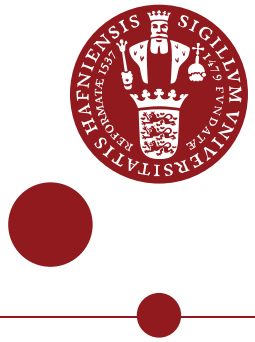
UNIVERSITY OF COPENHAGEN
DEPARTMENT OF COMPUTER SCIENCE



Information Propagation in Modular Language Modeling and Web Tracking

Zhan Su

Supervised by: Jakob Grue Simonsen, Rasmus Helles

This thesis has been submitted to the PhD School of The Faculty of Science,
University of Copenhagen

29th February 2024

Acknowledgements

This PhD thesis marks the finish of my doctoral journey, a journey that would have been impossible to complete without the support and encouragement of many. Foremost, I extend my deepest gratitude to my supervisors, Jakob Grue Simonsen and Rasmus Helles, whose guidance has been invaluable. Reflecting on my initial paper draft at the start of my PhD, I am astounded by the progress I've achieved through their mentorship.

My sincere thanks also go to the funding bodies that made my research possible: The Villum Foundation's Synergy Programme, The Carlsberg Foundation, and the University of Copenhagen's Data+ programme. Their support was crucial in my academic endeavor. Additionally, the vibrant research environment and the freedom to explore at the University of Copenhagen have been pivotal in maintaining my motivation and enthusiasm throughout my studies.

I am grateful for the opportunity to collaborate with remarkable colleagues—Lucas Caccia, Oleksiy Ostapenko, and Alessandro Sordoni at Microsoft, as well as the team at Mila Lab in Montreal. Their insights have significantly enriched my learning experience. My heartfelt thanks to my friends—Yuqin Zhou, Benyou Wang, and Qiuchi Li—for making this journey more enjoyable and fulfilling through their companionship and support.

Last but certainly not least, I owe a debt of gratitude to my family, Xingwei Ma, Guolin Su, and Jinmei Jia. Their unwavering support and encouragement have been my backbone throughout this challenging and rewarding period of my life.

This thesis is not just a reflection of my work, but a testament to the collective effort and support of all those mentioned above.

Abstract

Information propagation is the process through which data are transmitted within a system. The growth of large-scale web datasets has led to explosive growth in information, triggering new research areas such as large language models (Brown *et al.*, 2020a) or digital surveillance (Westerlund *et al.*, 2021).

In the realm of language modeling, this thesis studies information propagation in modular language modeling, where a subset of training parameters are treated as *modules*. Each *module* can be individually trained using a domain-specific dataset, leading to the creation of a *module* uniquely tailored to each domain. Following this, a *routing* function determines which module should be activated, and an *aggregation* function is then employed to integrate the outputs of the active modules. This study provides an in-depth analysis of information propagation in modular language modeling, examining three key aspects: *modules*, *routing*, and *aggregation*. Firstly, a novel *module* that incorporates tensor product representation is introduced, making a significant advancement in parameter efficiency by considerably reducing the number of parameters needed in the trained model. Secondly, the investigation explores a variety of routing functions in the context of multi-task learning, focusing on few-shot and zero-shot scenarios. thirdly, a new aggregation method is presented, designed from the ground up based on the newly proposed *modules*. Finally, we take steps to create modular language models by building and reusing a library of *modules*, paving the way for efficient and flexible utilization of language models across a wide array of tasks.

In the domain of digital surveillance, the research also delves into information propagation in web tracking, with a particular focus on the evolution of web tracking practices over the past decade. A comprehensive historical analysis of third-party web tracking practices is conducted by utilizing the Wayback Machine. This approach not only sheds light on the technical advancements in web tracking but also maps out the changing landscape of digital surveillance.

Resumé

Informationsformidling er den proces, hvorigennem data transmitteres inden for et system. Væksten af store webdatasæt har ført til en eksplosiv vækst i information, som har skabt nye forskningsområder såsom store sprogmodeller (Brown *et al.*, 2020a) og digital overvågning (Westerlund *et al.*, 2021).

Denne afhandling undersøger informationsformidling i modulær sprogmodellering, hvor en delmængde af træningsparametre behandles som *moduler*. Hvert *modul* kan trænes individuelt ved brug af et domænespecifikt datasæt, hvilket fører til skabelsen af et *modul*, der specifikt ertilpasset hvert domæne. Herefter bestemmer en *routing*-funktion, hvilket modul der skal aktiveres, og en *aggregerings*-funktion anvendes derefter til at integrere output fra de aktive moduler. Denne undersøgelse giver en dybdegående analyse af informationsformidling i modulær sprogmodellering, idet den undersøger tre nøgleaspekter: *moduler*, *routing* og *aggregering*. Først introduceres et nyt *modul*, der inkorporerer tensorproduktrepræsentation – et betydeligt fremskridt i parametereffektivitet ved betydeligt at reducere antallet af parametre, der er nødvendige i den trænede model. Dernæst udforskes en række routing-funktioner i kontekst af multi-opgavelæring med fokus på few-shot og zero-shot-scenarier. Dernæst præsenteres en ny aggregeringsmetode designet fra bunden baseret på de nyligt foreslåede *moduler*. Endelig tages der skridt til at skabe modulære sprogmodeller ved at bygge og genbruge et bibliotek af *moduler*, som baner vej for effektiv og fleksibel anvendelse af sprogmodeller på tværs af en bred vifte af opgaver.

Inden for digital overvågning undersøger afhandlingen information i webtracking med særligt fokus på udviklingen af webtrackingpraksisser over det seneste årti. En omfattende historisk analyse af tredjeparts-webtrackingpraksisser udføres ved at anvende Wayback Machine. Denne tilgang kaster ikke blot lys over de tekniske fremskridt inden for webtracking, men kortlægger også det skiftende landskab inden for digital overvågning.

Publications

The work presented in this thesis is organized by chapter, listed in order of their appearance.

- (Caccia *et al.*, 2022). Lucas Caccia, Edoardo Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, Alessandro Sordoni. Multi-Head Adapter Routing for Cross-Task Generalization. Advances in Neural Information Processing Systems (NeurIPS 2023). 2023.

- Mixture of LoRA Experts Using Tensor Product. (Manuscripts).

- Oleksiy Ostapenko*, Zhan Su*, Edoardo Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia*, Alessandro Sordoni* Towards Modular LMs by Building and Reusing a Library of LoRA Adapters. (ICML2024 submitted).

- (Su *et al.*, 2023). Zhan Su, Rasmus Helles, Ali Al-Laith, Antti Veilahti, Akrati Saxena, Jakob Grue Simonsen. Privacy Lost in Online Education: Analysis of Web Tracking Evolution. International Conference on Advanced Data Mining and Applications. 2023. pages 440–455.

The following papers or manuscripts are finished during my Ph.D. study, they are not highly related to the thesis.

- (Ostapenko *et al.*, 2023). Oleksiy Ostapenko, Lucas Caccia, Zhan Su, Nicolas Le Roux, Laurent Charlin, Alessandro Sordoni. A Case Study of Instruction Tuning with Mixture of Parameter-Efficient Experts. NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following. 2023.

- Zhan Su, Yuqin Zhou, Benyou Wang, Fengran Mo, Jian-Yun Nie, Jakob Grue Simonsen. Reinforced Query Expansion for Sparse Retrieval. (SIGIR2024 submitted)

- Zhan Su, Yuqin Zhou, Benyou Wang, Qiuchi Li, Jakob Grue Simonsen. Language Modeling Using Tensor Trains.

Contents

1 Introduction

Information propagation is roughly the process through which data, messages, or signals are distributed, transmitted, or conveyed within a system. It involves diverse applications spanning fields such as communication systems, computer networks, social sciences, and biology. The surge of large-scale web datasets has led to a marked increase in information, fostering new research areas in computer science and social networks (Chen, Castillo, *et al.*, 2022).

Information Propagation in Language modeling In natural language processing, the concept of transfer learning, where a model developed for one task is repurposed as the foundation for a model on a different task, represents a wide-ranging application of information propagation [1]. For instance, in natural language understanding tasks, the most commonly employed methods involve utilizing pre-trained models, which are typically trained with a language modeling objective (Devlin *et al.*, 2018). Transfer learning, leveraging pre-trained language models (PLMs), has emerged as the predominant approach, yielding strong performance on many NLP tasks (Devlin *et al.*, 2018; Qiu *et al.*, 2020). The most prevalent method of applying pre-trained language models (PLMs) to downstream tasks involves fine-tuning all the training parameters of the model (*Full-finetuning*). However, there are two notable limitations to this approach:

1. Fine-tuning PLMs requires duplicating the model for each individual task, which becomes prohibitively expensive when adapting to a large number of tasks (He *et al.*, 2021). Especially, the expansion of large-scale web datasets has been a pivotal factor in advancing language modeling (NLP) (Zhao *et al.*, 2023). This progression promotes a groundbreaking for language models such as Bert (Devlin *et al.*, 2018), GPT (Brown *et al.*, 2020b), LLama (Touvron *et al.*, 2023) and Mixtral (Jiang *et al.*, 2024). The advancement of Large Language Models (LLMs) has rendered the adaptation of *full finetuning* to multi-task transfer learning impractical and infeasible.

2. Training tasks independently can lead to the issue of *catastrophic forgetting* (McCloskey and Cohen, 1989). This phenomenon occurs when a model, upon learning from a new task, loses or 'forgets' the knowledge it acquired from previous tasks.

To tackle these challenges, modular language models have emerged as promising approaches. Both biological and artificial systems circumvent these weaknesses thanks to their inherent modularity (Ballard, 1986; Pfeiffer *et al.*, 2023). Expanding upon this concept, recent research has concentrated on the development of neural networks with

[1] This statement represents the author's own original conceptualization of information propagation within the language modeling.

explicit modularity (Jacobs, Jordan, and Barto, 1991; Rosenbaum *et al.*, 2017; Ponti, 2021; Pfeiffer, Kamath, *et al.*, 2020; Huang *et al.*, 2023; Ponti *et al.*, 2023a; Muqeeth *et al.*, 2024). The primary objective of this approach is to attain functional specialization and improve the reusability and composability of neural networks. These methods involve identifying 1) *modules* that can be updated independently and asynchronously, without impacting other parts of the model; 2) a *routing function* responsible for assigning a subset of relevant modules to each specific example or task; and 3) an *aggregation function* tasked with integrating the outputs of the active modules.

This thesis delves into the concept of modular language modeling. In Chapter 3, we introduce a novel module that employs tensor product representation. To investigate *routing functions*, we propose two latent skilled routing functions aimed at enhancing cross-task generalization, detailed in Chapter 3 and Chapter 2. For zero-shot tasks, we introduce several routing methods to make the modular LMs generalize to unseen tasks without training, as detailed in Chapter 4. The implementation of a tensor product-based *aggregation/routing* method is presented in Chapter 3. Finally, we present a novel modular LMs by building and resuing a library of *modules* in Chapter 4.

Information Propagation in web tracking Within the area of social networks, information propagation refers to the mechanism through which information spreads among individuals or groups. Take the example of Facebook, where a user named Sally updates her status or posts on a friend's wall about a new show she enjoyed. Information regarding this activity is usually shared with her friends. Consequently, the details of Sally's action have the potential to be transmitted transitively throughout the network, illustrating the dynamics of how information can ripple through a web of social connections. (Chen, Castillo, *et al.*, 2022). Research in this domain has seen a significant surge over the past decade (Wang *et al.*, 2015; Lerner *et al.*, 2016; Sims and Bamman, 2020; Chen, Castillo, *et al.*, 2022; Zhao *et al.*, 2022; Lee *et al.*, 2024). The surge in interest is primarily fueled by the swift growth of social media and online social networking platforms, which are expanding at an ever-increasing rate (Lerner *et al.*, 2016; Liu and He, 2020; He and Liu, 2020; Sims and Bamman, 2020; Jerez-Villota *et al.*, 2023; Joseph, 2023). Concurrently, alongside the remarkable expansion of social media and online social networking sites, digital tracking methods have become increasingly prevalent on the World Wide Web. This widespread adoption of tracking techniques raises serious concerns about the protection of personal privacy. A substantial amount of research focuses on digital surveillance, particularly through web tracking, as state-based web tracking methods are relatively easy to identify within the source code of a webpage (Helles *et al.*, 2020; Libert, 2015). However, studying digital surveillance historically presents several challenges:

1. Amassing a large-scale dataset of web tracking information from previous years is inherently challenging due to several factors. Primarily, the lack of comprehensive archives makes it difficult to retrieve past data (Lerner *et al.*, 2016; Karaj *et al.*, 2018). Additionally, privacy regulations and the proprietary interests of web services further restrict access to historical tracking data, creating significant gaps in the available information (Li *et al.*, 2019).

2. Exploring the historical development of web tracking techniques involves significant complexities. The rapid evolution of technology means that tracking mechanism become quickly outdated, replaced by new methods that are more sophisticated and harder to detect (Karaj *et al.*, 2018). These challenges underscore the difficulties in conducting a thorough historical examination of digital surveillance and web tracking, highlighting the need for innovative methodologies and adaptive strategies to overcome these obstacles (Agarwal and Sastry, 2022; Graux and Orlandi, 2022; Gandon and Hall, 2022; Halpin and Henshaw-Plath, 2022).

This thesis embarks upon a historical analysis of information propagation in web tracking. Specifically, we develop a historical tracker crawler utilizing the Wayback Machine (Lerner *et al.*, 2016; Karaj *et al.*, 2018). This innovative tool allows us to explore the evolution and development of third-party trackers over the past decade, providing insights into their progression and impact on web privacy. Our analysis concentrates on several crucial aspects: 1) The historical shifts in third-party tracker technologies and their adoption patterns across different countries. 2) The implications of these developments on user privacy and data security over time. This thesis provides a thorough understanding of the evolution and current landscape of web tracking. These insights are vital for formulating more effective approaches to digital privacy management and grasping the wider societal implications of web tracking practices.

1.1 Thesis Outline

The contributions of this thesis are categorized into distinct sections for clarity and organization.

In Section 1.2, we study information propagation within language modeling from three distinct perspectives. Firstly, under *module*, we study the implementation details of modules within the language models, examining their functionalities and the specific roles they play. Secondly, *routing* explores the criteria and mechanisms for selecting the appropriate modules for processing given inputs, highlighting the decision-making process within the model's architecture. Lastly, *aggregation* focuses on the methods employed to combine the outputs of active modules, detailing how these individual contributions are synthesized to produce a coherent output.

In section 1.3, we study information propagation in the context of web tracking. Specifically, we focus our analysis on the development of web-tracking practices within the past decade. This analysis employs statistical methods to understand the evolution and patterns of web tracking activities.

Information propagation in language modeling: ***Modules***. In language modeling, a *module* can be any component of a neural network architecture (Pfeiffer *et al.*, 2023). Section 1.2.3 unfolds the intricate variety of computation functions that constitute the backbone of modular design, specifically focusing on *parameter composition*, *input composition*, and *function composition*.

Subsequently, in paper 3, we introduce an innovative parameter composition strategy termed "tensorized modules". This approach leverages the tensor product operation to reduce training parameters (Panahi *et al.*, 2019). Tensorized modules represent a significant advancement in modular language models, offering a novel methodology for parameter-efficient finetuning (PEFT).

In paper 4, we introduce a novel approach to language modeling by proposing a modular language model (LM) that leverages a pre-trained library of modules. Our investigation spans various strategies for constructing a module library, aiming to enhance multi-task transfer capabilities and minimize task interference. Through our research, we have identified and implemented effective module selection techniques. Notably, our experimental findings demonstrate that clustering methods based on module (LoRA) similarity metrics can significantly optimize the process of module selection, thereby establishing a more efficient and versatile modular language model.

Routing Function: Selection of Active Modules. The mechanism of routing within language models delineates how active modules are selected for processing specific tasks, categorized into *fixed routing* and *learned routing*. *fixed routing* operates on a predefined schema, where the specialization of each module, along with the requisite module combinations for each task, is predetermined, known as *a priori*. Conversely, *learned routing* involves a dynamic selection process where the parameters governing the routing mechanism are optimized during the training. This method can further be subdivided into soft and hard routing. Soft routing is characterized by a distribution of routing scores. Hard routing, on the other hand, assigns a binary score to each module, categorically designating them as either active or inactive.

In contexts requiring adaptation to novel tasks with limited data, such as few-shot learning scenarios, we introduce routing strategies such as the multi-head routing function `MHR` detailed in Chapter 2, and the tensor product routing detailed in Chapter 3, specifically designed for managing latent-skilled experts. For zero-shot learning scenarios, where the model must adapt to tasks without any prior examples, a task predictor router and `Arrow` router are proposed (Chapter 4). This innovative approach enables the model to infer the most appropriate modules for unseen tasks, leveraging the underlying structure and semantics of the task description.

Aggregation Function: Integrating Outputs from Active Modules. The aggregation function plays a crucial role in synthesizing the outputs from active modules within a language model. It can be implemented through straightforward methods such as a weighted average or concatenation, or through more sophisticated approaches employing a learnable neural network that dynamically adapts based on the outputs of all engaged modules. Typically, the routing and aggregation functions are designed to work seamlessly together (Pfeiffer *et al.*, 2023). In alignment with our new tensor product routing strategy, we introduce a corresponding tensor product aggregation method (Chapter 3). This approach leverages the tensor product to combine module outputs in a manner that preserves the multidimensional relationships between different module contributions (Panahi *et al.*, 2019). Such a tensor-based aggregation is particularly well-suited to

complement the tensor product routing, ensuring that the selection and integration of module outputs are simultaneously optimized for complex task processing.

Finally, we target to build modular language models by building and reusing a library of *modules* (Chapter 4). We show the potential achievable by re-using independently or partially independently trained adapters (modules) with a zero-shot routing strategy. Overall. We compare different alternatives to build a library of adapters in such a way that they can perform well on the in-domain tasks and out-of-domain tasks. We investigate the strategic, modular augmentation of smaller (language) models, offering a promising direction for research that prioritizes efficiency, flexibility, and performance.

Information Propagation in Web tracking In Section 1.3, we investigate the dynamics of information propagation in the realm of web tracking, with a particular emphasis on the utilization of third-party trackers. This analysis is focused on understanding how web tracking mechanisms have evolved, specifically through the lens of third-party tracking entities.

In Chapter 5, we compare tracking activities on educational websites by contracting a sample of these sites with a control group. This comparison is meticulously designed to unravel the specific patterns and trends of web tracking within educational domains, offering insights into how these practices diverge from or align with broader web tracking behaviors. To facilitate this analysis, we developed a specialized historical crawler, *HTracker*, capable of extracting data on historical trackers from the vast archives of the Internet Archive's *WayBack Machine* (Lerner *et al.*, 2016). This tool enables us to traverse back in time and reconstruct the trajectory of third-party tracking practices, providing a longitudinal perspective on the evolution of web tracking. By employing statistical analysis methods, we dissect the development of third-party tracking on educational websites over time. This approach enriches our understanding of web tracking development and underscores the peculiarities of tracking in educational contexts compared to the wider web landscape.

1.2 Information Propagation in Language Modeling

In recent years, the de facto paradigm for natural language understanding (NLU) tasks has been to utilize pre-trained methods, particularly transformer models (Vaswani *et al.*, 2017a). Typically, these models are customized for particular tasks by fine-tuning all their parameters (Zhang and Yang, 2021; Ruder, 2017). However, this approach leads to the creation of a distinct set of fine-tuned model parameters for each task, becoming prohibitively resource-intensive in scenarios that involve numerous tasks (He *et al.*, 2021). Meanwhile, fine-tuning all network parameters for each specific task results in a model uniquely tailored for that task, inhibiting the utilization of shared information across different tasks (Ruder, 2017).

In multi-task scenarios, there are two main approaches for facilitating information sharing across various tasks. The first approach involves sequentially fine-tuning pre-trained

language models on each task, one after the other (Phang *et al.*, 2018). However, this approach encounters the problem of *catastrophic forgetting* wherein the model forgets previously learned information as it undergoes fine-tuning on subsequent tasks. The second approach, known as multi-task learning (Ruder, 2017; Vandenhende *et al.*, 2020; Sanh *et al.*, 2021a; Yang *et al.*, 2023; Hu *et al.*, 2024), involves simultaneously fine-tuning a pre-trained model on all target tasks by employing a weighted sum of the objective functions from each task. However, this approach requires simultaneous access to all tasks during training. Introducing new tasks requires a complete retraining process with all tasks jointly. Balancing multiple tasks also presents a significant challenge. As Lee *et al.* (2017) have demonstrated, these models tend to overfit on tasks with limited resources and underfit on those with more abundant resources (Pfeiffer, Simpson, *et al.*, 2020). These difficulties underscore the necessity for a more adaptable and efficient method of knowledge transfer within multi-task learning.

Parameter-efficient fine-tuning (PEFT) aims to resolve the above challenge by only training a small set of parameters while keeping most pre-trained parameters frozen (Lialin, Deshpande, *et al.*, 2023). Recently, A rich body of parameter-efficient fine-tuning (PEFT) approaches have been proposed (Houlsby *et al.*, 2019a; Zaken *et al.*, 2021; Karimi Mahabadi *et al.*, 2021; Li and Liang, 2021b; Zhang, Han, *et al.*, 2023). Methods like **Adapter** (He *et al.*, 2022), **Prefix Tuning** (Li and Liang, 2021b), **LoRA** (Hu *et al.*, 2021) and **IA3** (Liu, Tam, *et al.*, 2022a) often updating less than 1% of the original model parameters. These approaches have shown performance comparable to full-tuning in NLU tasks, allowing for separate and simultaneous training of modules for multiple tasks using the same backbone model. However, despite these advancements, PEFT approaches still grapple with difficulties in maximizing knowledge transfer across tasks, encountering similar limitations as seen in sequential fine-tuning and multi-task learning approaches (Pfeiffer, Kamath, *et al.*, 2020).

Modular deep learning has emerged as a s promising solution (Robbins, 2009; Ballard, 1986). Within this paradigm, computation units are crafted as parameter-efficient *modules*. Information is processed by directing it through a chosen subset of these modules and subsequently aggregating the outcomes (Pfeiffer *et al.*, 2023). Each module is tailored for a specific purpose, allowing for consistent reuse across different tasks. This modular architecture facilitates positive transfer and systematic generalization by decoupling computation from routing and updating modules locally (Pfeiffer, Kamath, *et al.*, 2020; Poth *et al.*, 2023; Muqeeth *et al.*, 2024). Such properties address the knowledge transfer challenges inherent in traditional PEFT and multi-task learning approaches, representing a significant advancement in the field of deep learning.

As a result, various studies have studied the concept of explicitly designing neural networks with a modular structure (Ponti, 2021; Rosenbaum *et al.*, 2017; Pfeiffer, Kamath, *et al.*, 2020; Vu *et al.*, 2021; Poth *et al.*, 2023; Huang *et al.*, 2024; Muqeeth *et al.*, 2024), These methods involve 1) *modules* (Section 1.2.3) which are sets of trainable parameters updated locally without impacting the other parameters. 2) A *Routing* (Section 1.2.4) mechanism that determine which *module* should be activated. 3) An aggregation function (Section 1.2.5) that decides how to integrate the outputs of these activated modules during the final forward pass.

This thesis advances the study of modular language models across three key dimensions. Initially, Section 1.2.1 lays the groundwork by detailing the foundational concepts and notations employed throughout the study. Given our emphasis on transformer-based language models, Section 1.2.2 delves into the fundamental architecture of these models. Subsequent sections are dedicated to exploring the modules, routing, and aggregation processes, each addressed in their respective section.

1.2.1 Concepts and Notation: The Basics

This section is dedicated to laying the groundwork for the concepts and notation. we employ a generalized notation designed to scaffold the conceptual framework connecting *modules* (Section 1.2.3), *routing* (Section 1.2.4), and *aggregation* (Section 1.2.5). This notation serves as a preliminary tool to elucidate the interrelationships and foundational principles underlying our study. we use these notations to facilitate an initial comprehension and to draw a broad outline of the theoretical landscape that our work navigates.

However, it is crucial to acknowledge that this introductory notation is not carried forward into the detailed discussions of the specific papers in this thesis. For the concrete analysis and examination of specific cases, theories, or models, we defer to the original notations as presented in the respective paper. Readers are encouraged to view the generalized notation introduced at the outset as a foundational bridge, a means to facilitate an initial understanding and to prepare for the deeper engagement that follows. As we progress through the thesis, the adoption of specific notation for each paper is intended to enhance clarity, foster accurate interpretation, and ensure that our discussion remains closely aligned with the established discourse within the field.

Notation	Definition
$x \in \mathcal{X}$	Input data
$y \in \mathcal{Y}$	Output data
$h \in \mathcal{H}$	Hidden representation
$t \in \mathcal{T}$	Task index
$f : \mathcal{X} \times \mathcal{H} \to \mathcal{Y} \times \mathcal{H}$	A computation function
θ	Shared parameters
$M = \{\phi_1, \ldots, \phi_M\}$	Set of module parameters
$\alpha \in \mathcal{A}$	Vector of routing scores
$r : \mathcal{X} \times \mathcal{H} \times \mathcal{T} \to \mathcal{A}$	Routing function
ρ	Routing parameters
g	Aggregation function
γ	Aggregation parameters

Table 1.1 Notation and definition of variables, functions, and operators used in this Section. We use the same notation used in Pfeiffer *et al.* (2023).

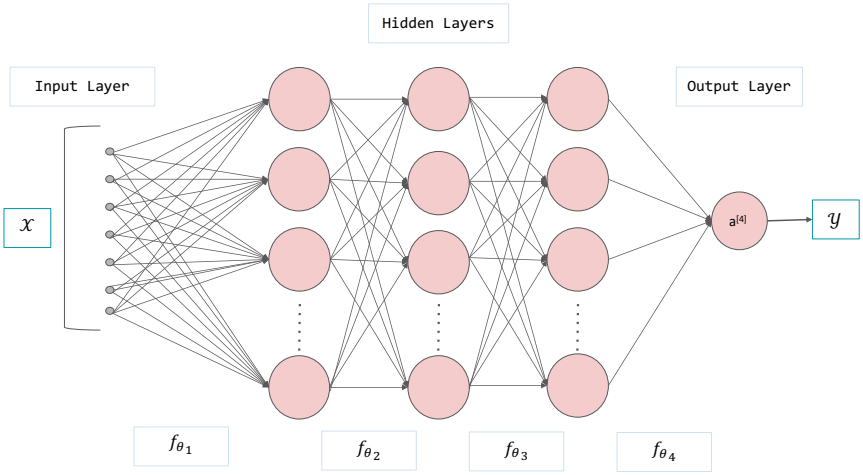


Figure 1.1 A linear chain representation of neural network $f_{\theta_1} \circ f_{\theta_2} \circ f_{\theta_3} \circ f_{\theta_4}$.

Consider a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ that can be decomposed into a graph consisting of sub-functions (Pfeiffer *et al.*, 2023). In its most basic form, this graph takes the shape of a linear chain, represented as $f_{\theta_1} \circ f_{\theta_2} \circ \cdots \circ f_{\theta_l}$, where $\circ$ denotes a function composition. The subfunction refers to the l layer within the model, each characterized by the uniquely indexed parameter $\theta_i, i = 1, ..., l$. These layers can be further broken down recursively into a network of their individual sub-functions. For instance, f_{θ_i} could be viewed as the i-th layer of a transformer model (Vaswani *et al.*, 2017a), which encompasses linear mappings for the query, key, value, and output, in addition to a nonlinear feed-forward network and residual connections (Section 1.2.2). The parameters at the beginning of the training process are denoted as θ^0, while the parameters after the training has been completed are denoted as θ^*.

Any i-th sub-function receiving input x can be adjusted by a module characterized by parameters ϕ selected from the inventory $M_i = \{\phi_1, ..., \phi_{|M|}\}$ in the following ways:

- Function composition: The revised sub-function is given by $f_{new} = f_\phi \circ f_{\theta_i}(x)$, wherein the output of the initial function is input into the subsequent function. This technique is exemplified by the introduction of adapter layers, as explored in the study by (Houlsby *et al.*, 2019b).

- Input composition: The updated sub-function is represented as $f_{new} = f_{\theta_i}([\phi, x])$, where $[\cdot, \cdot]$ denotes concatenation operation. This method finds application in techniques such as prefix tuning, as discussed by (Li and Liang, 2021b), and prompting tuning, as described by Lester *et al.* (2021).

- Parameter composition: The modified sub-function can be expressed as $f_{new} = f_{\theta_i \oplus \phi(x)}$, where $\oplus$ denotes an operation that amalgamates the existing parameters with new module parameters, thereby generating the updated training parameters. This approach is exemplified by the use of LoRA adapters as detailed in the work (Hu *et al.*, 2021).

For each i-th sub-function, module selection is governed by a routing function $r(\cdot)$, which assigns a score α_j to each module f_{ϕ_j} based on the data itself, such as a language token or a visual region x, or on metadata like the task identify $t \in \mathcal{T}$. The vector α can be predetermined through expert knowledge or dynamically learned via a specific parameterization $r_\rho(\cdot)$, where ϕ represents the learnable parameters of the routing function. The routing can take on one of two forms:

- Hard routing. $\alpha \in \{0,1\}^{|M|}$ is a discrete binary vector indicating the selection or exclusion of modules.
- Soft routing. $\alpha \in [0,1]^{|M|}$ is a continuous probability distribution, ensuring that $\sum_j \alpha_j = 1$, which allows for a weighted combination of module contributions.

The output for each module is combined using an aggregation function $g(\cdot)$. The simplest form of aggregation function, weighted averaging of outputs, has demonstrated significant performance benefits (Ponti *et al.*, 2023a; Fedus, Zoph, *et al.*, 2022b; Ostapenko *et al.*, 2023; Muqeeth *et al.*, 2024).

We present a popular modular language model, a sparse-gated mixture-of-experts(MoE) as discussed in (Shazeer *et al.*, 2017a). In their approach, a module can function as a feed-forward layer. A top-k routing taking a token representation x as input, subsequently routes it to the top-k modules selected from the available inventory $M_i = \{\phi_1, ..., \phi_{|M|}\}$ of M modules. The router function $h(x) = W_\rho x$ which are normalized via a softmax distribution over the $|M|$ modules. The routing score for module i is given by:

$$r_i(x) = \frac{e^{h(x)_i}}{\sum_j^M e^{h(x)_j}} \qquad \text{(MoE routing)}$$

where the router learns a probability distribution over the available modules. We denote the set of selected top-k experts indices as $\mathcal{K}$. Consequently, the routing and aggregation process can be described as follows:

$$g(x) = \sum_{\phi_j \in \mathcal{K}} r(\phi_j) f(x; \theta_i, \phi_j) \qquad \text{(MoE aggregation)}$$

where ϕ_j is the module j and θ_i is represents the parameter of sub layer i. Integrating the modules with the routing and aggregation functions yields a framework for a modular language model. We will present a modular language model based on transformer structure in Section 1.2.2.

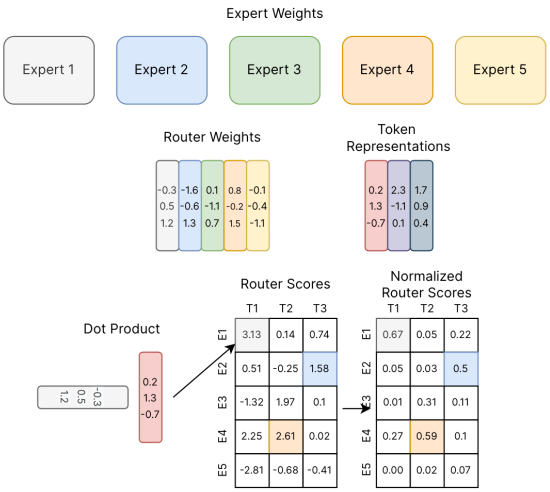


Figure 1.2 This is an example of the top-k token routing scheme over five experts and three input tokens (Fedus, Dean, *et al.*, 2022). Each expert is distinguished by a unique color code, and the router's weights include a specific representation for each expert. To determine the routing, the router weights perform a dot product with each token embedding x to produce the router scores. These scores are then normalized to ensure they sum to 1.

1.2.2 Transformer Structure

In this thesis, we examine language models that are built upon the transformer architecture. Before we present the specific *modules*, let us first review the foundational aspects of the transformer structure (Vaswani *et al.*, 2017a).

In the architecture of a standard transformer model, as described by (Vaswani *et al.*, 2017b), the model is composed of L sequential layers, each of which includes two primary components: a multi-head attention mechanism (MHA) and a fully connected feed-forward network (FFN). In the conventional attention function the queries $Q \in \mathbb{R}^{n \times d_k}$ and key-value pairs $K \in \mathbb{R}^{m \times d_k}$, $V \in \mathbb{R}^{m \times d_v}$ are mapped as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.1}$$

where n represents the number of queries and m denotes the number of key-value pairs. The multi-head attention mechanism executes the attention operation concurrently across N_h distinct heads, where each head is separately parameterized by $W_q^{(i)}$, $W_k^{(i)}$,$W_v^{(i)} \in \mathbb{R}^{d \times d_h}$ to project inputs to queries, keys and values. Given a sequence of m vectors $C \in \mathbb{R}^{m \times d}$, across which we aim to apply attention, and a query vector $x \in \mathbb{R}^d$, multi-head attention (MHA) calculates the output for each head and concatenates them:

$$\text{MHA}(X) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_o \tag{1.2}$$

$$\text{head}_i = \text{Attention}(xW_q^{(i)}, CW_k^{(i)}, CW_v^{(i)}) \tag{1.3}$$

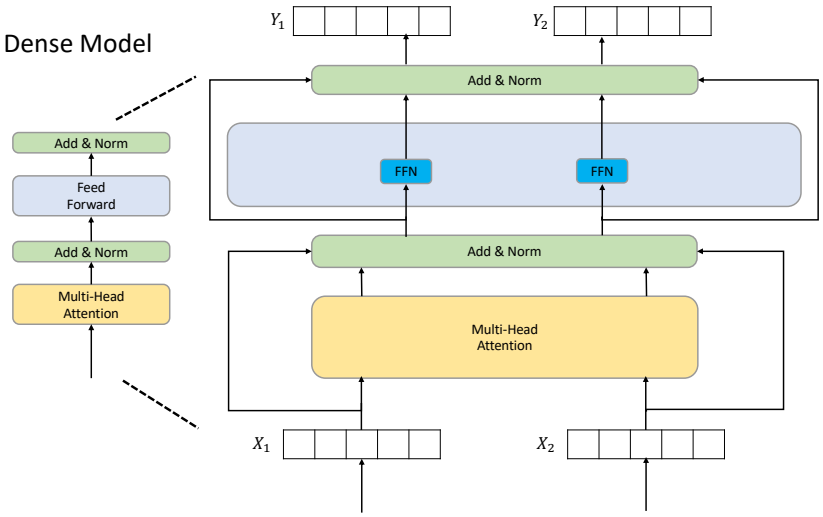


Figure 1.3 This figure presents a streamlined view of a transformer structure, highlighting the core components: Multi-head attention (MHA), Feed-forward network (FFN). The dense model sends both input tokens to the same FFN (Fedus, Dean, *et al.*, 2022).

where $W_o \in \mathbb{R}^{d \times d}$. d is the model dimension, and d_h is typically set to d/N_h to save parameters in MHA. Another crucial component is the fully connected feed-forward network (FFN), comprising two linear transformations with a ReLU activation function in between:

$$\text{FFN}(X) = \text{ReLU}(XW_{f_1} + b_1)W_{f_2} + b_2 \tag{1.4}$$

where $W_1 \in \mathbb{R}^{d \times d_m}$ and $W_2 \in \mathbb{R}^{d_m \times d}$. Transformers often employ a large model dimension, denoted as d_m, e.g. $d_m = 4d$. Subsequently, a residual connection is applied, followed by layer normalization (Ba *et al.*, 2016).

Recently, a substantial number of Transformer model variations have been introduced, demonstrating the model's versatility and impact across different domains. These range from earlier adaptations such as BERT (Devlin *et al.*, 2018), GPT (Radford *et al.*, 2019), T5 (Raffel *et al.*, 2020b), Palm (Chowdhery *et al.*, 2023) to more advancements in recent large language models like Gemini (Team *et al.*, 2023), GPT-3.5 Turbo (OpenAI, 2023) and GPT-4 (Achiam *et al.*, 2023). Additionally, openly released models such as LLaMA further enrich the landscape, showcasing the continuous innovation and broad applicability of Transformer-based architectures (Zhang, Han, *et al.*, 2023). Jiang *et al.* (2024) introduce Mixtral 8x7B, a modular language model with a Sparse Mixture of Experts (SMoE), which surpasses GPT-3.5 Turbo, Cemini Pro and LLama 70B. This development highlights the considerable potential for advancing modular language models in the future.

For all our papers about modular language models, all the backbone model is based on transformed structure, including T5 (Raffel *et al.*, 2020b) and T0 (Sanh *et al.*, 2021b) detailed in paper 3 and paper 2, GPTneo (Black *et al.*, 2021), Phi-2 (Microsoft Research, 2023), Stable-LM (Tow *et al.*, 2023) as detailed in paper 4.

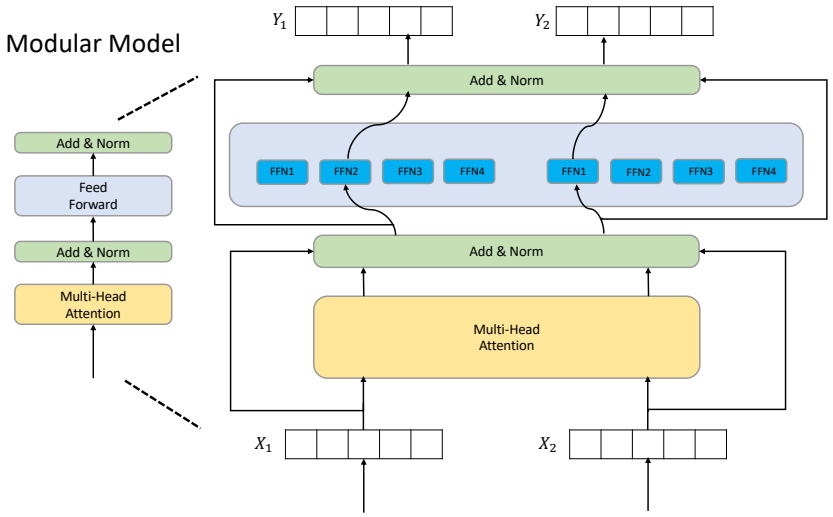


Figure 1.4 This figure presents a modular language model based on a transformer structure. In this case, modules are implemented as FFN. The modular model routes each input token independently among its four modules (experts).

