76.33	92.90	**5.40**	87.41	**2.36**	76.33
0		✓		97.11	**1.18**	85.57	**4.07**	52.25	96.93	**1.38**	88.16	**1.62**	52.25
1		✓		97.81	**0.47**	88.28	**1.36**	51.96	97.63	**0.68**	88.55	**1.22**	51.96
2		✓		81.18	**17.10**	76.56	**13.09**	47.50	88.71	**9.59**	84.51	**5.27**	47.50
-	-	-	✓	98.28	0	89.65	0	43.42	98.30	0	89.77	0	43.42

Table 4.16: Multi-task learning results on Multi-Fashion+MNIST dataset. LSK refers to turning the fist K layers into task-specific layers. FSK refers to turning the last K layers into task-specific layers. PD denotes the performance drop compared with Recon.

LSK	FSK	w/ Recon	CAGrad					PCGrad				
			Task 1		Task2		#P.	Task 1		Task2		#P.
			Acc↑	PD	Acc↑	PD		Acc↑	PD	Acc↑	PD	
✓			97.63	**0.66**	89.14	**0.50**	84.17	97.63	**0.65**	88.98	**0.66**	84.17
	✓		98.21	**0.07**	89.15	**0.50**	48.90	98.19	**0.09**	89.51	**0.13**	48.90
	-	✓	98.28	0	89.65	0	43.42	98.30	0	89.77	0	43.42

in Table 4.16. The results show that if we directly turn the top or the bottom of the neural network into task-specific parameters, it still will lead to performance degradation compared to Recon.

Ablation study on hyperparameters. We study the influence of the conflict severity S and the number of selected layers K on the performance of CAGrad w/ Recon on Multi-Fashion+MNIST. As shown in Fig. 4.6, a small K leads to a significant performance drop, which indicates that there are still some shared network

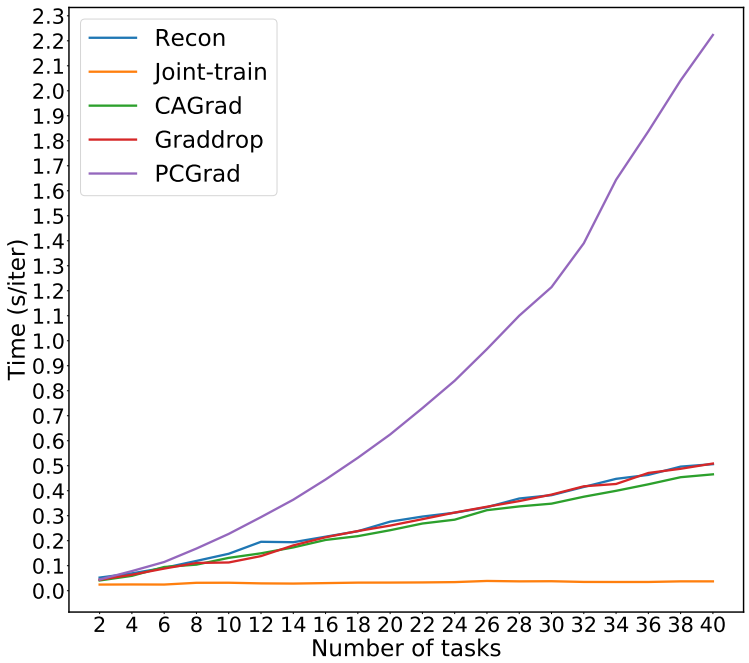


Figure 4.11: Comparison of running time (one iteration, excludes data fetching) on CelebA dataset.

layers suffering from severe gradient conflicts, while a large K will not lead to further performance improvement since severe conflicts have been resolved. For the conflict severity S, we find that a high value of S (e.g., 0.0) leads to performance drops since it includes too many gradient pairs with small conflicts, while some of them are helpful for learning common structures and should not be removed. In the meantime, a too small S (e.g., -0.15) also leads to performance degradation because it ignores too many gradient pairs with large conflicts, which may be detrimental to learning. While K and S are sensitive, we may only need to tune them once for a given network architecture, as discussed in Sec. 4.5.2.

Analysis of running time. We evaluate how Recon scales with the number of tasks on CelebA dataset, by comparing the running time of one iteration used by Recon in computing gradient conflict scores (the most time-consuming part of Recon) to that of the baselines. The results in Fig. 4.11 show that Recon is as fast as other gradient

manipulation methods such as CAGrad [140] and Graddrop [36], but much slower than joint-training especially when the number of tasks is large, which is natural since Recon needs to compute pariwise cosine similarity of task gradients. However, since Recon only needs to search for the conflict layers once for a given network architecture, as discussed above, the running time is not a problem.

4.6 Chapter Review

We propose an innovative method to address the issue of conflicting gradients in multi-task learning, thereby preventing negative transfer. Specifically, we analyze the angles between gradients corresponding to different tasks at each layer of the network during training. By identifying layers where gradient conflicts are severe (i.e., large angles), we convert these layers from shared to task-specific. This means that instead of sharing parameters, each task has its own parameters for these layers, which helps eliminate gradient conflicts in the remaining shared layers. In section 4.4.2, we provide theoretical proof that our algorithm can lead to smaller losses. We validate the effectiveness of our approach across various datasets and network architectures.

However, our method is not highly efficient because it requires two training phases: one to identify the conflicting layers and another to obtain the final model. This could make the implementation cumbersome. Despite this, our core idea provides valuable insights: rather than merely adjusting the direction of gradient updates to mitigate conflicts, it is more effective to eliminate the occurrence of gradient conflicts altogether. This fundamental approach can lead to significant improvements in overall performance.

Chapter 5

Understanding Layer Significance For LLM Alignment

5.1 Introduction

Aligning large language models (LLMs) with specific requirements is essential for enhancing their utility across diverse applications [156, 277, 157, 130, 145, 144, 60]. Fine-tuning LLMs during the alignment process can significantly improve the models' capabilities to meet targeted needs [19]. Typically, alignment involves fine-tuning the model on diverse datasets, which may include both human-curated [197] and LLM-generated [238] data, using approaches like instruction tuning [256] and preference learning [10, 196]. Given the significant cost associated with full parameter fine-tuning, parameter-efficient fine-tuning (PEFT) [94, 28, 182] methods have emerged as a popular alternative, offering a balance between performance and resource efficiency.

Understanding what LLMs actually learn during the alignment process is crucial. Zhou et al. [299] posits that the majority of knowledge and capabilities are developed during the pre-training phase, with alignment primarily serving to refine the model's conversational style and formatting. Using a well-selected set of 1,000 training ex-

amples for supervised fine-tuning (SFT), they successfully produced a high-quality aligned model. Similarly, Lin et al. [135] investigated the token distribution of LLMs before and after alignment and found that most changes were related to "stylistic tokens", such as discourse markers and transition words, while the knowledge-intensive content largely remained untouched, coming from the base pre-trained model. These findings imply that the alignment process mainly adjusts the model's presentation style rather than modifying its foundational knowledge.

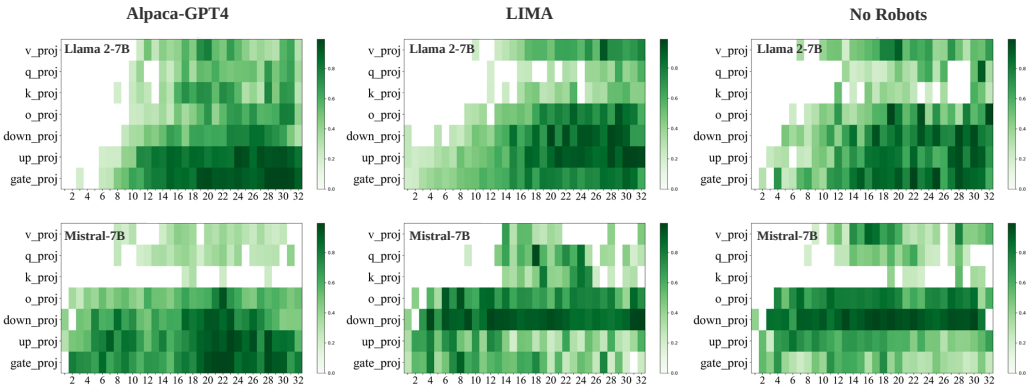


Figure 5.1: Layer importance rankings by our ILA algorithm for LLAMA 2-7B and Mistral-7B-v0.1 across Alpaca-GPT4, LIMA, and No Robots datasets. **Top 75% layers by score (s_i) are considered important**. X-axis: transformer block index; y-axis: linear layer names. The figure highlights two findings: (1) **High overlap (90%)** in important layers across datasets (Table5.2) suggests shared alignment needs, regardless of substantial differences in dataset content; (2) Important layers **differ by architecture**, reflecting model-specific dynamics.

To gain a deeper understanding of LLM alignment, we analyze this process at the level of model parameters. We conducted a pilot study to investigate the impact of different model components on alignment performance, by fine-tuning only specific layers and evaluating the resulting performance, as presented in Table 5.1 in Section 5.3. The results clearly indicate that fine-tuning different components of the LLM leads to considerable performance differences. For instance, fine-tuning the feed-forward net-

work (FFN) layers achieves performance similar to fine-tuning all linear layers (i.e., with LoRA), whereas focusing solely on the attention layers causes a notable drop in performance. This observation shows the complexity of layer-specific contributions to LLM alignment, highlighting the need for detailed analysis.

To address this, we propose ***identifying the layers that are most critical to alignment performance during the SFT process***. We develop a novel approach, ILA, for identifying the important layers for LLM alignment. Specifically, we learn a binary mask for the parameter changes in each layer during the fine-tuning process, which serves as an indicator of layer significance. A binary mask value of zero indicates that the corresponding layer has negligible influence during the process, while a value of one denotes that the layer is crucial. We use gradient descent to learn the binary mask effectively and offer a theoretical analysis of the optimization process. The main findings and significance of this work include:

- **Consistent layer importance ranking across different alignment datasets.** We observe similar rankings of important layers during alignment for the same pre-trained model (see Fig. 5.1), even though the alignment datasets vary significantly in both content and size. This suggests that the alignment process endows the model with similar capabilities, corroborating previous research findings and offers new insights into LLM alignment.

- **Enhancing performance by freezing unimportant layers.** We show that freezing about 25% of unimportant layers can improve performance and that *a single search* for layer importance ranking is sufficient for different alignment tasks using the same architecture.

- **Improving alignment efficiency through selective fine-tuning.** Our findings show that fine-tuning only 10-30% key layers achieves performance comparable to fine-tuning all linear layers. Additionally, integrating this approach with QLoRA allows tuning only 30-75% of key layers to maintain or enhance

performance while cutting resource costs.

- **Broader implications beyond LLM alignment.** Although our primary focus is on LLM alignment, the approaches and insights from this study have broader applicability. Our preliminary experiments on LLM reasoning reveal findings similar to those in alignment, showcasing the significant potential of our methods to enhance the reasoning capabilities of LLMs, particularly in achieving test-time scaling [179, 258, 223, 173].

5.2 Related Works

LLM Alignment. Pretrained language models encode general-purpose representations, enabling transfer across diverse tasks [194, 108, 178]. Alignment methods like instruction tuning [292, 233, 172] and preference learning [85, 72, 196, 226, 128] adapt these models to specific objectives. Recent studies have explored alignment mechanisms. LIMA [299] showed that fine-tuning on small datasets (e.g., 1,000 examples) shapes behavior without adding new knowledge, a finding echoed by others [32, 122, 73]. Duan et al. connected instruction tuning to in-context learning via hidden state analysis, while URIAL [135] revealed that alignment mainly modifies stylistic tokens, preserving knowledge-centric ones. These insights suggest alignment imparts narrow, targeted adjustments. Our work builds on this by identifying the specific layers most critical for alignment, offering a more fine-grained understanding of how adaptation occurs.