1.2.3 Modules

In language models, *modules* are most often incorporated into a base architecture whose parameters are fully shared. For language modeling based on transformer structure (Vaswani *et al.*, 2017a), recent studies have introduced several Parameter-Efficient Fine-tuning (PEFT) approaches that only update a minimal additional set of parameters while keeping the majority of the pre-trained model's parameters frozen (Houlsby *et al.*, 2019a; Li and Liang, 2021b; Hu *et al.*, 2021; Liu, Ji, *et al.*, 2022; Liu, Tam, *et al.*, 2022b; Zhang, Chen, Bukharin, *et al.*, 2023; Zaken *et al.*, 2021; Karimi Mahabadi *et al.*, 2021; Edalati *et al.*, 2022; Zhang, Han, *et al.*, 2023). PEFT models can be classified in multiple ways according to their underlying approach or conceptual framework (Lialin, Deshpande, *et al.*, 2023). In this thesis, we classify the PEFT approaches into three distinct categories: function composition, input composition, and parameter composition, as outlined by (Pfeiffer *et al.*, 2023). Each of these methods offers a pathway to extend the functionality and applicability of language models, enabling them to be tailored to a wide range of tasks and domains while maintaining efficiency and scalability.

Function Composition

Function composition methods are the most general as they augment the model with new task-specific functions: $f_i'(x) = f_{\theta_i}(x) \circ f_{\phi_i}(x) = f_{\phi_i}(f_{\theta_i}(x))$, where $\circ$ denote function composition.

The most commonly adopted method for function composition in this context is *adapter layer* (Houlsby *et al.*, 2019a). It is viewed as a module f_{ϕ_i} that integrates with the functions f_{θ_i} of an existing model. The adapter layer approach inserts small modules between transformer layers. It generally uses a down-projection with $W_{down} \in \mathbb{R}^{d \times r}$ to

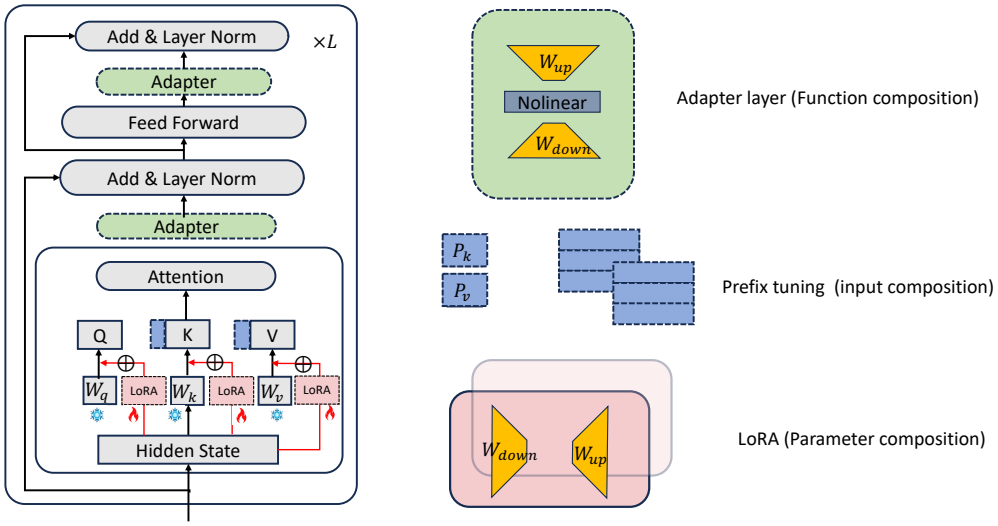


Figure 1.5 Different modular designs. we select three typical examples for each category. Function Composition is adapter layers (Houlsby *et al.*, 2019a) that are inserted in each layer that transforms the hidden representations. Input Composition is the prefix-tuning (Li and Liang, 2021b) extending the input sequence by pretending tunable matrices to the key and value in self-attention. Parameter Composition is LORA which updates the incremental low-rank matrix (Hu *et al.*, 2021).

project the input h to a lower-dimensional space specified by bottleneck dimension r, followed by a nonlinear activation function σ such as a ReLU unit, and an up-projection with $W_{up} \in \mathbb{R}^{r \times d}$. These adapters are surrounded by a residual connection, leading to a final form:

$$\begin{aligned} f_i^{'}(x) &= f_{\phi_i}(f_{\theta_i}(x)) \\ f_{\phi_i} &= \sigma(hW_{down})W_{up} \qquad \text{(adapter layer)} \\ h &= f_{\theta_i}(x) \end{aligned}$$

(Houlsby *et al.*, 2019a) places two adapters sequentially within each layer of the transformer, one after the multi-head attention and one after the FFN sub-layer.

Several studies have explored the different variants of adapter tuning (Rebuffi *et al.*, 2017; Pfeiffer, Kamath, *et al.*, 2020; He *et al.*, 2021; Karimi Mahabadi *et al.*, 2021; Liu, Tam, *et al.*, 2022a). For example, Karimi Mahabadi *et al.* (2021) introduce *Compacter*, a method designed for the fine-tuning of large-scale language models. This approach utilizes a hyper-complex, low-rank adapter to reparameterize W in the adapters as: $W = \sum_{i=1}^{n} A_i \otimes B_i$ where $A_i \in \mathbb{R}^{n \times n}$ is shared across layers (Karimi Mahabadi *et al.*, 2021; Pfeiffer *et al.*, 2023). IA3 Liu, Tam, *et al.* (2022a)) scales activations by learned vectors, attaining stronger performance while only introducing a tiny amount of new

parameters. This innovation highlights a significant advancement in the efficiency and effectiveness of model-tuning techniques.

Input Composition

Input composition methods wrap a model's input x by merging it with a learnable parameter vector ϕ_i : $f_i^{'}(x) = f_{\theta_i}[\phi_i, x]$. Typically, this augmentation applies to the model's initial input, directed into the first layer f_1, enriching the input space and potentially improving model performance by incorporating adaptable parameters directly into the input process. The conventional approach to prompting language models, as illustrated by ChatGPT (Brown *et al.*, 2020a; Achiam *et al.*, 2023), involves crafting a task-specific, discrete text prompt. This method is fundamental for guiding the model towards generating outputs aligned with the intended task requirements. However, this approach indeed exhibits sensitivity to how prompts are formulated (Lu *et al.*, 2021). This sensitivity underscores the importance of carefully designing prompts and selecting examples to guide models toward desired responses more effectively (Rubin *et al.*, 2021).

Instead, a continuous prompt vector ϕ can be learned directly and concatenated with the input (Lester *et al.*, 2021; Liu *et al.*, 2023; Hambardzumyan *et al.*, 2021). However, as ϕ is only concatenated with the first layer's input, the model has limited capacity to adapt to specific tasks. As a result, such continuous (also called soft) prompts perform poorly at smaller model sizes and on some harder tasks (Karimi Mahabadi *et al.*, 2021; Liu, Ji, *et al.*, 2022). To mitigate this, initialization via multi-task learning has been proposed (Vu *et al.*, 2021).

As an alternative, module vectors ϕ_i can be learned for each layer of the model: $f_i^{'}(x) = f_{\theta_i}([\phi_i, x])$ (Figure 1.5); (Li and Liang, 2021b; Liu, Ji, *et al.*, 2022). While this increases the number of parameters, it provides the model with more flexibility to adapt to a given task. In practice, module parameters in the form of prefix vectors $\phi_i = P_k^i, P_v^i \in \mathbb{R}^{l \times d}$ are prepended to the keys and values of every multi-head attention layer. Attention is defined as $f_i(x) = \text{Atten}(xW_q^i, CW_k^i, CW_v^i)$ where $W_q, W_k, W_v \in \mathbb{R}^{d \times d_h}$ are the projections that produce the queries, keys, and values and $C \in \mathbb{R}^{m \times d}$ is a sequence of context vectors. Prefix tuning thus takes the following forms:

$$f_i^{'}(x) = \text{Attn}(xW_q^i, [P_k^i, CW_k^i], [P_v^i, CW_v^i]) \qquad \text{(prefix tuning)}$$

Overall, input composition methods based on multi-layer prefix tuning (Li and Liang, 2021b; Yang and Liu, 2022) can be seen as a generalization of continuous prompting methods.

There are many variants of research for the input composition (Zhu *et al.*, 2023; Fang *et al.*, 2024; Wu *et al.*, 2024; Wang, Caccia, *et al.*, 2024; Razdaibiedina *et al.*, 2023). To

resist catastrophic forgetting, Razdaibiedina *et al.* (2023) introduce a *progressive prompt*, they learn a new soft prompt for each task to sequentially concatenate it with the previous learned prompts. Wang, Caccia, *et al.* (2024) introduce *planning tokens* and add their embeddings to the model parameters to improve the reasoning ability.

Parameter Composition

Parameter composition methods augment the function f_W of a base model with weights $W \in \mathbb{R}^{o \times i}$ with module parameters $\Phi \in \mathbb{R}^{o \times i}$, where i is the input dimensionality, and o is the output dimensionality (Pfeiffer *et al.*, 2023). The resulting function is parameterized as $f_{\theta \oplus \phi_i}$, where $\oplus$ stands for element-wise addition.

The standard parameter composition is the LoRA (Hu *et al.*, 2021). LoRA injects trainable low-rank matrices into transformer layers to approximate the weight updates. For the base model with weights $W \in \mathbb{R}^{o \times i}$, LoRA represents its update with a low-rank decomposition $W + \delta W = W + W_{down} W_{up}$, where $W_{down} \in \mathbb{R}^{d \times r}$, $W_{up} \in \mathbb{R}^{r \times k}$ are tunable parameters. As shown in Figure 1.5. For a specific input x to the linear projection in multi-head attention, LoRA modifies the projection output h as:

$$f_i^{'}(x) = f_\theta(x) + x W_{down} W_{up} \tag{LoRA}$$

Following the development of LoRA, numerous extended works have emerged, including significant contributions by Dettmers *et al.* (2024), who introduced QLoRA, an efficient finetuning approach by backpropagating gradients through a frozen, 4-bit quantized pre-trained language model into LoRA. Additionally, (Zhang, Chen, Bukharin, *et al.*, 2023) have introduced AdaLoRA, which adaptively allocates the parameter budget among weight matrices according to their importance score. Lialin, Muckatira, *et al.* (2023) introduces a novel method ReLoRA to train transformer language models and achieve comparable performance to regular neural network training. Huh *et al.* (2024) explores extending LoRA to model training and introduces LoRA-the-Explorer (LTE) to enable parallel training of multiple low-rank heads across computing nodes.

In this thesis, LoRA modules play a pivotal role. Specifically, in paper 3, we introduce TLoRA, a novel reparametrization of LoRA utilizing tensor products. In paper 2, we present the multi-head routing model, which applies fine-grained routing to the LoRA adapter. Further, in paper 4, we establish a module library built upon LoRA, subsequently devising various routing algorithms to facilitate the reuse of these LoRA adapters. Additionally, we explore the expansion of the module library through function composition and input composition, as elaborated in paper 4.

Hybrid Methods

Several approaches have integrated concepts from different categories of Parameter-Efficient Fine-Tuning (PEFT), as demonstrated in works by (He *et al.*, 2021; He *et al.*,

2022; Mao *et al.*, 2021). These methods blend techniques from function composition, input composition, and parameter composition to enhance model tuning and adaptation capabilities, showcasing the versatility and effectiveness of combining these strategies for improved performance in various tasks. For example, MAM (Mix-and-match) (He *et al.*, 2021) incorporates both Adapters and Prompt tuning. UniFELT (Mao *et al.*, 2021) add use different PEFT approaches as submodules and activate the best one based on a gate mechanism. Poth *et al.* (2023) introduce an open-source library that unifies 10 diverse adapter methods to address the challenges of conventional fine-tuning paradigms and prompting more efficient and modular and prompting.

A unique hybrid approach identified in our research involves treating the trainable parameters within the feed-forward and self-attention layers of transformer models as tensors, followed by applying tensor decomposition techniques (Jie and Deng, 2023; Pan *et al.*, 2024; Wang, Yang, *et al.*, 2024). These methods have been shown to significantly enhance parameter efficiency. However, its applicability is somewhat constrained by the necessity for compatibility with the structural specifics of the backbone model, thereby limiting its generalizability across different transformer-based architectures.

Contributions

In what follows, we outline the thesis's contributions to *modules*.

In our study (Chapter 3), we introduce a new hybrid *module* (TLoRA) optimization, using significantly fewer trained parameters. By reparameterizing LoRA's through a tensor product framework, our method extends beyond updating merely the self-attention layer, incorporating the feed-forward layer's reparameterization as well. This expansion is critical for enhancing overall performance, showcasing our approach's novelty and effectiveness in parameter-efficient model tuning. The training parameters in TLoRA are represented as finer-grained tensors, allowing for a substantial reduction in the number of training parameters—utilizing less than 1% compared to traditional full finetuning methods. In this way, we can reduce the training parameters significantly and use less than 1% of parameters compared to the full finetuning. TLoRA also induced the tensor product aggregation function introduced in Chapter 3.

In our Paper (Chapter 4), we build the *module* library to create our modular language model. Specifically, we find a positive correlation between the similarity of LoRA weights of tasks and transfer. We leverage the similarity in weight space, as observed through LoRA, among privately trained adapters as an indicator for identifying clusters of similar tasks. Subsequently, we train one *module* for each identified cluster. This approach provides deeper insights into how we can develop "new" modules from existing *modules*.

1.2.4 Routing Function: Selection of Active Modules

In the domain of modular language modeling, a diverse array of *modules* is accessible from a predefined inventory, denoted as $M = \{\phi_1, \ldots, \phi_{|M|}\}$. To effectively harness

Classification Category	Description
Soft Routing	Probabilistic Decision-making; inputs may be partially routed to multiple modules.
Hard Routing	Deterministic decision-making; inputs are directed to a single module.
Global Routing	Routing decisions affect the entire network.
Per Layer Routing	Routing decisions are made at each layer of the system.
Token Level Routing	Routing based on individual tokens in the input.
Example Level Routing	Routing decisions made for each input example.
Task Level Routing	Routing based on the type of task being performed.

Table 1.2 Overview of routing functions in modular language models. Routing functions can be categorized based on whether they are trained, the granularity of routing, or the inputs to the module. Each of these primary categories is further divided into several subcategories.

the capabilities of these modules, a critical decision-making mechanism is essential for selecting the active modules based on the input to the model. This selection mechanism is facilitated by a routing function, $r(\cdot)$, which allocates a score, α_i, to each module within the inventory M. The allocation of these scores is pivotal as it dictates the subset of modules that will be engaged in the computational process. Such a mechanism naturally leads to a sparse architectural framework, where only the selected (active) modules contribute to the model's computations. This strategic exclusion of inactive modules also enhances the model's capacity by focusing resources on the most relevant modules for a given task (Fedus, Dean, *et al.*, 2022).

Numerous approaches exist to categorize the various routing techniques (Fedus, Dean, *et al.*, 2022; Pfeiffer *et al.*, 2023). In general, the routing function $r(\cdot)$ can include *fixed routing* (or static routing) and *learnable routing*, each suited to different scenarios based on the availability and nature of metadata, such as expert knowledge about the subtasks or skills required for a task. *Fixed routing* is applicable when such metadata is available, allowing routing decisions to be made *a priori*, with each module's involvement predetermined based on the task's requirements.

In scenarios lacking prior information, the routing strategy must be learned (Pfeiffer *et al.*, 2023; Fedus, Dean, *et al.*, 2022). Learning routing can be subdivided into soft and hard routing. As depicted in Figure 1.6, *hard routing* involves learning a binary selection mechanism for modules, where a specific subset of modules is chosen for each decision-making step, effectively turning on or off certain modules based on the input.

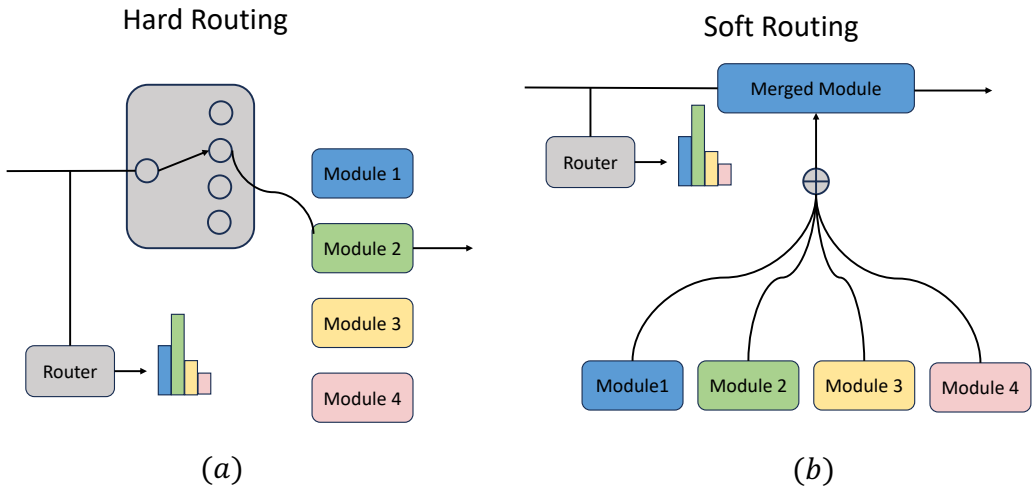


Figure 1.6 This example is explored in Muqeeth *et al.* (2023). (a) Hard learned Routing: learned hard selection modules. Each color symbolizes a distinct module. The router determines the module to engage by selecting the one with the highest probability. (b) Soft Learned Routing: soft selection and weighting of modules. The routing process produces a distribution over modules, and the contributions from all modules are aggregated according to their respective weights.

Conversely, *soft routing* aims to learn a probability distribution over modules, assigning weights to each module's contribution rather than making a binary choice. This approach allows for a more nuanced integration of module outputs, reflecting the probabilistic confidence in each module's relevance to the task.

Different routing methods can also be differentiated based on the nature of the input, as outlined by (Kudugunta *et al.*, 2021). Specifically, task-level routing involves directing the input to various experts based on task-specific information. In contrast, example-level routing dispatches the input to different experts based on the entire representation, absent any task-specific details. Token-level routing represents a more fine-grained approach, where each individual token is directed to distinct experts (Shen *et al.*, 2023).

Specifically, for transformer-based language models, additional methods exist to categorize routing based on the level of granularity. For instance, routing can be designed to utilize a shared routing function across all linear layers, a method known as global routing. Alternatively, finer-grained routing employs a distinct routing function for each linear layer. Block-wise routing involves using a separate routing function for each block of attention layers, whereas layer-wise routing applies a unique routing function to each individual linear layer, as discussed in Section 1.2.4.

Fixed Routing

Fixed routing makes discrete routing decisions a *priori*. Simplifying the routing function $r(\cdot)$ to the selection of a subset of modules $K \subset M$ for examples characterized by specific

metadata. This function can be implemented as a binary matrix $A \in \{0,1\}^{|\mathcal{T}|\times|M|}$ where each row corresponds to a possible task, and the number of columns corresponds to the size of the module inventory (Pfeiffer *et al.*, 2023; Ponti *et al.*, 2023a; Caccia *et al.*, 2022). (Roller *et al.*, 2021) show a *random* routing by hashing the input token can obtain a competitive performance with the learned routing.

In paper 2, we implement a fixed routing strategy during the fine-tuning process, opting for a uniform routing distribution across all layers. This approach yields results that are competitive with those achieved through learned routing. This conclusion is further supported by the findings in 4, where we apply uniform routing after constructing the module library, also achieving competitive outcomes in zero-shot scenarios.

Learned Routing

In the case of learned routing, a routing function is denoted as $r(\cdot)$, it can be implemented through training parameter ρ. This network takes as input the example x and outputs a routing score α (Pfeiffer *et al.*, 2023).

Hard Learned Routing Hard routing implies that the determination of a module's activity within the computation graph is binary, thereby excluding it or making it active. However, such discrete decisions pose challenges for vanilla gradient descent because it makes a discrete decision of which modules to select (Fedus, Dean, *et al.*, 2022). Consequently, alternative methods have been explored to facilitate inference, including **reinforcement learning** (Bengio, Bacon, *et al.*, 2015; Rosenbaum *et al.*, 2017; Rosenbaum *et al.*, 2019; Kirsch *et al.*, 2018; Chang *et al.*, 2018; Clark *et al.*, 2022), **evolutionary algorithms** (Fernando *et al.*, 2017; Fernando *et al.*, 2017; Gesmundo, 2022), and **stochastic re-parameterization** (Jang *et al.*, 2016; Maddison *et al.*, 2016). Hard routing can also be achieved through a continuous relaxation approach applied to the discrete latent variable, which determines module allocation. For instance, variable-size module routing can be achieved by learning a soft clustering or modules, also known as a soft partition of modules (Ponti *et al.*, 2023a; Caccia *et al.*, 2022; Pfeiffer *et al.*, 2023).

In Paper 2, we introduce a variable-size module routing mechanism that learns a finer-grained soft partition of modules. Subsequently, in Paper 3, we explore an alternative approach to variable-size module routing, employing tensor product techniques. These innovative routing mechanisms are designed to facilitate information sharing across tasks, enhancing the adaptability and efficiency of the modular language models.

Soft Learned Routing In *soft* routing methods, a *weighted combination* is employed to select and aggregate all modules, exemplified by the mixture of experts (MOE) model (Jacobs, Jordan, Nowlan, *et al.*, 1991a; Jordan and Jacobs, 1994). In this approach, the router learns a probability distribution over the available modules, denoted as $p(\mathcal{M}) = r_\rho(\cdot)$ (Pfeiffer *et al.*, 2023).

A rich body of works trains a continuous weighting of all modules (Eigen *et al.*, 2013; Meyerson and Miikkulainen, 2017; Wortsman *et al.*, 2020; Muqeeth *et al.*, 2023; Zadouri *et al.*, 2023). Since all the modules are activated in the forward pass, this may limit the degree of modularity. Additionally, activating *all* modules for each example significantly increases the computational cost (Pfeiffer *et al.*, 2023). To mitigate this, (Shazeer *et al.*, 2017a) only routed to top-k modules. Top-k MoEs stand between the hard routing and soft routing, as only a subset of modules is active as their average is weighted by the routing scores.

SwitchTransfomer Fedus, Zoph, *et al.* (2022b) demonstrates that even top-1 routing can achieve competitive results for language modeling. Muqeeth *et al.* (2023) present the Soft Merging of Experts with Adaptive Routing (SMEAR) technique, which eschews traditional hard-routing approaches in favor of creating a singular "merged" expert. This is achieved by calculating a weighted average of the parameters from all experts involved. Shen *et al.* (2023) utilizes Flan-MoE to integrate Mixture of Experts (MoE) with instruction tuning, effectively merging the advantages of instruction-finetuning with those of MoE. This strategy offers computational efficiency while also reducing memory demands. Ostapenko *et al.* (2023) explore the potential of using Mixture of Parameter-Efficient Experts (MoPEs) in instruction-tuning using large decoder-only language models. Their findings indicate that, given the relatively small datasets typically used for fine-tuning, the effectiveness of MoPEs is constrained.

In paper 4, our investigation encompasses a variety of soft routing strategies for both zero-shot and supervised fine-tuning contexts. The methods examined include uniform routing, task predictor routing, and `Arrow` routing, each offering distinct approaches to navigate and optimize model performance across different tasks and scenarios.

Routing with Different Granularity

In this subsection, we explore various routing methods categorized by their granularity, all of which are integrated into the Transformer architecture discussed in Section 1.2.2. As depicted in Figure 1.7, **coarse-grained** routing employs a singular routing function across all linear layers, facilitating a broader, less specific approach to module selection. In contrast, **fine-grained** routing applies a distinct routing function to each linear layer, allowing for more precise and tailored module engagement. Additionally, **block-wise** routing involves a routing strategy that is shared among attention layers and the feed-forward layer, promoting a medium level of granularity. Lastly, **layer-wise** routing employs separate routing functions for the self-attention layer and the feed-forward layer respectively, ensuring a highly detailed and focused routing mechanism.

Routing with Different Input Granularity

The performance of the routing function is significantly influenced by its input, a principle that holds considerable weight in the realm of natural language processing (NLP) tasks (Kudugunta *et al.*, 2021). Depending on the granularity of the input, routing functions

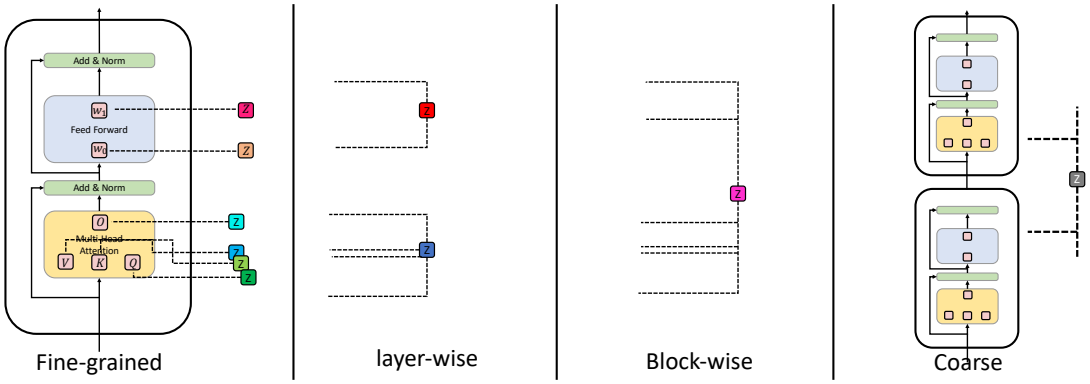


Figure 1.7 Different Routing Granularity. Z is the routing function. Coarse-grained routing shares the routing information across all linear layers. Conversely, Fine-grained routing applies a distinct routing to each linear layer. block-wise routing involves a routing that is shared among attention layers and the feed-forward layer while layer-wise routing employs separate routing functions for the self-attention layer and the feed-forward layer respectively.

within NLP can be categorized into three distinct levels: token level, which focuses on the smallest units of text; example level, which deals with individual instances or samples of data; and task level, which concerns itself with the overarching objectives or types of tasks being addressed.

Per-token routing Per-token routing is commonly employed in Mixture of Experts (MoE) models (Jacobs, Jordan, Nowlan, *et al.*, 1991b; Shazeer *et al.*, 2017b). In transformer-based language models, MoE models direct input tokens to a specific subset of Feed-Forward Network (FFN) modules. There are a rich body of MoE models using per-token routing (Shazeer *et al.*, 2017a; Lepikhin *et al.*, 2020; Fedus, Zoph, *et al.*, 2022b; Clark *et al.*, 2022; Yang *et al.*, 2021; Dua *et al.*, 2021; Rajbhandari *et al.*, 2022; Du *et al.*, 2022; Zhou, Yang, *et al.*, 2022; Zhou, Lei, *et al.*, 2022). After computing the routing scores, various types of per-token routing emerge based on the selection of tokens or experts. For example, 1) each token chooses top-k experts (Shazeer *et al.*, 2017a; Lepikhin *et al.*, 2020). 2) each expert chooses top-k tokens (Zhou, Lei, *et al.*, 2022; Lei *et al.*, 2024). 2) globally determine what tokens should be selected for each expert (Lewis *et al.*, 2021; Clark *et al.*, 2022).

In our paper 4, we propose a per-token routing `Arrow` which enables dynamic selection of the most relevant modules for new inputs without the need for retraining.

Example/task level Routing In the example level routing, all tokens of a single example can be routed to the same experts. Kudugunta *et al.* (2021) explore two variants of example-level routing for machine translation. In the case of sentence-level routing, they employ average pooling across token embeddings and subsequently condition the router on this aggregated representation. For task-level routing, a task-specific embedding is developed, upon which the router bases its learning of the distribution across modules (Pfeiffer *et al.*, 2023). Ponti *et al.* (2023a) and Caccia *et al.* (2022) introduce a latent skill routing function designed to disentangle and recombine various knowledge domains for

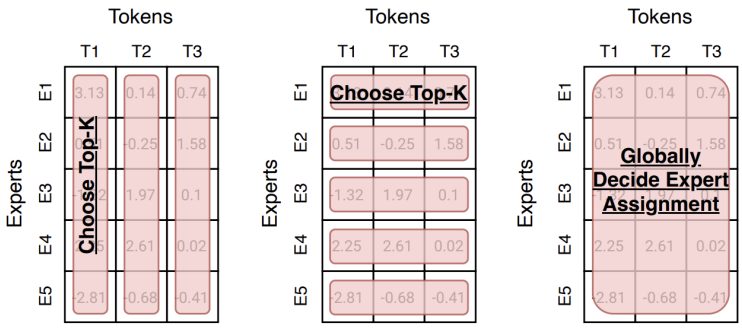


Figure 1.8 Figure 1.8 is from Fedus, Dean, *et al.* (2022). In this context, the term "experts" refers to modules. Upon obtaining the routing scores via an `Experts` × `tokens` matrix, we identify three prevalent categories of per-token routing algorithms. The first category, located on the left, is "Choose Top-k" which operates along the `Experts` axis, as discussed by (Shazeer *et al.*, 2017a). The middle category is "Choose Top-k" but operates along the `Tokens` axis, exemplified by (Zhou, Lei, *et al.*, 2022). The third category, on the right, is "Globally Decide Expert Assignment," akin to the approach used in the Base layer (Lewis *et al.*, 2021).

better generalization to new tasks. They develop a task-module allocation matrix, or routing mechanism, to determine the activation of specific modules for given tasks.

In our paper, as referenced in paper 4, within a zero-shot scenario, we introduce a task-predictor routing approach. Here, we conceptualize the routing strategies as a classifier, with the entire sentence serving as the input. This classifier then assigns a specific task ID based on the input.

Contributions

In what follows, we discuss this thesis' contributions to the *routing* functions.

In our paper 2, we propose a task-level routing function. We discuss Parameter-efficient fine-tuning (PEFT) for cross-task generalization, involving pre-training adapters on a multi-task training set before adapting them to test tasks with few-shot learning. In this paper, we want to answer three questions for *routing function*: Q1: Is the routing expressivity important? Q2: Why do routing-based PEFT methods yield superior performance? Q3: Is routing important in the pretrain or finetuning process? Our findings reveal that optimizing the routing function through fine-tuning alone can yield competitive results. In addition, the success of routing functions stems from better multi-task optimization. Our approach demonstrates the importance of fine-grained module selection for sample-efficient generalization and holds promise to improve other modular methods in future research.

In our paper 4, we investigate various *routing* functions designed to optimize the utilization of the pre-established module library in the zero-shot scenario. One straightforward *fix* routing method is to route to existing modules by setting the routing distribution to uniform for all layers. Another variant conceptualizes the routing process as a classifica-

tion problem with L distinct outcomes. In this model, the routing function is tasked with assigning each input to one of L predefined categories. Finally, we propose a `Arrow` routing which does not require access to training data for each expert (module). It can also parameterize a different routing distribution at every layer per token therefore potentially increasing overall model capacity. These routing approaches can take steps to create modular adaptable language models that can outperform traditional full-finetuning, paving the way for efficient and flexible utilization of LMs across a wide array of tasks.

1.2.5 Aggregation Function: How are the outputs of the active modules aggregated?

The routing function in a modular language model dictates which modules are activated. Following the selection of modules, an aggregation function is employed to integrate the outputs of all active modules. It is important to note that the processes of routing and aggregation are inherently inseparable (Pfeiffer *et al.*, 2023).

For a subset of active modules $K \subset M_i$, the aggregation of modular components can be implemented at various levels. At the *parameter* level, the aggregation is represented as $f_i^{'}(x) = f_{\phi_1 \oplus \cdots \oplus \phi_{|K|}}(x)$, where $\oplus$ denotes the operation of combining parameters from different modules. At the input level, it takes the form $f_i^{'}(x) = f_{\theta_i}([\phi_1, ..., \phi_{|K|}, x])$, incorporating the activated modules' outputs directly into the input. Lastly, at the function level, aggregation is achieved through sequential composition $f_i^{'}(x) = f_{\phi_1} \circ \cdots \circ f_{\phi_{|K|}}(x)$ (Pfeiffer *et al.*, 2023). These strategies for *aggregation* are reminiscent of previously discussed for *modules* (Section 1.2.3). These aggregation methods will be elaborated upon in the subsequent subsection.

Parameter Aggregation

An intuitive approach for aggregating information from various modules involves interpolating their weights (Pfeiffer *et al.*, 2023). For example, in the weight interpolation, Ansell, Ponti, Korhonen, *et al.* (2021) introduces Lottery Ticket Sparse Fine-Tuning (LT-SFT), where the task-specific modules can be aggregated by simply adding them to the base model. Hu *et al.* (2021) introduce LoRA, which represents its update with a low-rank decomposition $W + \delta W = W + W_{down} W_{up}$. (Liang *et al.*, 2023) presents the modular retrieval paradigm (REMOP), the modules are implemented as prefix prompt $P_1, ..., P_N$, Then the final module can be calculated as: $P_{final}(x) = P_{general} + \sum_i^N w_i P_i$. Huang *et al.* (2023) propose the LoRAHub which combined the LoRA modules using weighted averaging, the results on the Big-Bench Hard benchmark demonstrate that LoRAHub can match the performance of in-context learning in few-shot scenarios (Huang *et al.*, 2024). `Poly` employs a modular design to recombine different knowledge to generalize to new tasks. Once the routing function activates specific modules, these are then combined through a weighted average process, with the weights derived from the routing scores. Muqeeth *et al.*, 2023 introduce soft merging of experts with adaptive routing (SMEAR),

which uses a single "merged" expert constructed via a weighted average of module weights. All these approaches prove that parameter aggregation can achieve promising results. Collectively, these strategies underscore the efficacy of parameter aggregation in achieving notable outcomes, highlighting its potential to enhance the adaptability and performance of modular systems.

In this thesis, the technique of parameter aggregation is extensively applied across various chapters. Specifically, in our paper referenced as 3, for the implementations `TensorPoly-I` and `TensorPoly-II`, once the modules are activated, they are aggregated utilizing the principle of parameter aggregation. In our paper 2, the integration of multi-head modules is also achieved through the use of parameter aggregation. These approaches underscore the pivotal role of parameter aggregation in enhancing the functionality and adaptability of our proposed systems, demonstrating their effectiveness in synthesizing module outputs for improved performance.

Representation Aggregation

Representation aggregation focuses on combining the outputs from individual modules, as opposed to merging their weights. In the context of a linear function, the process of parameter aggregation aligns directly with that of representation aggregation. This highlights a method where the influence of each module is accounted for in the aggregated output through their respective contributions.

The most conventional method for representation aggregation is weighted representation averaging. In the context of the i-th sub-function of the model, where multiple modules $\phi \in M_i$, are present, the inputs are processed through the activated modules, yielding $|K_i|$ latent $h_1, ..., h_{|K_i|}$. An approach to aggregation involves learning the weights to interpolate across these hidden representations (Pfeiffer *et al.*, 2023):

$$f_i^{'}(x) = \sum_{j}^{K_i} \alpha_j h_j \tag{1.5}$$

where α_j is module-specific scalar weighting.

In this thesis, linear functions are predominantly employed to aggregate modules, indicating that parameter aggregation is directly congruent with representation aggregation in most instances. An exception to this norm is detailed in our paper 3. Within the `TensorPoly-II` approach, we introduce a more nuanced routing function that utilizes tensor product aggregation. Notably, tensor product aggregation at the representation level is not implemented in the current framework; however, it presents an intriguing avenue for future exploration and development.

Function Aggregation

Finally, aggregation can also occur at the function level, exemplified by $f_i^{'}(x) = f_{\phi_1} \circ f_{\phi_2}(x)$. The choice of aggregation method dictates whether the model employs a sequential (Pfeiffer, Vulić, *et al.*, 2020a; Pfeiffer, Vulić, *et al.*, 2020b) or hierarchical approach (Andreas *et al.*, 2016). For instance, in sequential aggregation, the process involves conducting a forward pass through multiple modules, with the output of one module serving as the input for the next. This results in the transformation of the hidden representations in a sequential manner: $f_{\phi_1}(f_{\phi_2}(...(f_{\phi_{|M|}}(x))))$ (Pfeiffer *et al.*, 2023).

In our paper 4, we construct a modular language model utilizing LoRA modules. In the present iteration of our model, we have opted not to incorporate function aggregation within our implementation framework. However, we posit that the integration of function aggregation could yield promising outcomes. The exploration of function aggregation as a component of our modular language model represents a significant opportunity for future research.

Contributions

Subsequent discussions will illuminate the distinct contributions made by this thesis to the domain of aggregation functions.

In paper 2, we employ routing scores to determine which modules are to be activated. The aggregation of these modules is then accomplished by averaging the products of the routing scores and their corresponding modules, with the routing scores serving as the determinant weight in this linear aggregation process. Contrastingly, in the paper 3, we adopt a tensor product aggregation approach, wherein the routing scores act as the modulating coefficient for the intricate tensor structure. Although the tensor product aggregation has the potential to extract more complex patterns from the dataset, our multi-task experimental findings indicate that a linear combination function may offer superior capabilities in terms of generalization.

1.3 Information Propagation in Web Tracking

The study of information propagation within social networks has increasingly captured the interest of researchers over recent years (Zhang and Ghorbani, 2020; Antonakaki *et al.*, 2021; Chen, Castillo, *et al.*, 2022; Cheng *et al.*, 2022). One distinguishing feature of online social networks is their capability to facilitate the dissemination of information through social connections. For instance, information can spread among friends within these networks, moving from one individual to the next, one hop at a time (Cha *et al.*, 2009). These interactions between users, often referred to as "word-of-mouth" exchanges, have the capacity to rapidly and extensively distribute content, ideas, or information across the network. A rich body of research has proposed viral marketing campaigns to

exploit the word-of-mouth effect (Dodds and Watts, 2005; Domingos and Richardson, 2001; Hartline *et al.*, 2008; Kempe *et al.*, 2003).

The study by (Adar and Adamic, 2005) explores the web tracking information flows in graphs, offering insights into the visualization and analysis of how information spreads. (Gruhl *et al.*, 2004) studies dynamics of information propagation within the realm of low-overhead personal publishing, utilizing an extensive dataset of weblogs over time to understand how information disseminates across such platforms. (Cha *et al.*, 2009) undertake a comprehensive collection and analysis of large-scale traces of information dissemination within the Flickr social network, examining the data of 2.5 million users and 11 million photos to investigate the breadth and speed of information spread in a social networking context. (Yang and Leskovec, 2010) propose viewing information flows as diffusion processes across social networks and developing a linear influence model to predict node influence dynamics, concentrating on how the global influence of a node affects the diffusion rate throughout the network.

With the proliferation of social networks, the dissemination of information has accelerated, necessitating a critical examination of the broader consequences. This rapid propagation of data within social networks is closely linked to the expansion of digital surveillance capabilities (Elharrouss *et al.*, 2021). Such surveillance is markedly enhanced by the vast amounts of data harvested from these platforms, enabling a level of monitoring that is far more extensive and invasive (Xiong *et al.*, 2020; Roger, 2022). The fluidity of information exchange on the web serves as a foundation for the development of advanced surveillance systems. These systems, in turn, possess the potential to significantly impinge upon individual privacy. Thus, as we scrutinize the intricacies of information flow in social networks, we must also grapple with the implications of such networks becoming tools for pervasive digital surveillance (Su *et al.*, 2023).

In the following sections, we will explore the mechanisms of digital surveillance in greater detail, examining how these practices are implemented, and their impact on user privacy.

1.3.1 Digital surveillance

Within the scope of digital surveillance in social networks, the transmission and handling of data related to individuals and collectives are essential for comprehending the mechanisms through which this information is accumulated, spread, and evaluated across a variety of technological platforms Christl and Spiekermann, 2016; Zuboff, 2023. The adoption of digital surveillance techniques, such as the use of cookies and tracking scripts, has become increasingly prominent on the World Wide Web, marking a significant shift in online privacy dynamics over the last decade (Solove, 2004; Zuboff, 2023; Easley, Kleinberg, *et al.*, 2010; Schneier, 2015; Russell, 2013). This rise in surveillance practices has introduced a multifaceted set of privacy concerns (Su *et al.*, 2023; Roger, 2022; Elharrouss *et al.*, 2021).

Digital surveillance is characterized by two main features: its *ubiquity* and its *opacity*. Ubiquity reflects the global reach of digital tracking; every nation with a digital infrastructure engages in a market for digital surveillance data, including both private and governmental sectors (Zuboff, 2023). Opacity, on the other hand, denotes the hidden nature of digital surveillance activities from those being monitored (Ho and Kallberg, 2017). The advancement of digital tracking technologies has notably increased the risk associated with privacy loss, which involves the unauthorized exposure, distribution, or access to personal information (Su *et al.*, 2023). Such privacy loss can lead to a range of negative consequences, from identity theft and financial fraud to reputational damage and discrimination(Craig and Ludloff, 2011).

Recently, a body of scholarly work has studied ethical, technological, and social dimensions of information privacy and security within digital ecosystems (Schneier, 2015; Tufekci, 2017; Lauer and Lipartito, 2021; Zuboff, 2023). These studies collectively underscore the critical need for ongoing research, policy development, and technological innovation to address the privacy challenges posed by digital surveillance in social networks and beyond.

1.3.2 Web tracking

Web tracking is a fundamental component of digital surveillance, acting as the technical mechanism through which data about individuals' online behaviors is systematically collected and analyzed (Mayer and Mitchell, 2012). Web tracking employs an array of technologies such as cookies, pixel tags, and fingerprinting to track and record users' internet activities (Lerner *et al.*, 2016). This data offers invaluable insights into user preferences, behaviors, and social interactions, facilitating targeted advertising, content personalization, and predictive analytics (Karaj *et al.*, 2018). There are several reasons that the web tracking can help us understand the digital surveillance. First, contrary to other surveillance methods, the technology of most state-based web tracking is easy to detect off the source code of the web page (Helles *et al.*, 2020; Libert, 2015). Also, historical trances of web tracking technology are stored in web archives as a "Wayback mahcines" (Helmond, Brügger, *et al.*, 2017; Lerner *et al.*, 2016). Therefore, the history of web tracking can be reconstructed for a longer period and at a level of detail far beyond other types of surveillance. Second, web tracking has expanded to the entire World Wide Web and presents itself as a good candidate for international (Samarasinghe and Mannan, 2019), comparative studies. As a result, web tracking can help us study digital surveillance from different cultures, politics, and regions (Bilić and Prug, 2021). Finally, web tracking remains an important source of surveillance information, and all major actors in the surveillance economy retain an active web tracking operation (Helles *et al.*, 2020; Libert and Nielsen, 2018).

Web Tracking Technologies

Web tracking involves various technologies designed to monitor and analyze user behavior online. Here are some key technologies used in web tracking. These technologies

underscore the variety of methods available for web tracking, each with its implications for user privacy and data security.

Cookies are small text files placed on a user's device by websites visited. They store data related to users' website visits, preferences, and activity, allowing for a personalized web experience and facilitating targeted advertising(Cahn *et al.*, 2016).

Pixel Tags(Web Beacons) are invisible, tiny images embedded in email and web pages. When accessed, they notify the server, allowing companies to track user behavior, email opens, and website visits(Shringarpure and Bustamante, 2015).

Browser Fingerprinting is a technique that collects information about a user's device and browser settings (such as screen resolution, operating system, and installed fonts) to create a unique identifier for tracking purposes. It is particularly insidious because it can track users across the web without relying on cookies (Acar *et al.*, 2013).

ETags(Entity tags) : Web cache validation tools that web servers use to understand if a user has a version of the web page saved in their cache. ETags can be exploited to track user uniquely by identifying their browser cache contents (Clausen, 2004).

LocalStorage and SessionStorage : part of the web storage API, these technologies allow websites to store data directly in a user's browser more expansively and flexibly than cookies. They can be used to save information across browsing sessions, aiding in tracking efforts (Mickens, 2010).

HTTP Referrer : This tracking method uses the HTTP header to determine which site a user previously visited, providing insights into the user's web navigation path. It helps in understanding traffic sources and user behavior(Mansoori *et al.*, 2016).

Social Media Widgets : Buttons like "Like" or "Share" on websites link to social media platforms and can track users, even if they don't interact with the widget but are logged into the social media servicec(Parker, 2008).

Tracking Scripts : JavaScript codes embedded in websites that can collect a wide range of data about user interaction on the site, such as clicks, page views, and form submissions. This data is then used for analytics and personalization purposes (Englehardt and Narayanan, 2016).

Geolocation Tracking : Uses IP addresses, WiFi, and GPS data to determine a user's physical location, allowing for location-specific content delivery and analytics (Musicki *et al.*, 2009).

Web tracking has catalyzed a significant and growing corpus of academic research within the realms of computer security and privacy. This burgeoning field of study aims to

comprehend, quantify, and develop defenses against such tracking practices (Acar *et al.*, 2014; Nikiforakis *et al.*, 2013; Akkus *et al.*, 2012; Bau *et al.*, 2013; Lerner *et al.*, 2016; Samarasinghe and Mannan, 2019; Xiong *et al.*, 2020; Shrotri *et al.*, 2021; Urman and Makhortykh, 2023; Goldberg *et al.*, 2024).

The study conducted by Acar *et al.* (2014) marks one of the pioneering large-scale investigations into the mechanisms of web tracking, establishing a foundation for future high-quality research in privacy measurement. Complementing this, Englehardt, Eubank, *et al.* (2015) introduced OpenWPM, a comprehensive and adaptable platform tailored for the analysis of web privacy. Leveraging OpenWPM, they further examined the implications of web tracking on surveillance in a follow-up study (Englehardt, Reisman, *et al.*, 2015). In a different vein, Han *et al.* (2012) investigated the realm of third-party tracking through mobile applications, specifically focusing on how user data is captured and shared across apps on Android smartphones. Nikiforakis *et al.* (2013) concentrated on the dynamics of web-based device fingerprinting, revealing the techniques websites use to track users without depending on conventional client-side identifiers like cookies. These studies collectively provide a broad and detailed perspective on the various dimensions and methods of web tracking, highlighting its implications for privacy and surveillance in the digital age.

Third-party Web Tracking

Building upon the foundational understanding of web tracking mechanisms and their privacy implications, we now turn our attention to a specific and pervasive form of tracking: third-party web tracking. This practice involves entities (trackers) that do not have a direct relationship with users but are nonetheless capable of collecting, storing, and analyzing user behavior across multiple sites (Li *et al.*, 2015). These trackers are embedded in websites through various means like scripts, images, or advertising networks, enabling the collection of detailed information about users without their direct interaction with the tracker's own site (Roesner *et al.*, 2012).

Third-party web tracking raises significant privacy concerns due to its invisible nature and the extensive data it can amass without explicit user consent (Lerner *et al.*, 2016; Bekos *et al.*, 2023; Müller and Bach, 2023). The implications of such tracking are far-reaching, influencing not only individual privacy rights but also broader societal norms regarding surveillance and data use. As users browse the web, their interactions with content, their search queries, and even the timing of their activities can be meticulously logged and analyzed, often without their knowledge or understanding of the extent of this surveillance (Mayer and Mitchell, 2012).

Moreover, the ecosystem of third-party tracking is complex and multifaceted, involving a range of actors from advertising networks and analytics companies to social media platforms (Mayer and Mitchell, 2012; Hoofnagle *et al.*, 2012; Yang *et al.*, 2022). These entities leverage tracked data for various purposes, from targeted advertising and market research to enhancing user experience and website functionality. However, the lack of transparency and control over personal data collection and use has sparked a significant

public and regulatory debate, emphasizing the need for more robust privacy protections and user empowerment in the digital age (Ermakova *et al.*, 2018). Lerner *et al.* (2016) presents a longitudinal study of third-party web tracking behaviors from 1996-2016, they propose a tool *TrackingExcavator* to collect the trackers from the Internet Archive's Wayback Machine. Schelter *et al.* (2018) utilized the extensive CommonCrawl dataset, as introduced by Patel and Patel (2020), to conduct a large-scale analysis of web tracking across the global internet landscape. This dataset, recognized as the most substantial empirical collection of web tracking data available, served as the foundation for their investigation. The findings from their analysis unequivocally demonstrate the pervasive presence of web trackers, highlighting the dominance of major entities like Google, Facebook, and Twitter in most regions.

In this thesis, our objective is to examine the evolution of web tracking over time. Obtaining historical data on dynamic tracking mechanisms like "cookies" proves challenging when relying on historical snapshots. As outlined in the section referenced as Paper 5, we adopt the use of third-party domains as proxies for trackers, a method commonly employed in the analysis of web tracking's history (Karaj *et al.*, 2018; Schelter *et al.*, 2018).

1.3.3 Historical Web Tracking

Examining the history of web tracking is critical for understanding how digital surveillance technologies have evolved in response to diverse political, economic, and cultural contexts across the globe (Zuboff, 2023). In the past decades, web tracking has witnessed significant shifts in the landscape of digital tracking, shaped by varying regulatory environments, technological advancements, and societal attitudes toward privacy and data security (Krishnamurthy and Wills, 2009; Karaj *et al.*, 2018; Gandon and Hall, 2022; Halpin and Henshaw-Plath, 2022). (Krishnamurthy and Wills, 2009) explored how third parties disseminate users' private information, presenting a longitudinal analysis over a four-year period to understand the dynamics of information diffusion. Karaj *et al.* (2018) introduced a methodology for measuring web tracking via a browser extension, utilizing data from 1.5 billion page loads collected over a 12-month period to analyze tracking practices. Agarwal and Sastry (2022) examined the top 100 Alexa-ranked websites using data from the Internet Archive (Wayback Machine) spanning 25 years, investigating the fluctuating popularity and categorization of these sites. Graux and Orlandi (2022) delved into a niche area of the web by analyzing papers accepted by the Web Conference and its annual gatherings, offering insights into the conference's evolution over time. Gandon and Hall (2022) traced the historical milestones in the development of the web, highlighting significant phases of its evolution. Halpin and Henshaw-Plath (2022) focused on tracking advancements in web history through the lens of status updates, providing a unique perspective on the web's progression.

The Role of Global Diversity in Web Tracking Evolution

The development of web tracking technologies has unfolded differently across countries and cultures (Rule and Greenleaf, 2010; Mansell and Raboy, 2011), reflecting local values, legal frameworks, and market dynamics. In the European Union, the advent of laws like the General Data Protection Regulation (GDPR) has forced a reevaluation of tracking practices, prompting greater transparency and user consent (Regulation, 2018; Li *et al.*, 2019; Voigt and Von dem Bussche, 2017; Peloquin *et al.*, 2020). Conversely, in countries with less regulatory oversight, tracking technologies have proliferated more freely, often raising concerns about user privacy and data protection.

Political and Economic Influences

The political climate of a region can significantly impact web tracking practices (Landau, 2011). For example, in authoritarian regimes, web tracking has been leveraged as a tool for surveillance and control, often without the knowledge or consent of the populace. This contrasts with more democratic societies, where surveillance through tracking is often done for commercial rather than political reasons (Moreira *et al.*, 2013; Deville and Van der Velden, 2015; Haddara *et al.*, 2023; González-Bailón *et al.*, 2023).

Economically, the growth of the digital advertising industry has been a major driver of web tracking technologies (Sevignani, 2015; Modi and Singh, 2023). The desire for more effective targeting and personalization of ads has spurred innovations in tracking methods, leading to more sophisticated ways of profiling and segmenting users. However, this has also led to a backlash among consumers and privacy advocates, prompting a reexamination of ethical practices within the industry.

Cultural Reflections Through Web Tracking

Cultural attitudes towards privacy have also influenced the acceptance and implementation of web tracking technologies (Acquisti *et al.*, 2007; Lyon, 2018; Stier *et al.*, 2020). In cultures where privacy is highly valued, there has been more resistance to invasive tracking practices, leading to the development of more privacy-centric technologies and policies. On the other hand, cultures with different conceptions of privacy may be more accepting of certain forms of surveillance, influencing the types of tracking technologies that are developed and deployed. For instance, while Germany and France exhibit a high sensitivity towards privacy concerns, their commercial web tracking practices are similar to those in Denmark, which may indicate a complex interplay between cultural perceptions of privacy and the technological landscape (Helles *et al.*, 2020).

The evolution of web tracking over the past two decades highlights the complex interplay between technology, society, politics, and the economy. It reflects broader debates about the balance between innovation and privacy, the role of government regulation, and the ethical considerations of digital surveillance. As we move forward, the history of web tracking serves as a reminder of the need for a nuanced approach to digital

technology—one that respects individual rights while embracing the benefits of the digital age.

Contributions

In what follows, we discuss this thesis' contributions to historical web tracking development.

In our paper 5, we study the privacy loss in online educational websites from 2012-2021. Our work highlights the pervasive threat of digital tracking to personal privacy, especially on educational websites, where users, including young individuals, may not fully understand the privacy implications. The paper's focus is on historical tracking practices on these websites and proposes a framework for comparing them to a control group. The findings show that while educational websites initially had lower tracking levels, they have grown significantly over the past decade, partly due to the increased use of interactive features and third-party services.

Our analysis raises concerns about privacy and independence in education. Privacy issues in educational websites should be prioritized, as they can lead to unauthorized disclosure of confidential information and loss of trust. Furthermore, researchers may wish to analyze privacy lost in other areas, such as news or sports, from a historical perspective. Our framework offers a convenient solution for creating comparable websites and collecting historical third-party tracker data in these domains.

Part I

Modular Language Modeling

2 Multi-Head Routing for Cross Tasks Generalization

2.1 Introduction

The ability to train effective models with a relatively small number of training data is of paramount importance due to the paucity of annotated examples for most tasks. One effective few-shot learning approach is to leverage large models pre-trained on a vast amount of unlabelled data and fine-tune them on the few examples available for each downstream task. To reduce the memory cost of duplicating the entire array of parameters for each downstream task, recent approaches resort to parameter-efficient fine-tuning (PEFT) methods, such as LoRA (Hu *et al.*, 2021), SFT (Ansell, Ponti, Korhonen, *et al.*, 2021), or $(IA)^3$ (Liu, Tam, *et al.*, 2022a). These only fine-tune adapters while leaving the pre-trained model 'frozen'.

Nevertheless, it remains unclear how to best exploit a set of *training* tasks to better generalize to a set of unseen *test* tasks in a sample-efficient fashion, based on just a few examples. One straightforward solution is to perform multi-task pre-training, i.e. first train the large model on the union of the examples from the training tasks, then fine-tune the obtained model to the test task (Liu, Tam, *et al.*, 2022a; Ye *et al.*, 2021a). However, this solution does not take into account that test tasks may require solving different combinations of sub-problems compared to training tasks (Vu *et al.*, 2020a), thus failing to achieve compositional generalization (Rosenbaum *et al.*, 2019; Ponti, 2021). Moreover, specializing the model towards different tasks during training may result in negative transfer, due to their corresponding gradients being misaligned (Wang *et al.*, 2021).

Several PEFT approaches have been proposed to enable better cross-task generalization by training adapters (or soft prompts) on each task independently (Pfeiffer, Kamath, *et al.*, 2020; Vu *et al.*, 2021; Asai *et al.*, 2022; Chronopoulou *et al.*, 2023). Given a new test task, parameters from similar training tasks are aggregated, which enables transfer. While solely having task-specific parameters is an effective strategy to mitigate interference across training tasks, it also inhibits any positive transfer within the same task pool. Polytropon (`Poly`) was recently proposed by Ponti *et al.* (2023b) to address these issues: the model assumes that task-specific adapters are learned combinations of a reusable inventory of basis adapters or *modules*. In practice, each module is implemented as a LoRA (Hu *et al.*, 2021) adapter, which modifies a large pre-trained model, such as T5 (Raffel *et al.*, 2020b). During both multi-task pre-training and few-shot adaptation, `Poly` learns both the inventory of adapters and a (continuously relaxed) binary task-module routing matrix, which determines which module is active for each task. While `Poly` shows promising results, several questions remain unanswered: 1) Does the expressivity of the routing function matter? 2) Why do routing-based PEFT methods

yield superior performance? 3) Is routing useful during both multi-task pre-training and few-shot adaptation?

To answer the first question, we propose a new routing function, `MHR`, that mixes adapters at a more granular level. Differently from `Poly`, where routing decisions are made for each adapter as a whole, in `MHR` we linearly combine blocks of the adapter dimensions (i.e. heads), each with different combination coefficients. We evaluate `MHR` and a series of competitive baselines for few-shot task adaptation on the T0 task suite (Sanh *et al.*, 2022) and Super-Natural Instructions (SuperNI; Wang, Mishra, *et al.*, 2022a). Based on our results, we report that `MHR` outperforms `Poly` and single adapter baselines. Additionally, we show that, thanks to the increased expressivity of the routing function, it becomes possible to fine-tune only the parameters of the routing function (and not the adapters) during few-shot adaptation: the resulting method, `MHR-`z, yields competitive performance while requiring orders of magnitude fewer parameters.

Regarding the second and third questions, we uncover that optimization during multitask pretraining plays a key role in explaining the downstream performance of routing-based PEFT approaches. Specifically, we find that `MHR` exhibits a higher cosine similarity between gradients from different tasks than `Poly` and single-adapter multi-task training. Hence, routing enables more knowledge transfer and less interference across tasks during multi-task pre-training. This finding led us to investigate whether routing is useful also during few-shot adaptation. It has been hypothesized (Ponti *et al.*, 2023b) that one of the reasons behind `Poly`'s performance resides in the inductive bias of the modular architecture, which allows test tasks to recombine and locally adapt the most relevant modules. To test this hypothesis, we propose `MHR-`μ, where the routing function is discarded and all available adapter parameters are averaged before a few-shot adaptation. We find that `MHR-`μ can recover the performance of `MHR`, hinting that `Poly`/`MHR` gains are only a result of better multi-task optimization. Finally, we show that `MHR-`μ can also be used as an effective zero-shot transfer method by training the average of the pre-trained adapters for a few additional steps on the multi-task training set. This yields gains up to 3% on absolute accuracy w.r.t. to strong baselines such as T0-11B.

2.2 Related Work

Multi-task learning is effective for low-resource tasks (Wei *et al.*, 2022; Aribandi *et al.*, 2022; Sanh *et al.*, 2022), as knowledge can be borrowed from similar tasks by sharing the model parameters. Multi-task learning has also been applied across languages and modalities (Ponti *et al.*, 2019; Bugliarello *et al.*, 2022). In the context of NLP, several families of methods enable learning new tasks from a limited set of labelled examples. Few-shot in-context learning (ICL; Brown *et al.*, 2020a), where examples of a new task are concatenated into an input prompt, enables models to generalize to *unseen* tasks without any gradient-based training. Such approaches are however sensitive to the prompt format and example ordering (Zhao *et al.*, 2021). More importantly, ICL methods incur a significant compute overhead, as for every prediction, the full set of examples must be processed by the model (Liu, Tam, *et al.*, 2022a). To remedy

this, many parameter-efficient fine-tuning (PEFT) methods have been proposed as an alternative to ICL, where a small number of new parameters are added over the frozen pre-trained network. To name a few, LoRA (Hu *et al.*, 2021) injects learnable low-rank matrices into each Transformer layer. Alternatively, the learnable matrix can be sparse, selecting nonzero shifts via the Lottery-Ticket hypothesis (Ansell, Ponti, Pfeiffer, *et al.*, 2021) or via their approximate Fisher information (Sung *et al.*, 2021). Finally, prefix-tuning methods (Li and Liang, 2021a) prepend learnable embeddings to the input or intermediate representations to specialize the model towards a downstream task.

Modular networks partition their parameters into several expert modules, each of them specialized to handle specific sub-tasks (Jacobs, Jordan, Nowlan, *et al.*, 1991a; Kirsch *et al.*, 2018). Modular networks are an appealing solution to the problem of adapting to unseen tasks (Corona *et al.*, 2021), as the model can leverage its existing modules and recombine them in a novel way, thus achieving systematic generalization (Bahdanau *et al.*, 2019). They have also been tested in learning scenarios with data presented sequentially (Ostapenko *et al.*, 2021), and with changing environments Goyal *et al.*, 2019. In NLP, mixture-of-experts (MoE) models (Shazeer *et al.*, 2017a; Fedus, Zoph, *et al.*, 2022a), where a learned gating mechanism routes token representations to appropriate experts (Feed-Forward layers), have shown success in scaling the number of parameters while retaining time efficiency. This results in higher performance when compared to their dense counterparts using a similar computing budget.

2.3 Background

In cross-tasks scenarios, we have a set of tasks $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_{|\mathcal{T}|}\}$. These tasks are partitioned into the train and test $\mathcal{T}_{train}$ and $\mathcal{T}_{test}$. The multi-task transfer learn is to transfer the knowledge in the $\mathcal{T}_{train}$ to the test tasks $\mathcal{T}$. There are two phases (*pretraining* and *finetuning*) for all the methods discussed, excluding the unsupervised pre-trained of the language model backbone on the large-scale training corpus. The first phase consists of multi-task pre-training using the dataset in tasks $\mathcal{T}_{train}$. The second consists of a few-shot adaptation, where the learned adapters are fine-tuned independently on each test task in $\mathcal{T}_{eval}$. We follow the procedure from (Raffel *et al.*, 2020b) and formulate each task as a text-to-text problem, enabling standard maximum-likelihood training with teacher forcing (Bengio, Vinyals, *et al.*, 2015) and a cross-entropy loss.

2.3.1 Adapters: LoRA & $(IA)^3$

LoRA (Hu *et al.*, 2021) and $(IA)^3$ (Liu, Tam, *et al.*, 2022a) are two recently proposed adapter architectures that achieve competitive trade-offs between performance and parameter efficiency (Mahabadi *et al.*, 2021; Liu, Tam, *et al.*, 2022a). For each linear transformation corresponding to the query (q), key (k), value (v) and output (o) of the self-attention layers, LoRA modifies the base model *parameters* as follows:

$$h^{q,k,v,o} = W_0^{q,k,v,o} x + s \cdot A^{q,k,v,o} (B^{q,k,v,o})^\top x, \qquad \text{(LoRA)}$$

where W_0 are the (frozen) weights of the pre-trained model (e.g. T5 (Raffel *et al.*, 2020b)). $A, B \in \mathbb{R}^{d \times r}$ are low-rank learnable parameters and $s \geq 1$ is a tunable scalar hyperparameter. $(IA)^3$, on the other hand, modifies key and value *representations* in self-attention element-wise, and also modifies the feed-forward MLP (f):

$$h^{k,v} = l^{k,v} \odot (W_0^{k,v} x); \; h^f = (l^f \odot \gamma(W_1^f x)) W_2^f, \qquad ((IA)^3)$$

where $l^{k,v,f} \in \mathbb{R}^d$ are learnable parameters, $W_{1,2}^f$ the frozen parameters of the feed-forward layer in the backbone, and γ a non-linearity. For clarity, we will drop the superscripts q, k, v, o in the rest of the paper.

2.3.2 Polytropon: Adapter Routing

Typical adapter methods either fully share adapters across tasks or train individual adapters for each task. `Poly` addresses the multi-task problem by softly sharing adapter parameters across tasks. Each `Poly` layer contains 1) an inventory of adapter modules $\mathcal{M} = \{\phi_1, \ldots, \phi_m\}$ with $|\mathcal{M}| \ll |\mathcal{T}|$; 2) a routing function $r(\cdot)$ that chooses which subset of the modules to combine for each task.

Each module corresponds to a LoRA adapter, where ϕ_i are its associated parameters $A^{(i)}, B^{(i)} \in \mathbb{R}^{d \times r}$. $r(\cdot)$ is implemented as a task–module routing matrix $Z \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{M}|}$. $z_\tau = Z_{\tau,:} \in \mathbb{R}^{|\mathcal{M}|}$ is a routing vector of task $\mathcal{T}_\tau$, with cell $Z_{\tau,j}$ being the probability logits of using module ϕ_j for task $\mathcal{T}_\tau$ in the current layer. Differently from mixture-of-experts (Fedus, Zoph, *et al.*, 2022a), which perform token-level top-k routing, Z converges to a binary matrix, defining a soft partition over modules. This is achieved by using a Gumbel-sigmoid distribution (Jang *et al.*, 2016) during training, with $\hat{Z}_{\tau,j} \sim \texttt{Gumbel}(Z_{\tau,j})$. At each forward pass, `Poly` can be defined as :

$$A^\tau = \sum_i \alpha_i A^{(i)}; \; B^\tau = \sum_i \alpha_i B^{(i)} \qquad \text{(Poly)}$$

where $\alpha_i = \frac{\hat{Z}_{\tau,i}}{\sum_j \hat{Z}_{\tau,j}}$, and $A^{(i)}, B^{(i)}, A^\tau, B^\tau \in \mathbb{R}^{d \times r}$. We normalize the mixing coefficients $\hat{Z}_{\tau,i}$ for each task to ensure that the number of active modules does not affect the norm of A^τ, B^τ. Overall, this approach enables different subsets of *modules* to be activated for the current layer and combined in a task-specific way. Following LoRA, the output of the `Poly` layer is added to the output of the original layer of the frozen backbone: $h = W_0 x + s A^\tau (B^\tau)^\top x$.

During multi-task pre-training, for each query, key, value, and output projection in self-attention layers, the parameters learned by `Poly` are the adapter parameters, $\{A_i, B_i\}_{i=1}^{|\mathcal{M}|}$, and the routing matrices Z. During fine-tuning, for each test task τ, `Poly` randomly initialize the routing vector $z_\tau \in \mathbb{R}^{1 \times |\mathcal{M}|}$ and fine-tunes both z_τ and all the modules parameters $\mathcal{M}$.

Method	Pre-Training	Fine-Tuning	Inference
`Full FT`	$d \times d$	$d \times d$	$d \times d$
`LoRA`	$d \times 2r$	$d \times 2r$	$d \times 2r$
`Poly`	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{T}\rvert \times \lvert\mathcal{M}\rvert$	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{M}\rvert$	$d \times 2r$
`Poly`-z	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{T}\rvert \times \lvert\mathcal{M}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{M}\rvert$
`MHR`-μ	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{T}\rvert \times \lvert\mathcal{M}\rvert$	$d \times 2r$	$d \times 2r$
`MHR`-z	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{T}\rvert \times \lvert\mathcal{M}\rvert \times h$	$\lvert\mathcal{M}\rvert \times h$	$\lvert\mathcal{M}\rvert \times h$
`MHR`	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{T}\rvert \times \lvert\mathcal{M}\rvert \times h$	$d \times 2r \times \lvert\mathcal{M}\rvert + \lvert\mathcal{M}\rvert \times h$	$d \times 2r$

Table 2.1 Number of parameters (per layer) used for each method. The calculation uses `LoRA` as the base adapter, modifying a linear transform in $\mathbb{R}^{d\times d}$. Note that the total number of parameters changed by `Full FT` is larger, given that the method also changes parameters for layers not modified by `LoRA`.

2.4 Multi-Head Adapter Routing

In `Poly`, module combination remains *coarse*: only linear combinations of modules are possible, and thus the resulting aggregated adapter remains a linear function of the modules. We propose to augment the expressivity of the module combination while keeping the parameter count similar. `MHR` (Fig. 2.1) takes inspiration from multi-head attention (Vaswani *et al.*, 2017c): it partitions the input dimensions into h different disjoint blocks, performs a separate `Poly`-style combination for each of them, and finally concatenates them. This corresponds to learning a different routing matrix Z for each block of input features, therefore enabling the model to select different adapters for different blocks of the input dimensions. This aggregation approach is *piecewise* linear (i.e., linear within disjoint intervals), which allows for more expressive combinations of modules.

In each `MHR` layer, the routing function is a third-order tensor $Z \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{M}|\times h}$, where $Z_{:,:,h} \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{M}|}$ is a 2D slice of the tensor Z. A slice represents the routing matrix for each of the h heads. Let us denote with $W[k] \in \mathbb{R}^{\frac{d}{h}\times r}$ the k-th partition along the rows of the matrix $W \in \mathbb{R}^{d\times r}$. The adapter parameters $A^\tau \in \mathbb{R}^{d\times r}$ for task τ, and for each adapter layer, are computed as (similarly for B^τ):

$$A_k^\tau = \sum_j A_j[k] \cdot \frac{\hat{Z}_{\tau,j,k}}{\sum_j \hat{Z}_{\tau,j,k}} \quad \text{with} \quad A_k^\tau \in \mathbb{R}^{\frac{d}{h}\times r}, \tag{MHR}$$

$$A^\tau = \texttt{concat}(A_1^\tau, \ldots, A_h^\tau)$$

where `concat` concatenates along the first dimension. Multi-task pre-training and fine-tuning are similar to `Poly`. Note that `MHR` results in only a negligible increase in the total amount of parameters, since most of the parameters are contained in the LoRA weights A, B (Tab. 2.1).

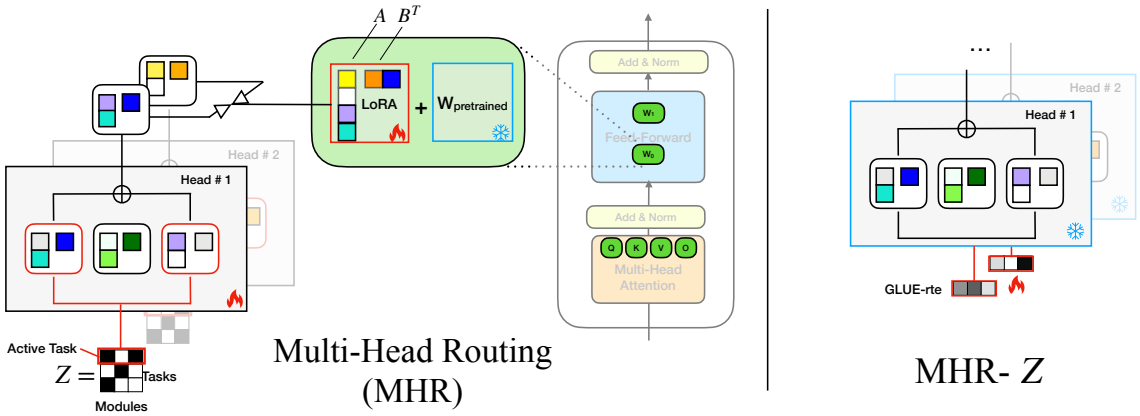
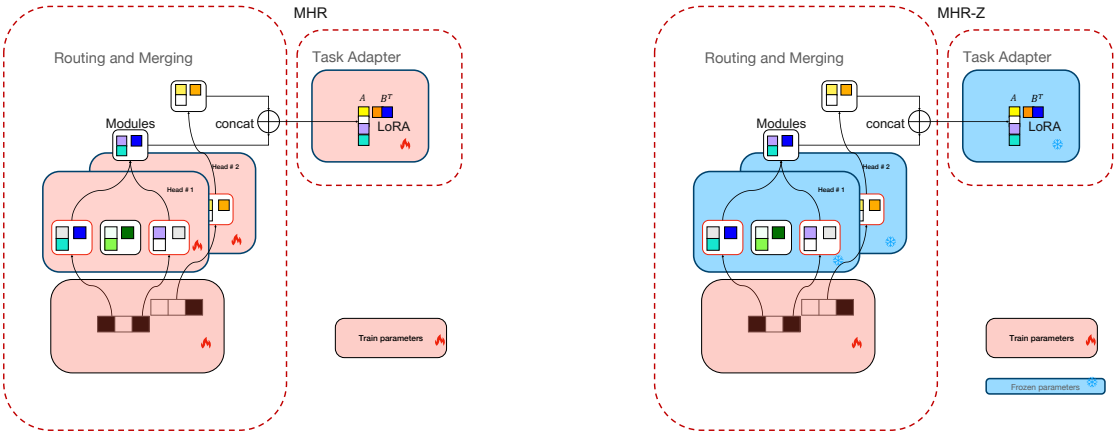


Figure 2.1 *Left:* A LoRA adapter with weight $AB^\top$ is trained on top of a frozen, pre-trained linear layer W. Our method `MHR` partitions the A, B parameter indexes into h blocks (or *heads*). For each block, a separate routing function selects the active modules for the current task among m copies with different parameter values, and combines them via averaging to form a task-specific head. The heads are then concatenated to form the LoRA adapter. Using multiple heads allows for more fine-grained mixing of task parameters with a negligible increase in overall parameter count. *Right:* During few-shot adaptation, one can fine-tune only the multi-head routing parameters (`MHR-`z), keeping the modules frozen, resulting in highly parameter-efficient adaptation.

Routing-Only Fine-Tuning (`MHR-`z) Prior work Shao *et al.* (2023, *inter alia*) has shown that compositional generalization can be achieved by learning to (re-)combine in novel ways pre-existing modules. We investigate whether fine-tuning the module parameters is really needed for few-shot adaptation in the context of both `Poly` and `MHR`. Therefore, we name `MHR-`z and `MHR-`z the variants that, during few-shot adaptation, keep the parameters of the modules learned during multi-task pre-training fixed and just update the routing parameters Z. Crucially, this enables highly parameter-efficient adaptation: for `LoRA` adapters, A and B matrices constitute the overwhelming majority of parameters. Therefore, by freezing the A, B matrices and only updating Z, we can significantly reduce the parameter cost when transferring knowledge to a new task.

Adapter Average Fine-Tuning (`MHR-`μ) To assess the importance of the routing parameters during few-shot adaptation, we propose an additional variant of `MHR`, `MHR-`μ, which shares the same multi-task pre-training procedure as `MHR`, but for each test task τ, fixes $z_\tau = (1/|\mathcal{M}|, \ldots, 1/|\mathcal{M}|)$ during few-shot adaptation. This is equivalent to discarding the routing parameters and averaging the module parameters into a single one before fine-tuning. Specifically, the adapter used during fine-tuning is initialized with (similarly for B^τ):

$$A^\tau = \frac{1}{|\mathcal{M}|} \sum_i A_i^*; \; A^\tau \in \mathbb{R}^{d \times r} \qquad (\texttt{MHR-}\mu)$$



(a) Finetuning process of `MHR`.

(b) Finetuning process of `MHR-z`.

Figure 2.2 In the finetuning process of `MHR`, there is no specific task id information. So the routing function become a matrix $Z \in \mathbb{R}^{|\mathcal{M}| \times h}$. `MHR-`$z$ keep the parameters of the modules learned during multi-task pre-training fixed and just updates the routing parameters Z.

where A_i^* are the parameters of the adapters after `MHR` multi-task pre-training. Note that, differently from `MHR`, `MHR-`μfine-tunes the same amount of parameters as the single adapter baseline. Thus, any difference in performance between the single adapter baseline and `MHR-`μcomes from differences in the adapter initialization and must be due to the optimization process taking place in the multi-task pre-training, before few-shot adaptation.

Routing Granularity In the original `Poly`, (Ponti *et al.*, 2023a) showed that learning a routing matrix Z for each model layer gave better performance than sharing a single Z matrix across all layers. We therefore investigate whether this holds true also for its multi-head counterpart, `MHR`. In addition, we explore intermediate approaches between one Z per layer and a single one shared for the entire model. In particular, we consider sharing Z 1) for the adapter layers belonging to the same Transformer block; or 2) for every block of l layers, which enables us to easily trade off expressivity for parameter efficiency. As we will demonstrate in section 2.6.1, this is an efficient mechanism to navigate this Pareto front in regimes of very small budgets of parameters per task.

2.5 Experiments

Our experimental evaluation aims to answer three research questions: 1) Does the expressivity of the routing function matter? 2) Why do routing-based PEFT methods yield superior performance? 3) Is routing useful during both multi-task pre-training and few-shot adaptation? We first present the baselines and datasets used in our evaluation and then discuss each question in turn.[1]

[1]We note that all experiments were run on a single NVIDIA A100 GPU.

T0 Dataset	Avg. Test
Backbone T5-XL-LM	
(IA)^3	$62.4_{0.4}$
AdapterSoup	$62.1_{1.0}$
LoRA	$66.0_{1.6}$
LoRA-big	$65.4_{0.9}$
Poly-z	$66.4_{0.3}$
Poly	$68.0_{1.0}$
MHR-z	$68.3_{0.8}$
MHR	$\underline{69.1}_{1.0}$
Backbone T0-3B	
Tfew Liu, Tam, *et al.*, 2022b	$66.2_{0.5}$
AdapterSoup	$66.1_{0.6}$
LoRA	$67.4_{0.8}$
Poly-z	$65.3_{1.0}$
Poly	$69.0_{0.8}$
MHR-z	$68.4_{1.2}$
MHR	$\underline{69.3}_{1.2}$

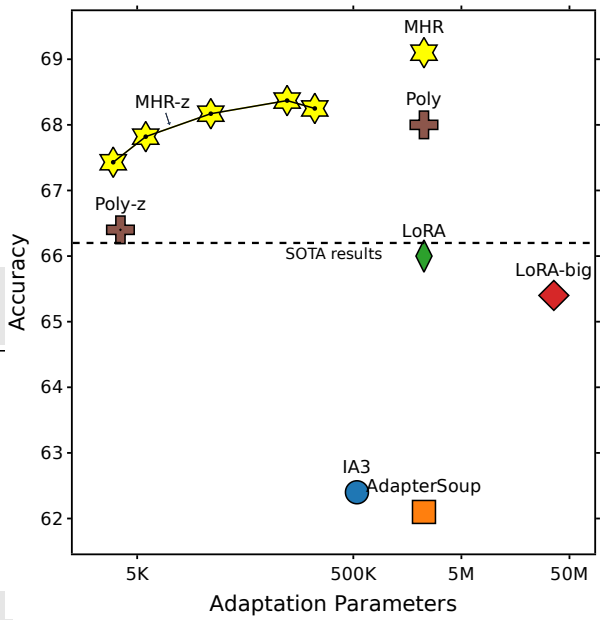


Figure 2.3 *Left:* Results of few-shot adaptation on T0 dataset (Sanh *et al.*, 2022). We report the mean of the best validation accuracy for each test task. Subscripts correspond to standard deviation. *Right:* Accuracy of PEFT methods on the T0 dataset when applied on top of T5-XL. The x-axis shows the parameter count during the fine-tuning process.

2.5.1 Baselines

In addition to Poly, we compare MHR to the following baselines for task-level generalization.

LoRA/(IA)^3 trains a single adapter common to all pre-training tasks, which is then fine-tuned on each test task separately. This is arguably the most widespread approach for parameter-efficient cross-task generalization (Liu, Tam, *et al.*, 2022a; Pfeiffer *et al.*, 2023).