Parameter Efficient Fine-Tuning (PEFT). Fine-tuning large language models with billions or trillions of parameters is computationally expensive [17, 59]. Parameter-efficient fine-tuning (PEFT) methods address this by updating specific components [281, 297, 5, 75] or using soft prompts [127, 131, 6]. Techniques such as BitFit [281], Adapters [93], LoRA [94], and their variants [291, 167] reduce cost while

maintaining transferability. Recent work [129, 97, 182, 265, 184] shows that selectively fine-tuning certain regions yields strong results, though random masking often lacks consistency. However, most PEFT approaches overlook parameter importance and lack prioritization. Our method addresses this by ranking layer importance, enabling targeted fine-tuning to improve performance with minimal cost.

Layer Analysis in Model Compression. Efforts in model compression leverage structured pruning [262, 151, 246] and layer analysis to improve efficiency. Approaches like Sheared LLaMA [263] and LLM-Streamline [34] demonstrate that selectively pruning layers, heads, and dimensions significantly reduces model size with minimal performance degradation. Studies on layer importance [294, 71] show the feasibility of removing less critical components, facilitating scalable LLMs.

While model compression studies have examined the importance of components like layers and heads for pruning, they aim to reduce model size rather than address the parameter changes needed for task-specific alignment. Our work, however, focuses on alignment fine-tuning. By emphasizing efficiency and prioritizing parameter updates through skill localization [185, 250], we enhance both the understanding and robustness of the alignment process.

5.3 Pilot Study

In this section, we conduct a pilot study to address the question: How does fine-tuning different regions of an LLM affect the alignment performance?

We use the LoRA algorithm [94] for fine-tuning and compare the following strategies: (1) **FFN**: fine-tuning all feed-forward networks (W_{up}, W_{down}, W_{gate}); (2) **ATT**: fine-tuning all attention layers including query/key/value projection layers (W_q, W_k, W_v) and the output projection layer (W_o); (3) **ATT2**: only fine-tuning the query/key/value projection layers (W_q, W_k, W_v); (4) **ALL**: fine-tuning all linear layers, including all

Table 5.1: Impact of fine-tuning different components of Llama 2-7B on alignment performance using the LIMA dataset. Evaluated on MMLU (5-shot) and GPT-4o scores for Vicuna and MT-Bench prompts. Tuned components include attention projections (W_q, W_k, W_v, W_o) and feed-forward layers (W_{up}, W_{down}, W_{gate}).

	ATT (W_q, W_k, W_v, W_o)	**ATT2** (W_q, W_k, W_v)	**FFN** (W_{up}, W_{down}, W_{gate})	**ALL (LoRA)**
MMLU ↑	42.03	42.65	43.06	**43.18**
Vicuna ↑	5.21	5.13	5.40	**5.43**
MT-Bench ↑	3.31	3.35	3.41	**3.45**

feed-forward networks and attention layers, which is equivalent to the LoRA algorithm itself; (5) **AdaLoRA** [291]: adaptively allocating parameter budget to the LoRA incremental weight matrices.

The results in Table 5.1 show that selecting layers based solely on type is suboptimal. Fine-tuning all linear layers yields the best performance, consistent with QLoRA's hyperparameter tuning [46]. Notably, fine-tuning FFN layers achieves similar results, while tuning only attention layers significantly degrades performance. These findings underscore the challenges of manual layer selection and motivate our approach to automatically identifying "important" layers for more effective alignment through targeted fine-tuning.

5.4 Layer Significance in LLM Alignment

To understand layer significance in LLM alignment, we propose ILA, a method to identify important layers by learning a binary mask that indicates each layer's significance.

Consider a pre-trained LLM model with parameters $\boldsymbol{\theta}_0$ composed of N layers, i.e., $\boldsymbol{\theta}_0 = \{\boldsymbol{\theta}_0^i\}_{i=1}^N$. The model is fine-tuned on an alignment dataset $\mathcal{D} = \{z_i\}_{i=1}^n$ with

a loss function $\mathcal{L}(\boldsymbol{\theta})$. After t training iterations, the model parameters are updated to $\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}_t$, where $\Delta\boldsymbol{\theta}_t$ represents the change in parameters till iteration t. Define a binary mask $\boldsymbol{\gamma}_t = \{\gamma_t^i | \gamma_t^i \in \{0,1\}\}_{i=1}^N$ that encodes layer-wise importance information. We apply $\boldsymbol{\gamma}_t$ to $\Delta\boldsymbol{\theta}_t$ and define

$$\boldsymbol{\theta}_t^{\text{mask}} = \boldsymbol{\theta}_0 + \boldsymbol{\gamma}_t \odot \Delta\boldsymbol{\theta}_t, \tag{5.1}$$

where $\odot$ is component-wise multiplication. The binary mask is applied to retain the changes in crucial layers while eliminating the rest. Below we provide a formal definition of the conditions under which training attains stability after an adequate number of iterations.

Definition 5.1 (ϵ-stable)**.** *$\forall \epsilon > 0$, the model is said to be ϵ-stable at iteration T if, for any $t > T$, the loss function satisfies the condition*

$$|\mathbb{E}_z[\mathcal{L}(z; \boldsymbol{\theta}_{t+1})] - \mathbb{E}_z[\mathcal{L}(z; \boldsymbol{\theta}_t)]| < \epsilon, \tag{5.2}$$

where $\mathbb{E}_z[\cdot]$ denotes the expectation with respect to the alignment dataset $\mathcal{D}$.

Once training stabilizes, we can identify the layers that are crucial for the alignment task.

Definition 5.2 (Layer Importance)**.** *The binary mask $\boldsymbol{\gamma}_t$ is defined as the solution to the following optimization problem:*

$$\boldsymbol{\gamma}_t = \arg\min_{\boldsymbol{\gamma}_t} \mathbb{E}_z[\mathcal{L}(z; \boldsymbol{\theta}_t^{\text{mask}})], \ s.t. \ \|\boldsymbol{\gamma}_t\| < H, \tag{5.3}$$

where H is a hyper-parameter that serves as a constraint to limit the number of important layers.

Efficiently Identifying the Importance Layers (Alg. 3). Due to the high cost of fine-tuning large models, to address the optimization problem in Eq. (5.3), we employ the LoRA [94] algorithm, which utilizes low-rank decomposition matrices to

Algorithm 3: Identify the Important Layers for Alignment (ILA)

Input: Pre-trained model parameters $\boldsymbol{\theta}_0$, learning rate α, the initial importance score vector $\boldsymbol{s}_0 = \{s_0^i\}_{i=1}^N$, the number of insignificant layers K, the low-rank matrices A_0, B_0 for the LoRA algorithm. (FFT is a special case of LoRA with full rank)

for iteration $i = 1, 2, \ldots$ **do**

  Update $A_t = A_{t-1} - \alpha \nabla_{A_{t-1}} \mathcal{L}(\boldsymbol{\theta}_t)$, $B_t = B_{t-1} - \alpha \nabla_{B_{t-1}} \mathcal{L}(\boldsymbol{\theta}_t)$ (LoRA);

  Or Update $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\boldsymbol{\theta}_{t-1})$ (FFT);

  if *Training has become stable* **then**

    Solve the optimization problem in Eq. (5.7) by gradient descent to find $\boldsymbol{s}_t = \{s_t^i\}_{i=1}^N$;

    Stop training;

  end

end

represent the change in model parameters till iteration t ($\Delta\boldsymbol{\theta}_t$). Specifically, LoRA utilizes two trainable low-rank matrices, $\boldsymbol{B}_t^i \in \mathbb{R}^{d_i \times r_i}$ and $\boldsymbol{A}_t^i \in \mathbb{R}^{r_i \times k_i}$, to estimate the change of the i^{th} layer:

$$\Delta\boldsymbol{\theta}_t^i = \beta \cdot \boldsymbol{B}_t^i \boldsymbol{A}_t^i, \tag{5.4}$$

where β is the scalar hyperparameter of LoRA. With the binary mask $\boldsymbol{\gamma}_t$, the i^{th} layer is updated by

$$\boldsymbol{\theta}_t^i = \boldsymbol{\theta}_0^i + \beta \cdot \gamma_t^i \cdot \boldsymbol{B}_t^i \boldsymbol{A}_t^i. \tag{5.5}$$

To ease the optimization of $\boldsymbol{\gamma}_t$, we re-parametrize each of its each components γ_t^i as the output of a Sigmoid function, i.e., $\gamma_t^i = \sigma(s_t^i)$. Then, the update of the i^{th} layer becomes

$$\boldsymbol{\theta}_t^i = \boldsymbol{\theta}_0^i + \beta \cdot \sigma(s_t^i) \cdot \boldsymbol{B}_t^i \boldsymbol{A}_t^i. \tag{5.6}$$

Let $\boldsymbol{s}_t = \{s_t^i\}_{i=1}^N$, $\boldsymbol{\theta}_t^{\text{M}} = \{\boldsymbol{\theta}_t^i\}_{i=1}^N$. The optimization problem in Eq. (5.3) becomes

$$\boldsymbol{s}_t = \arg\min_{\boldsymbol{s}_t} \mathbb{E}_z[\mathcal{L}(z; \boldsymbol{\theta}_t^{\text{M}})]. \tag{5.7}$$

We use gradient descent to optimize $\boldsymbol{s}_t$, yielding s_t^i as the importance score of the i^{th} layer. A larger value of s_t^i indicates γ_t^i is closer to one, signifying higher importance of the i^{th} layer.

Assumption 5.1 (Lipschitz-continuous)**.** *The loss function $\mathcal{L}(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and L-smooth with constant $L_1 > 0$ such that*

$$\|\mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}')\|_2 \leq L_1 \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|. \tag{5.8}$$

In addition, $\mathcal{L}(\boldsymbol{\theta})$ has an L-Lipschitz continuous gradient with constant $L_2 > 0$ such that

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\theta}')\|_2 \leq L_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|. \tag{5.9}$$

Assumption 5.2. *For any $t > T$, $\boldsymbol{\theta}_t$ is ϵ-stable. We assume there is a constant R such that*

$$\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|_2 \leq R\epsilon, \tag{5.10}$$

and there is a constant Q such that $\|\boldsymbol{\theta}_t\|_2 \leq Q$ for any $t > T$.