AdapterSoup (Chronopoulou *et al.*, 2023) trains a different adapter for each task. The method only averages the adapter weights of the training tasks most similar to a given test task, before proceeding with few-shot adaptation. To compute task relatedness, we measure the cosine similarity of sentence embeddings for each task averaged over their training dataset. Notably, unlike the methods proposed in this paper, there is no knowledge sharing (nor interference) during multi-task pre-training as task adapters are trained independently.

2.5.2 Datasets

We test our methods on the T0 (Sanh *et al.*, 2022) evaluation suite, following the same setup as Liu, Tam, *et al.*, 2022a, and `SuperNI` (Wang, Mishra, *et al.*, 2022a), a large-scale dataset with more than 1,600 training tasks.

T0 Tasks We follow the pre-training and fine-tuning procedure discussed in (Liu, Tam, *et al.*, 2022a), using hyper-parameters and losses suggested in the public codebase for T-Few.[2]

All methods were tested with T5-XL (Raffel *et al.*, 2020b) and T0-3B (Sanh *et al.*, 2022) as the backbone model. Crucially, T5 is simply pre-trained on (masked) language modelling, whereas T0 is further instruction tuned: in particular, the full model is fine-tuned on examples from multiple training tasks that have been augmented with task instructions. To ensure fairness for all methods, we report the median and standard deviation of the best validation accuracy for each test task across 3 seeds, when evaluated every 50 training epochs. We treat each data subset–template pair as a unique task, yielding a total of 313 tasks.

SuperNI To limit computational costs, we report the result on 20 out of 119 test tasks. Tasks were chosen at random, with the requirement that at least 300 examples were available, and were equally split into 100 training, 100 validation and 100 test examples. For every method, we perform early stopping on the validation set. We report results with Rouge-L averaged across 3 seeds. All methods use T5-XL (Raffel *et al.*, 2020b) as the backbone and not T0, as T0 training tasks and SuperNI test tasks may overlap.

2.6 Results and Discussion

2.6.1 Does the expressivity of the routing function matter?

`MHR` outperforms PEFT approaches We start our analysis by evaluating the effectiveness of our proposed technique when applied over a backbone that has not undergone prior training on instruction-following data (T5-XL). As indicated in the T0 benchmark results in the top table of Fig. 2.3, it is clear that multi-head routing techniques have a distinct advantage, outperforming both single-head routing `Poly` by **1.1%**, and surpassing standard `LoRA` approaches by an impressive **3.1%**. We also study the impact of performing instruction tuning of the full backbone before adapter training. To this end, we also experiment with T0-3B as a backbone. In the bottom table of Fig. 2.3, we can observe that while the relative gap between `MHR` and baselines is smaller, multi-head routing still manages to yield favourable results. Hence, the gains of `MHR` compound with other multi-task methods such as instruction tuning. Finally, we turn our attention

[2] https://github.com/r-three/t-few

towards the SuperNI dataset (Tab. 2.2). Here, MHR continues to surpass analogous baselines.

MHR-z facilitates extreme parameter efficiency Fig. 2.3 (right) reveals intriguing findings regarding MHR-z. When we restrict training to only the routing parameters Z in the original Poly, the results are unfortunately not up to par with its version where both routing and adapters are updated. However, when we apply the same constraint to MHR, the performance is significantly closer to the optimum achieved in this setting. In fact, MHR-z surpasses prior baselines while simultaneously necessitating fewer parameters for effective adaptation to new tasks. Moreover, by controlling the number of layers which share the same Z allocation (see sec. 2.4), MHR-z is able to trade-off performance for parameter efficiency, even surpassing Poly-z in settings with only 3K trainable parameters per test task (see also § 2.8.2 for a more in-depth analysis). This trend is similarly observed in the SuperNI benchmark (Tab. 2.2), where updates restricted to the routing parameters yield performance on par with standard fine-tuning. We therefore conclude that the MHR-z represents a robust approach for achieving extreme parameter efficiency in adaptation.

Additional routing heads is more beneficial than extra modules In the original Poly approach, a tradeoff between capacity and parameter efficiency can be achieved by adding extra modules for each adapter layer. However, this results in a linear increase in the number of multi-task parameters, which can become impractical. To explore a more effective tradeoff, we investigate the option of adding additional routing heads instead of extra modules. Fig 2.4 (right) presents the comparison between the two approaches. It demonstrates that increasing the number of routing heads leads to better performance compared to adding more modules. Importantly, the benefit of multi-head routing is twofold: it provides increased expressivity for the model, while also maintaining parameter efficiency. This finding highlights the advantage of multi-head routing as a more effective approach for balancing expressivity and parameter count in few-shot adaptation scenarios.

SuperNI Dataset	Rouge-L
LoRA	$67.6_{0.8}$
LoRA-big	$67.2_{0.7}$
Poly-z	$64.6_{0.3}$
Poly	$67.8_{0.8}$
MHR-z	$68.0_{0.2}$
MHR	$\underline{68.5}_{0.3}$

Table 2.2 Results on **SuperNI** dataset. Subscripts are standard deviation.

Routing-based methods also excel at the 11B scale We proceed to evaluate if Poly and MHR surpass established PEFT approaches when trained over a larger model backbone. To accomplish this, we employ the 11B version of T0. As depicted in Tab. 2.3, routing-based methods once again outshine standard adapter training, surpassing our reproduction of the previous state-of-the-art in Liu, Tam, *et al.*, 2022a by over 2%. We observe that Poly and MHR

T0 Dataset	Avg. Test
Backbone T0-11B	
T-Few	$72.5_{0.9}$
LoRA	$72.3_{1.0}$
Poly-z	$70.0_{0.6}$
Poly-	$\underline{74.9}_{0.6}$
MHR-z	$72.9_{0.8}$
MHR	$74.7_{0.6}$

Table 2.3 Few-shot results over 11B parameter backbones.

show similar performance in standard fine-tuning, but MHR z-tuning remains more performant in routing-only fine-tuning. Indeed, MHR-z (221K params) outperforms Poly-z (3.5K params) by 2.9%, while still remaining more parameter efficient than Liu, Tam, *et al.* (2022a) (1.1M params).

2.6.2 Why do routing-based PEFT methods yield superior performance?

While our proposed methods have demonstrated promising results across a broad spectrum of datasets and varying adaptation parameter budgets, the question of *why* routing-based PEFT exhibits superior performance remains unanswered. In this section, we aim to uncover the key components that drive MHR's superior performance.

Learning the Routing Function is essential Given that Poly and MHR have access to more parameters than standard adapters during multi-task pretraining, we investigate whether this, and not the routing mechanism, is responsible for their superior performance. To do so, we compare them to a baseline approach. Instead of learning the routing function, we randomly assign a binary module allocation to each data point in a minibatch, disregarding task information. This random routing approach, akin to Wang, Mukherjee, *et al.*, 2022, allows us to directly assess the influence of additional parameters during multi-task training. At test time, the learned modules are averaged into a single one before fine-tuning; we therefore refer to this baseline as Random-μ.

MHR fosters transfer and mitigates interference across pretraining tasks Recognizing the pivotal role of the multi-task pretraining step in bolstering Poly's performance, we explore the extent of transfer and interference across training tasks. By monitoring the average gradient alignment for each task pair (in terms of cosine similarity) throughout the training process, we are able to gauge the level of positive transfer. As Fig. 2.4 (left) shows, MHR displays a greater degree of gradient cosine similarity across tasks compared to other PEFT alternatives, including Poly. This finding suggests that the enhanced flexibility offered by multi-head routing may serve to mitigate interference across tasks to a larger extent than standard routing while simultaneously promoting positive transfer.

2.6.3 Is routing important for task generalization?

We assessed the importance of routing during pre-training. We now proceed to verify whether it is important to learn routing during few-shot adaptation, too. Polym and MHRm consistently outperform LoRA, and match the performance of Poly / MHR (Tab. 2.4). This demonstrates that, for few-shot adaptation, the average of the pre-trained modules provides a better initialization than learning an adapter shared across all the tasks during pre-training. The consistently superior performance of Polym with respect to Random-μ and AdapterSoup stresses the importance of routing during multi-task pre-training (but not during adaptation), which provides an effective adapter initialization for few-shot

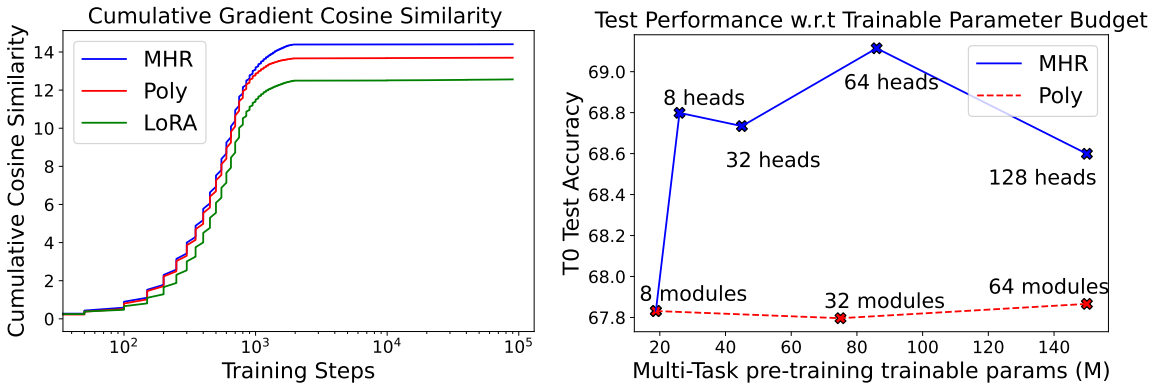


Figure 2.4 *Left:* Gradient alignment between tasks during multi-task pretraining. *Right:* Increasing the number of heads offers better scaling properties than increasing the number of modules.

learning. This finding could potentially inspire future work for improving meta-learning and weight-averaging approaches (Izmailov *et al.*, 2018).

MHR-μ excels at zero-shot learning For many downstream tasks of interest, additional labelled data may not be available. In such settings, it is unclear how to leverage MHR-μ and Poly-μ methods. To address this, we fine-tune the average of the multi-task trained adapters on the multi-task pre-training data (instead of using the downstream few-shot data), for an additional k steps. The results are presented in Table 2.5. We find that without any additional fine-tuning ($k = 0$), averaging the adapters does not yield good results. This is due to a potential mismatch between adapters learned via task-specific routing, and the uniform routing strategy. We can observe that when fine-tuning the average of the adapters on the multi-task pre-training data for an additional k steps, MHR-μ show strong performance when evaluated in a zero-shot manner. For a fair comparison, we also additionally fine-tune LoRA for the same number of additional steps. Our best model achieves a zero-shot performance of 64.5 on top of T0-11B, achieving an absolute gain of 3.5% accuracy points.

T0 Dataset	Test Acc.
LoRA	$66.0_{1.6}$
AdapterSoup	$62.1_{1.0}$
Poly	$68.0_{0.8}$
Poly-μ	$67.8_{0.6}$
MHR	$69.1_{1.1}$
MHR-μ	$69.1_{0.9}$
SuperNI	**Rouge-L**
LoRA	$67.6_{0.8}$
Poly	$67.8_{0.8}$
Poly-μ	$68.3_{0.5}$
MHR	$68.5_{0.6}$
MHR-μ	$68.5_{0.8}$

Table 2.4 Evaluating the impact of modular adaptation at test time.

2.7 Conclusions

In this paper, we tackle the challenge of generalizing to new tasks based on a few examples after multi-task pre-training. Specifically, we focus on Polytropon (Ponti *et al.*, 2023a), a model where each task is associated with a subset of adapters by a routing

Method	Zero-Shot Test with k-shot Extra Training			
	$k = 0$	$k = 1000$	$k = 5000$	$k = 10000$
Backbone T5-XL-LM	43.2			
`LoRA`	56.5	56.0	56.1	55.7
`Poly`-μ	46.0	53.0	56.8	56.3
`MHR`-μ	48.0	<u>58.0</u>	57.1	56.3
Backbone T0-11B (Sanh *et al.*, 2022)	61.0			
`LoRA`	61.2	61.6	61.5	61.5
`Poly`-μ	62.1	63.6	63.9	64.4
`MHR`-μ	63.5	<u>64.5</u>	64.5	64.4

Table 2.5 Zero-shot performance for MHR and the baselines, reported as the average over the 11 evaluation datasets from Sanh *et al.*, 2022. To obtain these zero-shot results, we average the learnt `Poly`/`MHR` adapters before performing k additional fine-tuning steps on the multi-task pretraining data. This effectively enables zero-shot transfer to downstream tasks using the same amount of parameters/flops as the baseline LoRA. `MHR` outperform baseline LoRA by up to 3% absolute accuracy points on T0-11B.

function. We investigate how varying the level of control afforded by the routing function impacts performance on two comprehensive benchmarks for multi-task learning, T0 and Super-Natural Instructions. First, a newly proposed variant of the routing function, where multiple heads are responsible for different blocks of input dimensions, improves consistently over all other baselines, including LoRA and $(IA)^3$ adapters. Second, we identify the cause of the success of routing in its ability to prevent interference between tasks, as it yields a better alignment between their gradients. Third, we find that simple averaging of all multi-task pre-trained adapters before few-shot adaptation to new tasks provides comparable performance, thus offering state-of-the-art performance for single-adapter few-shot learning. Multi-head routing demonstrates the importance of fine-grained adapter selection for sample-efficient generalization and holds promise to improve other modular methods, such as Mixtures of Experts (MoEs; Fedus, Zoph, *et al.*, 2022a) in future research.

2.8 Appendix

2.8.1 Additional Results

More detailed numbers on the T0 Sanh *et al.*, 2022 and SuperNI Wang, Mishra, *et al.*, 2022a datasets using different backbones, and different adapter layouts over the base model are found in Table 2.6. `Multi-Task params` is the number of additional parameters that must be conserved after multi-task pretraining to enable transfer to a downstream task. `Adaptation Params` refers to the number of parameters required to learn a new downstream task. For e.g. `Poly` and `MHR`, the multi-task parameters include the learned modules, but not the routing over the training tasks, as these are not required for transfer on a new task. Moreover, variants that average the learned modules prior to fine-tuning (`MHR`-μ and `Poly`-μ) will have both multi-task and adaptation parameters equal to that of a single shared adapter, since after multi-task pretraining one can average the modules.

2.8.2 Navigating the parameter efficiency / performance trade-off of tuning only the routing

Here we provide additional results on how different routing based methods can be more expressive when only learning a new routing function (over *frozen* modules) to adapt to a new task.

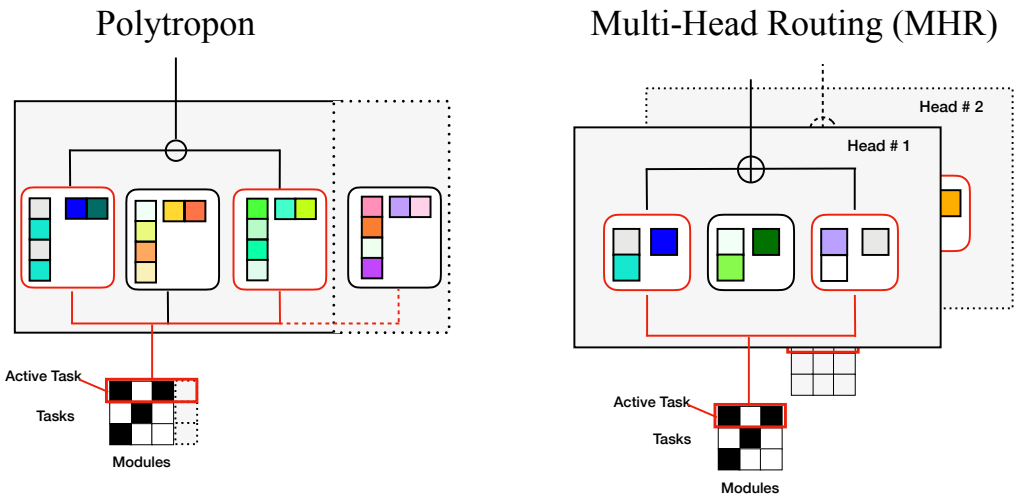


Figure 2.5 Different ways to control the expressivity of routing based methods. *Left :* In Polytropon, one can only add additional modules, resulting in a linear parameter increase. *Right :* In `MHR`, additional heads only introduce routing matrices Z, resulting in a negligible parameter increase.

In Fig. 2.5 (left), we see that in order to build more expressive routing functions Z, in `Poly` one can only do so by increasing the number of skills at each layer. However, this

has a significant impact on the number of multi-task parameters which much be kept in order to perform few-shot transfer. MHR on the other hand, can increase routing capacity in a much more parameter efficient way.

On the granularity of routing tensor in MHR

Here we provide additional results when modifying the granularity of Z for MHR. We see that one can easily trade-off more parameters for better performance.

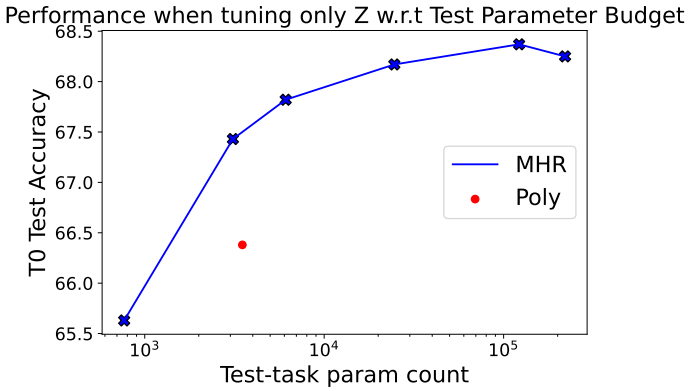


Figure 2.6 Routing-Only Fine-Tuning (MHR-z)

2.9 Broader Impact

In our work, we focus on advancing parameter-efficient fine-tuning methods for cross-task generalization. While our research primarily addresses technical challenges and performance improvements, when applying such methods, it is crucial to consider the potential negative societal impacts. Specifically, we believe that prior to applying our proposed adaptation method, critically examining the potential biases and ethical implications of the underlying large language model, and the data itself must be properly addressed. This includes issues related to fairness, privacy, and the spread of misinformation.

Model	Multi-Task Params	Adaptation Params	Avg. Test
T0 Dataset			
Backbone T5-XL-LM			
`Multi-Task Full Finetuning + LoRA`	2.8B	2.2M	$68.9_{x.x}$
`(IA)`^{3}	540K	540K	$62.4_{0.4}$
`AdapterSoup`	84M	2.2M	$62.1_{1.0}$
`LoRA`	2.2M	2.2M	$66.0_{1.6}$
`LoRA-big`	35M	35M	$65.4_{0.9}$
`Poly-`*z*	17M	3.5K	$66.4_{0.3}$
`Poly`	17M	2.2M	$68.0_{1.0}$
`MHR-`*z (64 h)*	17M	220K	$68.3_{0.8}$
`MHR` *(64 h)*	17M	2.2M	$\underline{69.1}_{1.0}$
Backbone T0-3B			
`T-Few` Liu, Tam, *et al.*, 2022a	540K	540K	$66.2_{0.5}$
`AdapterSoup`	84M	2.2M	$66.1_{0.6}$
`LoRA`	2.2M	2.2M	$67.4_{0.8}$
`LoRA-big`	35M	35M	$68.0_{0.8}$
`Poly-`*z*	17M	3.5K	$65.3_{1.0}$
`Poly`	17M	2.2M	$69.0_{0.8}$
`MHR`*z (64 h)*	17M	220K	$68.4_{1.2}$
`MHR` *(8 h)*	17M	2.2M	$\underline{69.3}_{1.2}$
Backbone T0-3B light version : (`k, v, ff` layers only)			
l-`LoRA` *(rank 1)*	934K	934K	$66.2_{0.9}$
l-`LoRA` *(rank 16)*	15M	15M	$67.6_{1.1}$
`AdapterSoup` (*l*-`LoRA`)	35M	934K	$64.9_{1.0}$
l-`Poly-`*z*	7.5M	2.1K	$62.9_{1.2}$
l-`Poly`	7.5M	934K	$68.0_{0.5}$
l-`MHR`*z (32 h)*	7.5M	74K	$66.8_{1.1}$
l-`MHR` *(8 h)*	7.5M	934K	$\underline{68.5}_{0.7}$
SuperNI Dataset			Rouge-L
Backbone T5-XL-LM light version : (`k, v, ff` layers only)			
l-`LoRA`	934K	934K	$67.6_{0.8}$
l-`LoRA-big`	18M	18M	$67.2_{0.7}$
l-`Poly-`*z*	7.5M	2.1K	$64.6_{0.3}$
l-`Poly`	7.5M	934K	$67.8_{0.8}$
l-`MHR`*z (64 h)*	7.5M	147K	$68.0_{0.2}$
l-`MHR` *(8 h)*	7.5M	934K	$\underline{68.5}_{0.3}$

Table 2.6 (top) Results on T0 dataset Sanh *et al.*, 2022, we report the mean of the best validation accuracy for each test task, when evaluated every 50 train epochs. `T-Few` is our reproduction of the results in Liu, Tam, *et al.*, 2022a. `LoRA-big` means a `LoRA` adapter with a larger rank. (bottom) Results on SuperNatural Instructions dataset.

3 Mixture of LoRA Experts Using Tensor Product

3.1 Introduction

Recently, the de facto paradigm for natural language understanding (NLU) tasks has centered on leveraging large language models (He *et al.*, 2021). These models are pre-trained on a vast corpus of unlabelled data and subsequently fine-tuned for specific tasks (Qiu *et al.*, 2020; Ye *et al.*, 2021b). While this approach has significantly advanced the field, it often requires substantial computational resources and may not efficiently transfer knowledge across diverse tasks. In addition, training tasks independently can lead to *negative transfer*, where the lack of shared information across tasks, fails to achieve compositional generalization (Ponti *et al.*, 2023a; Caccia *et al.*, 2022).

To address the aforementioned issues, there are two lines of research. To mitigate the computation and memory issue, several lightweight alternatives known as parameter-efficient finetuning (PEFT) have been proposed to update only a small number of extra parameters while keeping most pre-trained (Houlsby *et al.*, 2019b; Li and Liang, 2021b; Hu *et al.*, 2021). However, these solutions need to train each adapter for each task, which does not take into account that test tasks may require solving different combinations of sub-problems compared to training tasks (Vu *et al.*, 2020b), thus failing to achieve compositional generalization (Rosenbaum *et al.*, 2019; Ponti, 2021). Moreover, specializing the model toward different tasks during training can yield gradients that are poorly aligned, resulting in negative transfer (Wang *et al.*, 2020; Lialin, Deshpande, *et al.*, 2023).

To facilitate information sharing across multiple tasks, two primary approaches are employed: : *sequentially finetuning* and *multi-task learning*. Sequentially finetuning involves finetuning a pre-trained language model on one task after another in a specific order (Phang *et al.*, 2018). However, this process will suffer from *catastrophic forgetting* issue (McCloskey and Cohen, 1989; French, 1999), where language models will lose the knowledge trained by the previous task when training the new task. Multi-task learning (MTL) (Caruana, 1997; Zhang and Yang, 2021; Liu *et al.*, 2019) simultaneously trains the model on several tasks, allowing it to learn shared representations that benefit all tasks involved. However, MTL necessitates access to all training tasks during the training phase, meaning that incorporating new tasks requires retraining the model from scratch. This requirement significantly increases the computational burden and limits the flexibility of the model to adapt to new tasks efficiently.

A promising approach to address the above issues is the adoption of modular deep learning (Pfeiffer *et al.*, 2023). In this framework, computational units are typically implemented as parameter-efficient modules. Information flow is conditionally routed to a subset of these modules, where it is then aggregated. This design facilitates the positive transfer and systematic generalization (Pfeiffer *et al.*, 2023). Recently, Polytropon(Poly) was proposed as a novel framework designed to tackle diverse tasks by leveraging different combinations of latent modular skills (Ponti *et al.*, 2023a). This approach allows for the recombination of previously learned skills to solve test tasks more effectively, showcasing a significant advancement in task adaptability and learning efficiency. During both the multi-task pre-trained and few-shot finetuning, `Poly` concurrently optimizes the parameters of the skill inventory and a binary task-skill routing matrix. This matrix plays a critical role in determining which skills are activated for each training task.

In this paper, we propose a variant Poly model: `TensorPoly` using tensor product operation (Smolensky, 1990). A tensor product is an operation that takes two or more tensors and combines them to produce a new tensor. This process enables the capturing of higher-order interactions and structural relationships between the input tensors (Panahi *et al.*, 2019; Kye, 2023; Gan *et al.*, 2022). As depicted in Figure 3.1, we have reparameterized LoRA (Hu *et al.*, 2021) adapters by employing an "entangled" tensor structure. Consequently, the traditional training matrix $M \in \mathbb{R}^{d \times r}$ in LoRA is reparamerized into a more finer-grained tensor $\mathcal{L} \in \mathbb{R}^{N \times r \times \lceil \sqrt[N]{d} \rceil \times R)}$, named as TLoRA. This reparameterization allows for a more nuanced manipulation of the model's parameters, facilitating a more precise and efficient adapter process. The entangled tensor configuration introduces two critical hyper-parameters: the tensor rank (R) and the tensor order (N). Leveraging these parameters, we have developed two distinct routing functions designed to select modules with varying levels of granularity. As depicted in Figure 3.1, `TensorPoly-I` employs a routing mechanism that assigns distribution scores to different tensor ranks, facilitating the selection of modules based on their rank granularity. Further advancing this concept, we propose a more refined routing function, `TensorPoly-II`, which targets even finer-grained tensors as activated modules. Each module is associated with a specific order of the entangled tensor. Once modules are selected via the routing function, they are aggregated through a tensor product operation, enabling a sophisticated and dynamic assembly of modular skills.

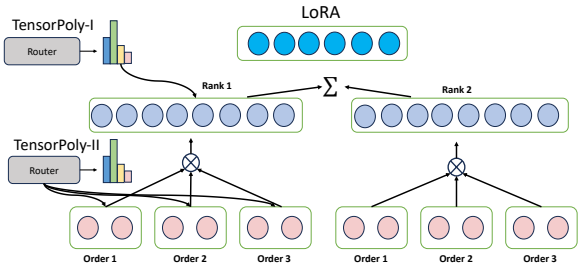


Figure 3.1 LoRA reparamarized by an entangled tensor with a rank = 2 and order = 3. Different granularity of modules induces two routing approaches, `TensorPoly-I` will select the rank of the tensor while `TensorPoly-II` select the finer-grained order of the entangled tensor.

We evaluate our methods against a series of competitive baselines for few-shot task adaptations benchmark (Victor *et al.*, 2022). The experimental outcomes reveal several key insights: 1) modular language models, specifically `Poly` and `TensorPoly` frameworks, consistently outperform traditional PEFT approaches, such as LoRA and its tensorized

variant TLoRA. This superiority underscores the effectiveness of our routing function in facilitating positive transfer across multi-task environments, as opposed to the more conventional dense model approaches. 2) `TensorPoly-I` demonstrates competitive results as `Poly`, while simultaneously reducing the number of training and adaptation parameters required. This efficiency gain highlights the benefits of our tensorized module approach in achieving high performance with lower parameter overhead. 3) A comparative analysis between `TensorPoly-I` and `TensorPoly-II` indicates that the latter's finer-grained routing mechanism does not contribute to improved performance in tensor product routing scenarios. This outcome suggests that while granularity in module selection is valuable (Caccia *et al.*, 2022), there is a complexity threshold beyond which additional granularity may not yield further benefits.

In summary, our contributions are as follows:

- We propose a novel modular language model `TensorPoly` using tensorized modules TLoRA, designed to enhance positive transfer and generalization across multi-task scenarios.
- TLoRA achieves competitive, and in some cases superior, results while utilizing only a 60% training parameters required by LoRA, highlighting our approach's high parameter efficiency.
- The findings from our T0 benchmark demonstrate that our modular language model significantly enhances the performance of a single module, underscoring the critical role of routing in scenarios involving multiple tasks.

3.2 Related Work

PEFT

Parameter efficient fine-tuning (PEFT) methods facilitate efficient adaptation of LLMs without updating all the training parameters, thereby reducing the memory and computation cost (Zhang, Han, *et al.*, 2023). One kind of PEFT approach focuses on adding some *modules* to LLMs and only these small *modules* will be trained, the backbone model is kept frozen and shared across tasks. For example, adapter tuning inserts small neural modules (adapters) between the layers of the basic model (Houlsby *et al.*, 2019a). Prefix tuning and Prompt tuning add some tunable vectors to the input or hidden layer of the base model (Li and Liang, 2021b; Lester *et al.*, 2021). These models can achieve comparable performance to full-finetuning while only using less than 1% training parameters. Another kind of research model is the incremental update of the pre-trained weights in a parameter-efficient way, without modifying the model architecture. Bit kit fixed all training parameters and only finetuned the additive bias term (Zaken *et al.*, 2021). Diff Pruning learns a task-specific "diff" vector that extends the original pre-trained parameters. As the number of tasks increases, diff pruning only requires storing only a small diff vector for each task (Guo *et al.*, 2020). Hu *et al.*

(2021) propose a method named LoRA, which parameterizes incremental weights Δ as a low-rank matrix by the product of the down projector matrix and up projector matrix. LoRA achieves comparable or even better performance than full fine-tuning (Hu *et al.*, 2021). Zhang, Chen, Bukharin, *et al.* (2023) demonstrates that weight matrices in the top layers are more important than those in the bottom layers. They propose a new method: Adaptive Low Rank Adaptation (AdaLoRA) which dynamically allocates the parameter budget among weight matrices during LoRA-like finetuning (Zhang, Chen, Bukharin, *et al.*, 2023). AdaLoRA adjusts the rank of incremental matrices for different layers. Critical matrices are assigned with high rank so that they can capture more task-specific information. Less important ones are pruned to have lower rank. In our approach, TLoRA reparameterize LoRA with tensor product. It utilizes the finer-grained tensors as modules.

MTL

To share the information across multiple tasks. AdapterSoup Chronopoulou *et al.* (2023) trains each adapter for each domain, then it performs weight-space averaging of adapters trained on different domains. (Huang *et al.*, 2023) introduce LoRAhub to aggregate the LoRA modules trained on diverse tasks. They first train a group of LoRA modules which are specialized in each task. Then they randomly select a subset of modules. They learn a set of weights to combine these LoRA models using gradient-free optimization. AdapterFusion (Pfeiffer, Kamath, *et al.*, 2020) proposes a two-stage algorithm that leverages knowledge from multiple tasks. The same as LoRAhub, they learn a group of task-specific adapters to encapsulate the task-specific information. In the second stage, they learn a fusion layer to combine the trained adapters. Ponti *et al.* (2023a) introduce a variable-size module routing mechanism, Poly, predicated on the notion that each task correlates with a specific subset of latent skills drawn from a comprehensive inventory of modules. They concurrently train both the modules and the routing function, facilitating a more dynamic and task-specific approach to model learning. Building upon Poly, (Caccia *et al.*, 2022) introduce a finer-grained multi-head routing function MHR, the experimental findings underscore the significance of the routing function during the pre-training phase. Remarkably, by solely fine-tuning the routing function, MHR attains competitive results, demonstrating exceptional parameter efficiency.

Model Merging Model merging can be seen as a parameter level aggregation, which is defined as combining multiple models into one single model in parameter space without access to data (Matena and Raffel, 2022). The simplest operation of merging is averaging the weights of different models. Wortsman *et al.* (2022) show that averaging the weights of multiple models finetuned with different hyperparameter configurations often improves accuracy and robustness. Matena and Raffel (2022) take the *aggregation* function can be seen as maximizing the joint likelihood of the posteriors of the model's parameters and propose a "Fisher merging" technique. Yadav *et al.* (2023) propose a TIES-Merging method to address interference due to redundant parameter values and disagreement on the sign of a given parameter's values across models. Ilharco *et al.* (2022) edits models with task arithmetic operations. (Jin *et al.*, 2022). Jin *et al.* (2022) propose a dataless knowledge fusion method (Regression Mean) that merges models in their parameter space.

3.3 Background

In scenarios involving multiple tasks, we define a set of tasks as $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_{|\mathcal{T}|}\}$. This set is divided into two subsets train $\mathcal{T}_{train}$ and test $\mathcal{T}_{test}$. The goal of multi-task transfer learning is to apply the knowledge from the training tasks $\mathcal{T}_{train}$ to the test tasks within $\mathcal{T}_{test}$. This process involves two main phases, The first phase consists of multi-task pre-training using the dataset in tasks $\mathcal{T}_{train}$. The second consists of a few-shot adaptation, where the learned adapters are fine-tuned independently on each test task in $\mathcal{T}_{test}$. We follow the procedure from (Raffel *et al.*, 2020b) and formulate each task as a text-to-text problem, enabling standard maximum-likelihood training with teacher forcing (Bengio, Vinyals, *et al.*, 2015) and a cross-entropy loss.

3.3.1 Module: LoRA

LoRA are recently proposed adapter architecture that achieves a competitive balance between performance and parameter efficiency (Hu *et al.*, 2021; Mahabadi *et al.*, 2021). For each linear transformation corresponding to the query (q), key (k), value (v), and output (o) of the self-attention layers, LoRA modifies the base model *parameters* as follows:

$$h^{q,k,v,o} = W_0^{q,k,v,o} x + s \cdot A^{q,k,v,o} (B^{q,k,v,o})^\top x, \tag{3.1}$$

where W_0 are the (frozen) weights of the pre-trained model (e.g. T5 (Raffel *et al.*, 2020b)). $A, B \in \mathbb{R}^{d \times r}$ are low-rank learnable parameters and $s \geq 1$ is a tunable scalar hyperparameter. We show the LoRA in Figure 3.2.

3.3.2 Tensor, Tensor Product, Entangled Tensor

Tensor. The tensor $\mathcal{A}$ is a multidimensional array of elements (called components) of $\mathbb{R}$, each denoted by its integer coordinates in the array; e.g., for a two-dimensional array, the component at position $i, j \in \mathbb{N}$ is denoted $A_{i,j}$. The *order* of a tensor is how many indices it has (e.g., a vector v is a first-order tensor, a matrix M is a second-order tensor, etc.)

Tensor Product. The tensor product $V \otimes W$ of two vector spaces $\mathcal{V}$ and $\mathcal{W}$ is a vector space to which is associated a bilinear map $V \times W \to V \otimes W$ that maps a pair (v, w), $v \in V, w \in W$ to an element of $V \otimes W$ denoted $v \otimes w$. We can create tensor product spaces by more than one application of the tensor product, $\mathcal{H} = \mathcal{U} \otimes \mathcal{V} \otimes \mathcal{W}$, with arbitrary bracketing, since the tensor product is associative. The tensor product space of the form is said to have tensor *order* of n.

$$\bigotimes_{j=1}^{n} \mathcal{H}_j = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes ... \otimes \mathcal{H}_n \tag{3.2}$$

Entangled Tensor. The n-order tensor product space $\otimes_{j=1}^{n}\mathcal{H}_j$ consists of vectors of the form $v = \otimes_{j=1}^{n} v_j$, where $v_j \in \mathcal{H}_j$, are called *simple tensor*. Vectors need to be represented as the sum of multiple simple tensors called *entangled tensors*:

$$\sum_{k=1}^{r}\bigotimes_{j=1}^{n}\mathcal{H}_j = \sum_{k=1}^{r}\mathcal{H}_1 \otimes \mathcal{H}_2 \otimes ... \otimes \mathcal{H}_n \tag{3.3}$$

where tensor rank r is the smallest number of simple tensors that sum up to v. For example, $\frac{\psi_0\otimes\phi_0+\psi_1\otimes\phi_1}{\sqrt{2}}$ is a tensor of rank 2.

3.3.3 Tensorized Training Parameters with Tensor Product

Any training parameters $v \in \mathbb{R}^d$ can be expressed as an entangled tensor of rank r and order n by:

$$v = \sum_{k=1}^{r}\bigotimes_{j=1}^{n} v_{jk}, \tag{3.4}$$

Here, $v_{jk} \in \mathbb{R}^q$, yielding a resultant vector v of dimension $p = q^n$. Its storage requirements are efficiently managed, consuming only $rnq = O(rq \log p/q)$. If $q^n > d$, the excess part of the generated vector will be cut off. The number of vector parameters can be reduced from d to $rn\sqrt{d}$. For example, when $d = 512$, $q = 8$, $n = 3$, and $r = 2$, the number of parameters of a vector can be reduced from 512 to 48.

3.4 Methods: TensorPoly

In section 2.3.2, we introduced the `Poly` model. This section delves into two variant models, `TensorPoly-I` and `TensorPoly-II`. `TensorPoly-I` employs a routing mechanism that assigns distribution scores to tensor ranks while `TensorPoly-II` assigns distribution scores to the finer-grained tensor order.

3.4.1 TensorPoly-I

As illustrated in Figure 3.2, the `TensorPoly` model innovatively incorporates the Tensorized Low-Rank Adaptation (TLoRA) as its core module. This approach adopts finer-grained tensors for training parameters, thereby enabling a more nuanced parameter manipulation. The routing matrix $Z \in \mathbb{R}^{|\mathcal{T}|\times R}$ plays a pivotal role in this framework, determining which rank within the entangled tensor is to be activated for a given task. Upon activation of the selected rank, the model performs a linear combination of the selected tensor, weighted by a factor α:

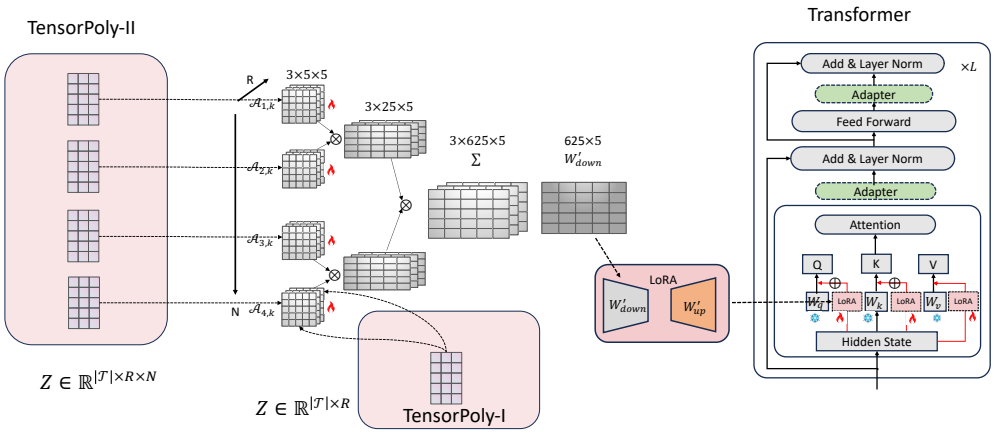


Figure 3.2 `TensorPoly-I` and `TensorPoly-II`. We illustrate how to reparametrized the LoRA matrix $\mathbb{R}^{625\times5}$ with 4 tensor $\mathcal{A} \in \mathbb{R}^{3\times5\times5}$. In this case, the tensor rank $R = 3$, tensor order $N = 4$. For `TensorPoly-I`, the routing function Z is designed to select which rank of the entangled tensor is activated for a given task. Conversely, `TensorPoly-II` introduces a more granular control by selecting tensor rank and tensor order. The Blue color indicates a non-trainable component and the reddish color indicates a trainable component.

$$A^{\tau} = \sum_{i} \alpha_i A^{(i)};\ B^{\tau} = \sum_{i} \alpha_i B^{(i)} \tag{3.5}$$

$$A^{\tau} = \sum_{k=1}^{R} \alpha \bigotimes_{i=1}^{N} \mathcal{A}_{i,k};\ B^{\tau} = \sum_{k=1}^{R} \alpha \bigotimes_{i=1}^{N} \mathcal{B}_{i,k} \tag{3.6}$$

where R is the rank of the entangled tensor. N is the order of the tensors. $\alpha = \frac{\hat{z}_{\tau,i}}{\sum_j \hat{z}_{\tau,j}}$. $\mathcal{A}_{i,k} \in \mathbb{R}^{N\times r\times \lceil \sqrt[N]{d} \rceil}$ is a third-order tensor.

3.4.2 TensorPoly-II

In `TensorPoly-II`, we implement a fine-grained routing function that selects both the tensor rank and tensor order, enabling precise control over the model's adaptation to specific tasks. This sophisticated routing mechanism facilitates the selection of finer-grained tensor elements, which are then aggregated through a tensor product operation followed by a linear combination. The routing itself is conceptualized as a third-order tensor $Z \in \mathbb{R}^{|\mathcal{T}|\times R\times N}$, which offers an unprecedented level of granularity in directing the model's focus across different ranks and orders of the tensor space.

$$A^{\tau} = \sum_{k=1}^{R} \bigotimes_{i=1}^{N} \alpha\mathcal{A}_{i,k};\ B^{\tau} = \sum_{k=1}^{R} \bigotimes_{i=1}^{N} \alpha\mathcal{B}_{i,k} \tag{3.7}$$

3.5 Experiments

Model	Natural Language Inference					Sentence Completion			Co-reference		WSD	ACC	Param
	RTE	CB	ANLI1	ANLI2	ANLI3	COPA	H-SWAG	Story	WSC	Wino	WiC		
Baselines (w/o pretrain)													
FullFT	79.8	87.5	46.6	41.3	40.0	81.0	46.4	93.8	65.4	56.5	57.7	63.3	3B
BitFit (with LN)	72.2	57.1	36.5	35.3	36.6	75.0	29.5	88.6	61.5	56.6	51.7	54.6	1.3M
LayerNorm	71.8	57.1	36.5	35.1	36.3	76.0	29.6	88.7	63.5	49.4	52.2	54.2	250K
Adapter	76.2	87.5	45.1	40.4	35.3	84.0	41.9	91.7	65.4	54.7	55.5	61.6	12.9M
Compacter	75.8	82.1	40.8	37.4	35.8	84.0	46.4	93.5	64.4	55.5	55.2	61.0	807K
Compacter++	76.9	82.1	41.7	38.3	36.9	86.0	46.3	93.5	65.4	55.1	54.1	61.5	540K
Prompt(10)	52.7	66.1	34.2	33.5	33.5	67.0	29.9	84.2	54.8	51.9	51.6	50.9	41K
Prompt(100)	48.0	53.6	33.4	33.8	33.3	60.0	26.8	74.0	60.6	51.1	50.0	47.7	409K
Prefix tuning	68.6	84.0	43.3	37.5	36.5	71.0	42.1	90.2	56.7	52.0	54.2	57.8	576K
FishMask (0.2%)	76.9	83.9	43.7	39.7	37.2	82.0	44.1	94.2	63.5	54.5	52.5	61.1	6M
FishMask (0.02%)	75.5	76.8	39.9	38.1	36.2	84.0	38.2	93.6	61.5	53.9	53.5	59.2	600K
SAID	69.0	80.4	40.4	35.4	35.5	77.0	36.7	89.3	61.5	52.7	55.0	57.5	500K
SAID	66.1	83.9	41.3	38.5	35.8	76.0	38.3	89.7	55.8	50.9	55.3	57.4	20K
LoRA	78.3	85.7	45.1	41.0	39.5	88.0	47.1	93.6	60.6	56.8	55.2	62.8	9.1M
$(IA)^3$	78.0	87.5	48.6	40.8	40.8	87.0	49.4	94.7	68.3	59.8	56.0	64.6	540K
w/ pretrain													
LoRA	81.9	89.3	41.2	40.3	41.3	93.7	59.8	96.2	66.0	67.9	56.8	66.8	2.2M
TLoRA	80.7	90.5	39.9	40.9	41.2	93.0	54.4	95.3	66.3	67.4	57.3	66.1	1.4M
`Poly`	84.7	89.3	46.0	42.8	42.7	93.0	63.3	96.6	68.9	70.1	59.9	68.8	17M
`MHR`	85.2	90.5	44.7	42.3	42.8	94.7	63.3	96.7	70.5	70.6	59.8	69.2	17M
`TensorPoly-I`	85.2	91.7	45.0	42.5	42.5	96.7	63.1	96.6	68.6	69.8	60.6	**69.3**	27.8M
`TensorPoly-II`	86.9	90.5	44.9	41.6	42.2	93.0	58.3	96.3	68.3	66.7	58.3	67.9	13.3M

Table 3.1 Results on the T0 few-shot benchmark. All the results in our implementation are the median score of 3 seeds [0,1024,42]. For all the baseline scores, we report the results from Liu, Tam, *et al.*, 2022a. The **bold** is the best score. The underline is the second best in the same training setting.

To test the effectiveness of our approach, we conduct experiments on multi-task transfer learning in few-shot scenarios.

3.5.1 Backbone, Datasets and Evaluation

Backbone To ensure our model retains high performance with a limited number of labeled examples after fine-tuning, it is crucial to select an appropriate pre-trained model as the backbone. To facilitate a fair comparison with baseline methodologies, we have chosen the T0 model, consistent with the approach described in the IA3 paper by Liu, Tam, *et al.* (2022a). T0 was created by finetuning the T5 model on a multi-task mixture of datasets (Raffel *et al.*, 2020b). Each dataset is associated with multiple prompt templates that are used to format the example to (input, target) pairs. Examples in the datasets to train the T0 were prompted by applying the prompt templates from the Public Pool of Prompts (P3) (Bach *et al.*, 2022). As a result, all the different types of natural language tasks can trained with sequence-to-sequence structure (Victor *et al.*, 2022).

Datasets To evaluate the generalization capabilities of our models, we adopt the same benchmarking strategy as (Liu, Tam, *et al.*, 2022a), utilizing a subset of tasks designated as held-out from the multitask training. This benchmark encompasses a diverse array of

tasks, including sentence completion (COPA (Roemmele *et al.*, 2011), H-SWAG(Zellers *et al.*, 2019) and Story Cloze (Sharma *et al.*, 2018) datasets), natural language inference (ANLI (Nie *et al.*, 2019), CB (De Marneffe *et al.*, 2019) and RTE (Dagan *et al.*, 2005)), coreference resolution (WSC (Levesque *et al.*, 2012), Winogrande(Sakaguchi *et al.*, 2021)), and word sense disambiguation (WIC (Pilehvar and Camacho-Collados, 2018)). For each task, our evaluation strategy involves constructing sets of five few-shot training examples, which are generated by sampling subsets from each dataset using different seeds. We then report the median performance. It is noted that the prompt examples from each dataset using the prompt templates from P3 (Bach *et al.*, 2022), using a randomly-sampled prompt template for each example.

Evaluation For the evaluation of our models, we employ the rank classification methodology as outlined by the Liu, Tam, *et al.* (2022a) study. This approach involves ranking the model's log probabilities for all possible label strings associated with each task. The model's prediction is deemed correct if the label string with the highest log probability ranking corresponds to the correct answer. This method allows for a nuanced assessment of the model's predictive accuracy by examining its ability to prioritize the correct label over others based on their calculated log probabilities, offering a precise measure of its understanding and processing of the task at hand.

3.5.2 Baselines

In our comparative analysis, we initially set the benchmark by evaluating the performance of the TLoRA model against the traditional full fine-tuning approach, referred to as `FullFT`. In the `FullFT` scenario, we do not freeze any parameters of the pre-trained model, nor do we insert any adapters, allowing for a comprehensive update of the model's parameters during fine-tuning. Subsequently, we contrast our method against a suite of established parameter-efficient fine-tuning (PEFT) baselines to study the efficiency and effectiveness of each in terms of training parameter utilization. These baselines include:

- **Adapter**, as introduced by Houlsby *et al.* (2019b), which involves inserting trainable layers while keeping the pre-trained model's parameters fixed.
- **BitFit** by Zaken *et al.* (2021), which only fine-tunes the bias terms within the model.
- **LoRA** proposed by Hu *et al.* (2021), adjusting the low-rank adaptations of the weight matrices.
- **Compacter** and **Compacter++** by Karimi Mahabadi *et al.* (2021), which extend the adapter methodology with compact and efficient training strategies.
- **Prompt tuning** (Lester *et al.*, 2021) and Prefix tuning Li and Liang, 2021b add some tunable vectors to the input or hidden layer of the base model.

- **FishMask** by Sung *et al.* (2021), identifying and training a subset of parameters.
- **Intrinsic SAID** as described by Aghajanyan *et al.* (2020), focusing on intrinsic sparse activations.
- **IA3** (Liu, Tam, *et al.*, 2022a), emphasizing adaptability and efficiency.

We undertake a comparative analysis between routing approaches `TensorPoly`, `Poly` and employing a single expert mechanism without routing, specifically TLoRA and LoRA. This comparison aims to evaluate the impact of routing techniques on model performance and efficiency. By contrasting these models, we seek to understand how the dynamic allocation of tasks to specific experts in `TensorPoly` and `Poly` compares to the single expert models without shared information across tasks.

3.5.3 Results

Method	Pre-Training	Fine-Tuning
FullFT	$d \times d$	$d \times d$
LoRA	$2 \times d \times r$	$2 \times d \times r$
TLoRA	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R$	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R$
Poly	$2 \times d \times r \times \lvert S \rvert + \lvert \mathcal{T} \rvert \times \lvert S \rvert$	$2 \times d \times r \times \lvert S \rvert + \lvert S \rvert$
TensorPoly-I	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R + \lvert \mathcal{T} \rvert \times R$	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R + R$
TensorPoly-II	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R + \mathcal{T} \times R \times N$	$2 \times N \times r \times \lceil \sqrt[N]{d} \rceil \times R + R \times N$

Table 3.2 Number of parameters (per layer) used for each method. d is the input and output dimension of the training parameters. We assume they are identical. r is the rank in the LoRA, where $r \ll d$. N and R are the order and rank of entangled tensors respectively. S is the number of modules in Poly.

Table. 3.1 presents the mean downstream accuracy for 11 held-out tasks in the T0 benchmark. Notably, we did not train all these PEFT approaches, all the results are reported from (Liu, Tam, *et al.*, 2022a). When evaluating the performance of various PEFT approaches against single expert performances, it is observed that many PEFT strategies achieve similar outcomes while utilizing a significantly smaller subset of training parameters compared to the FULLFT (Full Fine-Tuning) method. Additionally, the implementation of Low-Rank Adaptation (LoRA) within our training framework further illustrates the efficiency of these methods. Specifically, TLoRA achieves a competitive score of 66.1, closely trailing the original LoRA's score of 66.8, while requiring only about 60% of the training parameters used by LoRA. This demonstrates that TLoRA not only matches the effectiveness of LoRA in terms of performance but also surpasses it in terms of parameter efficiency.

In our analysis, we initially contrast the modular model against the dense model, followed by a comparison of routing-based approaches with a single adapter strategy. Within this context, we utilize both LoRA and TLoRA as baselines for these routing techniques. According to the results presented in Table 3.1, the `Poly` model demonstrates superior

performance over LoRA by a margin of 2.0 points. Moreover, TLoRA exhibits an improvement of 3.2 points over the base TLoRA model, underscoring the efficacy of routing in enhancing multi-task generalization within the realm of multi-task transfer learning.

When evaluating various `Poly` variants, `TensorPoly-I` stands out by not only surpassing recent state-of-the-art achievements but also by outperforming `TensorPoly-II`, despite the latter employing a more granular routing function. This finding is particularly noteworthy, as it suggests that the increased specificity of the routing function in `TensorPoly-II` does not necessarily translate to superior performance. We will discuss this in Section 3.5.6.

Rank and Order Analysis

The rank of an entangled tensor serves as a measure of its capacity, denoting the fewest number of rank-one tensors required for its original tensor (§3.3.2). We focus on examining the relationship between the rank of tensor and the associated training parameters. Table 3.2 illustrates how the tensor rank R and tensor order N impact the selection of training and fine-tuning parameters. Illustrated in Figure 3.3, when the tensor rank is set to 1, it corresponds to the original TLoRA configuration. An increase in tensor rank necessitates a larger number of training parameters. Concomitantly, there is a notable enhancement in performance. This progression underscores the direct correlation between the tensor rank and the model's capability.

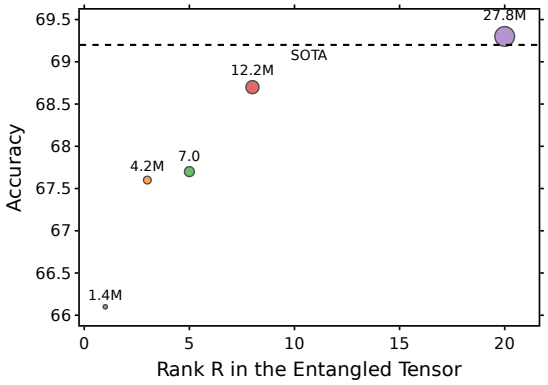


Figure 3.3 Explore the tensor Rank in the Entangled Tensor. We set order $N = 2$ in this setting.

Then tensor order N, correlates with the granularity of training parameters; a tensor of order 4 yields a finer-grained module compared to one of order 2. Our research examines the impact of varying tensor orders on performance outcomes. For simplicity, we constrain our analysis to tensors of order 2 and 4. Our results demonstrate that a tensor of order 2 outperforms one of order 4 by a margin of 1.7 for the `TensorPoly-I`. For `TensorPoly-II`, an order-2 tensor exceeds the performance of an order-4 tensor by 4.8, suggesting a balance must be struck between the size of experts and its efficacy. While higher-order tensors may conserve training parameters, this comes at the cost of diminished performance.

3.5.4 Routing Analysis

In the paper referenced as Caccia *et al.* (2022), the `MHR` study demonstrates that fine-tuning solely the routing function can yield competitive outcomes. This insight provides

Model	Adaptation Params	ACC
LoRA[r=1]	2.2M	66.8
TLoRA [r=16]	1.4M	66.1
Poly[r=1]	17 M	68.8
TensorPoly-I[N=2]	12.2M	68.7
TensorPoly-I[N=4]	4.3M	66.9
TensorPoly-II[N=2]	13.3M	67.9
TensorPoly-II[N=4]	7.6M	63.1

Table 3.3 Results on T0 dataset, with T0-3B. The Adaptation parameter is the number of parameters in the finetuning process.

a valuable perspective for our investigation into various routing strategies within the TensorPoly framework. In line with this approach, we focus on fine-tuning exclusively the routing function during the few-shot adaptation process, indicated by the notation -z. This methodological choice allows us to isolate the impact of the routing function's optimization on the overall performance of the TensorPoly model, thereby offering a clearer understanding of how dynamic routing contributes to the adaptability and efficiency of the model in few-shot learning scenarios.

As detailed in Table 3.4, an initial observation reveals that the routing parameters necessitate a remarkably small number of training parameters. The performance metrics of Poly-z, TensorPoly-I-z, and TensorPoly-II-z lag behind their counterparts where both the modules and the routing function undergo fine-tuning. This disparity highlights the critical role that modules play in the fine-tuning process for TensorPoly. The findings suggest that while optimizing the routing function alone can contribute extreme parameter efficiency, the integration of module fine-tuning is indispensable for achieving the best possible performance. This underlines the synergy between modules and routing functions in the TensorPoly architecture.

Model	Adaptation Params	ACC	Δ
Poly-z	3.5k	65.4	-3.4
MHR-z	220K	68.3	-0.9
TensorPoly-I-z[N=2]	3.5k	64.3	-4.4
TensorPoly-I-z[N=4]	3.5k	62.7	-0.4
TensorPoly-II-z[N=2]	6.9k	62.5	-5.4
TensorPoly-II-z[N=4]	13.8k	60.3	-2.8

Table 3.4 Results on T0 dataset, with T0-3B. We only fine-tuning the routing function. The symbol Δ represents the difference in ACC (accuracy) scores between the 'modules+routing' configuration and the 'only routing' configuration.

3.5.5 Flop Analysis

TLoRA aims to reduce the number of training parameters by parameterizing the original LoRA architecture, yet it does not alter LoRA's computational process. This implies that TLoRA might incur additional computational overhead compared to LoRA. To understand this, we analyze the floating-point operations (FLOPs) involved in TLoRA.

Considering the tensor product of a vector a of size $n \times 1$ with a vector b of size $1 \times n$, we obtain a matrix C of size $n \times n$. This operation involves n^2 multiplications due to the n elements in vector a and n elements in vector b, each contributing to an entry of the resulting matrix. As outlined in Table 3.2 and illustrated in Figure 3.2, the additional computational effort in TLoRA stems from the tensor product operations, which are dictated by the order of the tensor. Specifically, for parameterizing a dimension d with rank r and tensor rank R, the extra computation required can be approximated as $d \times r \times R$. This calculation helps quantify the additional computational resources necessary for TLoRA.

3.5.6 Discussion

In modular language models, tensor product routing is poised to construct more intricate patterns. This capability arises from its method of aggregating activated modules through tensor products, as opposed to the conventional linear routing approach. However, our experimental results diverged from theoretical expectations, revealing that tensor product routing did not contribute to improved final performance in our models. This discrepancy prompts a future line of inquiry: we plan to explore whether there exist specific benchmarks or conditions under which tensor product routing could demonstrate its purported benefits. Identifying such scenarios will be crucial for harnessing the potential advantages of tensor product routing in modular language models.

In the current methodology, each tensor-based module is not specialized towards any particular domain. In the future, we intend to explore a more tailored training strategy. This will involve dedicating each tensor to a specific domain and subsequently aggregating these domain-specialized tensors using tensor product operations. Our objective is to assess whether this domain-specific aggregation approach can yield superior generalization capabilities.

3.6 Conclusion

In this paper, we introduce a novel modular language model named `TensorPoly`. This model incorporates tensorized modules, specifically TLoRA, to significantly reduce the number of training parameters required by the traditional LoRA approach. We employ two distinct strategies for aggregating activated modules: a linear routing function, referred to as `TensorPoly-I`, and a more finely trained tensor product routing, named `TensorPoly-II`. Our evaluation across various multi-task learning scenarios reveals

that modular language models, such as `TensorPoly`, surpass the performance of single-adapter models. This underscores the importance of sharing task information through a routing function in multi-task learning contexts. Notably, `TensorPoly-I` achieves state-of-the-art results, highlighting the effectiveness of the `TensorPoly` framework. However, `TensorPoly-II` does not outperform `TensorPoly-I` in our experimental settings, suggesting areas for further investigation in future research.

4 Towards Modular LMs by Building and Reusing a Library of LoRA Adapters

4.1 Instruction

The multiplicity of adaptations of a base language model (LM) via parameter-efficient *adapters* calls for studying whether reusing such trained adapters can improve performance for new tasks or new inputs. In this paper, we study how to best build a *library* of adapters given multi-task data and study techniques for *zero-shot* inference and effective task adaptation through *routing* in such library. We benchmark existing approaches to *build* this library and introduce a clustering-based method `MBC` ("model-based clustering") which groups tasks based on the similarity of their adapter weights, indirectly optimizing for transfer across the multi-task dataset. To *re-use* this library of adapters, we present a novel zero-shot routing mechanism, `Arrow`, which enables dynamic selection of the most relevant adapters for new inputs without the need for retraining. We make steps towards creating modular, adaptable LMs that can outperform traditional full-finetuning, paving the way for efficient and flexible utilization of LMs across a wide array of tasks.

Adapting pretrained Language Models (LMs) to downstream tasks, domains, or user profiles is of paramount importance given the recent democratization of their usage, catalyzed by the release of open-weight LMs (Zhang, Han, *et al.*, 2023; Microsoft Research, 2023, *inter alia*). Most adaptation techniques rely on an *adapter*, a parameter-efficient mechanism built on top of the base LM (Hu *et al.*, 2021; Liu, Tam, *et al.*, 2022a; Li and Liang, 2021a; Zhang, Han, *et al.*, 2023).

With the growing number of adaptations of a base model (Beck *et al.*, 2021; Mangrulkar *et al.*, 2022), it is natural to ask how to re-use such adaptations to process new inputs or to solve new tasks. Prior works show that previously trained adapters can accelerate supervised adaptation for unseen tasks (Ponti *et al.*, 2023b; Vu *et al.*, 2021; Huang *et al.*, 2024). If one considers these trained adapters as *experts* at their respective tasks, there is an opportunity to build a more expressive, "modular" LM that can also route to pre-existing adapters zero-shot, i.e. for new inputs. In contrast to standard mixture-of-experts approaches (MoE) (Fedus, Zoph, *et al.*, 2022a), there is the need to devise post-hoc routing strategies that can handle multiple adapters without requiring joint training of the routing mechanism and expert weights.

In this paper, we study techniques to create a modular LM "end-to-end" by first building and then re-using a *library* of adapters, *both* for supervised adaptation and zero-shot

generalization. We first identify a base LM, such as Phi2 (Microsoft Research, 2023) or StableLM (Tow *et al.*, 2023). Then, we investigate building a library of adapters by leveraging a multi-task dataset such as Flan-v2, which is composed of 264 tasks.[1] We focus on LoRA adapters (Hu *et al.*, 2021) and leave the extension to other adapter types and combinations thereof for future work. Once the adapters have been built, we devise routing strategies to deal with both *zero-shot* generalization on 10 downstream tasks comprising common-sense reasoning and coding (ARC (Clark *et al.*, 2018), MBPP Austin *et al.*, 2021, *inter alia*) and 12 SuperNatural Instructions (SNI) tasks (Wang, Mishra, *et al.*, 2022b), which we also use to study *supervised adaptation*.

How to build the adapter library? One straightforward approach is to operate in a *private* scenario, in which one trains one adapter for each task in the multi-task data and mix those adapters for unseen tasks (Chronopoulou *et al.*, 2023; Vu *et al.*, 2021; Huang *et al.*, 2024). This method is useful when the multi-task data cannot be shared for privacy concerns but adapters can (Mireshghallah *et al.*, 2020). To favour transfer between training tasks, recent approaches compress the multi-task data into a smaller set of reusable, composable adapters (Ponti *et al.*, 2023b; Caccia *et al.*, 2023). Additionally to benchmarking these approaches in our setting, we propose a simple two-stage approach to build a library of adapters. We find a positive correlation between the similarity of LoRA weights of tasks and transfer. Therefore, we first exploit LoRA similarity in weight space between privately trained adapters as a proxy for detecting clusters of similar tasks, then train one adapter for each cluster. Our approach empirically improves performance and surpasses the state-of-the-art while matching compute budget. We refer to this method with `MBC` (Model-Based Clustering).

How to reuse the library for new scenarios? Given a library of trained adapters, we examine strategies on how to re-use the library in two settings: *zero-shot generalization* and parameter-efficient *supervised adaptation* to new tasks. If the latter has received extensive attention from the research community (Ponti *et al.*, 2023b; Vu *et al.*, 2021; Huang *et al.*, 2024), the former remains relatively unexplored. Re-using adapters in a zero-shot manner is significantly more challenging because we do not have downstream labelled data to learn a routing mechanism to select which adapter to use. In this respect, we experiment with a routing mechanism that automatically routes new inputs to relevant adapters without re-training. For each layer in the base LM, we compute an adapter representation by computing the direction of maximum variance induced by the adapter weights. This is computed only once after adapter training. At inference time, we route each hidden state by computing its alignment with each adapter representation.

We dub our method `Arrow` (↗) routing.

In summary, our contributions are the following:

- We study how to create a parameter-efficient, modular LM "end-to-end". We demonstrate that we can outperform full-finetuning and competitive baselines on a set of 10 well-established downstream tasks.

[1]We held-out SNI tasks to test supervised adaptation.

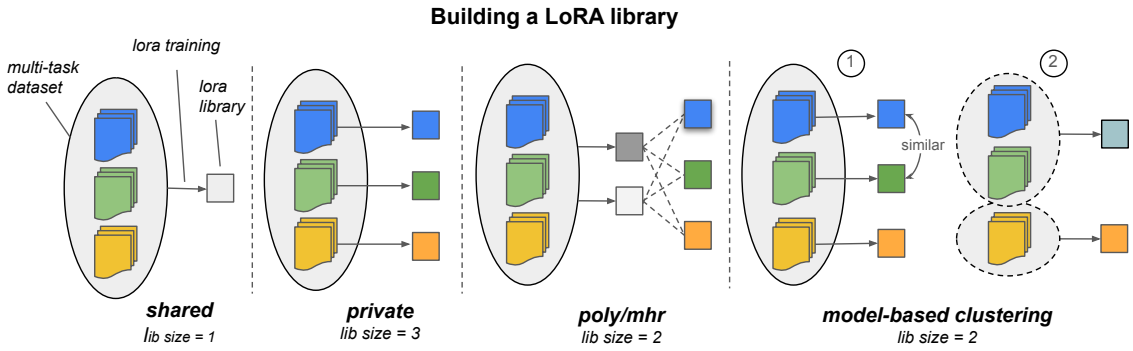


Figure 4.1 Different solutions for building a library of LoRA adapters. `Shared` trains one adapter via multi-task training. `Private` trains one adapter per task. `Poly`/`MHR` train a set of basis adapters (2 in the figure) that can be recombined to obtain task adapters. `MBC` i) trains private adapters, ii) clusters tasks based on LoRA similarity, iii) trains one adapter per cluster ensuring transfer between similar tasks.

- We propose a clustering approach (`MBC`) to train a library of adapters given multi-task data which improves 1.7% absolute accuracy in zero-shot generalization over existing baselines, and 3.3% over Phi-2 and 4.3% over StableLM.

- We propose a zero-shot routing approach to select which adapters to use from the library (`Arrow`). This improves performance further by 0.6% accuracy for Phi-2 and enables the routing of independently trained experts without requiring access to their training data.

4.2 Preliminaries

We are given a set of tasks $\mathcal{T} = \{t_1, \ldots, t_T\}$, with each task t_i a dataset containing a set of samples $\mathcal{D}_i = \{(x_1, y_1), ..., (x_n, y_n)\}$. The union of the training sets constitutes our multi-task dataset $\mathcal{D}$, i.e. Flan (Longpre *et al.*, 2023). In order to create our library of task adapters, we use LoRA (Hu *et al.*, 2021). LoRA achieves competitive trade-offs between performance and parameter efficiency (Mahabadi *et al.*, 2021) by modifying the linear transformations in a base LM. For each linear transformation in the base LM, LoRA modifies the base model parameters as follows:

$$h = Wx + s \cdot AB^\top x, \qquad \text{(LoRA)}$$

where W are the (frozen) weights of the base LM and $A, B \in \mathbb{R}^{d \times r}$ are low-rank learnable parameters and $s \geq 1$ is a tunable scalar hyperparameter. LoRA achieves parameter efficiency because of the reduced rank r ($\ll d$).

4.3 Building the LoRA Library

We present different alternatives to build a library $\mathcal{L}$ of adapters in such a way that they can perform well on the tasks they have been trained on but also such that they are versatile enough to be effective for other inputs and tasks (e.g. obtain high performance). To do so, one should seek methods that enhance multi-task transfer while reducing task interference (Wang *et al.*, 2021; Chen, Shen, *et al.*, 2022).

`Private` Adapters One straightforward solution is to train separate adapters on each of the training tasks, i.e. the library will be composed of T adapters (see Fig. 4.1). Several existing methods operate in this setting, such as LoraHub (Huang *et al.*, 2024), AdapterSoup (Chronopoulou *et al.*, 2023) and SPoT (Vu *et al.*, 2021). Although this solution does not exploit multi-task training, it is useful in settings where the task data is private, e.g., user data, and cannot be shared (Mireshghallah *et al.*, 2020).

`Shared` Adapter To encourage transfer, one solution is to train a single adapter on all the multi-task training data. One possible shortcoming is the reduced capacity allowed to fit the multi-task training data and the possibility of interference between the multitude of training tasks (Ponti *et al.*, 2023b). Training a single adapter may result in negative transfer because task gradients are misaligned (Wang *et al.*, 2021). An obvious solution to reduce the amount of interference is to augment the number of trainable parameters, e.g. to fine-tune the whole base LM on the multi-task data (Liu, Tam, *et al.*, 2022a).

`Poly` / `MHR` Adapters Polytropon (Poly) and Multi-Head Routing (MHR) (Ponti *et al.*, 2023b; Caccia *et al.*, 2023) explore intermediate approaches between private and shared, where $K < T$ "basis" adapters are trained on the multi-task training data. These K adapters can be considered "latent skills", as each task adapter in the multi-task training set can be expressed as a linear combination of these basis adapters. If private training for all the tasks learns a matrix of parameters $\Phi \in \mathbb{R}^{T \times D}$, where D is the dimensionality of the LoRA adapters, Poly decomposes $\Phi = U\hat{\Phi}$, where $U \in \mathbb{R}^{T \times K}, \hat{\Phi} \in \mathbb{R}^{K \times D}$, $\hat{\Phi}$ storing the latent skills and U the linear combination coefficients for each task which specify the task-specific routing w.r.t. the latent skills. Both U and $\hat{\Phi}$ are trained jointly on the multi-task training set by gradient descent. Note that the skills $\hat{\Phi}$ do not correspond to specific tasks and therefore it is not clear how to re-use them for zero-shot generalization (Caccia *et al.*, 2023).

Model-Based Clustering In this paper, we propose another approach to compress multi-task data into a set of reusable adapters; we cluster tasks based on their similarity and then train one adapter per task cluster. Ideally, one wants the similarity between two tasks to be correlated with the benefit of training a single model on both tasks compared to having two independent models (Fifty *et al.*, 2021; Vu *et al.*, 2020a). One intuition is that we can approximate the amount of transfer between a pair of tasks by computing the similarity of their private LoRA parameter vectors. To confirm this, we devise the following experiment: we sample pairs of tasks $(t_i, t_j), t \in \mathcal{T}$ from the multi-task dataset, and we train both a) a LoRA on each task independently b) a LoRA on the union

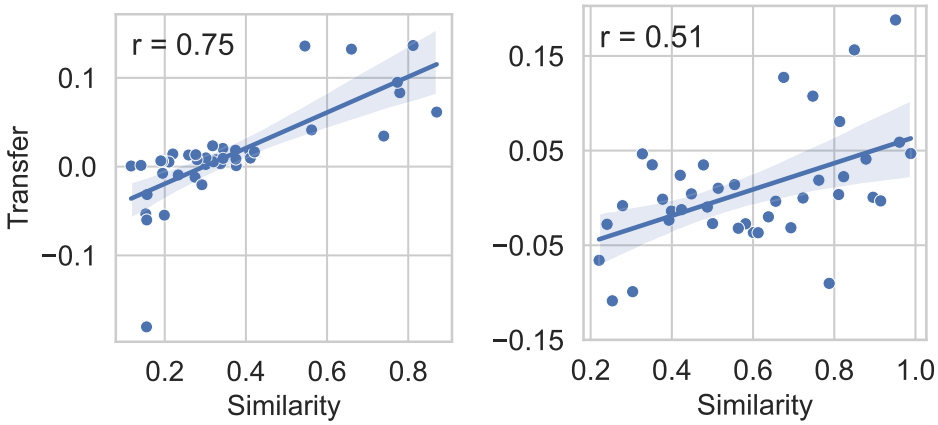


Figure 4.2 We report the cosine similarity of LoRAs adapters for pairs of tasks vs. the transfer (see text). Similarity between LoRAs correlates with transfer, indicating that if LoRA embeddings are dissimilar, then we should abstain from multi-task training.

Algorithm 2 Model-Based Clustering

Input: Multi-task data $\mathcal{D}_1, \ldots, \mathcal{D}_T$, base model `LM`, number of library adapters K
Output: Library $\mathcal{L}$
$\mathcal{L} = \{\}$, $AB = \{\}$ *# LoRA parameters*
for $t = 1$ **to** T **do**
  $A_t, B_t = \text{Train}(\mathcal{D}_t; \texttt{LM})$ *# Train LoRA on task t*
  $AB = AB \cup \{\text{concat}(\text{flatten}(A_t), \text{flatten}(B_t))\}$
end for
$\hat{AB} = \text{SVD}(AB)$ *# Reduce LoRA dim*
$S = \text{cos\_similarity}(\hat{AB}, \hat{AB}^T)$ *# $T \times T$ similarities*
$c_1, \ldots, c_K = \text{K-Means}(S, K)$ *# Cluster similarities*
for $k = 1$ **to** K **do**
  $\mathcal{D}_k = \bigcup \mathcal{D}_t, \forall t \in c_k$ *# Join datasets in cluster*
  $A_k, B_k = \text{Train}(\mathcal{D}_k; \texttt{LM})$ *# Train LoRA on cluster*
  $\mathcal{L} = \mathcal{L} \cup \{(A_k, B_k)\}$
end for
Returns $\mathcal{L}$ =0

of the tasks' training datasets. We then compute the cosine *similarity* between the flattened LoRA parameters. We quantify *transfer* as the difference in the average test log-likelihood induced by the joint and private models when evaluated on the test set of the two tasks. In Fig. 4.2, we see that, for two different base models (`GPT-Neo` and `Phi-2`), the closer the tasks are in parameter space, the more transfer we observe when we train on the joint dataset.

The previous observation warrants our simple two-stage training procedure illustrated in Fig. 4.1 (right-most). Given a fixed computation training budget of N training steps per task, we use the first n steps to train private LoRAs. We then use these LoRA parameters to group tasks into K clusters by running a standard clustering algorithm (K-Means). In the second stage of training, we train one adapter per cluster for an additional $N - n$ training steps, which keeps the total amount of computation similar to

other approaches. We refer to this method as Model-Based Clustering (MBC) as it uses the model-based information encoded in the weights to determine a similarity metric between tasks (see Alg. 2).

4.4 Re-Using the LoRA Library

We study the re-use of a trained library $\mathcal{L}$ in two scenarios: for new inputs x^*, i.e. *0-shot*, and in a *supervised adaptation* setting, where new tasks t^* come equipped with their training data $\mathcal{D}_{t^*}$. While the latter has been addressed in recent works (Huang *et al.*, 2024; Caccia *et al.*, 2023; Vu *et al.*, 2021), the former scenario remains less explored. We first devise routing strategies in the 0-shot setting and supervised setting, and then we briefly describe how to aggregate the contributions of adapters selected by the routing strategies.

4.4.1 Routing

Let us denote the hidden state for any token at each transformer layer produced by the input token x^* as h^*. Similar to MoE approaches, we seek to parameterize a routing distribution for each layer in the transformer model that prescribes which adapters to use. We denote this categorical distribution over L outcomes as $p(a \mid h^*, x^*)$, where we drop the dependence on the layer for simplicity. For example, in standard MoE approaches (Fedus, Zoph, *et al.*, 2022a), $p(a \mid h^*, x^*) = \text{softmax}(\lambda W h^*)$, where λ is a temperature parameter, $W \in \mathbb{R}^{L \times D}$ are routing parameters jointly learnt with the expert library during pre-training and L is the size of the library $\mathcal{L}$. As we will see in Section 4.2, we only consider averaging schemes in this paper, but one could imagine sampling from the routing distribution instead. Given that we relax the assumption that the routing and the library should be trained together, we must devise ways to learn such the routing distribution a posteriori.

Zero-Shot Example Routing

μ Routing One straightforward method to route to existing experts is to set the routing distribution to uniform for all layers, $p(a_\ell \mid h^*_\ell, x^*) = [1/L, \ldots, 1/L]$. This type of routing has shown to be quite effective in recent work (Caccia *et al.*, 2023; Chronopoulou *et al.*, 2023) and, due to the linearity of the LoRA adapters, effectively boils down to averaging uniformly weights of the trained adapters.

TP Routing Another variant is to treat routing as an L-way classification problem. Specifically, given an input x belonging to task t in our multi-task training set, we train a task predictor f by minimizing $y = -\log f(x)[t]$, where $f(x)$ is a probability distribution obtained by softmax, and set $p(a_\ell | h^*_\ell, x^*) = f(x^*)$ for every layer at inference time. We use a smaller T5 model to parameterize f (Raffel *et al.*, 2020a). We call this predictor TP (Task Predictor).

Algorithm 3 `Arrow` Routing ↗ initialization

Input: LoRA library $\mathcal{L}$, layer ℓ
Output: Routing parameters for layer ℓ, W_ℓ
for $i = 1$ **to** L **do**
  $A_i, B_i = \mathcal{L}[i, \ell]$ *# Get weights for expert i*
  $U, D, V = \text{SVD}(A_i B_i^T)$
  $W_\ell[i] = V[:, 0]$ *# First right singular vector*
end for
Returns W_ℓ =0

`CM` **Routing** Centroid Matching (CM) routing amounts to representing the expert's prototype as the average hidden representation $\bar{h}_l$ calculate on the i's expert's dataset: $\bar{h}_l = \frac{1}{|D_i|} \sum_{j=1}^{|D_i|} h(x_j)$. The routing distribution is then calculated per-token by normalizing the vector of cosine similarities between the new hidden state and expert's prototype. This routing is similar in spirit to existing works (Jang *et al.*, 2023; Belofsky, 2023).

`Arrow` **Routing** ↗ The rows of the routing matrix W_ℓ of standard MoE routing can be interpreted as expert "prototypes". `Arrow` prescribes a way to learn such routing matrix in a 0-shot fashion with no further training. Let's denote by $\{A_i, B_i\}$ the parameters for expert i at layer ℓ, where we drop the dependency on ℓ. LoRA expert i transforms each token's hidden state h^* as $h_i^* = A_i B_i^T h^*$. Therefore, a prototype for the expert i can be found by decomposing with SVD the outer product $A_i B_i^T$ and taking the right first singular vector of this transformation (see Alg. 3). This determines the direction of most variance induced by expert i in the space of hidden states h. If the LoRA adapters are of rank 1, i.e. $A_i, B_i \in \mathbb{D}^{D \times 1}$ the prototype for the expert i will be equal to the normalized B_i vector, i.e. $\text{argmax}_{v, \|v\|_2 = 1} \|(A_i B_i^T v)\|_2 = B_i / \|B_i\|_2$. The advantages of such per-token routing include a) it doesn't require access to training data for each expert which is useful when experts are added from heterogeneous sources; b) it can parameterize a different routing distribution at every layer per token therefore potentially increasing overall model capacity; and c) it is compute efficient since it requires no further training and SVD decomposition can be computed efficiently for low-rank matrices Elhage *et al.*, 2021; Nakatsukasa, 2019. In Section 4.8.1, we provide empirical evidence that indeed, $\|A_i B_i^T v\|_2$ is larger when v belongs to task i, thus motivating this routing approach.