Theorem 5.1. *For a sufficiently small ϵ, $\boldsymbol{\theta}_T$ is ϵ-stable, thus Assumption 5.1 and Assumption 5.2 are satisfied. For any $t > T$, we assume that $\forall i, \gamma_t^i \in [0, 1]$. Let $\boldsymbol{\gamma}_t'$ denote the result of $\boldsymbol{\gamma}_t$ after one step of gradient descent, i.e., $\boldsymbol{\gamma}_t' = \boldsymbol{\gamma}_t - \beta\nabla_{\boldsymbol{\gamma}_t}\mathcal{L}(\boldsymbol{\theta}_t^{\text{mask}})$. Then we have*

$$\|\boldsymbol{\gamma}_t' - \boldsymbol{\gamma}_{t+1}'\|_2 \leq \beta(QL_2 + L_1)R\epsilon. \tag{5.11}$$

This theorem demonstrates that when $\boldsymbol{\theta}_T$ is ϵ-stable, solving the optimization problem in Eq. (5.3) for any $t > T$ yields similar results. This is because, after one step of gradient descent, the difference between γ_t and γ_{t+1} is smaller than a sufficiently small number. The proof is given below:

Proof. Let $\hat{\boldsymbol{\gamma}}$ be the initial values of $\boldsymbol{\gamma}_t$ and $\boldsymbol{\gamma}_{t+1}$. Then we have

$$\boldsymbol{\gamma}_t' = \hat{\boldsymbol{\gamma}} - \beta\nabla_{\boldsymbol{\gamma}_t}\mathcal{L}(\boldsymbol{\theta}_t^{\text{mask}}), \tag{5.12}$$

$$\boldsymbol{\gamma}'_{t+1} = \hat{\boldsymbol{\gamma}} - \beta \nabla_{\boldsymbol{\gamma}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1}). \tag{5.13}$$

The difference of $\boldsymbol{\gamma}'_t$ and $\boldsymbol{\gamma}'_{t+1}$ is

$$\begin{aligned}
\|\boldsymbol{\gamma}'_t - \boldsymbol{\gamma}'_{t+1}\|_2 &= \|(\hat{\boldsymbol{\gamma}} - \beta \nabla_{\boldsymbol{\gamma}_t} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_t)) \\
&\quad - (\hat{\boldsymbol{\gamma}} - \beta \nabla_{\boldsymbol{\gamma}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1}))\|_2 \\
&= \beta \|\nabla_{\boldsymbol{\gamma}_t} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_t) - \nabla_{\boldsymbol{\gamma}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1})\|_2 \\
&= \beta \|\boldsymbol{\theta}_t \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_t} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_t) \\
&\quad - \boldsymbol{\theta}_{t+1} \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1})\|_2 \\
&\leq \beta \|\boldsymbol{\theta}_t \odot (\nabla_{\boldsymbol{\theta}^{\text{mask}}_t} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_t) - \nabla_{\boldsymbol{\theta}^{\text{mask}}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1}))\|_2 \\
&\quad + \beta \|(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}) \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1})\|_2.
\end{aligned} \tag{5.14}$$

Because $\mathcal{L}(\boldsymbol{\theta})$ has an L-Lipschitz continuous gradient with constant $L_2 > 0$, and $\|\boldsymbol{\theta}_t\| \leq Q$,

$$\begin{aligned}
&\|\boldsymbol{\theta}_t \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_t} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_t) - \boldsymbol{\theta}_{t+1} \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1})\|_2 \\
&\leq Q L_2 \|\boldsymbol{\theta}^{\text{mask}}_t - \boldsymbol{\theta}^{\text{mask}}_{t+1}\|_2 \\
&= Q L_2 \|\Delta\boldsymbol{\theta}_{t+1} - \Delta\boldsymbol{\theta}_t\|_2 \\
&= Q L_2 \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|_2.
\end{aligned} \tag{5.15}$$

Because $\mathcal{L}(\boldsymbol{\theta})$ is L-smooth with constant L_1,

$$\|(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}) \odot \nabla_{\boldsymbol{\theta}^{\text{mask}}_{t+1}} \mathcal{L}(\boldsymbol{\theta}^{\text{mask}}_{t+1})\|_2 \leq L_1 \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|. \tag{5.16}$$

Therefore,

$$\|\boldsymbol{\gamma}'_t - \boldsymbol{\gamma}'_{t+1}\|_2 \leq \beta(QL_2 + L_1)\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|_2. \tag{5.17}$$

According to the Assumption 5.2, we have $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|_2 \leq R\epsilon$, hence,

$$\|\boldsymbol{\gamma}'_t - \boldsymbol{\gamma}'_{t+1}\|_2 \leq \beta(QL_2 + L_1)R\epsilon. \tag{5.18}$$

□

Leveraging Layer Importance Rankings. The identified rankings of layer importance can be leveraged to enhanc both the performance and efficiency of LLM alignment. To maximize performance, prioritize fine-tuning the significant layers while freezing those deemed less important. For efficiency, focus on the layers most critical to model success. Detailed experiments and analyses are presented in Sec. 5.5.

5.5 Experiments and Findings

5.5.1 Experimental Setup

Datasets. (1) Alpaca-GPT4 contains 52K instruction-following data generated by GPT-4, utilizing prompts from Alpaca [238]. (2) LIMA contains only 1K carefully curated prompts and responses. (3) No Robots contains 10K instructions and demonstrations created by skilled human annotators.

Models and Baselines. We use four different models as the base for our experiments: Llama 2-7B [243], Llama 2-13B, Llama 3.1-8B [54], and Mistral-7B-v0.1 [107]. Our baselines are as follows: (1) **LoRA**[94]: Trainable rank decomposition matrices are added in parallel to existing weight matrices, including query/key/value projection (W_q, W_k, W_v), output projection (W_o) in self-attention, feed-forward networks (W_{up}, W_{down}, W_{gate}), and the output layer (W_{head}). (2) **AdaLoRA**[290]: It dynamically adjusts the rank of incremental matrices to control the parameter budget, with AdaLoRA modules added to all linear layers, similar to LoRA. (3) **Full**

Fine-tune: All model parameters, initialized from pre-trained weights and biases, undergo gradient updates during fine-tuning.

Evaluation and Training Setup. We assess language model alignment across two key dimensions: (1) **Language Understanding Ability**: Evaluated using **MMLU** [86] for specialized knowledge and **Hellaswag** [284] for commonsense reasoning. (2) **Conversational Ability**: Measured using **MT-Bench** [298] (multi-turn) and **Vicuna** [40] (single-turn), with responses graded by **GPT-4o**. All evaluations are performed three times, and the average scores are reported. **We conduct hyperparameter searches for LoRA and full fine-tuning to establish strong baselines.**

Targeted Performance. (1) **Language Understanding Ability**: Recent research [51, 234, 54] suggests that the learning of language understanding tasks essentially occurs during the pre-training phase of the base model. Therefore, significant performance improvements in language understanding tasks (i.e., MMLU, Hellaswag) after alignment are not expected. However, *it is crucial to ensure the model retains the learned knowledge during alignment.* (2) **Conversational Ability**: Without alignment, the pre-train model's conversational ability is poor. For example, LLAMA 2-7B often produces incorrect or irrelevant responses on the Vicuna dataset. However, *its conversational ability can be significantly improved through the alignment process.*

For all experiments, we follow fine-tuning hyperparameters: we use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and weight decay of 0.1. The scheduler employed is a cosine scheduler with a warmup ratio of 0.01. For LoRA baselines, we set the hyperparameter rank r as 32.

5.5.2 Layer Importance Rankings in LLM Alignment

In this subsection, we applied ILA to rank important layers during alignment across three datasets—No Robots, LIMA, and Alpaca-GPT4 (Fig.5.1). We also analyzed

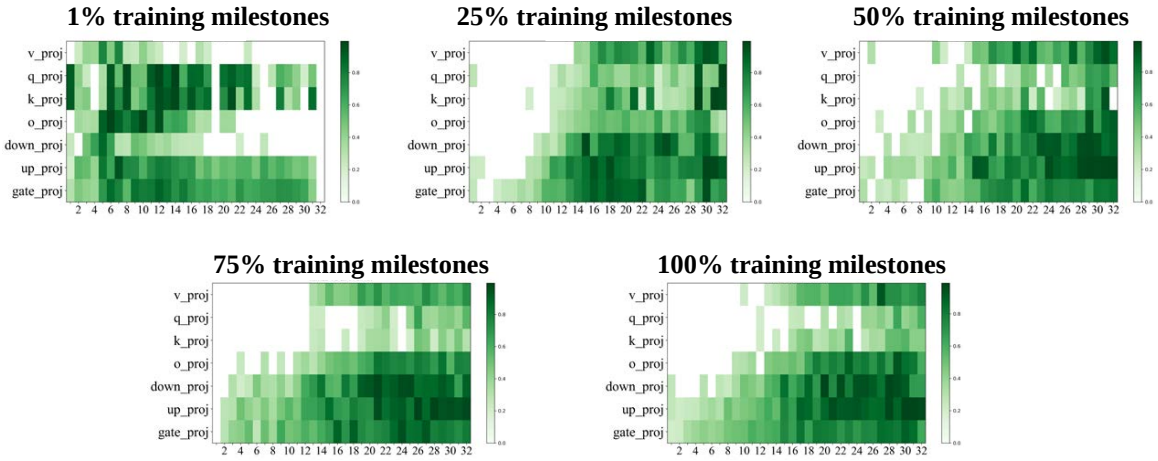


Figure 5.2: Layer importance rankings of LLAMA 2-7B during fine-tuning on LIMA at 1%, 25%, 50%, 75%, and 100% milestones. X-axis: transformer block index; y-axis: linear layer names. Jaccard similarities are provided in Table 5.4.

Table 5.2: Jaccard similarities of the top 75% highest-scoring layers identified as important during fine-tuning of LLAMA 2-7B and Mistral-7B on various datasets.

Datasets	LLAMA 2-7B			Mistral-7B		
	LIMA	No Robots	Alpaca-GPT4	LIMA	No Robots	Alpaca-GPT4
LIMA	-	-	-	-	-	-
No Robots	0.91	-	-	0.90	-	-
Alpaca-GPT4	0.90	0.90	-	0.89	0.93	-

layer importance rankings at different training milestones (Fig.5.2). To quantify similarity between sets of important layers, we used the Jaccard similarity coefficient, defining the top 75% highest-scoring layers as the important set $\boldsymbol{S}$. The similarity between two sets, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, is given by: $J(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{|\boldsymbol{S}_1 \cap \boldsymbol{S}_2|}{|\boldsymbol{S}_1 \cup \boldsymbol{S}_2|}$. where $J = 1$ indicates identical sets, and $J = 0$ indicates no overlap.

Consistency in Layer Importance Rankings Across Different Datasets. Our findings show strong consistency in layer importance rankings: (1) highly similar important layers are identified across different alignment datasets, (Fig.5.1, Table5.2);

Table 5.3: Jaccard similarities of the top 75% important layers in LLAMA 2-7B fine-tuned on the LIMA dataset using different random seeds.

Random Seed	seed1	seed2	seed3
seed1	-	-	-
seed2	0.92	-	-
seed3	0.91	0.91	-

Table 5.4: Jaccard similarities of the top 75% important layers identified at different stages of LLAMA 2-7B fine-tuning on the LIMA dataset.

Training Milestones	1%	25%	50%	75%	100%
1%	-	-	-	-	-
25%	0.69	-	-	-	-
50%	0.70	0.91	-	-	-
75%	0.69	0.90	0.92	-	-
100%	0.69	0.91	0.92	0.93	-

(2) the rankings remain stable across different random seeds for γ (Table 5.3); and (3) similar layers can be identified even at the beginning stages of training, such as completion of 25% (Fig.5.2, Table5.4).

These results confirm the robustness of ILA, which consistently identifies stable and overlapping layers across datasets. This aligns with recent findings that alignment largely involves stylistic token shifts [135]. In essence, alignment seeks similar capabilities, as *evidenced by our observation that important layers remain stable across different datasets*. This underscores the relevance of our algorithm to the fundamental objectives of alignment.

5.5.3 Enhancing Alignment Performance through Freezing Unimportant Layers

To leverage layer importance rankings, we excluded less important layers that could negatively impact fine-tuning, removing approximately of unimportant layers. The main results on **No Robots** are in Table 5.5, with additional results for LLAMA 2-13B (see Table 5.8) and main results on Alpaca-GPT4 (see Table 5.6), and LIMA (see Table 5.7) datasets. Key observations include: (1) **Freezing unimportant**

layers can enhance performance. ILA consistently outperformed LoRA and full fine-tuning on most metrics, with freezing 25% of unimportant layers yielding better results than tuning all layers. (2) **A single search for layer importance ranking suffices for a given architecture.** Rankings were stable across alignment tasks, allowing us to compute it on the No Robots dataset and apply it to others.

These results show that ILA improves fine-tuning efficiency by focusing on significant layers. Compared to AdaLoRA, even though we explored a narrow range of the hyperparameter t_r (target average rank of incremental matrices), our method performed better, suggesting that adjusting LoRA's matrix rank alone doesn't guarantee superior results, as also noted in [46].