Supervised Task Routing

When generalizing to a new task, we can learn the optimal routing given the task data $\mathcal{D}^*$. This setting is similar to previous task generalization works (Ponti *et al.*, 2023b; Caccia *et al.*, 2023; Huang *et al.*, 2024). We compare results in this supervised setting to both Poly (Ponti *et al.*, 2023b) and LoraHub (Huang *et al.*, 2024).

`Poly` **Routing** treats the distribution over experts at each layer as an L-dimensional parameter that is learned by minimizing the LM-loss on the new task $\mathcal{D}^*$. It optimizes the merging coefficients of LoRA adapters for the new task, i.e. $A^* = \sum_{i=1}^{L} w^i A_i$, $\{A_i, B_i\} \in \mathcal{L}$ together with parameters $\{A_i, B_i\}$. Here $p(a|h^*, x) = (w^1, \ldots, w^n)$ is the (input-independent) learnable routing distribution for a given layer.

`LoraHub` **Routing** is similar to Poly with the exception that a) it recurs to gradient-free optimization to learn routing coefficients and b) it doesn't fine-tune the library adapters incurring in the linear combination (Huang *et al.*, 2024).

4.4.2 LoRA Composition

Given the routing weights $w = p(a \mid h^*, x)$ obtained using any routing method, we linearly combine adapters in the library. Unless stated otherwise, we combine A and B of LoRA separately before the outer product, i.e. $A^* = \sum_{i=1}^{L} w_i A_i$, $B^* = \sum_{i=1}^{L} w_i B_i$ and use the resulting adapter to perform inference at every patched layer of the base LM (Ponti *et al.*, 2023b; Huang *et al.*, 2024).

4.5 Experiments

Our experimental evaluation aims to answer the following questions: 1) How do methods for building LoRA library compare? 2) How large is the gap between privately trained libraries and libraries which assume access to multi-task data? 3) To what extent is routing important when working with a library of LoRA adapters?

Multi-Task Dataset We train expert modules on 264 tasks built from the original FlanV2 dataset Longpre *et al.*, 2023. We excluded the SNI tasks (> 1000 tasks) Wang, Mishra, *et al.*, 2022b for computational reasons induced by training `Private` libraries. We reserved 12 SNI tasks for downstream out-of-domain evaluation. After removing our downstream evaluation tasks from the training set, our multi-task training set results in 256 training tasks in total. Similarly to Wang *et al.* (2023), we sub-sampled 10,000 examples per task to ensure computational feasibility. Within these samples, 1,000 are allocated for validation and early-stopping. We will release our dataset for reproducibility.

Evaluation For our supervised adaptation study, we use 12 held-out SNI tasks, each corresponding to a different SNI category, and we cut the number of training examples to 10,000 examples per task. We evaluate performance with RougeL scores Lin and Hovy, 2003. For 0-shot evaluation, we use ten tasks commonly used in the literature, including common-sense reasoning (WinoGrande Sakaguchi *et al.*, 2021, HellaSwag Zellers *et al.*, 2019, piqa Bisk *et al.*, 2020), Q&A (boolQ Clark *et al.*, 2019, OpenbookQA Mihaylov *et al.*, 2018, ARC-easy and hard Clark *et al.*, 2018) coding (HumanEval Chen *et al.*, 2021, mbpp Austin *et al.*, 2021) as well as BBH[2] tasks Suzgun *et al.*, 2022. We remove any overlap between the downstream evaluation task and the Flan multi-task training set (boolQ, ARC, winogrande, hellaswag, openbookqa and piqa).

Models & Training This work focuses on augmenting LMs with a library of adapters to transform them into multi-task learners. Our primary focus is on the Phi2 model

[2]We test on a subset of randomly sampled 1000 examples to reduce evaluation costs.

Microsoft Research, 2023, a state-of-the-art pretrained model with 2.7 billion parameters, leading its class of models with parameter counts below 3 billion, according to the open leaderboard Beeching *et al.*, 2023. Additionally, we conducted experiments using the GPT-Neo 1.3B Black *et al.*, 2021 model, a smaller and less powerful pretrained transformer model. We also experimented with the StableLM 3B Tow *et al.*, 2023, which yields capabilities between GPT-Neo and Phi2. For all models, we only patch attention layers with LoRA adapters. Unless stated otherwise, for all our multi-task training and single-task adaptation scenarios, we use LoRA rank of 4, dropout of 0.05 and learning rate of 1e-4. Unless specified, we set the number of clusters for MBC to 10, resulting in the best performance for both Phi2 and StableLM models as demonstrated in Fig. 4.3.

Methods In both 0-shot and adaptation scenarios, we consider the following methods. `Base` – the base model tested without any adaptation; `Shared` – a single expert LoRA finetuned on the joint training set of all tasks (e.g. 256 in case of Phi2 model) on top of the base model using multi-task learning; `FullFT` – same as `Shared` but the full model is finetuned. We adopt the following naming convention for the models using a library of multiple experts: "`library type`"-"`routing`". For the library type, we study `Poly`, `MHR`, `Private` and `MBC` libraries described in Sec. 4.3. For `MBC`, we match the total amount of compute, meaning that we use half of the training steps to compute the LoRA clustering and the other half to compute the final cluster embeddings. For routing, we use μ, `TP`, `Arrow` in the 0-shot scenario and `Poly` and `LoraHub`[3] for the supervised scenario, described in Sec. 4.4. We also include oracle baselines: `1-Oracle` – a single expert that performs the best on all downstream tasks; `N-Oracle` – we select best performing expert for each tasks.

4.5.1 Zero-Shot Results

In the 0-shot scenario downstream tasks are evaluated without further fine-tuning. Tab. 4.1 presents the mean downstream accuracy for 10 held-out Flan tasks, while Tab. 4.5 provides the complete per-task view. Focusing on the Phi-2 model, we observe that `MHR`-μ delivers a strong 0-shot model achieving competitive performance w.r.t. `Shared` and `FullFT`, in line with the results of in Caccia *et al.* (2023). Interestingly, training one adapter per task and then taking the average, `Private`-μ, still achieves gains w.r.t. `Base`, albeit falling short on multi-task training, highlighting the power of uniform adapter averaging (Chronopoulou *et al.*, 2023). Comparing the performance of our proposed MBC approach for library construction (`MBC`-μ) to previous approaches, we notice a sizable bump in performance of 1.2% absolute accuracy points over the strongest baseline. We make similar observations for the Stable LM model as well as 0-shot performance of Phi2 on 12 SNI tasks in the Tab. 4.9.

Next, we analyze whether a more sophisticated routing strategy can improve performance w.r.t. uniform routing. We see that both `TP`, `CP` and `Arrow` routing improve

[3]For `LoraHub`, we match the amount of compute used by SGD. Assuming the backward pass is twice the compute of a forward pass, and since nevergrad (NG) Rapin and Teytaud, 2018 only does forward passes, to match the compute of 5 SGD training epochs, we perform 30 epochs of NG with 1/2 of the training data used by SGD methods.

	Library	Routing	L	Mean Acc.
Phi2 (2.8B)	Base	-	-	63.8
	FullFT	-	-	65.6
	Shared	-	1	65.5
	Poly	μ	1	65.7
	MHR	μ	1	65.9
	Private	μ	256	64.8
	Private	TP	256	65.7
	Private	CM	256	65.7
	Private	↗	256	66.0
	MBC	μ	10	67.1
	MBC	TP	10	66.8
	MBC	CM	10	67.1
	MBC	↗	10	<u>67.7</u>
	Private	1-Oracle	256	66.6
	Private	n-Oracle	256	68.2
StableLM (3B)	Base	-	-	51.6
	Shared	-	1	52.5
	Private	μ	100	54.8
	Private	TP	100	53.2
	Private	↗	100	55.2
	MBC	μ	10	55.9
	MBC	TP	10	55.9
	MBC	↗	10	55.9

Table 4.1 0-shot results for Phi-2 and StableLM base models on 10 held-out Flan tasks (see Tab. 4.5 for the full per-task view and Tab. 4.9 for SNI tasks). L denotes the library size. We only train 100 experts for StableLM for computational reasons in the Private scenario. The best results are underlined.

the performance over μ routing for the Private library, gaining 0.9% and 1.2% points, surpassing the performance of FullFT and 1-Oracle[4]. This highlights the importance of routing for larger libraries. Interestingly, inline with Jang *et al.* (2023) we find that 1-Oracle outperforms both FullFT and Shared. Notably, arrow routing almost reaches the quasi-upper bound of N-Oracle[5].

TP routing decreases performance when compared to uniform routing for MBC, while MBC-↗ improves over MBC-μ by 0.9% accuracy points on average and appear as a more robust routing method for both Private and MBC libraries. Overall, MBC-↗ improves over 3.9% absolute accuracy points over the base model and 2.1% absolute over FullFT.

For StableLM, we see a similar trend with MBC libraries achieving the best performance. Arrow routing contributes in a 0.4% increase in average performance over uniform routing

[4]The best performing expert here corresponds to the task ropes_plain_no_background.
[5]Its only an upper bound for top-1 hard selection.

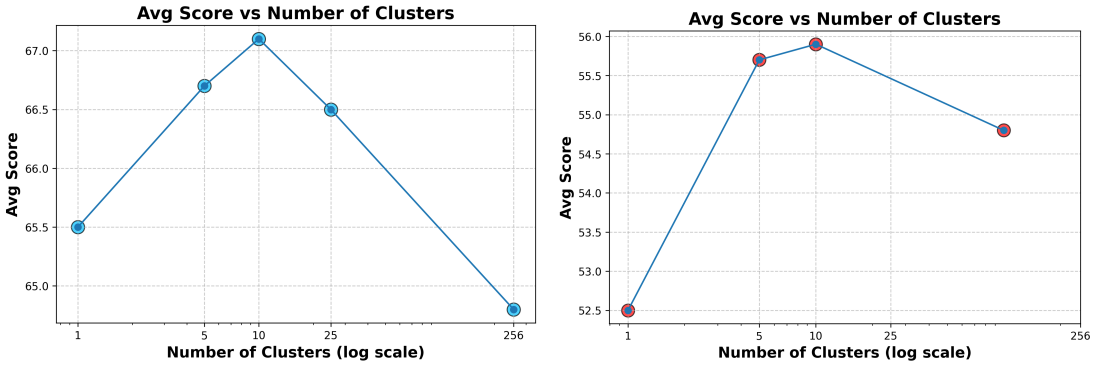


Figure 4.3 0-shot performance on 10 held-out Flan tasks plotted against the number of MBC clusters for Phi2 and StableLM.

when using a `Private` library (`Private-`↗ vs. `Private-`μ). We do not see any gains from using other routing methods for 10 experts in `MBC` library in this case.

Analysis `MBC` enhances performance of the library across all our results. To investigate further, we compare different clustering techniques. First, we compare to clusters obtained by randomly selecting examples (`RandomExamples`). This is equivalent to randomly partitioning the joint multi-task dataset. Then, we compare to clusters obtained by randomly choosing tasks from the entire set of training tasks (`RandomTask`). Finally, we cluster task embeddings, which are obtained by forwarding task-specific examples in the model, and averaging their representation at the model's penultimate layer (`Embeddings`). For all the methods, we set the number of clusters to 10. The results are in Table 4.2. `RandomTask` surpasses `RandomExamples` by 1.6%, which indicates that grouping tasks rather than task examples is crucial for transfer ability. `Embeddings` underperform `MBC` and supports our observation that the cosine similarity between the weights of privately-trained LoRA correlates better than using representation similarity for 0-shot generalization. Further evidence of `MBC` effectiveness is given when analyzing 0-shot generalization for SNI (see Table 4.7), where `MBC` outperforms `RandomTask` by 2.4 Rouge-L points. Additionally, we also report average pairwise cluster "similarity" (as measured by the cosine similarity of the cluster LoRA weights) and observe a tendency that expert clusters with lower similarity, i.e. higher diversity, tend to result in higher performance. We hypothesize that this is an artifact of different clusters contributing distinct features to the joint model and leave further investigation in this direction for future work (Jolicoeur-Martineau *et al.*, 2023). Finally, we assess the the importance of model-based clustering by looking at the in-distribution performance on the test sets of the tasks that *are* in the library assuming known task ids. As shown in Fig, 4.4, `MBC` also results in a significant increment in the in-distribution performance for all clusters. Interestingly, we observe that in-distribution `Shared` under-performs `Private`, which can be attributed to negative task interference observed also by e.g. Jang *et al.* (2023) for T5 Raffel *et al.*, 2020a fine-tuning.

Clustering	Mean Acc.	Similarity
RandExamples*-μ	64.8	0.82
RandTask*-μ	66.4	0.58
RandTask-μ	66.4	0.58
Embeddings*-μ	66.1	0.37
MBC*-μ	66.7	0.37
MBC-μ	67.1	0.27

Table 4.2 Analysis for different task clustering approaches: random tasks clusters, random examples, embedding-based clustering and model-based clustering. '*' means we train these clustering approaches with one epoch to save computation. We also report average pairwise similarity of cluster adapters.

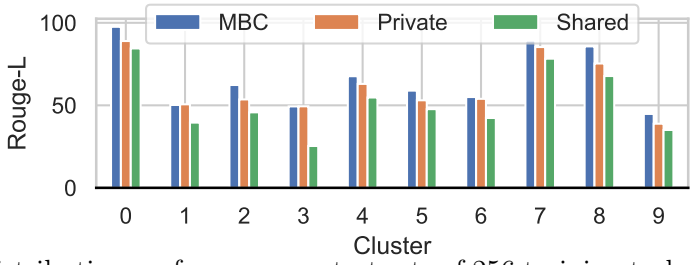


Figure 4.4 In-distribution performance on test sets of 256 training tasks for Shared, Private and MBC baselines on Phi2 library.

4.5.2 Supervised Adaptation

In Table 4.8 we present our detailed supervised adaptation results for Phi2, additionally in Table 4.4 we summarize the results for two additional models. First, for all models (Phi2, GPT-Neo and StableLM) we observe a significant performance boost coming from using Private and MBC libraries w.r.t. No Library, which optimizes a LoRA for each downstream task by starting from a random initialization, and Shared, which starts from the multi-task trained LoRA solution. Secondly, similarly to 0-shot results, we observe that MBC significantly boosts the performance with both Poly and μ routing: for Phi2 the performance of MBC-μ tops Private-μ. Additionally, we see that randomly grouping tasks RandomTask-Poly outperforms the non library baselines but doesn't quite match MBC-based clustering for all the models with exception of StableLM, where it performs on par.

The low performance of LoraHub can be attributed to the fact that LoraHub does not fine-tune the LoRA experts' weights but only their routing coefficients (due to gradient free optimization). Refer to App. 4.8.3 for more analyses on this point. Finally, MBC-μ performs similarly to MBC-Poly echoing results in (Caccia *et al.*, 2023).

Method	L	Rouge-L
No Library	1	75.5
Shared	1	75.8
Poly	10	75.8
Private-μ	256	76.9
MBC-μ	256	78.8
MBC-LoraHub	10	45.3
RandTask-Poly	10	76.7
MBC-Poly	10	78.9

Table 4.3 Rouge-L supervised adaptation results averaged over 12 held-out SNI tasks for different libraries built on top of Phi-2. Complete per-task results are presented in Table 4.8 in the appendix.

	StableLM 3B	GPT Neo 1.3B
Base	15.0	11.7
No Library	80.9	67.9
Shared	81.9	71.1
RandTask-Poly	82.3	70.9
MBC-Poly	82.3	71.7

Table 4.4 Adaptation results on SNI12 for StableLM and GPT-Neo.

4.5.3 Summary of Results

The first takeaway from our results is that, when appropriately routed (Private-↗), independently trained adapter experts can match zero-shot performance of multi-task training. This was a rather surprising result given that sharing data and re-training from scratch might be a hard constraint to meet in many realistic settings (Mireshghallah *et al.*, 2020). If sharing is possible, then clustering tasks by their similarity and averaging the LoRA adapters obtained through MBC (MBC-μ) constitutes a very effective model. Our zero-shot and supervised adaptation results underscore the importance of task-based clustering. For supervised adaptation, training both adapters are routing coefficients seem crucial. Although MBC outperforms random task clustering in most scenarios, the effectiveness of the latter merits further investigation. Our zero-shot routing MBC-↗ helped push performance further for Phi-2, outperforming 0.6% accuracy points over uniform averaging. Overall, if routing seems beneficial for large libraries of adapters, the gains for smaller libraries seems diminishing. This seems to be in contrast with standard MoE models, where routing is crucial (Jiang *et al.*, 2024). This could be due to the linearity of LoRA experts. For MLP experts (Fedus, Zoph, *et al.*, 2022a), uniform averaging may not be as effective as a simple routing strategy. We leave this investigation for future work.

4.6 Related Work

Multi-task learning involves training on a joint set of all tasks Caruana, 1997, potentially leading to performance degradation due to task interference Zhao *et al.*, 2018. Extensive literature studies how to partition learnable parameters into shared and task-specific Ding *et al.*, 2023; Strezoski *et al.*, 2019; Bragman *et al.*, 2019; Zaremoodi *et al.*, 2018; Wallingford *et al.*, 2022; Fifty *et al.*, 2021. We operate in the parameter-efficient multi-task learning setting (Ponti *et al.*, 2023b; Vu *et al.*, 2021; Chronopoulou *et al.*, 2023; Pfeiffer, Kamath, *et al.*, 2020). Vu *et al.* (2021) train one prefix adapter (Li and Eisner, 2019) per task and learn to re-use them for other tasks based on the adapter similarities. `MBC` can be seen as an extension of this approach where we cluster tasks based on the weight similarity to ensure more transfer during multi-task pre-training.

Mixture of experts (MoEs), when coupled with sparse routing, are notable for augmenting model capacity with minimal computational overhead Fedus, Zoph, *et al.*, 2022a. Among the most important differences in this work i) adapter experts are not trained during base model pre-training, ii) they are parameter-efficient and iii) they correspond to models tailored to specific tasks instead of being opaque computation units whose specialization is not easily interpretable (Jiang *et al.*, 2024). Regarding ii), recent work by Caccia *et al.* (2023), Ponti *et al.* (2023b), and Ostapenko *et al.* (2023) investigate the effectiveness of densely routed adapter experts trained end-to-end with an expert library for MTL fine-tuning. For expert aggregation, we employ parameter-space weighted averaging Wortsman *et al.*, 2022 with weights induced by a learned router, a technique akin to those in previous works Ostapenko *et al.*, 2023; Zadouri *et al.*, 2023. Several recent works have also proposed techniques for learning how to route queries to specialized pretrained open-source LLMs Lu *et al.*, 2023; Shnitzer *et al.*, 2023.

Model ensembling techniques aim to enhance model robustness and generalization by integrating multiple distinct models. Parameter space averaging of independent models Frankle *et al.*, 2020; Wortsman *et al.*, 2022 serves as an efficient ensembling method for full models Ilharco *et al.*, 2022; Ainsworth *et al.*, 2022 and adapters Zhang, Chen, Liu, *et al.*, 2023, requiring only a single forward pass through the model, unlike output space ensembling Dietterich, 2000; Breiman, 1996, that requires many forward passes. Efficient output ensembling techniques that can be applied in conjunction with our work are in Wen *et al.*, 2020. Similarly, Pfeiffer, Kamath, *et al.* (2020) proposes ensembling bottleneck style adapters with the subsequent fine-tuning step. In contrast to the existing works that usually study the merging of a small number of models (typically ≤ 10), here we tackle the problem of ensembling of over 200 experts.

Data Clustering for LMs have been proposed to improve performance and decrease task interference Fifty *et al.*, 2021; Gururangan *et al.*, 2023. These methods include clustering using similarities computed by tf-idf and neural embeddings, K-means clustering with balanced linear assignment, and soft clustering with GMMs (Gross *et al.*, 2017; Chronopoulou *et al.*, 2023; Chronopoulou *et al.*, 2021; Gururangan *et al.*, 2023; Duan *et al.*, 2021; Caron *et al.*, 2018). Recent work by Zhou, Xu, *et al.* (2022) observes the potential of adapter parameters as effective task embeddings for clustering purposes, a

concept we leverage in this work. A similar observation, but regarding task gradients, has been made by Vu *et al.* (2020b).

Building libraries of composable experts has been envisioned in several previous works. Beck *et al.* (2021) and Poth *et al.* (2023) orchestrated a framework for assembling diverse adapters, offering flexibility in both training and inference. Most related to this work, Huang *et al.* (2023) build LoRaHub, a library of task-specific LoRas that can be combined for few-shot generalization. We extend and complement these works by i) analyzing performance with different libraries of experts, ii) proposing novel methods to build a library, and iii) proposing techniques for zero-shot post-hoc routing independently trained adapters.

4.7 Conclusion

We investigate how to build and re-use a library of adapters "end-to-end". We show the potential achievable by re-using independently or partially independently trained adapters with a zero-shot routing strategy. Overall, we investigate the strategic, modular augmentation of smaller (language) models, offering a promising direction for research that prioritizes efficiency, flexibility, and performance.

The current investigation focuses on LoRA adapters. We are excited by the exploration of a heterogeneous "universe" of adapters (including soft and hard prompts (Lester *et al.*, 2021; Wen *et al.*, 2023), MLPs (Houlsby *et al.*, 2019b), etc.) and combinations thereof. Different adapter types might be more suited for different capabilities (reasoning, knowledge, etc.) The scalability of our approach to a broader range of tasks and the optimization of routing mechanisms for even greater accuracy remain for further investigation.

4.8 Appendix

4.8.1 Analyzing $\|AB^Tv\|_2$ for in-distribution and out-of-distribution samples

In this section, we analyze whether the motivation behind `Arrow` routing holds in practice. Recall that at each layer, `Arrow` routing initializes prototypes in the linear router for expert i with the unit vector v_i maximizing $\|AB^Tv\|_2$. Concretely, we hypothesize that for a hidden activation h computed from $x \in \mathcal{D}_i$, we have $\|A_iB_i^Tv\|_2 > \|A_jB_j^Tv\|_2$, for experts i, j. In other words, the norm of the linearly transformed prototype will be higher under the expert belonging to the same task as the input h.

To test this hypothesis, we run the following experiment. Let h_l denote the input to the expert at layer l, and $(AB^T)_l^i$ denote the linear transformation of expert i at layer l. We first sample 5000 examples from the multitask dataset. Then, for a given input $x \in \mathcal{D}_i$ at each layer l, we compute both $\|(AB^T)_l^i \cdot h_l\|_2$ and $\|(AB^T)_l^j \cdot h_l\|_2$ where j is another randomly sampled expert such that $i \neq j$. We then compute the average norm ratio r across all layers, i.e.

$$r = \sum_l^L \frac{1}{L} \frac{\|(AB^T)_l^i \cdot h_l^i\|_2}{\|(AB^T)_l^j \cdot h_l^i\|_2}.$$

Note that the random expert j is sampled at every layer, and the output of the in-distribution expert is propagated to the next layer. As such, $r > 1$ indicates that on average, the in-distribution expert produces a higher norm output, which would validate the use of the norm-maximizing initialization that `Arrow` routing uses. In figure 4.5, we see that for all the points considered, this ratio is positive, indicating that in-distribution experts tend to be maximize the norm of the linearly transformed input.

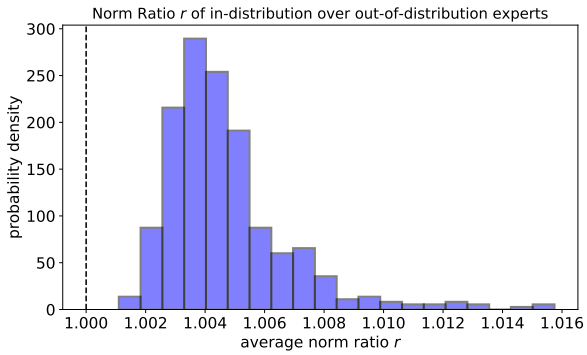


Figure 4.5 Histogram of the ratios r computed over 5000 samples.

	Library	Routing	L	piqa	boolq	wgrande	hswag	arcE	arcC	HE	oqa	bbh	mbpp	Acc.
Phi2 (2.8B)	Base	-	-	79.2	82.7	75.7	72.5	77.5	52.9	45.1	49.8	48.0	56.0	63.8
	FullFT	-	-	80.3	80.8	77.0	73.2	83.5	57.9	50.0	48.0	47.7	57.2	65.6
	Shared	-	1	80.4	82.4	76.6	73.4	83.2	55.8	46.3	50.4	48.4	58.4	65.5
	Poly	μ	1	80.6	82.3	76.7	71.7	82.7	55.3	48.2	50.4	49.8	59.1	65.7
	MHR	μ	1	80.1	83.0	77.1	70.4	83.2	55.5	46.3	53.4	52.0	58.0	65.9
	Private	μ	256	79.5	83.2	76.0	73.1	81.4	54.5	43.9	47.8	48.5	59.9	64.8
	Private	TP	256	79.5	83.4	76.9	73.1	83.0	55.4	47.6	48.8	49.4	59.5	65.7
	Private	CM	256	79.5	83.3	76.5	73.1	81.7	55.2	47.0	48.4	53.0	59.1	65.7
	Private	↗	256	79.6	83.4	76.6	73.2	82.8	56.3	48.2	50.8	48.5	60.7	66.0
	MBC	μ	10	80.3	85.1	77.3	73.1	84.3	57.7	48.8	50.2	51.6	62.3	67.1
	MBC	TP	10	80.2	84.7	78.2	73.1	84.3	57.6	50.0	50.4	47.1	62.3	66.8
	MBC	CM	10	79.8	84.8	77.7	73.1	84.5	58.0	50.6	50.8	50.0	61.	67.1
	MBC	↗	10	79.7	84.8	77.7	72.9	84.8	57.2	51.8	52.0	53.3	63.0	67.7
	Private	1-Oracle	256	79.7	84.4	77.0	71.3	85.2	56.7	52.8	48.2	48.0	61.1	66.6
	Private	n-Oracle	256	80.9	85.3	78.1	75.2	86.1	58.5	53.0	53.2	49.7	61.8	68.2
StableLM (3B)	Base	-	-	78.2	73.1	66.6	73.7	59.6	41.5	18.3	37.6	34.7	32.3	51.6
	Shared	-	1	79.4	80.3	68.0	71.3	74.7	42.1	11.6	38.0	38.3	21.0	52.5
	Private	μ	100	79.1	76.2	67.6	74.4	72.7	43.9	18.3	42.4	39.6	33.9	54.8
	Private	TP	100	79.3	75.2	65.6	74.1	76.6	45.3	14.0	41.8	35.2	24.5	53.2
	Private	↗	100	79.4	77.2	66.6	74.9	74.8	46.1	17.1	41.8	38.9	35.0	55.2
	MBC	μ	10	80.3	80.3	67.6	74.8	76.6	47.4	15.2	42.8	39.0	35.0	55.9
	MBC	TP	10	80.4	80.2	68.3	74.7	76.6	47.1	16.5	42.6	38.0	34.6	55.9
	MBC	↗	10	80.5	80.5	68.0	74.8	76.3	46.9	15.9	43.2	39.9	33.5	55.9

Table 4.5 **0-shot** results for `Phi-2` and `StableLM` base models. L denotes the library size. We only train 100 experts for StableLM for computational reasons in the `Private` scenario. The best results are underlined. Here `(ma)` stands for "Merging After" and refers to aggregation strategy where instead of merging LoRA's A and B separately, we merge them after performing the outer product, resulting in a higher maximal rank of the resulting weight matrix as discussed in Ostapenko *et al.*, 2023.

4.8.2 Zero-shot routing extended results

In Tab. 4.5 we present the extended view of the zero-shot routing results. We also include merging variant that merges the LoRA aprameters after performing the outer produce. This variant is denoted with `(ma)` – "Merging After". Formally, this variant modifies base model as $h = Wx + s \cdot \overline{AB}x$, , where $\overline{AB} = \frac{1}{L}\sum_{i=1}^{L} A_i B_i^\top$. As discussed in Ostapenko *et al.*, 2023, merging after the outer product requires more memory, since the weight matrix has to be computed explicitly. However, it allows for a the maximum rank of the resulting matrix $\overline{AB}$ to be bounded by the sum of the ranks of the individual $A_i B_i^\top$ summands (which is $L \cdot r$), whereas in case of separate merging of A and B the resulting rank is always r. We observe that this strategy does not result in any improvement for a compressed `MBC` μ `(ma)` library, yet leads to a 1 rouge point improvement for the `Private` library highlighting the effectiveness of the `(ma)` approach for larger libraries.

4.8.3 Few-shot adaptation

We apply some of the proposed methods to a data scarce setting with up to only 0.5% of the original training data per task (approx. 40 examples per task). We show the results in Table 4.6. Even in this setting gradient based method `MBC-Poly` considerably outperforms

LoraHub, where the LoraHub is given compute equivalent to training gradient based methods on full dataset. Additionally, we observe that MBC-PolyZ, a method similar to MBC-Poly that only updates the routings and not the expert's weights, performs similarly to LoraHub. Interestingly, when data amount is lowered, the oerfromance of MBC-PolyZ is reduced by a relatively smaller margin than MBC-Poly which can be explained by a smaller amount of updated parameters.

Method	L	SNI Tasks												Rouge-L
		202	304	614	613	362	242	1728	1557	035	1356	039	1153	
Full data														
MBC-LoraHub	10	41.5	21.9	37.4	17.5	78.1	68.3	48.0	82.0	62.6	21.2	33.5	31.1	45.3
MBC-Poly	10	96.9	84.4	67.2	53.9	96.4	97.8	60.2	87.9	91.3	29.4	81.7	99.7	78.9
MBC-PolyZ	10	37.7	27.9	36.2	12.6	75.9	74.4	48.7	81.3	58.9	22.5	36.1	31.1	45.3
10%														
MBC-Poly	10	89.6	53.3	64.5	44.5	93.5	98.5	58.5	75.7	87.3	27.2	65.6	66.8	68.8
MBC-PolyZ	10	34.3	27.5	36.2	12.4	76.5	74.3	47.5	86.2	57.9	22.7	35.3	31.4	45.2
5%														
MBC-Poly	10	87.0	43.0	61.3	41.7	92.0	95.2	55.3	77.3	89.0	25.4	59.1	47.8	64.5
MBC-PolyZ	10	32.6	28.6	36.0	13.0	76.4	73.9	47.3	86.3	57.7	22.7	36.6	31.0	45.2
0.5%														
MBC-Poly	10	49.7	30.4	43.7	20.3	77.3	78.4	48.2	86.5	72.2	23.2	43.0	29.1	50.2
MBC-PolyZ	10	32.0	27.4	34.3	12.6	77.6	78.1	47.2	86.5	53.2	22.2	37.0	29.1	44.8

Table 4.6 Rouge-L score for Phi2 model after adaptation with different portions of data per task ranging from full dataset down to 10% and 5% of data per task.

Method	PIQA	Boolq	Wgrande	Hswag	Arc-E	Arc-C	HE	OQA	BBH (200b)	Mbpp	AVG
Base	70.7	61.4	54.7	47.4	44.4	27.4	4.9	32.6	35.0	5.8	38.4
Private-μ	70.9	61.0	54.9	48.3	50.8	27.9	6.1	33.0	31.1	5.4	39.0
MBC-μ	71.3	62.9	55.0	48.6	59.6	28.9	4.3	33.2	31.0	5.4	40.0

Table 4.7 Our preliminary results on GPt-Neo 0-shot evaluation on 10 Flan tasks.

Method	L	SNI Tasks												Rouge-L
		202	304	614	613	362	242	1728	1557	035	1356	039	1153	
No Library	1	93.2	74.2	64.9	51.4	95.9	96.2	59.3	81.4	90.5	26.9	73	99.1	75.5
Shared	1	93.1	73.4	65.0	48.9	96.0	95.9	58.4	86.8	91.2	29.0	73.4	98.4	75.8
Poly	10	96.2	69.3	65.7	47.2	96.4	99.3	57.7	87.3	90.8	28.2	73.7	98.0	75.8
Private-μ	256	93.6	78.1	65.0	50.7	94.8	97.8	59.7	87.9	90.7	28.1	76.4	99.7	76.9
MBC-μ	256	96.4	83.2	67.6	53.5	96.2	98.0	60.5	88.2	90.7	29.8	82.3	99.5	78.8
MBC-LoraHub	10	41.5	21.9	37.4	17.5	78.1	68.3	48.0	82.0	62.6	21.2	33.5	31.1	45.3
RandTask-Poly	10	96.4	77.1	66.5	48.6	96.7	98.9	59.9	85.1	90.7	28.6	73.9	97.5	76.7
MBC-Poly	10	96.9	84.4	67.2	53.9	96.4	97.8	60.2	87.9	91.3	29.4	81.7	99.7	78.9

Table 4.8 Rouge-L supervised adaptation results for Phi-2 on 12 held-out SNI tasks given different libraries. LoraHub follows the original implementation and optimizes the weighting coefficients for the adapters in the library with a non-gradient based optimizer.

Method	L	SNI Tasks												Rouge-L
		202	304	614	613	362	242	1728	1557	035	1356	039	1153	
Base	-	4.0	3.3	26.4	3.5	16.2	32.5	35.2	62.5	54.2	12.8	8.2	7.6	22.2
Shared	1	38.3	17.9	36.4	11.5	77.2	39.4	45.8	84.5	40.7	21.5	34.3	24.1	39.3
Private-μ	256	10.6	16.1	35.6	9.6	64.8	58.2	42.6	72.2	61.7	17.5	25.1	24.0	36.5
MBC-μ	10	31.8	26.9	33.9	12.7	77.6	77.9	47.2	86.0	49.2	22.4	37.0	29.8	44.4
RandTask-μ	10	35.9	25.1	33.8	12.6	78.8	44.6	44.4	81.9	54.0	23.6	40.2	29.1	42.0

Table 4.9 Zero-shot performance on 12 SNI tasks for library built arount the Phi2 base model. Similar to the main paper, we observe a significant performance boost induced by applying MBC over both Shared, Private-μ as well as RandTask-μ baselines.

Part II

Information Propagation in Web Tracking

5 Privacy Lost in Online Education: Analysis of Web Tracking Evolution

5.1 Introduction

Privacy lost occurs when an individual's personal information or data is disclosed, shared, or accessed by others without their permission, which can result in various negative consequences, such as identity theft, financial fraud, damage to reputation, and discrimination. To investigate these issues, researchers may examine the historical practice of third-party web tracking, as described by Lerner *et al.* (2016). Third-party web tracking involves third-party entities, such as advertisers, social media widgets, and website analytics engines, that are embedded in the first-party sites that users directly visit and are capable of re-identifying users across domains while they browse the web. The proliferation of web tracking has spurred a growing body of research in the computer security and privacy community, which seeks to understand, quantify, and counteract these privacy risks posed by tracking companies compiling lists of websites that users have visited (Lerner *et al.*, 2016; Agarwal and Sastry, 2022; Amos *et al.*, 2021).

As the education industry transitions from traditional offline models to online or hybrid models, the need for privacy protection on educational websites is becoming increasingly prominent. This issue is crucial because the loss of privacy on educational websites can undermine the fundamental principles of privacy and security that are essential for individuals to feel safe and empowered while using the internet for educational purposes. By protecting users' privacy, educational websites can promote trust, openness, and responsibility, which are essential for fostering a positive and inclusive online learning experience. Therefore, several researchers have started studying the practice of web tracking in educational websites (Vlajic *et al.*, 2018; Jordan, 2018; Saxena *et al.*, 2019; Jarke and Breiter, 2019).

To deepen our comprehension of the nature and progression of tracking on educational websites, we propose an analytical framework that enables a comparative analysis of tracking on a specific type of site (in this case, education) in relation to a control group of sites with comparable traffic levels but of different types. The framework involves three steps: we construct a sample of educational websites, and a control sample of non-educational websites that have similar levels of traffic (Section 5.3.1). We then retrieve the historical websites from the Internet Archive's Wayback Machine[1] for both samples. Third, we scan the HTML file snapshots of the collected websites using the

[1] https://archive.org/

Wayback Machine (Section 5.3.2), and extract third-party trackers embedded in the HTML files (Section 5.3.3).

We aim to answer the following research questions, which we present along with our main findings:

RQ1: How has the use of trackers on educational websites evolved from 2012 to 2021?

In Section 5.4.1, we examine the average number of trackers from 2012 to 2021 and observe a general trend of tracker growth. Until 2018, both educational and non-educational sites sees substantial growth, but they diverge around the time of the introduction of the GDPR in 2018: at this point there is a minor drop in tracking on non-educational sites, which is not seen on educational sites, where the development merely stagnates.

RQ2: How does the evolution of the use of trackers differ between educational and non-educational websites?

Section 5.4.1 also addresses differences between educational and non-educational websites in the evolution of tracking between 2012 and 2021. The results show that despite the similarity of the underlying trend, the intensity of tracking has grown relatively more on educational sites and that the growth has not similarly reverted as on non-educational websites after the introduction of the GDPR. The results are further supported by a Wilcoxon signed rank (WS) test conducted in Section 5.4.2, demonstrating that the intensity of tracking on educational sites surpassed that of non-educational sites in 2017.

RQ3: Is there a qualitative difference in what kind of trackers are used on educational and non-educational websites?

The quantitative difference between tracking on educational and non-educational sites that we find in the average number of trackers also shows up in the different compositions of the portfolios of trackers found at the two types of sites. We substantiate this statistically by using the Kolmogorov-Smirnov test (KS) to compare the distribution of trackers in these two groups of sites. To investigate the source of these differences, Section 5.4.3 examines the occurrence of some of the most popular trackers, demonstrating that the use of Twitter, Youtube, and Facebook has evolved very differently between educational and non-educational websites. In addition, Section 5.4.5 compares the presence of trackers presenting particular categories, demonstrating that tracking related to enhancing customer interaction in particular seems to have become relatively more common on educational websites over the past few years.

Our contributions can be summarized in two main points: (I) We develop a list of both educational and non-educational websites to investigate the issue of *privacy lost* in online education. The complete code and dataset we compiled can be accessed at [2]. (II) We

[2] https://github.com/shuishen112/Privacy_Lost.git

conduct a quantitative and qualitative analysis of third-party tracking on educational websites, focusing on third-party services from 2012 to 2021. Our findings highlight potential concerns regarding the autonomy and fairness of education.

5.2 Related Work

Tracking through third-party cookies and scripts has been extensively studied from various perspectives. A significant portion of this research has focused on mapping the prevalence of trackers across samples of websites, such as those found on the Alexa top lists (Acar *et al.*, 2014; Englehardt and Narayanan, 2016; Mathur *et al.*, 2019; Englehardt and Narayanan, 2016). Other studies have investigated tracking on different platforms, including the mobile ecosystem (Binns, Lyngs, *et al.*, 2018; Binns, Zhao, *et al.*, 2018).