Additionally, as discussed in Sec. 5.5.2, the stability of the layer importance ranking across datasets means a single search is often sufficient. In our experiments, we computed the layer importance ranking using full training iterations on the No Robots dataset, and then directly applied this ranking to other datasets. Though dataset-specific rankings can further improve results (Table 5.14 in Sec. 5.9), the strong cross-dataset performance with one ranking demonstrates our approach's robustness and generalizability.

5.5.4 Enhancing Alignment Efficiency by Fine-tuning Only the Most Critical Layers

To investigate this issue, we fine-tune the top 10%, 20%, and 30% of the important layers of Mistral-7B-v0.1, as identified by ILA, on the No Robots dataset, and compare the results with those of the LoRA algorithm. The results demonstrate clear benefits in focusing on a subset of important layers:

(1) Fine-tuning a small subset of the most important layers achieves competitive performance and enhances efficiency. Fine-tuning the top 10%, 20%,

Table 5.5: Comparison of LLAMA 2-7B, Mistral-7B-v0.1, and Llama 3.1-8B fine-tuned on the No Robots dataset, evaluated on **MMLU** (5-shot), **Hellaswag** (0-shot), and **GPT-4o scores** for **Vicuna** and **MT-Bench** prompts. Vicuna and MT-Bench results are averaged over **three runs**. Grey cells indicate improvements over the base model; best scores are in bold.

Models	**Methods**	**Language Understanding**		**Conversational Ability**	
		MMLU ↑	**Hellaswag ↑**	**Vicuna ↑**	**MT-Bench ↑**
Llama 2-7B	AdaLoRA	45.23	57.30	5.81	4.01
	Full Fine-tune	45.72	57.69	6.12	4.18
	Full Fine-tune w/ ILA	**45.98**	57.87	**6.35**	**4.37**
	LoRA	44.58	59.46	5.78	4.02
	LoRA w/ ILA	45.78	**59.65**	5.90	4.33
Mistral-7B-v0.1	AdaLoRA	62.13	61.68	6.21	4.69
	Full Fine-tune	61.05	**64.26**	6.32	4.55
	Full Fine-tune w/ ILA	61.75	64.21	**6.51**	4.78
	LoRA	61.95	62.90	6.25	4.68
	LoRA w/ ILA	**62.14**	62.98	6.42	**4.87**
Llama 3.1-8B	AdaLoRA	64.85	62.85	6.51	5.08
	Full Fine-tune	64.44	63.65	6.50	5.11
	Full Fine-tune w/ ILA	65.00	**63.69**	**6.61**	**5.23**
	LoRA	64.95	60.77	6.33	4.58
	LoRA w/ ILA	**65.43**	60.95	6.45	4.69

or 30% of layers results in only a slight performance drop compared to full fine-tuning. Fine-tuning 30% of layers nearly matches full fine-tuning (Table 5.9), demonstrating that focusing on the most important layers ensures efficient fine-tuning with minimal performance loss.

(2) Our method can be applied to enhance QLoRA, further reducing costs. When combined with QLoRA, our method fine-tunes only 30% or 75% of the most important layers while maintaining or improving performance (Table 5.10), highlight-

Table 5.6: Comparative evaluation of LLAMA 2-7B, Mistral-7B-v0.1, and Llama 3.1-8B models fine-tuned on the Alpaca-GPT4 Dataset. Evaluated using MMLU (5-shot), Hellaswag (0-shot), **GPT-4o scores** on Vicuna prompts, and MT-Bench prompts. **The evaluations are performed three times, and the average scores are reported.** Cells highlighted in grey indicate that ILA has enhanced the performance of the base model. The best result is marked in bold.

Models	Methods	Language Understanding		Conversational Ability	
		MMLU ↑	Hellaswag ↑	Vicuna ↑	MT-Bench ↑
llama-7B	AdaLoRA	46.13	57.85	6.89	3.78
	Full Fine-tune	45.91	57.73	6.78	3.72
	Full Fine-tune w/ ILA	**46.23**	57.67	6.99	3.85
	LoRA	43.66	**58.49**	6.96	3.80
	LoRA w/ ILA	44.69	58.22	7.17	3.99
Mistral-7B-v0.1	AdaLoRA	**62.48**	62.08	7.25	4.77
	Full Fine-tune	60.56	62.80	7.19	4.78
	Full Fine-tune w/ ILA	60.88	**62.91**	7.35	4.91
	LoRA	61.82	62.70	7.23	4.89
	LoRA w/ ILA	62.14	62.80	7.33	5.02
Llama 3.1-8B	AdaLoRA	**65.82**	61.02	7.48	5.39
	Full Fine-tune	63.58	61.58	7.33	5.32
	Full Fine-tune w/ ILA	64.61	61.74	7.57	5.42
	LoRA	65.40	61.72	7.65	5.43
	LoRA w/ ILA	65.76	**61.81**	**7.79**	**5.55**

ing the efficiency of our approach in achieving comparable or better results with fewer layers.

These findings highlight the effectiveness of our layer selection strategy, optimizing resource use with minimal performance trade-offs. Our integration with QLoRA shows that fine-tuning a targeted subset of important layers improves both performance and memory efficiency during fine-tuning.

For a clearer understanding of GPU memory savings, we measured memory consumption for QLoRA, LoRA, Full Fine-Tuning, and versions fine-tuning only the key layers

Table 5.7: Comparative evaluation of LLAMA 2-7B, Mistral-7B-v0.1, and Llama 3.1-8B models fine-tuned on the LIMA Dataset. Evaluated using MMLU (5-shot), Hellaswag (0-shot), **GPT-4o scores** on Vicuna prompts, and MT-Bench prompts. **The evaluations are performed three times, and the average scores are reported.** Cells highlighted in grey indicate that ILA has enhanced the performance of the base model. The best result is marked in bold.

Models	Methods	Language Understanding		Conversational Ability	
		MMLU ↑	Hellaswag ↑	Vicuna ↑	MT-Bench ↑
Llama 2-7B	AdaLoRA	44.21	59.85	5.22	3.51
	Full Fine-tune	**46.36**	62.06	5.83	3.71
	Full Fine-tune w/ ILA	46.32	**62.18**	**5.98**	**3.85**
	LoRA	43.18	54.52	5.43	3.45
	LoRA w/ ILA	44.13	54.55	5.62	3.72
Mistral-7B-v0.1	AdaLoRA	**62.40**	61.52	6.64	4.49
	Full Fine-tune	60.11	63.76	6.88	4.63
	Full Fine-tune w/ ILA	61.01	64.01	6.95	**4.77**
	LoRA	60.83	65.42	6.70	4.58
	LoRA w/ ILA	61.52	**65.51**	**6.98**	4.69
Llama 3.1-8B	AdaLoRA	63.55	62.65	6.50	4.73
	Full Fine-tune	64.31	65.64	7.09	5.12
	Full Fine-tune w/ ILA	**64.73**	**65.98**	**7.17**	**5.23**
	LoRA	62.33	62.92	6.57	4.79
	LoRA w/ ILA	63.31	63.01	6.61	4.93

identified by ILA. As shown in Table 5.11, it demonstrates that the GPU memory usage and average training time per iteration for various fine-tuning approaches, including LoRA, QLoRA, full fine-tune, and their modified versions where only 30% of the important layers identified by ILA are fine-tuned. Both LoRA and QLoRA show substantial reductions in memory usage when restricted to tuning only 30% of important layers, compared to the full-layer fine-tuning approaches. These results indicate that selectively fine-tuning a small set of critical layers is highly effective in reducing GPU memory consumption, particularly for efficient methods like QLoRA. This sug-

Table 5.8: Fine-tuning results of Llama 2-13B on the LIMA and No Robots datasets. Evaluated using MMLU (5-shot), Hellaswag (0-shot), **GPT-4o scores** on Vicuna prompts, and MT-Bench prompts. Cells highlighted in grey indicate that ILA has improved the performance of the base model.

Datasets	Methods	Language Understanding		Conversational Ability	
		MMLU ↑	Hellaswag ↑	Vicuna ↑	MT-Bench ↑
LIMA	LoRA	53.85	63.08	6.16	3.79
	LoRA w/ ILA	54.33	62.04	6.25	3.91
No Robots	LoRA	54.08	61.73	5.72	4.24
	LoRA w/ ILA	54.45	61.13	5.88	4.37

Table 5.9: Fine-tuning results of Mistral-7B-v0.1 on the No Robots dataset, evaluated on MMLU (5-shot), Hellaswag (0-shot), and GPT-4o scores for Vicuna and MT-Bench prompts (averaged over three runs). Percentages in parentheses denote the fraction of fine-tuned linear layers. Best results are in bold.

Models	Methods	Language Understanding		Conversational Ability	
		MMLU ↑	Hellaswag ↑	Vicuna ↑	MT-Bench ↑
Mistral-7B-v0.1	LoRA	**61.95**	**62.90**	6.25	4.68
	LoRA w/ ILA (10%)	62.09	61.94	5.99	4.39
	LoRA w/ ILA (20%)	61.83	62.16	6.12	4.53
	LoRA w/ ILA (30%)	61.89	62.79	**6.27**	**4.75**

gests that targeted fine-tuning can enhance computational efficiency while preserving model performance, which is especially beneficial when scaling large language models with limited hardware resources.

5.5.5 Ablation Study

Observation 1: Our layer importance ranking algorithm is effective. We evaluated our algorithm by comparing it to a baseline that fine-tunes all layers and three alternatives: (1) **RL 1** and **RL 2**, which randomly freeze top-K layers; (2)

Table 5.10: Comparison of QLoRA fine-tuning on Llama 2-7B vs. selectively fine-tuning important layers identified by ILA. Evaluated on MMLU (5-shot), Hellaswag (0-shot), and GPT-4o scores for Vicuna and MT-Bench prompts (averaged over three runs). Grey cells indicate improvements over the base model by ILA.

Datasets	**Methods**	**Language Understanding**		**Conversational Ability**	
		MMLU ↑	**Hellaswag ↑**	**Vicuna ↑**	**MT-Bench ↑**
LIMA	QLoRA	43.06	55.47	5.31	2.98
	QLoRA w/ ILA (75%)	43.48	55.95	5.56	3.19
	QLoRA w/ ILA (30%)	44.01	55.82	5.17	3.01

Table 5.11: GPU memory usage for LoRA, QLoRA, Full Fine-tune and LoRA/QLoRA/Full Fine-tune with only 30% of important layers fine-tuned. Batch size is set to 2, and the maximum token length is 1024. Percentages in parentheses indicate the proportion of linear layers fine-tuned.

	GPU Memory Usage (MiB)	**Training time (ms)**
Full Fine-tune (100%)	81276	396
Full Fine-tune w/ ILA (30%)	33458	304
LoRA (100%)	32752	403
LoRA w/ ILA (30%)	28586	359
QLoRA (100%)	26238	523
QLoRA w/ ILA (30%)	17912	423

FL, freezing the first K layers; and (3) **LL**, freezing the last K layers. As shown in Table 5.12, these naive strategies underperform. In contrast, our method effectively identifies and freezes the least critical layers, yielding notable gains in both efficiency and performance.

Observation 2: The important scores calculated using LoRA are similar to those obtained through full fine-tuning. To assess whether LoRA-based approximations differ from full fine-tuning (FFT), we compared parameter updates

Table 5.12: Performance comparison of ILA, random, and position-based layer selection for fine-tuning Llama 2-7B on the No Robots dataset. **RL1**/**RL2** freeze K randomly selected layers (different seeds); **FL** and **LL** freeze the first and last K layers, respectively. Blue highlights indicate lower performance than ILA.