Karaj et al. (Karaj *et al.*, 2018) proposed a method for measuring web tracking using a browser extension, resulting in a dataset covering 1.5 billion page loads collected over 12 months period from real users. Krishnamurthy and Wills (Englehardt and Narayanan, 2016) presented a dataset on tracking based on a crawl of the top 1 million websites. They developed an open-source web privacy measurement tool called OpenWPM, which allows researchers to detect, quantify, and characterize emerging online tracking behaviors. Our work is related to several general areas:

Historical web tracking. Krishnamurthy and Wills provided early insights into web tracking, demonstrating the evolution of third-party organizations between 2005 and 2008 (Krishnamurthy and Wills, 2009). Lerner et al. presented longitudinal measurements of third-party web tracking behaviors from 1996-2016 (Lerner *et al.*, 2016). Karaj et al. conducted a large-scale and long-term measurement of online tracking based on real users (Karaj *et al.*, 2018). Agarwal and Sastr analyzed the top 100 Alexa websites over 25 years using data from the Internet Archive, studying changes in website popularity and examining different categories of websites and their popularity trends over time (Agarwal and Sastry, 2022). Amos et al. curated a dataset of 1,071,488 English language privacy policies spanning over two decades and encompassing more than 130,000 different websites (Amos *et al.*, 2021).

Web tracking after GDPR. Numerous studies have investigated web tracking following the implementation of the GDPR (General Data Protection Regulation) in the EU in May 2018, which imposed constraints on online data collection. These studies generally indicate a pattern of diminished tracking activity (Dabrowski *et al.*, 2019; Sørensen and Kosta, 2019; Sanchez-Rola *et al.*, 2019), but they also reveal that most sites appear unable or unwilling to fully comply with regulations (Kretschmer *et al.*, 2021; Urban *et al.*, 2020; Hu and Sastry, 2019), and tracking companies can still likely monitor user behavior (Sanchez-Rola *et al.*, 2019).

Web tracking in educational websites. A body of research focuses explicitly on educational websites, which are known to have a higher incidence of tracking technology than sites aimed at minors (Vlajic *et al.*, 2018). In particular, university websites exhibit

a substantial prevalence of major tracking companies (e.g., Google, Facebook) (Jordan, 2018). While several recent papers discuss the implications of tracking on educational websites, there seems to be a lack of studies investigating third-party tracking on substantial samples of educational websites post-2018 or examining the development of tracking over time for these websites.

5.3 Data collection

We provide a concise overview of our data collection framework, which comprises three main components. Firstly, we discuss the process of gathering educational and non-educational websites, as detailed in Section 5.3.1. Secondly, we present the methodology for scanning historical snapshots from Internet Archive's Wayback machine, which is described in Section 5.3.2. Finally, we discuss the approach for extracting third-party trackers from HTML files, which is outlined in Section 5.3.3.

5.3.1 Collecting Websites

To understand the evolution of web tracking in educational websites, we compare them to a control set of non-educational websites to see whether there are any changes related to education in particular. The comparison set is explicitly controlled for popularity so that the two sets consisting of educational and non-educational websites have equal rank distribution. The studied websites must also have available historical data stored in Internet Archive.

We construct the two rank-matched sets of educational and non-educational websites as follows:

Step 1. We extract the educational websites from DMOZ[3]. DMOZ is a large communally maintained open directory that categorizes websites based on webpage content, and we use the DMOZ classification of educational websites. There are 146,941 websites in the DMOZ database labeled as educational websites.

Step 2. Next, we limit the set of educational websites to those occurring on the Open PageRank Initiative[4], which maintains a list of the top 10 million websites ranked based on their Open PageRank. There are 55,390 educational websites present among the top 10 million. This filtering is done so that we can create a comparable control set.

Step 3. We use the Internet Archive's Wayback Machine[5] for archived data. Therefore, the set of educational websites is further limited to those with at least one snapshot per year in every year from 2012 through 2021 to ensure that annual comparisons are balanced. This results in 17,975 educational websites altogether.

[3] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OMV93V
[4] https://www.domcop.com/top-10-million-websites
[5] https://archive.org

Step 4. Based on the list of educational websites from Step 3, we construct a set of non-educational websites with rank (Open Pagerank) distribution matching educational websites. Starting with the educational website of the highest rank, this is done recursively by choosing for each educational website a non-educational website that satisfies the following three conditions:

For instance, if there are two educational websites of ranks 19 and 20, then ranks 21 and 22 would be chosen to the control set of non-educational sites, provided that they are not educational websites and have archived versions available. We study the rank gap (The rank of non-educational websites minus the rank of matching educational websites) distribution. The mean rank gap is 2.86, while the maximum gap is 255. We also find that 99% of the rank gap is below 16.

5.3.2 Scanning the Historical Snapshot

There are two primary methods for scanning historical snapshots: the Wayback CDX Server[6] and the waybackpy Python library [7]. The Wayback CDX Server is a standalone HTTP servlet that serves the index used by the Wayback Machine to search for captures. The second method involves using the WaybackMachineCDXServerAPI provided by the waybackpy library to retrieve historical snapshots at specific times. For our research, we opted to use the Wayback CDX Server as our scanning method.

5.3.3 Extraction of Third-party Trackers

Each website is examined for requests to other URLs initiated during the website's loading. These requests will always be embedded in three HTML-elements: *"iframe"* , *"script"* and *"img"*. We only consider the requests generated automatically without user action. That is why we omitted the *"a"*-element[8].

The list of third-party services (TPSs) was compiled by extracting all URLs found in the three HTML elements mentioned earlier across the entire dataset. For each website and URL, we checked whether the main domain of the linked URL (e.g., 'google' in 'www.google.dk') differed from the main domain of the website. If the domains were different, the URL was considered a 'third party' and the domain (e.g., 'google') along with the suffix (e.g., 'dk') were added to the list of TPSs.

To clarify our terminology, we will use the term 'trackers' instead of 'third-party services' for the remainder of this paper. While many third-party services serve various functions, such as providing weather data or chat services, some are solely designed for tracking and provide data that is used for personalized banner ads. However, even third-party services that seemingly provide non-tracking functionality have the potential to gather

[6] https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server
[7] https://akamhy.github.io/waybackpy/
[8] A "ping"-attribute in the "a"-element allows requests to be made to multiple URLs without the user being aware of this, but there were no ping-attributes used in the data used in this study

valuable data from users, such as their timestamped IP addresses and the websites they visit when the third-party service is activated. This information may be used by the third-party provider or sold to data brokers, or both. As all third-party services can track users, we refer to all such services invoked through websites as trackers (Libert, 2015).

We utilized the trackers list[9], which covers the period between May 2017 and August 2022(Karaj *et al.*, 2018). The trackers on the list are ranked according to their *tracker reach*, which is a metric defined in the aforementioned paper (Karaj *et al.*, 2018). It should be noted that each tracker corresponds to multiple tracker domains. For instance, Doubleclick is associated with three tracker domains: '2mdn.net', 'doubleclick.net', and 'invitemedia.com'. In total, the tracker list comprises 1,285 tracker domains.

5.4 Analysis and Discussion

Our analysis focuses on the changes in web tracking between 2012 and 2021, with a particular emphasis on the qualitative and quantitative differences in tracker usage between educational and non-educational websites.

5.4.1 Evolution of tracker domains per website

To begin our analysis, we computed the average number of trackers per website for each year. Figure 5.1 displays the trends in tracker usage on educational and non-educational websites between 2012 and 2021. In general, a striking 94.5% increase in the average number of trackers on educational websites was observed, while the control group experienced a comparatively modest 31.3% increase from 2012-2021. Specifically, when observing the trends of growth each year, we notice a plateau or slight reversal in growth occurring after 2017. Notably, the vertical line in Figure 5.1 represents the formal implementation of GDPR in 2018. It is interesting to observe that the number of trackers on non-educational sites experienced a slight decline, whereas tracker usage on educational sites appeared to taper off around the same time.

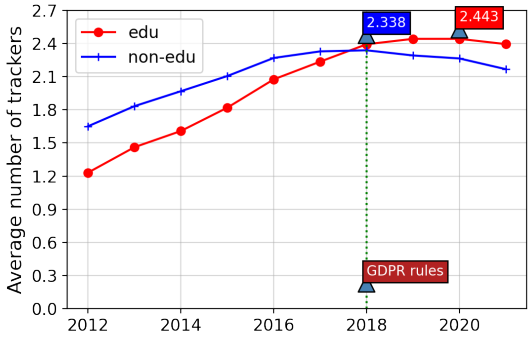
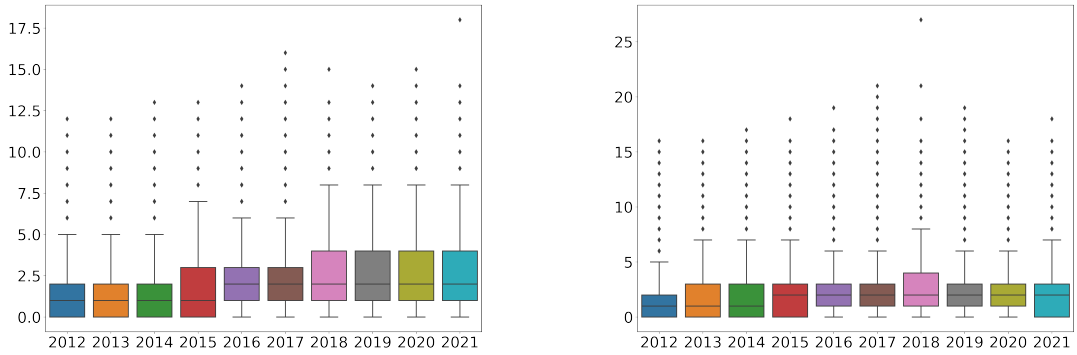


Figure 5.1 Evolution of the average number of tracker domains per webiste.

Figure 5.2a shows a box plot of the number of trackers per year for educational websites. The plot suggests that the evolution in the average number of trackers after 2018 observed in Figure 5.1 is driven by an increased dispersion in tracking across distinct educational sites, as the third quartile increases from 2017-2018, while the median (the horizontal

[9]https://whotracks.me/trackers.html

line in each box) remains almost the same in the period 2017-2021. As a comparison, Figure 5.2b shows a boxplot for non-educational websites. The plot also suggests the evolution in the average number of trackers in Figure 5.1. The third quartile increased from 2017-2018 but has dropped since 2019. Especially, the first quartile decrease in 2021.



(a) educational websites. **(b)** non-educational websites.

Figure 5.2 Number of trackers in each year for educational websites and non-educational websites.

According to the findings, it appears that users who browse educational websites are at a higher risk of having their online behavior information collected and potentially utilized by various services and websites. This disparity between educational and non-educational websites highlights the potential inadequacy of the GDPR in addressing privacy concerns specific to educational sites.

5.4.2 The number of trackers on educational and non-educational websites

The tracking trends presented in Section 5.4.1 are purely descriptive, hence we conducted a statistical test to determine if there is a significant difference between educational and non-educational websites. Specifically, we performed a matched-pairs Wilcoxson signed rank(WS)[10] test to evaluate if the medians of the educational and non-educational samples are different for each year. This test is appropriate for paired data, which is the case for our study due to the rank-based construction of the data, and does not make any assumptions about the underlying distributions, making it a non-parametric test.

Table 5.1 Summary of the WS-test showing differences in each year, N = 17975.

	WS-test	
Year	p-value	Z
2012	5.2×10^{-86}	-19.66
2013	6.8×10^{-55}	-15.6
2014	1.6×10^{-46}	-14.32
2015	6.7×10^{-28}	-10.95
2016	7.1×10^{-11}	-6.52
2017	6.6×10^{-2}	-1.84
2018	1.5×10^{-5}	-4.33
2019	8.7×10^{-18}	-8.59
2020	3.2×10^{-20}	-9.21
2021	2.3×10^{-30}	-11.45

[10] http://www.biostathandbook.com/wilcoxonsignedrank.html

The input of the WS test is the number of tracker domains for each educational and non-educational website. The results of the tests are summarized in Table 5.1; as usual, small p-values indicate statistical significance; for all years, except 2017 $p < 0.01$ for both tests. The sole exception is 2017, where $p > 0.05$ for the WS-test, consistent with the prevalence curves (see Figure 5.1) crossing that year.[11]

5.4.3 Evolution of usage rate for the most common trackers

To understand how tracking development differs between educational and non-educational sites, we compare how the ten most commonly occurring trackers have changed during the measurement period. We compute the usage of trackers based on the usage rate and select the top ten most used trackers in educational websites in 2012. The top ten trackers are on the vertical axis in Figure 5.3.

We define the *usage rate* of each tracker as $f(t) = \frac{\mathrm{N}(t)}{\mathrm{N}(w)}$ where $\mathrm{N}(t)$ is the total number of websites where tracker t occurs, and $N(w)$ is the total number of websites in the sample. We calculate the relative increase I in usage rates of the top ten trackers most common on educational websites from 2012 to 2012 as $I = \frac{f(t)_{2021} - f(t)_{2012}}{f(t)_{2012}}$. The relative change of usage rate is shown in Figure 5.3. We observe that the overall usage rate increased for the five top trackers on educational websites, including the social media sites Twitter, Facebook, and Youtube. It also increased for two Google-related trackers.

It decreased for five trackers, including Twimg (operated by Twitter) decrease by 66.4%, Addthis (-46.9%), Google-analytics (-32.5%), Adobe (-31.6%) and Googlesyndication (-1.1%) on educational websites. For non-educational websites, the usage rate increased only for the three Alphabet-operated trackers (Youtube, Googleapis and Google) and saw the largest decrease for Twimg, by 78.4%.

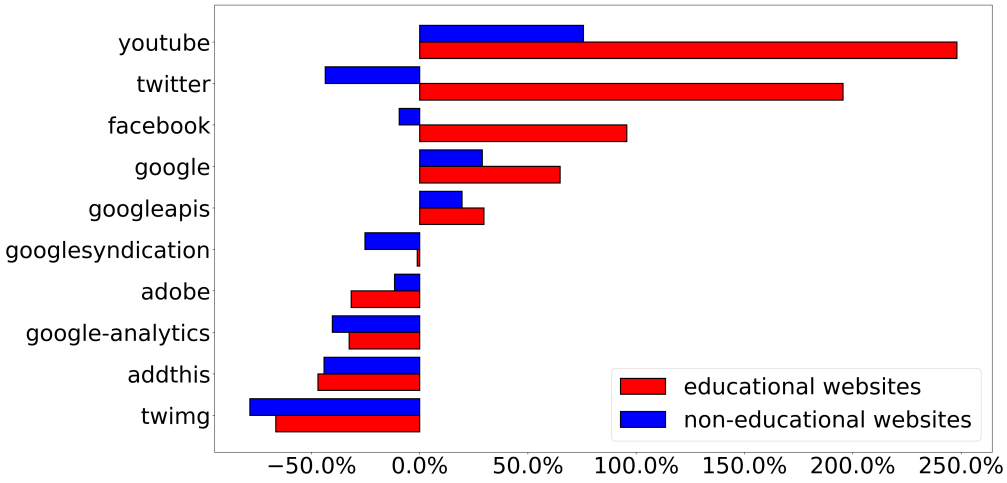


Figure 5.3 Top usage rate change of trackers. X-axis is the percentage change of usage rate. Y-axis is the tracker name.

[11] Note that the level of statistical significance in the test means that correcting the alpha level for multiple comparisons does not alter the finding. This also holds for table 2.

Notably, the use of Twitter, Youtube, and Facebook has evolved very differently between educational and non-educational websites. All three have an increased presence on educational sites, whereas their use has declined (Twitter and Facebook) or grown much slower (Youtube) on non-educational sites. The presence of trackers from these companies on educational sites helps target ads at the users when they visit the platforms and can also help educational sites to advertise their services to new, potential users with profiles similar to their existing users.

5.4.4 Distribution of trackers in educational and non-educational sites

The results in Section 5.4.1 show that the level of tracking differs significantly between educational and non-educational sites. We will investigate if the difference also relates to the composition of trackers used on the two groups of sites and differences in intensity. As stated in 5.3.3, all third-party services may collect data that can be used for tracking. However, the value proposition to the website owner differs between different services since they provide various functionalities to the site. Therefore, an analysis of the other functionalities also indicates what the site owner has sought to gain from embedding the service (or tracker), irrespective of the tracking of user behavior it enables. This analysis will, in turn, show if educational sites have followed a different path in integrating trackers than other sites.

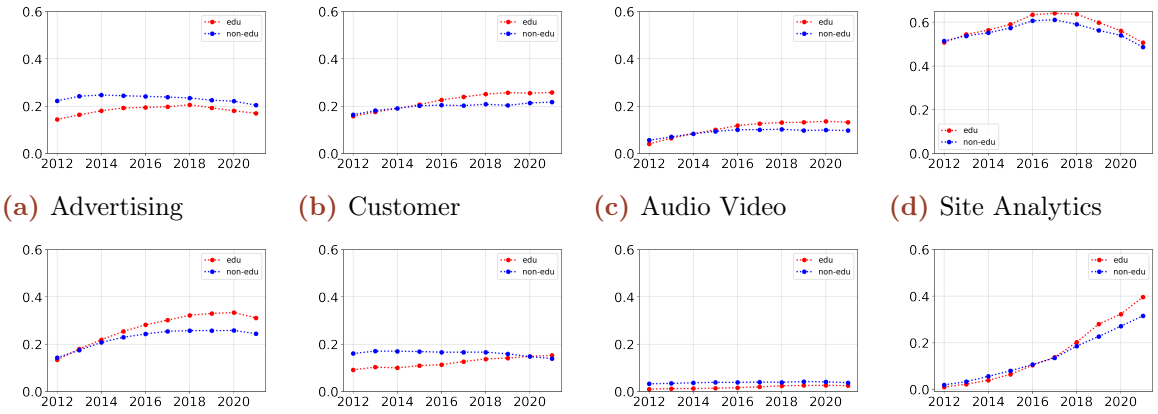
For each year, we employ the two-sample Kolmogorov-Smirnov(KS) test–the standard non-parametric test for comparing distributions–to test whether the educational, resp. non-educational samples are drawn from the same underlying distribution. The test is suitable for paired data, similar to the WS test reported before. As indicated in Table 5.2, there is a significant difference in the distribution of trackers found on the two groups of sites in each year between 2012 and 2021. This indicates that in addition to the different quantitative trends, there appears to be a qualitative difference in the kind of trackers used on educational and non-educational websites.

Table 5.2 Summary of the KS-test showing differences in each year.

	KS-test	
Year	p-value	Statistic
2012	4.7×10^{-6}	0.25
2013	6.0×10^{-5}	0.22
2014	4.1×10^{-5}	0.23
2015	2.7×10^{-3}	0.18
2016	1.4×10^{-3}	0.19
2017	1.4×10^{-3}	0.19
2018	2.8×10^{-4}	0.21
2019	1.2×10^{-3}	0.19
2020	4.4×10^{-4}	0.2
2021	3.3×10^{-3}	0.17

5.4.5 Evolution of different categories of trackers

To understand how the overall differences in tracker distribution identified in the previous section relate more closely to different purposes of web functionality, we look at the



(a) Advertising **(b)** Customer **(c)** Audio Video **(d)** Site Analytics

(e) Cdn **(f)** Social Media **(g)** Misc **(h)** Essential

Figure 5.4 Evolution of different categories of trackers from 2012 to 2021.

changes across different types of trackers. Since no exhaustive categorization of trackers exists, we use the tracker typology made available by the WhoTracksMe initiative in June 2022, which to our knowledge, is the most comprehensive and up-to-date list, that is openly available [12].

This typology matches 1285 trackers in our dataset. While this represents only a subset of the total number of trackers, the list coincides with 213 of the 1285 most common trackers in our analysis. In the following, we examine the distribution of trackers across categories but only do so for the subset of the most common trackers. The WhoTracksMe list categorizes most common trackers into one of *Site Analytics*, *Customer Interaction*, *Advertising*, *Cdn*, *Social Media*, *Audio Video Player*, *Essential*, *Misc*. A more detailed explanation of the eight tracker categories is found in Appendix 5.6.1.

For each category c of trackers, we calculate usage rate $f(c)$ as $f(c) = \frac{\mathrm{N}(c)}{\mathrm{N}(w)}$ where $\mathrm{N}(c)$ is the number of websites this type of trackers occur on, and $\mathrm{N}(w)$ is the total number of websites in the sample. We calculated the usage rate $f(c)$ for each category c of trackers as $f(c) = \frac{\mathrm{N}(c)}{\mathrm{N}(w)}$, where $\mathrm{N}(c)$ is the number of websites on which this type of tracker occurs, and $\mathrm{N}(w)$ is the total number of websites in the sample. The evolution of different tracker categories in Figure 5.4 indicates that, when comparing the levels in 2012 and 2021, educational sites have increased their use of all trackers except for *Site Analytics*. Even though *Advertising* and *Cdn* trackers have dropped slightly since their peak, they are still at a higher level than in 2012. In contrast, the usage rate for *Site Analytics*, *Advertising*, and *Social Media* is lower in 2021 than in 2012 for non-educational sites.

When comparing the two groups, two significant trends are apparent. Firstly, educational sites exhibit higher growth in the use rate of trackers related to interactive site features and audio-visual content. For instance, the *Audio Video Player* category witnessed a growth of 225.0% on educational websites from 2012-2021 compared to 73.1% for non-educational websites. While the increase slowed down in 2018 or was even reverted for non-educational websites, it has increased again since 2019 and decreased since

[12] https://whotracks.me/trackers.html

2020. Additionally, *Cdn* services and *Customer Interaction* trackers have become more commonly used and grown more on educational sites than non-educational sites during the period. The increase from 2012-2021 was 63.7% for educational websites and 32.5% for non-educational websites in the case of *Customer Interaction* trackers. This growth in interactive features and audio-visual content on educational sites is consistent with the evolution of online learning, which has become more interactive and audio-visually engaging over the years (Elisabeta and Alexandru, 2018). Moreover, these trends make the sites more bandwidth-consuming, which is also in line with the growth of *Cdn* services.

Second, the use of *Social Media* (increased by 66.2%) and *Advertising* (18.0%) related trackers grow for educational sites but display a net drop for non-educational sites. Compared to the level in 2012, the use of both trackers is higher in 2021, whereas both are lower on non-educational sites. For both categories, the educational sites begin at a lower level than the non-educational sites. In both categories, the difference becomes less pronounced over time, and for *Social Media* ends up at the same level. This indicates that purely commercial tracking on educational sites has evolved from being comparatively less common than on other types of sites to be similar. For both types of sites, the use of trackers for *Advertising* has gone down in recent years, but for educational sites, the peak is more recent (2018) than for non-educational sites (2014). For non-educational sites, this is consistent with the overall development in commercial tracking, which has seen a general trend toward concentration around a few major players. In 2012, the market for commercial tracking was less dominated by monopolies such as Alphabet and Meta than it has since become (Bilić and Prug, 2021). The market domination of fewer players is consistent with the falling trend in the use rate. For educational sites, the continued growth is consistent with them catching up to the market standard for commercial tracking, which is also suggested by the strong growth of trackers from the top market players observed in section 5.4.3.

5.4.6 Discussion

GDPR implementation in the EU influenced web tracking trends as expected, but its effect on educational websites was less than on non-educational ones. While non-educational sites saw a decrease in tracker use post-GDPR, tracker usage on educational sites not only increased but remained consistently higher. Commercial tracking, like ads and social media, is now nearly as common on educational sites as it is on non-educational ones.

Third-party tracking services offer functionalities like chat features on websites, while also collecting user data. The rise in tracking on educational websites means users are more likely to have their online behavior information gathered and possibly used across various platforms. Notably, this raises privacy concerns as the use of educational sites is often mandatory, making consent to tracking less meaningful. These sites are used across educational systems and for corporate training. The growing use of both commercial and other third-party trackers suggests increasing commercial exploitation of learning activities.

The trends of tracking we have identified also suggest that the associated business model(s) remain active and are of increasing relevance in the online education sector. Our paper particularly raises the concern whether the GDPR in its current form suitably addresses privacy issues related to websites whose use is not predominantly voluntary, such as educational websites, but also many other sites like public websites, where the trends of tracking form an important research question on its right and should be addressed in studies in the future.

Reviewing the results, we do not find convincing evidence that tracking on educational websites has been substantially impacted by the COVID-19 pandemic. Despite the fact that additional traffic to these types of sites during the lock-down periods would represent a valuable asset for site owners, no trends in the data meaningfully relate to this. This may be related to the fact that educational sites had already adjusted their portfolio of commercial tracking in particular to facilitate monetization of increased traffic, e.g. through re-targeting of potential students on social media.

5.5 Conclusions

In this paper, we present a framework for examining historical web tracking within a defined set of sites and apply it to a sample of educational websites. We constructed a sample comprising educational websites and a control group of non-educational websites that shared similar ranking positions. Utilizing the Internet Archive's Wayback Machine, we analyze 17,975 pairs of websites and their corresponding controls, spanning the period from 2012 to 2021. We observed a notable overall rise in tracking activities on educational sites.

Then we conduct a quantitative and qualitative analysis of third-party tracking on educational websites. We discover that the growth rate of educational websites has surpassed the control group from 2012-2021. Our investigation into the relative expansion of various tracker types suggests that the accelerated growth of tracking on educational websites may be attributed to the rising use of customer interaction, audio-visual content, and social media integration within these platforms.

Our analysis raises concerns about privacy and independence in education. Privacy issues in educational websites should be prioritized, as they can lead to unauthorized disclosure of confidential information, loss of trust, legal consequences, and intellectual property compromise. Furthermore, researchers may wish to analyze privacy lost in other areas, such as news or sports, from a historical perspective. Our framework offers a convenient solution for creating comparable websites and collecting historical third-party tracker data in these domains.

5.6 Appendix

5.6.1 Tracker Categories

Trackers differ both in the technologies they use, and the purpose they serve. Based on the the service they provide to the site owner, we have categorized the trackers in the following:

Advertising Provides advertising or advertising-related services such as data collection, behavioral analysis or re-targeting.

Customer Interaction Includes chat, email messaging, customer support, and other interaction tools

Essential Includes tag managers, privacy notices, and technologies that are critical to the functionality of a website

Site Analytics Collects and analyzes data related to site usage and performance. Social Media Integrates features related to social media sites

Audio Video Player Enables websites to publish, distribute, and optimize video and audio content

CDN (Content Delivery Network) Content delivery network that delivers resources for different site utilities and usually for many different customers.

Misc (Miscellaneous) This tracker does not fit in other categories.

Essential Includes tag managers, privacy notices, and technologies that are critical to the functionality of a website

Bibliography

6

Acar, Gunes, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz (2014). „The web never forgets: Persistent tracking mechanisms in the wild“. In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 674–689.

Acar, Gunes, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel (2013). „FPDetective: dusting the web for fingerprinters“. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1129–1140.

Achiam, Josh, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, *et al.* (2023). „Gpt-4 technical report“. In: *arXiv preprint arXiv:2303.08774*.

Acquisti, Alessandro, Stefanos Gritzalis, Costos Lambrinoudakis, and Sabrina di Vimercati (2007). *Digital privacy: theory, technologies, and practices*. CRC Press.

Adar, Eytan and Lada A Adamic (2005). „Tracking information epidemics in blogspace“. In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*. IEEE, pp. 207–214.

Agarwal, Vibhor and Nishanth Sastry (2022). „“Way back then”: A Data-driven View of 25+ years of Web Evolution“. In: *Proceedings of the ACM Web Conference 2022*, pp. 3471–3479.

Aghajanyan, Armen, Luke Zettlemoyer, and Sonal Gupta (2020). „Intrinsic dimensionality explains the effectiveness of language model fine-tuning“. In: *arXiv preprint arXiv:2012.13255*.

Ainsworth, Samuel K, Jonathan Hayase, and Siddhartha Srinivasa (2022). „Git re-basin: Merging models modulo permutation symmetries“. In: *arXiv preprint arXiv:2209.04836*.

Akkus, Istemi Ekin, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke (2012). „Non-tracking web analytics“. In: *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 687–698.

Amos, Ryan, Gunes Acar, Eli Lucherini, Mihir Kshirsagar, Arvind Narayanan, and Jonathan Mayer (2021). „Privacy policies over time: Curation and analysis of a million-document dataset“. In: *Proceedings of the Web Conference 2021*, pp. 2165–2176.

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell, and Dan Klein (2016). „Neural module networks“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48.

Ansell, Alan, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić (2021). „Composable sparse fine-tuning for cross-lingual transfer“. In: *arXiv preprint arXiv:2110.07560*.

Ansell, Alan, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen (Nov. 2021). „MAD-G: Multilingual Adapter Generation for

Efficient Cross-Lingual Transfer“. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4762–4781.

Antonakaki, Despoina, Paraskevi Fragopoulou, and Sotiris Ioannidis (2021). „A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks“. In: *Expert Systems with Applications* 164, p. 114006.

Aribandi, Vamsi, Yi Tay, Tal Schuster, *et al.* (2022). „ExT5: Towards Extreme Multi-Task Scaling for Transfer Learning“. In: *International Conference on Learning Representations*.

Asai, Akari, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi (2022). „Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts“. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6655–6672.

Austin, Jacob, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, *et al.* (2021). „Program Synthesis with Large Language Models“. In: *arXiv preprint arXiv:2108.07732*.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML].

Bach, Stephen H, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, *et al.* (2022). „Promptsource: An integrated development environment and repository for natural language prompts“. In: *arXiv preprint arXiv:2202.01279*.

Bahdanau, Dzmitry, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville (2019). „Systematic Generalization: What Is Required and Can It Be Learned?“ In: *International Conference on Learning Representations*.

Ballard, Dana H (1986). „Cortical connections and parallel processing: Structure and function“. In: *Behavioral and brain sciences* 9.1, pp. 67–90.

Bau, Jason, Jonathan Mayer, Hristo Paskov, and John C Mitchell (2013). „A promising direction for web tracking countermeasures“. In: *Proceedings of W2SP*.

Beck, Tilman, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych (2021). „Adapterhub playground: Simple and flexible few-shot learning with adapters“. In: *arXiv preprint arXiv:2108.08103*.

Beeching, Edward, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf (2023). *Open LLM Leaderboard*. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

Bekos, Paschalis, Panagiotis Papadopoulos, Evangelos P Markatos, and Nicolas Kourtellis (2023). „The Hitchhiker's Guide to Facebook Web Tracking with Invisible Pixels and Click IDs“. In: *Proceedings of the ACM Web Conference 2023*, pp. 2132–2143.

Belofsky, Joshua (2023). „Token-level Adaptation of LoRA Adapters for Downstream Task Generalization“. In: *arXiv preprint arXiv:2311.10847*.

Bengio, Emmanuel, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup (2015). „Conditional computation in neural networks for faster models“. In: *arXiv preprint arXiv:1511.06297*.

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer (2015). „Scheduled sampling for sequence prediction with recurrent neural networks“. In: *Advances in neural information processing systems* 28.

Bilić, Paško and Toni Prug (2021). *The Political Economy of Digital Monopolies: Contradictions and Alternatives to Data Commodification*. Policy Press.

Binns, Reuben, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt (2018). „Third party tracking in the mobile ecosystem“. In: *Proceedings of the 10th ACM Conference on Web Science*, pp. 23–31.

Binns, Reuben, Jun Zhao, Max Van Kleek, and Nigel Shadbolt (2018). „Measuring third-party tracker power across web and mobile“. In: *ACM Transactions on Internet Technology (TOIT)* 18.4, pp. 1–22.

Bisk, Yonatan, Rowan Zellers, Jianfeng Gao, Yejin Choi, *et al.* (2020). „Piqa: Reasoning about physical commonsense in natural language“. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05, pp. 7432–7439.

Black, Sid, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman (Mar. 2021). *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. Version 1.0. If you use this software, please cite it using these metadata.

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

Breiman, Leo (1996). „Bagging predictors“. In: *Machine learning* 24, pp. 123–140.

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

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

Bugliarello, Emanuele, Fangyu Liu, Jonas Pfeiffer, Siva Reddy, Desmond Elliott, Edoardo Maria Ponti, and Ivan Vulić (17–23 Jul 2022). „IGLUE: A Benchmark for Transfer Learning across Modalities, Tasks, and Languages“. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 2370–2392.

Caccia, Lucas, Edoardo Ponti, Lucas Liu, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni (2022). „Multi-Head Adapter Routing for Data-Efficient Fine-Tuning“. In: *arXiv preprint arXiv:2211.03831*.

Caccia, Lucas, Edoardo Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni (2023). *Multi-Head Adapter Routing for Cross-Task Generalization*. arXiv: 2211.03831 [cs.AI].

Cahn, Aaron, Scott Alfeld, Paul Barford, and Shanmugavelayutham Muthukrishnan (2016). „An empirical study of web cookies“. In: *Proceedings of the 25th international conference on world wide web*, pp. 891–901.

Caron, Mathilde, Piotr Bojanowski, Armand Joulin, and Matthijs Douze (2018). „Deep clustering for unsupervised learning of visual features“. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149.

Caruana, Rich (1997). „Multitask learning“. In: *Machine learning* 28, pp. 41–75.

Cha, Meeyoung, Alan Mislove, and Krishna P Gummadi (2009). „A measurement-driven analysis of information propagation in the flickr social network“. In: *Proceedings of the 18th international conference on World wide web*, pp. 721–730.

Chang, Michael B, Abhishek Gupta, Sergey Levine, and Thomas L Griffiths (2018). „Automatically composing representation transformations as a means for generalization“. In: *arXiv preprint arXiv:1807.04640*.

Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, *et al.* (2021). „Evaluating large language models trained on code“. In: *arXiv preprint arXiv:2107.03374*.

Chen, Wei, Carlos Castillo, and Laks VS Lakshmanan (2022). *Information and influence propagation in social networks*. Springer Nature.

Chen, Zitian, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik Learned-Miller, and Chuang Gan (2022). *Mod-Squad: Designing Mixture of Experts As Modular Multi-Task Learners*. arXiv: 2212.08066 [cs.CV].

Cheng, Ruizhi, Nan Wu, Matteo Varvello, Songqing Chen, and Bo Han (2022). „Are we ready for metaverse? A measurement study of social virtual reality platforms“. In: *Proceedings of the 22nd ACM Internet Measurement Conference*, pp. 504–518.

Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, *et al.* (2023). „Palm: Scaling language modeling with pathways“. In: *Journal of Machine Learning Research* 24.240, pp. 1–113.

Christl, Wolfie and Sarah Spiekermann (2016). „Networks of Control: A Report on Corporate Surveillance“. In: *Digital Tracking, Big Data & Privacy*.

Chronopoulou, Alexandra, Matthew E Peters, and Jesse Dodge (2021). „Efficient hierarchical domain adaptation for pretrained language models“. In: *arXiv preprint arXiv:2112.08786*.

Chronopoulou, Alexandra, Matthew E Peters, Alexander Fraser, and Jesse Dodge (2023). „Adaptersoup: Weight averaging to improve generalization of pretrained language models“. In: *arXiv preprint arXiv:2302.07027*.

Clark, Aidan, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, *et al.* (2022). „Unified scaling laws for routed language models“. In: *International Conference on Machine Learning*. PMLR, pp. 4057–4086.

Clark, Christopher, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova (2019). „BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions“. In: *NAACL*.

Clark, Peter, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord (2018). „Think you have solved question answering? try arc, the ai2 reasoning challenge“. In: *arXiv preprint arXiv:1803.05457*.

Clausen, Lars (2004). „Concerning etags and datestamps“. In: *4th International Web Archiving Workshop (IWAW'04)*. Citeseer.

Corona, Rodolfo, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell (June 2021). „Modular Networks for Compositional Instruction Following“. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 1033–1040.

Craig, Terence and Mary E Ludloff (2011). *Privacy and big data: the players, regulators, and stakeholders*. " O'Reilly Media, Inc."

Dabrowski, Adrian, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar Weippl (2019). „Measuring cookies and web privacy in a post-gdpr world". In: *International Conference on Passive and Active Network Measurement*. Springer, pp. 258–270.

Dagan, Ido, Oren Glickman, and Bernardo Magnini (2005). „The pascal recognising textual entailment challenge". In: *Machine learning challenges workshop*. Springer, pp. 177–190.

De Marneffe, Marie-Catherine, Mandy Simons, and Judith Tonhauser (2019). „The commitmentbank: Investigating projection in naturally occurring discourse". In: *proceedings of Sinn und Bedeutung*. Vol. 23. 2, pp. 107–124.

Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer (2024). „Qlora: Efficient finetuning of quantized llms". In: *Advances in Neural Information Processing Systems* 36.

Deville, Joe and Lonneke Van der Velden (2015). „Seeing the invisible algorithm: The practical politics of tracking the credit trackers". In: *Algorithmic Life*. Routledge, pp. 87–106.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). „Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dietterich, Thomas G (2000). „Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer, pp. 1–15.

Ding, Chuntao, Zhichao Lu, Shangguang Wang, Ran Cheng, and Vishnu Naresh Boddeti (2023). „Mitigating Task Interference in Multi-Task Learning via Explicit Task Routing With Non-Learnable Primitives". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7756–7765.

Dodds, Peter Sheridan and Duncan J Watts (2005). „A generalized model of social and biological contagion". In: *Journal of theoretical biology* 232.4, pp. 587–604.

Domingos, Pedro and Matt Richardson (2001). „Mining the network value of customers". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 57–66.

Du, Nan, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, *et al.* (2022). „Glam: Efficient scaling of language models with mixture-of-experts". In: *International Conference on Machine Learning*. PMLR, pp. 5547–5569.

Dua, Dheeru, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan (2021). „Tricks for training sparse translation models". In: *arXiv preprint arXiv:2110.08246*.

Duan, Zhibin, Hao Zhang, Chaojie Wang, Zhengjue Wang, Bo Chen, and Mingyuan Zhou (2021). „EnsLM: Ensemble language model for data diversity by semantic clustering". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2954–2967.

Easley, David, Jon Kleinberg, *et al.* (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Vol. 1. Cambridge university press Cambridge.

Edalati, Ali, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh (2022). „Krona: Parameter efficient tuning with kronecker adapter“. In: *arXiv preprint arXiv:2212.10650*.

Eigen, David, Marc'Aurelio Ranzato, and Ilya Sutskever (2013). „Learning factored representations in a deep mixture of experts“. In: *arXiv preprint arXiv:1312.4314*.

Elhage, Nelson, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, *et al.* (2021). „A mathematical framework for transformer circuits“. In: *Transformer Circuits Thread* 1.

Elharrouss, Omar, Noor Almaadeed, and Somaya Al-Maadeed (2021). „A review of video surveillance systems“. In: *Journal of Visual Communication and Image Representation* 77, p. 103116.

Elisabeta, Păduraru Monica and Mihăilă Robert Alexandru (2018). „Comparative Analysis of E-Learning Platforms on The Market“. In: *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, pp. 1–4.

Englehardt, Steven, Chris Eubank, Peter Zimmerman, Dillon Reisman, and Arvind Narayanan (2015). „OpenWPM: An automated platform for web privacy measurement“. In: *Manuscript. March*.

Englehardt, Steven and Arvind Narayanan (2016). „Online tracking: A 1-million-site measurement and analysis“. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1388–1401.