Methods	**Language Understanding**		**Conversational Ability**	
	MMLU ↑	**Hellaswag** ↑	**Vicuna** ↑	**MT-Bench** ↑
LoRA	44.58	59.46	5.78	3.98
LoRA w/ RL 1	44.23	59.71	5.72	3.96
LoRA w/ RL 2	43.98	59.11	5.62	3.89
LoRA w/ FL	44.02	59.32	5.58	3.71
LoRA w/ LL	44.61	59.21	5.65	3.99
LoRA w/ ILA	45.78	59.65	5.90	4.15

Table 5.13: Jaccard Similarity between important layers selected using Full Fine-Tuning and LoRA for Llama 2-7B. Top 75% highest-scoring layers are determined as important layers.

Datasets	**LIMA (FFT)**	**No Robots (FFT)**	**Alpaca-GPT4 (FFT)**
LIMA (LoRA)	0.84	0.76	0.83
No Robots (LoRA)	0.78	0.80	0.81
Alpaca-GPT4 (LoRA)	0.82	0.83	0.86

from LoRA (i.e., Eq. (5.4)) and FFT (i.e., $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$). For both methods, we derived layer importance scores and selected the top 75% of layers, then calculated the Jaccard similarity between the layers. As shown in Table 5.13, LoRA achieves nearly 83% overlap with the important layers identified by FFT, reducing computational overhead while effectively ranking layer importance. The results show that LoRA provides a strong approximation of $\Delta\boldsymbol{\theta}_t$ compared to $\boldsymbol{\theta}_t - \boldsymbol{\theta}_0$.

Observation 3: Cross-dataset evaluation of layer importance enhances performance. Different datasets highlight subtle differences in important layers (Table 5.2). By intersecting the top-K least important layers from LIMA, No Robots, and Alpaca-GPT4 and freezing them during fine-tuning (Table 5.14), we found that

Table 5.14: Results of fine-tuning Mistral-7B on the LIMA dataset using ILA to identify important layers from various datasets. **Dataset (Imp. Layers)** indicates the datasets utilized to search for the important layers. **Intersection** represents freezing the layers that are the intersection of the top-K least important layers found from the LIMA, No Robots, and Alpaca GPT4 datasets.

Dataset (Imp. Layers)	**Dataset (Finetune)**	**MMLU ↑**	**Hellaswag ↑**	**Winogrande ↑**	**Vicuna ↑**	**MT-Bench ↑**
LIMA	LIMA	**61.82**	65.48	72.01	6.99	5.38
No Robots	LIMA	61.52	65.51	71.66	6.92	5.34
Alpaca-GPT4	LIMA	61.23	65.20	71.59	7.03	5.21
Intersection	LIMA	61.49	**65.62**	**72.20**	**7.10**	**5.49**

cross-dataset evaluation yields better results than dataset-specific fine-tuning. As shown in Table 5.2, different datasets reveal subtle variations in the layers identified as important. This suggests that layers consistently deemed unimportant across multiple datasets are likely genuinely non-essential. To validate this, we intersect the top-K least important layers identified from three datasets (LIMA, No Robots, and Alpaca-GPT4) to derive a set of universally non-critical layers. The results are presented in Table 5.14.

Our analysis reveals that a holistic consideration of layer importance across multiple datasets yields superior results compared to dataset-specific approaches. For instance, identifying important layers within the LIMA dataset and fine-tuning on the No Robots dataset is less effective than an integrated approach. Similarly, finding important layers and fine-tuning exclusively on the No Robots dataset do not perform as well as the comprehensive method. This suggests that a cross-dataset evaluation of layer importance can lead to more robust and effective fine-tuning strategies.

Observation 4: Cross-model transfer of layer importance rankings is feasible but less effective than cross-dataset transfer. Models sharing architecture but trained on different datasets show strong agreement in important layers (Jaccard

Table 5.15: Jaccard similarities of the top 75% important layers across different models.

	Llama 2-7B (LIMA)	**Llama 2-7B (NoRobots)**	**Llama 2-7B (Alpaca-GPT4)**
Mistral-7B-v0.1 (LIMA)	0.67	-	-
Mistral-7B-v0.1 (NoRobots)	0.70	0.71	-
Mistral-7B-v0.1 (Alpaca-GPT4)	0.71	0.66	0.75

Table 5.16: Experimental results for Mistral-7B-v0.1 on the No Robots dataset, using layer importance rankings derived from Llama 2-7B.

Methods	**MMLU**	**Hellaswag**	**Vicuna**	**MT-Bench**
LoRA	61.95	62.90	6.25	4.68
LoRA w/ ILA (75%) (cross-model transfer)	62.10	**63.21**	6.29	4.72
LoRA w/ ILA (75%)	**62.14**	62.80	**6.42**	**4.87**
LoRA w/ ILA (30%) (cross-model transfer)	61.77	**63.16**	6.11	4.60
LoRA w/ ILA (30%)	**61.89**	62.79	**6.27**	**4.75**

similarity of 0.90 for the top 75%, Table 5.2). This drops to 0.70 across architectures (Table 5.15), indicating reduced transferability. Nonetheless, significant overlap suggests cross-architecture transfer remains viable. Fine-tuning Mistral-7B-v0.1 using rankings from Llama 2-7B on No Robots (Table 5.16) confirms that cross-model transfer can still perform well.

These findings suggest that cross-model transfer of layer importance rankings is possible, though less effective than using rankings from the same architecture. Fine-tuning the top 75% of layers based on cross-model transfer shows some improvement, while fine-tuning only the top 30% achieves comparable performance.

Observation 5: ILA is robust to the initialization of layer importance scores. We evaluated the effect of different initial layer importance scores on the con-

Table 5.17: The Jaccard similarities of top 75% important layers identified during fine-tuning of Llama 2-7B on the LIMA dataset with varying initial scores.

Initial Scores	**4.0**	**2.0**	**1.0**
4.0	-	-	-
2.0	0.83	-	-
1.0	0.78	0.88	-

sistency of identified important layers. The scores were initialized to $s_0 = 4.0, 2.0, 1.0$. The consistency was measured using the Jaccard similarity of the top 75% important layers identified during fine-tuning of LLama 2-7B on the LIMA dataset. As demonstrated in Table 5.17, our algorithm ILA is resilient to varying initial importance scores ($s_0 = 4.0, 2.0, 1.0$), with minimal impact on final rankings. The stable Jaccard similarities for the top 75% of layers during Llama 2-7B fine-tuning on LIMA confirm reliable convergence regardless of initialization.

Observation 6: The computation cost of ILA is low. ILA runs in two stages: Stage 1 trains the model with LoRA until ϵ-stability, and Stage 2 tunes importance weights (γ_t) with the backbone and LoRA frozen. For both LLAMA 2-7B and Mistral-7B-v0.1 (225 linear layers), Stage 1 takes 6671 ms per iteration, and Stage 2 takes 5343 ms. Stage 2 finishes in 11 minutes (128 batches). Most cost lies in Stage 1, but Table 5.4 shows only 25–50% of training milestones are needed for strong performance.

5.5.6 Beyond LLM Alignment: LLM Reasoning

Our findings indicate that the alignment process of LLM imparts similar capabilities despite data variations. This complements prior research by revealing layer-specific roles and improving efficiency through strategic tuning and freezing of layers. Nonetheless, the approaches and insights derived from this study extend beyond LLM

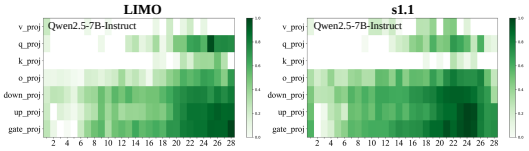


Figure 5.3: Layer-wise importance rankings for Qwen2.5-7B-Instruct fine-tuned using the LIMO and s1.1 datasets, respectively.

Table 5.18: Performance of Qwen2.5-7B-Instruct on mathematical reasoning benchmarks after fine-tuning with the LIMO dataset.

Methods	MATH500	AIME
FFT	77.00	13.33
FFT w/ ILA	**79.00**	**16.67**

alignment.

LLM Reasoning and Test-time Scaling. Advanced models like o1 [179], Deepseek R1 [74], and Kimi 1.5 [239] have exhibited strong reasoning capabilities. Similar to alignment, reasoning seeks to *further activate the knowledge* acquired during pre-training. While alignment ensures that outputs align with human values, reasoning *drives the model toward deeper inference for enhanced accuracy*. Rather than scaling model size or training data, recent studies such as LIMO [275] and s1 [173] investigate *test-time scaling*—boosting performance by increasing the number of input tokens used for reasoning. Their findings show that even limited high-quality training data with chain-of-thought (CoT) examples can effectively enhance LLMs' reasoning capabilities.

To evaluate the effectiveness of ILA on reasoning tasks, we conduct experiments using the following setup.

Datasets. (1) **LIMO [275]:** This dataset comprises 817 carefully selected problems drawn from an initial pool of tens of millions. The final selection meets strict quality standards and covers a broad range of mathematical reasoning tasks. High-quality solutions are provided by both human experts and AI systems like DeepSeek R1 [74]. (2) **s1.1 [173]:** This dataset includes 1,000 questions paired with reasoning traces, curated based on three rigorously validated criteria: difficulty, diversity, and quality.

The chain-of-thought solutions are generated by DeepSeek R1 [74].

Evaluation. (1) AIME24: This set contains 30 problems from the 2024 American Invitational Mathematics Examination (AIME), administered on January 31 and February 1, 2024. (2) MATH500 [86]: A benchmark consisting of competition-level math problems spanning a range of difficulties.

Consistency in Layer Importance Across Datasets. We applied our proposed ILA algorithm to identify layer importance rankings on both the LIMO and s1.1 datasets using the Qwen2.5-7B-Instruct model [269]. The resulting rankings showed strong consistency, with a Jaccard similarity of ***0.86*** (see Fig.5.3), suggesting that LLMs tend to acquire similar reasoning-related knowledge across datasets. We hypothesize that much of this knowledge is already learned during pretraining—as also suggested by LIMO [275]—and that fine-tuning primarily serves to activate the model's latent reasoning abilities.

Based on the identified importance rankings, we further conducted experiments on LIMO by freezing approximately 25% of the least important layers. As shown in Table 5.18, fine-tuning only the top 75% most important layers led to a slight improvement in performance, indicating that selective tuning can help enhance the model's reasoning capabilities.

5.6 Chapter Review

To better understand LLM alignment, we introduce ILA, a method that identifies critical layers in the alignment process by learning binary masks over weight matrices. ILA consistently highlights important layers across diverse datasets, suggesting that alignment imparts similar capabilities regardless of variations in training data. This finding complements prior research by shedding light on the layer-specific roles involved in alignment.

Beyond alignment, we observe that ILA reveals a similar pattern of important layers across datasets for LLM reasoning tasks as well. Notably, freezing less important layers not only reduces computational overhead but also improves performance in both alignment and reasoning scenarios.

Overall, ILA provides a unified and efficient approach to understanding and optimizing LLMs by revealing transferable layer importance across tasks and datasets, contributing both theoretical insights and practical efficiency gains.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

This thesis addresses several challenges in applying knowledge transfer to real-world scenarios, such as enabling models to adapt to dynamic environments while retaining previously learned knowledge, learning new tasks without forgetting old ones, handling multiple tasks simultaneously to overcome potential negative transfer between tasks, and improving the efficiency of knowledge transfer. Our research primarily focuses on the contexts of Continual Learning and Multi-Task Learning, proposing algorithms to tackle their core difficulties. We also explore methods for effectively aligning Large Language Models (LLMs), which serves as an example of a Multi-Task Learning challenge. These algorithms aim to make knowledge transfer more applicable to practical scenarios.

Firstly, in Chapter 3, we explore knowledge transfer in the context of Continual Learning. We propose that instead of addressing the issue of catastrophic forgetting when learning new tasks, it is more effective to consider it during the training of the base model. By ensuring the model converges to a flat local minimum, any fine-tuning within this region is unlikely to forget the knowledge acquired during the

initial training phase. This approach allows the model to learn new knowledge without forgetting previous knowledge, thereby enhancing the effectiveness of knowledge transfer in dynamically changing real-world environments.

In Chapter 4, we examine knowledge transfer in the context of Multi-Task Learning. The current trend emphasizes the ability of models to handle multiple tasks simultaneously, as seen with LLMs. Learning multiple tasks not only addresses the issue of insufficient training data for single tasks but also leverages the correlations between tasks to mutually enhance performance. This chapter focuses on mitigating the problem of negative transfer between tasks. By converting shared layers that often result in conflicting gradients into task-specific layers, we eliminate the occurrence of conflicting gradients, ensuring that tasks do not negatively impact each other. This method effectively resolves conflicts between tasks, enabling better model performance whether training a pre-trained base model or fine-tuning a model for a multi-task downstream application.

Finally, in Chapter 5, we investigate the alignment of LLMs, which is a crucial step in applying pre-trained LLMs to downstream tasks. Due to the high parameter count of large models, fine-tuning can be resource-intensive. In this chapter, we identify key layers that are crucial during the alignment process. By fine-tuning only these specific layers, we achieve a more efficient use of resources while still enhancing model performance. This approach significantly reduces the computational resources required for knowledge transfer in large-scale models and helps to mitigate overfitting to some extent.

Overall, the significance of this thesis lies in its contribution to making knowledge transfer more effective, scalable, and applicable in complex, real-world settings. By addressing key limitations in Continual Learning and Multi-Task Learning—such as catastrophic forgetting, negative transfer, and resource inefficiency—we help bridge the gap between theoretical advancements and practical deployment. Our work enables models to adapt continuously to evolving environments without retraining from

scratch, to leverage task interdependencies while avoiding harmful interference, and to harness the power of large-scale models with reduced computational overhead. These advancements not only improve model robustness and generalization but also expand the feasibility of deploying intelligent systems in dynamic, multi-task, and resource-constrained scenarios. In doing so, this thesis lays important groundwork for the future of adaptive, lifelong, and efficient machine learning systems.

6.2 Future works

Building on the foundational work presented in this thesis, future research will focus more deeply on the unique challenges and opportunities presented by Large Language Models (LLMs). As LLMs become increasingly central to modern AI systems, their scale, versatility, and potential for multi-task capabilities make them a natural extension of the knowledge transfer methods discussed here. However, their size and complexity introduce new demands in terms of alignment, efficiency, and adaptability. Future work will explore how to extend and refine knowledge transfer strategies to better suit LLMs—ensuring they can continuously learn, generalize across diverse tasks, and be efficiently fine-tuned or aligned for specific applications while minimizing resource overhead.

6.2.1 Continual Learning for Large Language Models

One promising research direction is the development of continual learning (CL) techniques specifically tailored to LLMs. While current LLMs are pretrained on large static datasets and then fine-tuned for individual tasks, real-world deployment often requires models to learn from streaming data that evolves over time. Retraining from scratch each time new data appears is both impractical and resource-intensive.

Future work will explore scalable CL strategies for LLMs, such as applying flat minima

optimization during pretraining and integrating knowledge preservation mechanisms to mitigate catastrophic forgetting. A particularly promising idea is to maintain a curated subset of high-quality, representative samples from past tasks—essentially a dynamic memory buffer—that can be combined with new data for continual training. This hybrid dataset approach could enable LLMs to retain past knowledge while acquiring new information more efficiently.

6.2.2 Multi-Task Learning for Large Language Models

Another critical area is improving multi-task learning (MTL) for LLMs. These models are frequently used in settings where they must perform well across a variety of tasks—often with imbalanced data, varying task complexity, and conflicting objectives. Traditional MTL approaches may falter under these conditions due to challenges like long-tail distributions and gradient interference.

To address this, future research will focus on advanced task-balancing techniques, conflict mitigation strategies, and adaptive learning frameworks. One promising solution involves leveraging the Mixture-of-Experts (MoE) architecture. MoE models dynamically route input (e.g., tokens or tasks) to specialized ”experts”—subnetworks trained to handle particular tasks or types of data. This design is well-suited for handling gradient conflicts in MTL, as discussed in this thesis.

By integrating gradient conflict analysis into expert routing decisions, future MoE-based LLMs could allocate conflicting or dissimilar tasks to separate experts. This would reduce negative transfer and improve model stability. Dynamic expert activation based on input features or task metadata may further enhance performance and allow more efficient use of model capacity.

6.2.3 Optimizing Training and Inference for LLMs

Despite recent advances in parameter-efficient fine-tuning (PEFT), LLMs still demand significant computational resources for both training and inference. These resource constraints limit their deployment in edge environments, mobile devices, or low-latency applications.

Future work will investigate strategies to further improve the efficiency and scalability of LLMs. This includes developing better model compression techniques (e.g., pruning, quantization), smarter adapter selection mechanisms, and sparsity-aware fine-tuning algorithms. Reducing memory usage and compute costs is especially crucial for enabling real-time applications such as conversational agents and code generation systems.

An especially promising direction is leveraging larger, high-capacity LLMs to generate high-quality, information-rich synthetic data for training smaller models—including MoE-based architectures. This teacher-student paradigm could be one of the most effective ways to accelerate training and inference without compromising performance. By combining synthetic data generation with modular architectures and PEFT techniques, it may be possible to create lightweight LLMs that are fast, accurate, and easy to deploy.

References

[1] OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Is abella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin,

Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Ma teusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Adeola Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack W. Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek,

Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. 2023. URL `https://api.semanticscholar.org/CorpusID:257532815`.

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7120–7129, 2016. URL `https://api.semanticscholar.org/CorpusID:914027`.

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[4] Maryam Aminian, Mohammad Sadegh Rasooli, and Mona T. Diab. Mutlitask learning for cross-lingual transfer of semantic dependencies. *ArXiv*, abs/2004.14961, 2020. URL `https://api.semanticscholar.org/CorpusID:216867238`.

[5] Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*, 2021.

[6] Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. *arXiv preprint arXiv:2205.11961*, 2022.

[7] Isabelle Augenstein and Anders Søgaard. Multi-task learning of keyphrase boundary classification. *ArXiv*, abs/1704.00514, 2017. URL https://api.semanticscholar.org/CorpusID:8002441.

[8] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *ArXiv*, abs/2310.10631, 2023. URL https://api.semanticscholar.org/CorpusID:264172303.

[9] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[10] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[11] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proc. Natl. Acad. Sci. USA*, 117(1):161–170, 2020. doi: 10.1073/pnas.1908636117. URL https://doi.org/10.1073/pnas.1908636117.

[12] Himanshu Batra, Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. Bert-based sentiment analysis: A software engineering perspective. In *International Conference on Database and Expert Systems Applications*, 2021. URL https://api.semanticscholar.org/CorpusID:235352765.

[13] Shawn L. E. Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O. Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. In

European Conference on Artificial Intelligence, 2020. URL `https://api.semanticscholar.org/CorpusID:211259472`.

[14] Marcel Bollmann and Anders Søgaard. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *International Conference on Computational Linguistics*, 2016. URL `https://api.semanticscholar.org/CorpusID:215824896`.

[15] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[16] Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C Alexander, and Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1385–1394, 2019.

[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] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search for resource-efficient branched multi-task networks. *arXiv preprint arXiv:2008.10292*, 2020.

[19] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[20] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple

baseline. *ArXiv*, abs/2004.07211, 2020. URL https://api.semanticscholar.org/CorpusID:215768806.

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

[22] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[23] Francisco Manuel Castro, Manuel J. Marín-Jiménez, Nicolás Guil Mata, Cordelia Schmid, and Alahari Karteek. End-to-end incremental learning. In *European Conference on Computer Vision*, 2018. URL https://api.semanticscholar.org/CorpusID:50785377.

[24] Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun, and Hoa T. Le. Multi-task dialog act and sentiment recognition on mastodon. *ArXiv*, abs/1807.05013, 2018. URL https://api.semanticscholar.org/CorpusID:49742988.

[25] Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. Zero-shot text-to-sql learning with auxiliary task. *ArXiv*, abs/1908.11052, 2019. URL https://api.semanticscholar.org/CorpusID:201667588.

[26] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[27] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

[28] Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. Revisiting parameter-efficient tuning: Are we really there yet? *arXiv preprint arXiv:2202.07962*, 2022.

[29] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating forgetting in online continual learning via instance-aware parameterization. *Advances in Neural Information Processing Systems*, 33, 2020.

[30] Kuilin Chen and Chi-Guhn Lee. Incremental few-shot learning via vector quantization in deep embedded space. In *International Conference on Learning Representations*, 2021.

[31] Lichang Chen, Heng Huang, and Minhao Cheng. Ptp: Boosting stability and performance of prompt tuning with perturbation-based regularizer. *ArXiv*, abs/2305.02423, 2023. URL `https://api.semanticscholar.org/CorpusID:258479677`.

[32] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

[33] Tairui Chen. Going deeper with convolutional neural network for intelligent transportation. *Worcester Polytechnic Institute: Worcester, MA, USA*, 2015.

[34] Xiaodong Chen, Yuxuan Hu, and Jing Zhang. Compressing large language models by streamlining the unimportant layer. *arXiv preprint arXiv:2403.19135*, 2024.

[35] 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.

[36] 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. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

[37] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2016. URL `https://api.semanticscholar.org/CorpusID:264145480`.

[38] Xin Cheng, Yankai Lin, Xiuying Chen, Dongyan Zhao, and Rui Yan. Decouple knowledge from paramters for plug-and-play language modeling. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL `https://api.semanticscholar.org/CorpusID:263867307`.

[39] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2534–2543, 2021.

[40] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

[41] Alexandra Chronopoulou, Matthew E. Peters, Alexander M. Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pre-trained language models. *ArXiv*, abs/2302.07027, 2023. URL `https://api.semanticscholar.org/CorpusID:256846453`.

[42] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele.

The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572, 2013.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[45] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, Jan Peters, et al. Sharing knowledge in multi-task deep reinforcement learning. In *8th International Conference on Learning Representations*, pages 1–11. OpenReview. net, 2020.

[46] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL `https://api.semanticscholar.org/CorpusID:52967399`.

[48] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5133–5141, 2018. URL `https://api.semanticscholar.org/CorpusID:53776855`.

[49] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[50] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, 2020. URL `https://api.semanticscholar.org/CorpusID:220665451`.

[51] Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*, 2020.

[52] Hanyu Duan, Yixuan Tang, Yi Yang, Ahmed Abbasi, and Kar Yan Tam. Exploring the relationship between in-context learning and instruction tuning. *arXiv preprint arXiv:2311.10367*, 2023.

[53] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Cantón Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab A. AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriele Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guanglong Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Laurens Geffert, Jana

Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Ju-Qing Jia, Kalyan Vasuden Alwala, K. Upasani, Kate Plawiak, Keqian Li, Ken-591 neth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline C. Muzzi, Mahesh Babu Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melissa Hall Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri S. Chatterji, Olivier Duchenne, Onur cCelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Chandra Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag,

Yashesh Gaur, Yasmine Babaei, Yiqian Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zhengxu Yan, Zhengxing Chen, Zoe Papakipos, Aaditya K. Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adi Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Ben Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Shang-Wen Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzm'an, Frank J. Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory G. Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Han Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie,

Jonathan Torres, Josh Ginsburg, Junjie Wang, Kaixing(Kai) Wu, U KamHou, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, A Lavender, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollár, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sung-Bae Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal

Mangla, Vlad Ionescu, Vlad Andrei Poenaru, Vlad T. Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xia Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024. URL `https://api.semanticscholar.org/CorpusID:271571434`.

[54] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[55] Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL `http://auai.org/uai2017/proceedings/papers/173.pdf`.

[56] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *European Conference on Computer Vision*, 2020. URL `https://api.semanticscholar.org/CorpusID:214612169`.

[57] Ali Edalati, Marzieh S. Tahaei, Ivan Kobyzev, V. Nia, James J. Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *ArXiv*, abs/2212.10650, 2022. URL `https://api.semanticscholar.org/CorpusID:254926823`.

[58] Mehrdad Farajtabar, Navid Azizan, Alexander Mott, and Ang Li. Orthogonal gradient descent for continual learning. *ArXiv*, abs/1910.07104, 2019. URL `https://api.semanticscholar.org/CorpusID:204734380`.

[59] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[60] Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. Towards llm-driven dialogue state tracking. *arXiv preprint arXiv:2310.14970*, 2023.

[61] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

[62] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *ArXiv*, abs/2109.04617, 2021. URL `https://api.semanticscholar.org/CorpusID:237485414`.

[63] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[64] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[65] Jhair Gallardo, Tyler L. Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. In *British Machine Vision Conference*, 2021. URL `https://api.semanticscholar.org/CorpusID:232352548`.

[66] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality

reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214, 2019.

[67] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11543–11552, 2020.

[68] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[69] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[70] Jianping Gou, B. Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789 – 1819, 2020. URL `https://api.semanticscholar.org/CorpusID:219559263`.

[71] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

[72] Lin Guan, Karthik Valmeekam, and Subbarao Kambhampati. Relative behavioral attributes: Filling the gap between symbolic goal specification and reward learning from human preferences. *arXiv preprint arXiv:2210.15906*, 2022.

[73] Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

[74] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[75] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

[76] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 270–287, 2018.

[77] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International Conference on Machine Learning*, pages 3854–3863. PMLR, 2020.

[78] Yiduo Guo, Wenpeng Hu, Dongyan Zhao, and Bing Liu. Adaptive orthogonal projection for batch and online continual learning. In *AAAI Conference on Artificial Intelligence*, 2022. URL `https://api.semanticscholar.org/CorpusID:250290748`.

[79] Rujun Han, Xiang Ren, and Nanyun Peng. Econet: Effective continual pretraining of language models for event temporal reasoning. In *Conference on Empirical Methods in Natural Language Processing*, 2020. URL `https://api.semanticscholar.org/CorpusID:237497382`.

[80] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

[81] Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. *ArXiv*,

abs/1910.02509, 2019. URL https://api.semanticscholar.org/CorpusID:203837695.

[82] Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2549–2560, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01d8bae291b1e4724443375634ccfa0e-Abstract.html.

[83] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[84] Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. Mera: Merging pretrained adapters for few-shot learning. *ArXiv*, abs/2308.15982, 2023. URL https://api.semanticscholar.org/CorpusID:261339605.

[85] Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W Bradley Knox, and Dorsa Sadigh. Contrastive prefence learning: Learning from human feedback without rl. *arXiv preprint arXiv:2310.13639*, 2023.

[86] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[87] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In Lenny Pitt, editor,

Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993, pages 5–13. ACM, 1993. doi: 10.1145/168304.168306. URL https://doi.org/10.1145/168304.168306.

[88] Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022. URL https://api.semanticscholar.org/CorpusID:249145348.

[89] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 529–536. MIT Press, 1994. URL http://papers.nips.cc/paper/899-simplifying-neural-nets-by-discovering-flat-minima.

[90] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 4003–4014, 2019.

[91] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

[92] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. *ArXiv*, abs/1902.00751, 2019. URL https://api.semanticscholar.org/CorpusID:59599816.

[93] 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.

[94] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[95] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International Conference on Learning Representations*, 2018.

[96] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3957–3966, 2021.

[97] Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Weiran Xu, Yu Sun, and Hua Wu. Hft: Half fine-tuning for large language models. *arXiv preprint arXiv:2404.18466*, 2024.

[98] Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *ArXiv*, abs/1910.06562, 2019. URL `https://api.semanticscholar.org/CorpusID:202783008`.

[99] Julio Hurtado, Alain Raymond-Sáez, and Álvaro Soto. Optimizing reusable knowledge for continual learning via metalearning. In *Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:235390470`.

[100] Ahmet Iscen, Jeffrey O. Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *European*

Conference on Computer Vision, 2020. URL https://api.semanticscholar.org/CorpusID:214775266.

[101] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

[102] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. *ArXiv*, abs/2110.03215, 2021. URL https://api.semanticscholar.org/CorpusID:238419458.

[103] Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:248476156.

[104] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. In *ICLR*, 2022.

[105] Khurram Javed and Martha White. Meta-learning representations for continual learning. *ArXiv*, abs/1905.12588, 2019. URL https://api.semanticscholar.org/CorpusID:168169908.

[106] Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *ArXiv*, abs/2307.04657, 2023. URL https://api.semanticscholar.org/CorpusID:259501579.

[107] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel,

Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[108] Guanying Jiang, Lingyong Yan, Haibo Shi, and Dawei Yin. The real, the better: Aligning large language models with online human behaviors. *arXiv preprint arXiv:2405.00578*, 2024.

[109] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJgIPJBFvH`.

[110] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2019.

[111] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark A. Hasegawa-Johnson, Sung Ju Hwang, and Chang Dong Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, 2022. URL `https://api.semanticscholar.org/CorpusID:250340593`.

[112] Ta-Chu Kao, Kristopher T. Jensen, Gido M. van de Ven, Alberto Bernacchia, and Guillaume Hennequin. Natural continual learning: success is a journey, not (just) a destination. *ArXiv*, abs/2106.08085, 2021. URL `https://api.semanticscholar.org/CorpusID:235435811`.

[113] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

[114] 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.

[115] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

[116] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[117] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6129–6138, 2017.

[118] Andreas Kopf, Yannic Kilcher, Dimitri von Rutte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Rich'ard Nagyfi, ES Shahul, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. Openassistant conversations - democratizing large language model alignment. *ArXiv*, abs/2304.07327, 2023. URL `https://api.semanticscholar.org/CorpusID:258179434`.

[119] Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. *ArXiv*, abs/2309.10105, 2023. URL `https://api.semanticscholar.org/CorpusID:262054014`.

[120] Richard Kurle, Botond Cseke, Alexej Klushyn, Patrick van der Smagt, and Stephan Günnemann. Continual learning with bayesian neural networks for non-stationary data. In *International Conference on Learning Representations*, 2020. URL https://api.semanticscholar.org/CorpusID:211091599.

[121] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3358–3365, 2013.

[122] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[123] Janghyeon Lee, Donggyu Joo, Hyeong Gwon Hong, and Junmo Kim. Residual continual learning. *ArXiv*, abs/2002.06774, 2020. URL https://api.semanticscholar.org/CorpusID:211132923.

[124] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *ArXiv*, abs/1703.08475, 2017. URL https://api.semanticscholar.org/CorpusID:5160005.

[125] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, Yu Zhang, and Ming-Wei Chang. Conditional adapters: Parameter-efficient transfer learning with fast inference. *ArXiv*, abs/2304.04947, 2023. URL https://api.semanticscholar.org/CorpusID:258060039.

[126] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://api.semanticscholar.org/CorpusID:233296808.

[127] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[128] Aaron J Li, Satyapriya Krishna, and Himabindu Lakkaraju. More rlhf, more trust? on the impact of human preference alignment on language model trustworthiness. *arXiv preprint arXiv:2404.18870*, 2024.

[129] Haoling Li, Xin Zhang, Xiao Liu, Yeyun Gong, Yifan Wang, Yujiu Yang, Qi Chen, and Peng Cheng. Gradient-mask tuning elevates the upper limits of llm performance. *arXiv preprint arXiv:2406.15330*, 2024.

[130] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

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

[132] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217, 2022. URL `https://api.semanticscholar.org/CorpusID:249192356`.

[133] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[134] Baohao Liao, Yan Meng, and Christof Monz. Parameter-efficient fine-tuning without introducing new latency. *ArXiv*, abs/2305.16742, 2023. URL `https://api.semanticscholar.org/CorpusID:258947572`.

[135] Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking

spell on base llms: Rethinking alignment via in-context learning. *arXiv preprint arXiv:2312.01552*, 2023.

[136] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. *ArXiv*, abs/2202.02931, 2022. URL `https://api.semanticscholar.org/CorpusID:246634117`.

[137] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

[138] Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong yan Mei. Lora dropout as a sparsity regularizer for overfitting control. *ArXiv*, abs/2404.09610, 2024. URL `https://api.semanticscholar.org/CorpusID:269149525`.

[139] Geert J. S. Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017. URL `https://api.semanticscholar.org/CorpusID:2088679`.

[140] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

[141] Hao Liu and Huaping Liu. Continual learning with recursive gradient optimization. *ArXiv*, abs/2201.12522, 2022. URL `https://api.semanticscholar.org/CorpusID:246430145`.

[142] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021.

[143] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *ArXiv*, abs/2310.18339, 2023. URL `https://api.semanticscholar.org/CorpusID:271065459`.

[144] Qijiong Liu, Jieming Zhu, Quanyu Dai, and Xiao-Ming Wu. Boosting deep ctr prediction with a plug-and-play pre-trainer for news recommendation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2823–2833, 2022.

[145] Qijiong Liu, Jieming Zhu, Yanting Yang, Quanyu Dai, Zhaocheng Du, Xiao-Ming Wu, Zhou Zhao, Rui Zhang, and Zhenhua Dong. Multimodal pretraining, adaptation, and generation for recommendation: A survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6566–6576, 2024.

[146] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019.

[147] Xialei Liu, Marc Masana, Luis Herranz, Joost van de Weijer, Antonio M. López, and Andrew D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268, 2018. URL `https://api.semanticscholar.org/CorpusID:44868747`.

[148] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *ArXiv*, abs/2103.10385, 2021. URL `https://api.semanticscholar.org/CorpusID:232269696`.

[149] Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empir-

ical bayes for few-shot learning. In *European Conference on Computer Vision*, pages 404–421. Springer, 2020.

[150] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12245–12254, 2020.

[151] Yijin Liu, Fandong Meng, and Jie Zhou. Accelerating inference in large language models with a unified layer skipping strategy. *arXiv preprint arXiv:2404.06954*, 2024.

[152] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[153] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S Yu. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems*, 30, 2017.

[154] Noel Loo, Siddharth Swaroop, and Richard E. Turner. Generalized variational continual learning. *ArXiv*, abs/2011.12328, 2020. URL `https://api.semanticscholar.org/CorpusID:227162606`.

[155] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Neural Information Processing Systems*, 2017. URL `https://api.semanticscholar.org/CorpusID:37308416`.

[156] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[157] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

[158] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939, 2018.

[159] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018. URL `https://api.semanticscholar.org/CorpusID:50770252`.

[160] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations*, 2021. URL `https://api.semanticscholar.org/CorpusID:247958259`.

[161] Gaurav Maheshwari and Michaël Perrot. Fairgrad: Fairness aware gradient descent. *Trans. Mach. Learn. Res.*, 2023, 2022. URL `https://api.semanticscholar.org/CorpusID:249926774`.

[162] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision*, 2018. URL `https://api.semanticscholar.org/CorpusID:3977226`.

[163] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive

single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1851–1860, 2019.

[164] Pratik Mazumder, Pravendra Singh, and Piyush Rai. Few-shot lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2337–2345, 2021.

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

[166] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *J. Mach. Learn. Res.*, 24:214:1–214:50, 2021. URL `https://api.semanticscholar.org/CorpusID:245329773`.

[167] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

[168] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11): 2624–2637, 2013.

[169] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017.

[170] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[171] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014.

[172] Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*, 2023.

[173] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[174] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Fast scene understanding for autonomous driving. *ArXiv*, abs/1708.02550, 2017. URL `https://api.semanticscholar.org/CorpusID:10322553`.

[175] Ansong Ni, Srini Iyer, Dragomir R. Radev, Ves Stoyanov, Wen tau Yih, Sida I. Wang, and Xi Victoria Lin. Lever: Learning to verify language-to-code generation with execution. *ArXiv*, abs/2302.08468, 2023. URL `https://api.semanticscholar.org/CorpusID:256900680`.

[176] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021. URL `https://api.semanticscholar.org/CorpusID:231979499`.

[177] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards

photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:245335086.

[178] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[179] OpenAI. Learning to reason with LLMs, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

[180] Oleksiy Ostapenko, Mihai Marian Puscas, Tassilo Klein, Patrick Jähnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11313–11321, 2019. URL https://api.semanticscholar.org/CorpusID:102353035.

[181] Oleksiy Ostapenko, Timothée Lesort, Pau Rodr'iguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *CoLLAs*, 2022. URL https://api.semanticscholar.org/CorpusID:250265129.

[182] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*, 2024.

[183] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010. URL https://api.semanticscholar.org/CorpusID:740063.

[184] Ashwinee Panda, Berivan Isik, Xiangyu Qi, Sanmi Koyejo, Tsachy Weissman,

and Prateek Mittal. Lottery ticket adaptation: Mitigating destructive interference in llms. *arXiv preprint arXiv:2406.16797*, 2024.

[185] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *International Conference on Machine Learning*, pages 27011–27033. PMLR, 2023.

[186] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR (Poster)*, 2016.

[187] 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 *BioNLP@ACL*, 2019. URL `https://api.semanticscholar.org/CorpusID:189762009`.

[188] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *ArXiv*, abs/2005.00247, 2020. URL `https://api.semanticscholar.org/CorpusID:218470208`.

[189] Quang Hong Pham, Chenghao Liu, and Steven C. H. Hoi. Dualnet: Continual learning, fast and slow supplementary materials. In *35th Conference on Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:244837226`.

[190] Fabrizio Pittorino, Carlo Lucibello, Christoph Feinauer, Gabriele Perugini, Carlo Baldassi, Elizaveta Demyanenko, and Riccardo Zecchina. Entropic gradient descent algorithms and wide flat minima. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=xjXg0bnoDmS`.

[191] Barbara Plank and Héctor Martínez Alonso. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Confer-*

ence of the European Chapter of the Association for Computational Linguistics, 2016. URL https://api.semanticscholar.org/CorpusID:2418468.

[192] Qi Qin, Han Peng, Wen-Rui Hu, Dongyan Zhao, and Bing Liu. Bns: Building network structures dynamically for continual learning. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:245011415.

[193] Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *ArXiv*, abs/2307.16789, 2023. URL https://api.semanticscholar.org/CorpusID:260334759.

[194] Yifu Qiu, Zheng Zhao, Yftah Ziser, Anna Korhonen, Edoardo M Ponti, and Shay B Cohen. Spectral editing of activations for large language model alignment. *arXiv preprint arXiv:2405.09719*, 2024.

[195] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290, 2023. URL https://api.semanticscholar.org/CorpusID:258959321.

[196] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[197] Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No robots. https://huggingface.co/datasets/HuggingFaceH4/no_robots, 2023.

[198] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13588–13597, 2020.

[199] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022. URL `https://api.semanticscholar.org/CorpusID:251648120`.

[200] Rahul Ramesh and Pratik Chaudhari. Model zoo: A growing brain that learns continually. In *International Conference on Learning Representations*, 2021. URL `https://api.semanticscholar.org/CorpusID:245007201`.

[201] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

[202] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[203] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard S Zemel. Incremental few-shot learning with attention attractor networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 5275–5285, 2019.

[204] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2018.

[205] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *ArXiv*, abs/1805.07810, 2018. URL `https://api.semanticscholar.org/CorpusID:29169199`.

[206] H. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 2007.

[207] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021. URL `https://api.semanticscholar.org/CorpusID:245335280`.

[208] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

[209] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[210] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

[211] Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL `https://api.semanticscholar.org/CorpusID:186206211`.

[212] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *2023 IEEE/CVF Conference on Computer Vision*

and Pattern Recognition (CVPR), pages 22500–22510, 2022. URL https://api.semanticscholar.org/CorpusID:251800180.

[213] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016. URL https://api.semanticscholar.org/CorpusID:15350923.

[214] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *ArXiv*, abs/2103.09762, 2021. URL https://api.semanticscholar.org/CorpusID:232257725.

[215] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[216] Jonathan Schwarz, Wojciech M. Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *ArXiv*, abs/1805.06370, 2018. URL https://api.semanticscholar.org/CorpusID:21718339.

[217] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[218] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:2157345.

[219] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical

image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017. URL https://api.semanticscholar.org/CorpusID:207614217.

[220] Jiayi Shen, Xiantong Zhen, Marcel Worring, and Ling Shao. Variational multi-task learning with gumbel-softmax priors. *Advances in Neural Information Processing Systems*, 34:21031–21042, 2021.

[221] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip H. S. Torr, Song Bai, and Vincent Y. F. Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16701–16710, 2021. URL https://api.semanticscholar.org/CorpusID:245005663.

[222] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. In *Neural Information Processing Systems*, 2020. URL https://api.semanticscholar.org/CorpusID:227178110.

[223] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[224] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[225] Anders Søgaard and Joachim Bingel. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2017. URL https://api.semanticscholar.org/CorpusID:3127682.

[226] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 18990–18998, 2024.

[227] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[228] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Learning task relatedness in multi-task learning for images in context. *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019. URL `https://api.semanticscholar.org/CorpusID:102352117`.

[229] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1375–1384, 2019. URL `https://api.semanticscholar.org/CorpusID:85544221`.

[230] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *North American Chapter of the Association for Computational Linguistics*, 2021. URL `https://api.semanticscholar.org/CorpusID:248405974`.

[231] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[232] Qianru Sun, Yaoyao Liu, Zhaozheng Chen, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning through hard tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[233] Xianghui Sun, Yunjie Ji, Baochang Ma, and Xiangang Li. A comparative study between full-parameter and lora-based fine-tuning on chinese instruction data for instruction following large language model. *arXiv preprint arXiv:2304.08109*, 2023.

[234] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*, 2021.

[235] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35:1299–1312, 2016. URL `https://api.semanticscholar.org/CorpusID:32710`.

[236] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9629–9638, 2021. URL `https://api.semanticscholar.org/CorpusID:234742637`.

[237] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12183–12192, 2020.

[238] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[239] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[240] Marvin Teichmann, Michael Weber, Johann Marius Zöllner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020, 2016. URL `https://api.semanticscholar.org/CorpusID:5064446`.

[241] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pages 489–497, 1996.

[242] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 266–282. Springer, 2020.

[243] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[244] A. Triki, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1329–1337, 2017. URL `https://api.semanticscholar.org/CorpusID:12253672`.

[245] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-

free low-rank adaptation. *ArXiv*, abs/2210.07558, 2022. URL https://api.semanticscholar.org/CorpusID:252907428.

[246] Tycho FA van der Ouderaa, Markus Nagel, Mart Van Baalen, Yuki M Asano, and Tijmen Blankevoort. The llm surgeon. *arXiv preprint arXiv:2312.17244*, 2023.

[247] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. *ArXiv*, abs/2001.06902, 2020. URL https://api.semanticscholar.org/CorpusID:210838646.

[248] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[249] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3637–3645, 2016.

[250] Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. *arXiv preprint arXiv:2309.04827*, 2023.

[251] Zhen Wan, Yating Zhang, Yexiang Wang, Fei Cheng, and Sadao Kurohashi. Reformulating domain adaptation of large language models as adapt-retrieve-revise. *ArXiv*, abs/2310.03328, 2023. URL https://api.semanticscholar.org/CorpusID:263672135.

[252] Jianxun Wang and Yixiang Chen. A review on code generation with llms: Application and evaluation. *2023 IEEE International Conference on Medi-*

cal Artificial Intelligence (MedAI), pages 284–289, 2023. URL https://api.semanticscholar.org/CorpusID:267337676.

[253] Liyuan Wang, Xingxing Zhang, Qian Li, Mingtian Zhang, Hang Su, Jun Zhu, and Yi Zhong. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nat. Mac. Intell.*, 5:1356–1368, 2023. URL https://api.semanticscholar.org/CorpusID:261276620.

[254] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:258685677.

[255] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime G. Carbonell. Characterizing and avoiding negative transfer. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11285–11294, 2018. URL https://api.semanticscholar.org/CorpusID:53748459.

[256] 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. *arXiv preprint arXiv:2109.01652*, 2021.

[257] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3:1–40, 2016.

[258] Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

[259] Junda Wu, Tong Yu, Rui Wang, Zhao Song, Ruiyi Zhang, Handong Zhao, Chaochao Lu, Shuai Li, and Ricardo Henao. Infoprompt: Information-theoretic

soft prompt tuning for natural language understanding. *ArXiv*, abs/2306.04933, 2023. URL https://api.semanticscholar.org/CorpusID:259108694.

[260] Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*, 2022. URL https://api.semanticscholar.org/CorpusID:247717213.

[261] Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefan 0 Soatto. Class-incremental learning with strong pre-trained models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9591–9600, 2022. URL https://api.semanticscholar.org/CorpusID:248005986.

[262] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

[263] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

[264] Dan Xu, Wanli Ouyang, Xiaogang Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018. URL https://api.semanticscholar.org/CorpusID:21670200.

[265] Jing Xu and Jingzhao Zhang. Random masking finds winning tickets for parameter efficient fine-tuning. *arXiv preprint arXiv:2405.02596*, 2024.

[266] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective

and generalizable fine-tuning. *ArXiv*, abs/2109.05687, 2021. URL https://api.semanticscholar.org/CorpusID:237491053.

[267] Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. Meta-attention for vit-backed continual learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 150–159, 2022. URL https://api.semanticscholar.org/CorpusID:247596966.

[268] Prateek Yadav, Qing Sun, Hantian Ding, Xiaopeng Li, Dejiao Zhang, Ming Tan, Xiaofei Ma, Parminder Bhatia, Ramesh Nallapati, Murali Krishna Ramanathan, Mohit Bansal, and Bing Xiang. Exploring continual learning for code generation models. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:259341641.

[269] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[270] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[271] Yunzhi Yao, Peng Wang, Bo Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:258833129.

[272] Qianru Sun Yaoyao Liu, Bernt Schiele. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[273] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020.

[274] Han-Jia Ye, Hexiang Hu, and De-Chuan Zhan. Learning adaptive classifiers synthesis for generalized few-shot learning. *International Journal of Computer Vision*, pages 1–24, 2021.

[275] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

[276] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *ArXiv*, abs/1708.01547, 2017. URL `https://api.semanticscholar.org/CorpusID:3693512`.

[277] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[278] Lu Yu, Bartlomiej Twardowski, X. Liu, Luis Herranz, Kai Wang, Yong mei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6980–6989, 2020.

[279] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[280] Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. Interventional few-shot learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[281] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

[282] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022.

[283] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[284] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[285] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1:364 – 372, 2018. URL `https://api.semanticscholar.org/CorpusID:52908642`.

[286] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[287] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12203–12213, 2020.

[288] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning. *arXiv e-prints*, pages arXiv–2003, 2020.

[289] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12455–12464, 2021.

[290] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[291] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

[292] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[293] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675, 2019. URL `https://api.semanticscholar.org/CorpusID:127986044`.

[294] Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. *arXiv preprint arXiv:2409.14381*, 2024.

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

[296] Hanbin Zhao, Yongjian Fu, Mintong Kang, Qi Tian, Fei Wu, and Xi Li. Mgsvf: Multi-grained slow versus fast framework for few-shot class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(3): 1576–1588, 2021.

[297] Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. *arXiv preprint arXiv:2004.12406*, 2020.

[298] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

[299] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment, 2023.

[300] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130:2337 – 2348, 2021. URL `https://api.semanticscholar.org/CorpusID:237386023`.

[301] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109:43–76, 2019. URL `https://api.semanticscholar.org/CorpusID:207847753`.