Englehardt, Steven, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten (2015). „Cookies that give you away: The surveillance implications of web tracking“. In: *Proceedings of the 24th International Conference on World Wide Web*, pp. 289–299.

Ermakova, Tatiana, Benjamin Fabian, Benedict Bender, and Kerstin Klimek (2018). „Web tracking-A literature review on the state of research“. In.

Fang, Taoran, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen (2024). „Universal prompt tuning for graph neural networks“. In: *Advances in Neural Information Processing Systems* 36.

Fedus, William, Jeff Dean, and Barret Zoph (2022). „A review of sparse expert models in deep learning“. In: *arXiv preprint arXiv:2209.01667*.

Fedus, William, Barret Zoph, and Noam Shazeer (2022a). „Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity“. In: *Journal of Machine Learning Research* 23.120, pp. 1–39.

Fedus, William, Barret Zoph, and Noam Shazeer (2022b). „Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity“. In: *The Journal of Machine Learning Research* 23.1, pp. 5232–5270.

Fernando, Chrisantha, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra (2017). „Pathnet: Evolution channels gradient descent in super neural networks“. In: *arXiv preprint arXiv:1701.08734*.

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

Frankle, Jonathan, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin (2020). „Linear mode connectivity and the lottery ticket hypothesis“. In: *International Conference on Machine Learning*. PMLR, pp. 3259–3269.

French, Robert M (1999). „Catastrophic forgetting in connectionist networks“. In: *Trends in cognitive sciences* 3.4, pp. 128–135.

Gan, Guobing, Peng Zhang, Sunzhu Li, Xiuqing Lu, and Benyou Wang (2022). „Morphte: Injecting morphology in tensorized embeddings“. In: *Advances in Neural Information Processing Systems* 35, pp. 33186–33200.

Gandon, Fabien and Wendy Hall (2022). „A never-ending project for humanity called "the Web"“. In: *Proceedings of the ACM Web Conference 2022*, pp. 3480–3487.

Gesmundo, Andrea (2022). „A Multi-Agent Framework for the Asynchronous and Collaborative Extension of Multitask ML Systems“. In: *arXiv preprint arXiv:2209.14745*.

Goldberg, Samuel G, Garrett A Johnson, and Scott K Shriver (2024). „Regulating privacy online: An economic evaluation of the GDPR“. In: *American Economic Journal: Economic Policy* 16.1, pp. 325–358.

González-Bailón, Sandra, David Lazer, Pablo Barberá, Meiqing Zhang, Hunt Allcott, Taylor Brown, Adriana Crespo-Tenorio, Deen Freelon, Matthew Gentzkow, Andrew M Guess, *et al.* (2023). „Asymmetric ideological segregation in exposure to political news on Facebook“. In: *Science* 381.6656, pp. 392–398.

Goyal, Anirudh, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf (2019). „Recurrent independent mechanisms“. In: *arXiv preprint arXiv:1909.10893*.

Graux, Damien and Fabrizio Orlandi (2022). „Through the Lens of the Web Conference Series: A Look Into the History of the Web“. In: *Proceedings of the ACM Web Conference 2022*, pp. 3458–3464.

Gross, Sam, Marc'Aurelio Ranzato, and Arthur Szlam (2017). „Hard mixtures of experts for large scale weakly supervised vision“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6865–6873.

Gruhl, Daniel, Ramanathan Guha, David Liben-Nowell, and Andrew Tomkins (2004). „Information diffusion through blogspace“. In: *Proceedings of the 13th international conference on World Wide Web*, pp. 491–501.

Guo, Demi, Alexander M Rush, and Yoon Kim (2020). „Parameter-efficient transfer learning with diff pruning“. In: *arXiv preprint arXiv:2012.07463*.

Gururangan, Suchin, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer (2023). „Scaling Expert Language Models with Unsupervised Domain Discovery“. In: *arXiv preprint arXiv:2303.14177*.

Haddara, Moutaz, Ab Salazar, and Marius Langseth (2023). „Exploring the Impact of GDPR on Big Data Analytics Operations in the E-Commerce Industry“. In: *Procedia Computer Science* 219, pp. 767–777.

Halpin, Harry and Evan Henshaw-Plath (2022). „From Indymedia to Tahrir Square: The Revolutionary Origins of Status Updates on Twitter“. In: *Proceedings of the ACM Web Conference 2022*, pp. 3465–3470.

Hambardzumyan, Karen, Hrant Khachatrian, and Jonathan May (2021). „Warp: Word-level adversarial reprogramming“. In: *arXiv preprint arXiv:2101.00121*.

Han, Seungyeop, Jaeyeon Jung, and David Wetherall (2012). „A study of third-party tracking by mobile apps in the wild“. In: *Univ. Washington, Tech. Rep. UW-CSE-12-03* 1.

Hartline, Jason, Vahab Mirrokni, and Mukund Sundararajan (2008). „Optimal marketing strategies over social networks“. In: *Proceedings of the 17th international conference on World Wide Web*, pp. 189–198.

He, Daobing and Xiaoyang Liu (2020). „Novel competitive information propagation macro mathematical model in online social network“. In: *Journal of Computational Science* 41, p. 101089.

He, Junxian, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig (2021). „Towards a unified view of parameter-efficient transfer learning“. In: *arXiv preprint arXiv:2110.04366*.

He, Shwai, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao (2022). „Sparseadapter: An easy approach for improving the parameter-efficiency of adapters“. In: *arXiv preprint arXiv:2210.04284*.

Helles, Rasmus, Stine Lomborg, and Signe Sophus Lai (2020). „Infrastructures of tracking: Mapping the ecology of third-party services across top sites in the EU“. In: *New Media & Society* 22.11, pp. 1957–1975.

Helmond, Anne, Niels Brügger, *et al.* (2017). „Historical website ecology. Analyzing past states of the web using archived source code“. In.

Ho, Monte and Jan Kallberg (2017). *Black Code: Surveillance, Privacy, and the Dark Side of the Internet*.

Hoofnagle, Chris Jay, Ashkan Soltani, Nathaniel Good, and Dietrich J Wambach (2012). „Behavioral advertising: The offer you can't refuse“. In: *Harv. L. & Pol'y Rev.* 6, p. 273.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly (2019a). „Parameter-efficient transfer learning for NLP“. In: *International Conference on Machine Learning*. PMLR, pp. 2790–2799.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly (2019b). „Parameter-efficient transfer learning for NLP“. In: *International Conference on Machine Learning*. PMLR, pp. 2790–2799.

Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen (2021). „Lora: Low-rank adaptation of large language models“. In: *arXiv preprint arXiv:2106.09685*.

Hu, Xuehui and Nishanth Sastry (2019). „Characterising third party cookie usage in the EU after GDPR“. In: *Proceedings of the 10th ACM Conference on Web Science*, pp. 137–141.

Hu, Yuzheng, Ruicheng Xian, Qilong Wu, Qiuling Fan, Lang Yin, and Han Zhao (2024). „Revisiting scalarization in multi-task learning: A theoretical perspective“. In: *Advances in Neural Information Processing Systems* 36.

Huang, Chengsong, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin (2023). „Lorahub: Efficient cross-task generalization via dynamic lora composition“. In: *arXiv preprint arXiv:2307.13269*.

Huang, Chengsong, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin (2024). *LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition*. arXiv: 2307.13269 [cs.CL].

Huh, Minyoung, Brian Cheung, Jeremy Bernstein, Phillip Isola, and Pulkit Agrawal (2024). „Training Neural Networks from Scratch with Parallel Low-Rank Adapters“. In: *arXiv preprint arXiv:2402.16828*.

Ilharco, Gabriel, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi (2022). „Editing models with task arithmetic“. In: *arXiv preprint arXiv:2212.04089*.

Izmailov, Pavel, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson (2018). „Averaging Weights Leads to Wider Optima and Better Generalization“. In: *CoRR* abs/1803.05407. arXiv: 1803.05407.

Jacobs, Robert A, Michael I Jordan, and Andrew G Barto (1991). „Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks“. In: *Cognitive science* 15.2, pp. 219–250.

Jacobs, Robert A, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton (1991a). „Adaptive mixtures of local experts“. In: *Neural computation* 3.1, pp. 79–87.

Jacobs, Robert A, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton (1991b). „Adaptive mixtures of local experts“. In: *Neural computation* 3.1, pp. 79–87.

Jang, Eric, Shixiang Gu, and Ben Poole (2016). „Categorical reparameterization with gumbel-softmax“. In: *arXiv preprint arXiv:1611.01144*.

Jang, Joel, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo (2023). „Exploring the benefits of training expert language models over instruction tuning“. In: *arXiv preprint arXiv:2302.03202*.

Jarke, Juliane and Andreas Breiter (2019). „The datafication of education“. In: *Learning, Media and Technology* 44.1, pp. 1–6.

Jerez-Villota, Eleana, Francisco Jurado, and Jaime Moreno-Llorena (2023). „Understanding the Role of the User in Information Propagation on Online Social Networks: A Literature Review and Proposed User Model“. In: *International Conference on Ubiquitous Computing and Ambient Intelligence*. Springer, pp. 304–315.

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, *et al.* (2024). *Mixtral of Experts*. arXiv: 2401.04088 [cs.LG].

Jie, Shibo and Zhi-Hong Deng (2023). „Fact: Factor-tuning for lightweight adaptation on vision transformer“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 1, pp. 1060–1068.

Jin, Xisen, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng (2022). „Dataless knowledge fusion by merging weights of language models“. In: *arXiv preprint arXiv:2212.09849*.

Jolicoeur-Martineau, Alexia, Emy Gervais, Kilian Fatras, Yan Zhang, and Simon Lacoste-Julien (2023). *PopulAtion Parameter Averaging (PAPA)*. arXiv: 2304.03094 [cs.LG].

Jordan, Katy (2018). „Degrees of intrusion? A survey of cookies used by UK Higher Education institutional websites and their implications“. In: *A Survey of Cookies Used by UK Higher Education Institutional Websites and Their Implications (March 16, 2018)*.

Jordan, Michael I and Robert A Jacobs (1994). „Hierarchical mixtures of experts and the EM algorithm“. In: *Neural computation* 6.2, pp. 181–214.

Joseph, Nimish (2023). „Understanding information propagation in social media“. PhD thesis. IIT Delhi.

Karaj, Arjaldo, Sam Macbeth, Rémi Berson, and Josep M Pujol (2018). „WhoTracks. Me: Shedding light on the opaque world of online tracking“. In: *arXiv preprint arXiv:1804.08959*.

Karimi Mahabadi, Rabeeh, James Henderson, and Sebastian Ruder (2021). „Compacter: Efficient low-rank hypercomplex adapter layers“. In: *Advances in Neural Information Processing Systems* 34, pp. 1022–1035.

Kempe, David, Jon Kleinberg, and Éva Tardos (2003). „Maximizing the spread of influence through a social network“. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146.

Kirsch, Louis, Julius Kunze, and David Barber (2018). „Modular networks: Learning to decompose neural computation“. In: *Advances in neural information processing systems* 31.

Kretschmer, Michael, Jan Pennekamp, and Klaus Wehrle (2021). „Cookie banners and privacy policies: Measuring the impact of the GDPR on the web“. In: *ACM Transactions on the Web (TWEB)* 15.4, pp. 1–42.

Krishnamurthy, Balachander and Craig Wills (2009). „Privacy diffusion on the web: a longitudinal perspective“. In: *Proceedings of the 18th international conference on World wide web*, pp. 541–550.

Kudugunta, Sneha, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat (2021). „Beyond distillation: Task-level mixture-of-experts for efficient inference“. In: *arXiv preprint arXiv:2110.03742*.

Kye, Seung-Hyeok (2023). „Compositions and tensor products of linear maps between matrix algebras“. In: *Linear Algebra and its Applications* 658, pp. 283–309.

Landau, Susan (2011). *Surveillance or security?: The risks posed by new wiretapping technologies*. Mit Press.

Lauer, Josh and Kenneth Lipartito (2021). *Surveillance capitalism in America*. University of Pennsylvania Press.

Lee, Charles Cheolgi, Jafar Afshar, Arousha Haghighian Roudsari, Woong-Kee Loh, and Wookey Lee (2024). „A bitwise approach on influence overload problem“. In: *Data & Knowledge Engineering* 150, p. 102276.

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

Lei, Tao, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, *et al.* (2024). „Conditional adapters: Parameter-efficient transfer learning with fast inference“. In: *Advances in Neural Information Processing Systems* 36.

Lepikhin, D, H Lee, Y Xu, D Chen, O Firat, Y Huang, M Krikun, N Shazeer, and Z Gshard (2020). „Scaling giant models with conditional computation and automatic sharding“. In: *arXiv preprint arXiv:2006.16668*.

Lerner, Ada, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner (2016). „Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016“. In: *25th USENIX Security Symposium (USENIX Security 16)*.

Lester, Brian, Rami Al-Rfou, and Noah Constant (2021). „The power of scale for parameter-efficient prompt tuning“. In: *arXiv preprint arXiv:2104.08691*.

Levesque, Hector, Ernest Davis, and Leora Morgenstern (2012). „The winograd schema challenge“. In: *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Lewis, Mike, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer (2021). „Base layers: Simplifying training of large, sparse models“. In: *International Conference on Machine Learning*. PMLR, pp. 6265–6274.

Li, He, Lu Yu, and Wu He (2019). *The impact of GDPR on global technology development*.

Li, Tai-Ching, Huy Hang, Michalis Faloutsos, and Petros Efstathopoulos (2015). „Trackadvisor: Taking back browsing privacy from third-party trackers“. In: *Passive and Active Measurement: 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings 16*. Springer, pp. 277–289.

Li, Xiang Lisa and Jason Eisner (Nov. 2019). „Specializing Word Embeddings (for Parsing) by Information Bottleneck“. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 2744–2754.

Li, Xiang Lisa and Percy Liang (Aug. 2021a). „Prefix-Tuning: Optimizing Continuous Prompts for Generation“. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, pp. 4582–4597.

Li, Xiang Lisa and Percy Liang (2021b). „Prefix-tuning: Optimizing continuous prompts for generation“. In: *arXiv preprint arXiv:2101.00190*.

Lialin, Vladislav, Vijeta Deshpande, and Anna Rumshisky (2023). „Scaling down to scale up: A guide to parameter-efficient fine-tuning“. In: *arXiv preprint arXiv:2303.15647*.

Lialin, Vladislav, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky (2023). „ReLoRA: High-Rank Training Through Low-Rank Updates“. In: *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Liang, Juhao, Chen Zhang, Zhengyang Tang, Jie Fu, Dawei Song, and Benyou Wang (2023). *Modular Retrieval for Generalization and Interpretation*. arXiv: 2303.13419 [cs.IR].

Libert, Timothy (2015). „Exposing the hidden web: An analysis of third-party HTTP requests on 1 million websites“. In: *arXiv preprint arXiv:1511.00619*.

Libert, Timothy and R Nielsen (2018). „Third-party web content on eu news sites: Potential challenges and paths to privacy improvement“. In: *Reuters Institute for the Study of Journalism*.

Lin, Chin-Yew and Eduard Hovy (2003). „Automatic evaluation of summaries using n-gram co-occurrence statistics“. In: *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pp. 150–157.

Liu, Haokun, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel (2022a). „Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning“. In: *Advances in Neural Information Processing Systems* 35, pp. 1950–1965.

Liu, Haokun, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel (2022b). „Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning“. In: *Advances in Neural Information Processing Systems* 35, pp. 1950–1965.

Liu, Xiao, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang (2022). „P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks“. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68.

Liu, Xiao, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang (2023). „GPT understands, too“. In: *AI Open*.

Liu, Xiaodong, Pengcheng He, Weizhu Chen, and Jianfeng Gao (2019). „Multi-task deep neural networks for natural language understanding“. In: *arXiv preprint arXiv:1901.11504*.

Liu, Xiaoyang and Daobing He (2020). „Information propagation and public opinion evolution model based on artificial neural network in online social network“. In: *The Computer Journal* 63.11, pp. 1689–1703.

Longpre, Shayne, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, *et al.* (2023). „The flan collection: Designing data and methods for effective instruction tuning“. In: *arXiv preprint arXiv:2301.13688*.

Lu, Keming, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou (2023). „Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models“. In: *arXiv preprint arXiv:2311.08692*.

Lu, Yao, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp (2021). „Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity“. In: *arXiv preprint arXiv:2104.08786*.

Lyon, David (2018). *The culture of surveillance: Watching as a way of life*. John Wiley & Sons.

Maddison, Chris J, Andriy Mnih, and Yee Whye Teh (2016). „The concrete distribution: A continuous relaxation of discrete random variables“. In: *arXiv preprint arXiv:1611.00712*.

Mahabadi, Rabeeh Karimi, Sebastian Ruder, Mostafa Dehghani, and James Henderson (2021). „Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks“. In: *arXiv preprint arXiv:2106.04489*.

Mangrulkar, Sourab, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan (2022). *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. https://github.com/huggingface/peft.

Mansell, Robin and Marc Raboy (2011). „Introduction: Foundations of the theory and practice of global media and communication policy“. In: *The handbook of global media and communication policy*, pp. 1–20.

Mansoori, Masood, Yuichi Hirose, Ian Welch, and Kim-Kwang Raymond Choo (2016). „Empirical analysis of impact of HTTP referer on malicious website behaviour and delivery“. In: *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, pp. 941–948.

Mao, Yuning, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa (2021). „Unipelt: A unified framework for parameter-efficient language model tuning“. In: *arXiv preprint arXiv:2110.07577*.

Matena, Michael S and Colin A Raffel (2022). „Merging models with fisher-weighted averaging“. In: *Advances in Neural Information Processing Systems* 35, pp. 17703–17716.

Mathur, Arunesh, Gunes Acar, Michael J Friedman, Elena Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan (2019). „Dark patterns at scale: Findings from a crawl of 11K shopping websites“. In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW, pp. 1–32.

Mayer, Jonathan R and John C Mitchell (2012). „Third-party web tracking: Policy and technology“. In: *2012 IEEE symposium on security and privacy*. IEEE, pp. 413–427.

McCloskey, Michael and Neal J Cohen (1989). „Catastrophic interference in connectionist networks: The sequential learning problem“. In: *Psychology of learning and motivation*. Vol. 24. Elsevier, pp. 109–165.

Meyerson, Elliot and Risto Miikkulainen (2017). „Beyond shared hierarchies: Deep multitask learning through soft layer ordering“. In: *arXiv preprint arXiv:1711.00108*.

Mickens, James (2010). „Silo: Exploiting {JavaScript} and {DOM} Storage for Faster Page Loads“. In: *USENIX Conference on Web Application Development (WebApps 10)*.

Microsoft Research (2023). *Phi-2: The Surprising Power of Small Language Models*. URL: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/.

Mihaylov, Todor, Peter Clark, Tushar Khot, and Ashish Sabharwal (2018). „Can a suit of armor conduct electricity? a new dataset for open book question answering“. In: *arXiv preprint arXiv:1809.02789*.

Mireshghallah, Fatemehsadat, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh (2020). „Privacy in deep learning: A survey“. In: *arXiv preprint arXiv:2004.12254*.

Modi, Nandini and Jaiteg Singh (2023). „Understanding online consumer behavior at e-commerce portals using eye-gaze tracking“. In: *International Journal of Human–Computer Interaction* 39.4, pp. 721–742.

Moreira, Silvio, David S Batista, Paula Carvalho, Francisco M Couto, and Mario J Silva (2013). „Tracking politics with POWER“. In: *Program* 47.2, pp. 120–135.

Müller, Philipp and Ruben L Bach (2023). „Populist alternative news use and its role for elections: Web-tracking and survey evidence from two campaign periods“. In: *New media & society* 25.10, pp. 2663–2683.

Muqeeth, Mohammed, Haokun Liu, Yufan Liu, and Colin Raffel (2024). „Learning to Route Among Specialized Experts for Zero-Shot Generalization“. In: *arXiv preprint arXiv:2402.05859*.

Muqeeth, Mohammed, Haokun Liu, and Colin Raffel (2023). „Soft Merging of Experts with Adaptive Routing“. In: *arXiv preprint arXiv:2306.03745*.

Musicki, Darko, Regina Kaune, and Wolfgang Koch (2009). „Mobile emitter geolocation and tracking using TDOA and FDOA measurements“. In: *IEEE transactions on signal processing* 58.3, pp. 1863–1874.

Nakatsukasa, Yuji (2019). „The low-rank eigenvalue problem“. In: *arXiv preprint arXiv:1905.11490*.

Nie, Yixin, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela (2019). „Adversarial NLI: A new benchmark for natural language understanding“. In: *arXiv preprint arXiv:1910.14599*.

Nikiforakis, Nick, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna (2013). „Cookieless monster: Exploring the ecosystem of web-based device fingerprinting“. In: *2013 IEEE Symposium on Security and Privacy*. IEEE, pp. 541–555.

OpenAI (2023). *GPT-3.5 Turbo: Fine-Tuning and API Updates*. https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates.

Ostapenko, Oleksiy, Lucas Caccia, Zhan Su, Nicolas Le Roux, Laurent Charlin, and Alessandro Sordoni (2023). „A Case Study of Instruction Tuning with Mixture of Parameter-Efficient Experts“. In: *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Ostapenko, Oleksiy, Pau Rodriguez, Massimo Caccia, and Laurent Charlin (2021). „Continual Learning via Local Module Composition“. In: *Advances in Neural Information Processing Systems* 34.

Pan, Yu, Ye Yuan, Yichun Yin, Zenglin Xu, Lifeng Shang, Xin Jiang, and Qun Liu (2024). „Reusing Pretrained Models by Multi-linear Operators for Efficient Training“. In: *Advances in Neural Information Processing Systems* 36.

Panahi, Aliakbar, Seyran Saeedi, and Tom Arodz (2019). „word2ket: Space-efficient word embeddings inspired by quantum entanglement“. In: *arXiv preprint arXiv:1911.04975*.

Parker, Philip M (2008). *The 2009-2014 World Outlook for Advertising for Social Media and Widgets*. Icon Group International.

Patel, Jay M and Jay M Patel (2020). „Introduction to common crawl datasets“. In: *Getting Structured Data from the Internet: Running Web Crawlers/Scrapers on a Big Data Production Scale*, pp. 277–324.

Peloquin, David, Michael DiMaio, Barbara Bierer, and Mark Barnes (2020). „Disruptive and avoidable: GDPR challenges to secondary research uses of data“. In: *European Journal of Human Genetics* 28.6, pp. 697–705.

Pfeiffer, Jonas, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych (2020). „AdapterFusion: Non-destructive task composition for transfer learning“. In: *arXiv preprint arXiv:2005.00247*.

Pfeiffer, Jonas, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti (2023). „Modular deep learning“. In: *arXiv preprint arXiv:2302.11529*.

Pfeiffer, Jonas, Edwin Simpson, and Iryna Gurevych (2020). „Low Resource Multi-Task Sequence Tagging–Revisiting Dynamic Conditional Random Fields“. In: *arXiv preprint arXiv:2005.00250*.

Pfeiffer, Jonas, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder (2020a). „Mad-x: An adapter-based framework for multi-task cross-lingual transfer“. In: *arXiv preprint arXiv:2005.00052*.

Pfeiffer, Jonas, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder (2020b). „UNKs everywhere: Adapting multilingual language models to new scripts“. In: *arXiv preprint arXiv:2012.15562*.

Phang, Jason, Thibault Févry, and Samuel R Bowman (2018). „Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks“. In: *arXiv preprint arXiv:1811.01088*.

Pilehvar, Mohammad Taher and Jose Camacho-Collados (2018). „WiC: the word-in-context dataset for evaluating context-sensitive meaning representations“. In: *arXiv preprint arXiv:1808.09121*.

Ponti, Edoardo (2021). „Inductive Bias and Modular Design for Sample-Efficient Neural Language Learning“. PhD thesis. University of Cambridge.

Ponti, Edoardo Maria, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen (2019). „Modeling language variation and universals: A survey on typological linguistics for natural language processing“. In: *Computational Linguistics* 45.3, pp. 559–601.

Ponti, Edoardo Maria, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy (2023a). „Combining Parameter-efficient Modules for Task-level Generalisation“. In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 687–702.

Ponti, Edoardo Maria, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy (May 2023b). „Combining Parameter-efficient Modules for Task-level Generalisation“. In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, pp. 687–702.

Poth, Clifton, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer (2023). „Adapters: A Unified Library for Parameter-Efficient and Modular Transfer Learning“. In: *arXiv preprint arXiv:2311.11077*.

Qiu, Xipeng, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang (2020). „Pre-trained models for natural language processing: A survey“. In: *Science China Technological Sciences* 63.10, pp. 1872–1897.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, *et al.* (2019). „Language models are unsupervised multitask learners“. In: *OpenAI blog* 1.8, p. 9.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu (2020a). „Exploring the limits of transfer learning with a unified text-to-text transformer“. In: *The Journal of Machine Learning Research* 21.1, pp. 5485–5551.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, *et al.* (2020b). „Exploring the limits of transfer learning with a unified text-to-text transformer.“ In: *J. Mach. Learn. Res.* 21.140, pp. 1–67.

Rajbhandari, Samyam, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He (2022). „Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale“. In: *International Conference on Machine Learning*. PMLR, pp. 18332–18346.

Rapin, J. and O. Teytaud (2018). *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad.

Razdaibiedina, Anastasia, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi (2023). „Progressive prompts: Continual learning for language models“. In: *arXiv preprint arXiv:2301.12314*.

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi (2017). „Learning multiple visual domains with residual adapters“. In: *Advances in neural information processing systems* 30.

Regulation, General Data Protection (2018). „General data protection regulation (GDPR)“. In: *Intersoft Consulting, Accessed in October* 24.1.

Robbins, Philip (2009). „Modularity of mind“. In.

Roemmele, Melissa, Cosmin Adrian Bejan, and Andrew S Gordon (2011). „Choice of plausible alternatives: An evaluation of commonsense causal reasoning“. In: *2011 AAAI Spring Symposium Series*.

Roesner, Franziska, Tadayoshi Kohno, and David Wetherall (2012). „Detecting and Defending Against {Third-Party} Tracking on the Web“. In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 155–168.

Roger, Alexis (2022). „A review of modern surveillance techniques and their presence in our society“. In: *arXiv preprint arXiv:2210.09002*.

Roller, Stephen, Sainbayar Sukhbaatar, Jason Weston, *et al.* (2021). „Hash Layers For Large Sparse Models“. In: *Advances in Neural Information Processing Systems* 34, pp. 17555–17566.

Rosenbaum, Clemens, Ignacio Cases, Matthew Riemer, and Tim Klinger (2019). „Routing networks and the challenges of modular and compositional computation“. In: *arXiv preprint arXiv:1904.12774*.

Rosenbaum, Clemens, Tim Klinger, and Matthew Riemer (2017). „Routing networks: Adaptive selection of non-linear functions for multi-task learning“. In: *arXiv preprint arXiv:1711.01239*.

Rubin, Ohad, Jonathan Herzig, and Jonathan Berant (2021). „Learning to retrieve prompts for in-context learning“. In: *arXiv preprint arXiv:2112.08633*.

Ruder, Sebastian (2017). „An overview of multi-task learning in deep neural networks“. In: *arXiv preprint arXiv:1706.05098*.

Rule, James B and Graham William Greenleaf (2010). *Global privacy protection: the first generation*. Edward Elgar Publishing.

Russell, Matthew A (2013). *Mining the social web: data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more*. " O'Reilly Media, Inc."

Sakaguchi, Keisuke, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi (2021). „Winogrande: An adversarial winograd schema challenge at scale“. In: *Communications of the ACM* 64.9, pp. 99–106.

Samarasinghe, Nayanamana and Mohammad Mannan (2019). „Towards a global perspective on web tracking“. In: *Computers & Security* 87, p. 101569.

Sanchez-Rola, Iskander, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos (2019). „Can i opt out yet? gdpr and the global illusion of cookie control“. In: *Proceedings of the 2019 ACM Asia conference on computer and communications security*, pp. 340–351.

Sanh, Victor, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, *et al.* (2021a). „Multitask prompted training enables zero-shot task generalization“. In: *arXiv preprint arXiv:2110.08207*.

Sanh, Victor, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, *et al.* (2021b). „Multitask prompted training enables zero-shot task generalization“. In: *arXiv preprint arXiv:2110.08207*.

Sanh, Victor, Albert Webson, Colin Raffel, *et al.* (2022). „Multitask Prompted Training Enables Zero-Shot Task Generalization“. In: *The Tenth International Conference on Learning Representations*.

Saxena, Akrati, Pratishtha Saxena, Harita Reddy, and Ralucca Gera (2019). „A survey on studying the social networks of students“. In: *arXiv preprint arXiv:1909.05079*.

Schelter, Sebastian *et al.* (2018). „On the ubiquity of web tracking: Insights from a billion-page web crawl“. In: *The Journal of Web Science* 4.

Schneier, Bruce (2015). *Data and Goliath: The hidden battles to collect your data and control your world*. WW Norton & Company.

Sevignani, Sebastian (2015). *Privacy and capitalism in the age of social media*. Routledge.

Shao, Nan, Zefan Cai, Hanwei xu, Chonghua Liao, Yanan Zheng, and Zhilin Yang (2023). „Compositional Task Representations for Large Language Models“. In: *The Eleventh International Conference on Learning Representations*.

Sharma, Rishi, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh (2018). „Tackling the story ending biases in the story cloze test“. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 752–757.

Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean (2017a). „Outrageously large neural networks: The sparsely-gated mixture-of-experts layer“. In: *arXiv preprint arXiv:1701.06538*.

Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean (2017b). „Outrageously large neural networks: The sparsely-gated mixture-of-experts layer“. In: *arXiv preprint arXiv:1701.06538*.

Shen, Sheng, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, *et al.* (2023). „Mixture-of-experts meets instruction tuning: A winning combination for large language models“. In: *arXiv preprint arXiv:2305.14705*.

Shnitzer, Tal, Anthony Ou, Mırian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin (2023). „Large Language Model Routing with Benchmark Datasets“. In: *arXiv preprint arXiv:2309.15789*.

Shringarpure, Suyash S and Carlos D Bustamante (2015). „Privacy risks from genomic data-sharing beacons“. In: *The American Journal of Human Genetics* 97.5, pp. 631–646.

Shrotri, Madhumita, Tui Swinnen, Beate Kampmann, and Edward PK Parker (2021). „An interactive website tracking COVID-19 vaccine development“. In: *The Lancet Global Health* 9.5, e590–e592.

Sims, Matthew and David Bamman (2020). „Measuring information propagation in literary social networks“. In: *arXiv preprint arXiv:2004.13980*.

Smolensky, Paul (1990). „Tensor product variable binding and the representation of symbolic structures in connectionist systems“. In: *Artificial intelligence* 46.1-2, pp. 159–216.

Solove, Daniel J (2004). *The digital person: Technology and privacy in the information age*. Vol. 1. NyU Press.

Sørensen, Jannick and Sokol Kosta (2019). „Before and after gdpr: The changes in third party presence at public and private european websites“. In: *The World Wide Web Conference*, pp. 1590–1600.

Stier, Sebastian, Nora Kirkizh, Caterina Froio, and Ralph Schroeder (2020). „Populist attitudes and selective exposure to online news: A cross-country analysis combining web tracking and surveys“. In: *The International Journal of Press/Politics* 25.3, pp. 426–446.

Strezoski, Gjorgji, Nanne van Noord, and Marcel Worring (2019). „Many task learning with task routing“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1375–1384.

Su, Zhan, Rasmus Helles, Ali Al-Laith, Antti Veilahti, Akrati Saxena, and Jakob Grue Simonsen (2023). „Privacy Lost in Online Education: Analysis of Web Tracking Evolution“. In: *International Conference on Advanced Data Mining and Applications*. Springer, pp. 440–455.

Sung, Yi-Lin, Varun Nair, and Colin A Raffel (2021). „Training neural networks with fixed sparse masks“. In: *Advances in Neural Information Processing Systems* 34, pp. 24193–24205.

Suzgun, Mirac, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, *et al.* (2022). „Challenging big-bench tasks and whether chain-of-thought can solve them". In: *arXiv preprint arXiv:2210.09261*.

Team, Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, *et al.* (2023). „Gemini: a family of highly capable multimodal models". In: *arXiv preprint arXiv:2312.11805*.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, *et al.* (2023). „Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971*.

Tow, Jonathan, Marco Bellagente, Dakota Mahan, and Carlos Riquelme (2023). *StableLM 3B 4E1T*.

Tufekci, Zeynep (2017). *Twitter and tear gas: The power and fragility of networked protest*. Yale University Press.

Urban, Tobias, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann (2020). „Measuring the impact of the gdpr on data sharing in ad networks". In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 222–235.

Urman, Aleksandra and Mykola Makhortykh (2023). „You are how (and where) you search? Comparative analysis of web search behavior using web tracking data". In: *Journal of Computational Social Science*, pp. 1–16.

Vandenhende, Simon, Stamatios Georgoulis, Marc Proesmans, Dengxin Dai, and Luc Van Gool (2020). „Revisiting multi-task learning in the deep learning era". In: *arXiv preprint arXiv:2004.13379* 2.3.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017a). „Attention is all you need". In: *Advances in neural information processing systems* 30.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017b). „Attention is all you need". In: *Advances in neural information processing systems* 30.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017c). „Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762.

Victor, Sanh, Webson Albert, Raffel Colin, Bach Stephen, Sutawika Lintang, Alyafeai Zaid, Chaffin Antoine, Stiegler Arnaud, Raja Arun, Dey Manan, *et al.* (2022). „Multitask prompted training enables zero-shot task generalization". In: *International Conference on Learning Representations*.

Vlajic, Natalija, Marmara El Masri, Gianluigi M Riva, Marguerite Barry, and Derek Doran (2018). „Online Tracking of Kids and Teens by Means of Invisible Images: COPPA vs. GDPR". In: *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, pp. 96–103.

Voigt, Paul and Axel Von dem Bussche (2017). „The eu general data protection regulation (gdpr)". In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676, pp. 10–5555.

Vu, Tu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer (2021). „Spot: Better frozen model adaptation through soft prompt transfer“. In: *arXiv preprint arXiv:2110.07904*.

Vu, Tu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer (Nov. 2020a). „Exploring and Predicting Transferability across NLP Tasks“. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 7882–7926.

Vu, Tu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer (2020b). „Exploring and predicting transferability across NLP tasks“. In: *arXiv preprint arXiv:2005.00770*.

Wallingford, Matthew, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto (2022). „Task adaptive parameter sharing for multi-task learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7561–7570.

Wang, Haixin, Xinlong Yang, Jianlong Chang, Dian Jin, Jinan Sun, Shikun Zhang, Xiao Luo, and Qi Tian (2024). „Parameter-efficient tuning of large-scale multimodal foundation model“. In: *Advances in Neural Information Processing Systems* 36.

Wang, Xinyi, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni (2024). *Guiding Language Model Math Reasoning with Planning Tokens*. arXiv: 2310.05707 [cs.CL].

Wang, Yaqing, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao (2022). „AdaMix: Mixture-of-Adapter for Parameter-efficient Tuning of Large Language Models“. In: *arXiv preprint arXiv:2205.12410*.

Wang, Yizhong, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, *et al.* (2023). „How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources“. In: *arXiv preprint arXiv:2306.04751*.

Wang, Yizhong, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, *et al.* (2022a). „Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks“. In: *arXiv preprint arXiv:2204.07705*.

Wang, Yizhong, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, *et al.* (2022b). „Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks“. In: *arXiv preprint arXiv:2204.07705*.

Wang, Zhefeng, Enhong Chen, Qi Liu, Yu Yang, Yong Ge, and Biao Chang (2015). „Maximizing the coverage of information propagation in social networks“. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Wang, Zirui, Yulia Tsvetkov, Orhan Firat, and Yuan Cao (2020). „Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models“. In: *arXiv preprint arXiv:2010.05874*.

Wang, Zirui, Yulia Tsvetkov, Orhan Firat, and Yuan Cao (2021). „Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models“. In: *International Conference on Learning Representations*.

Wei, Jason, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le (2022). „Finetuned Language Models are Zero-Shot Learners“. In: *International Conference on Learning Representations*.

Wen, Yeming, Dustin Tran, and Jimmy Ba (2020). *BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning*. arXiv: 2002.06715 [cs.LG].

Wen, Yuxin, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein (2023). „Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery“. In: *arXiv preprint arXiv:2302.03668*.

Westerlund, Mika, Diane A Isabelle, and Seppo Leminen (2021). „The acceptance of digital surveillance in an age of big data“. In.

Wortsman, Mitchell, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, *et al.* (2022). „Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time“. In: *International Conference on Machine Learning*. PMLR, pp. 23965–23998.

Wortsman, Mitchell, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi (2020). „Supermasks in superposition“. In: *Advances in Neural Information Processing Systems* 33, pp. 15173–15184.

Wu, Junda, Tong Yu, Rui Wang, Zhao Song, Ruiyi Zhang, Handong Zhao, Chaochao Lu, Shuai Li, and Ricardo Henao (2024). „Infoprompt: Information-theoretic soft prompt tuning for natural language understanding“. In: *Advances in Neural Information Processing Systems* 36.

Xiong, Jiangmei, Yulin Hswen, and John A Naslund (2020). „Digital surveillance for monitoring environmental health threats: A case study capturing public opinion from twitter about the 2019 Chennai water crisis“. In: *International journal of environmental research and public health* 17.14, p. 5077.

Yadav, Prateek, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal (2023). „TIES-Merging: Resolving Interference When Merging Models“. In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Yang, An, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, *et al.* (2021). „M6-t: Exploring sparse expert models and beyond“. In: *arXiv preprint arXiv:2105.15082*.

Yang, Enneng, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo (2023). „Adatask: A task-aware adaptive learning rate approach to multi-task learning“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 9, pp. 10745–10753.

Yang, Jaewon and Jure Leskovec (2010). „Modeling information diffusion in implicit networks“. In: *2010 IEEE international conference on data mining*. IEEE, pp. 599–608.

Yang, Zhiju, Weiping Pei, Monchu Chen, and Chuan Yue (2022). „Wtagraph: Web tracking and advertising detection using graph neural networks“. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 1540–1557.

Yang, Zonghan and Yang Liu (2022). „On robust prefix-tuning for text classification“. In: *International Conference on Learning Representations*.

Ye, Qinyuan, Bill Yuchen Lin, and Xiang Ren (Nov. 2021a). „CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP“. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7163–7189.

Ye, Qinyuan, Bill Yuchen Lin, and Xiang Ren (2021b). „Crossfit: A few-shot learning challenge for cross-task generalization in nlp“. In: *arXiv preprint arXiv:2104.08835*.

Zadouri, Ted, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker (2023). „Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning“. In: *arXiv preprint arXiv:2309.05444*.

Zaken, Elad Ben, Shauli Ravfogel, and Yoav Goldberg (2021). „Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models“. In: *arXiv preprint arXiv:2106.10199*.

Zaremoodi, Poorya, Wray Buntine, and Gholamreza Haffari (2018). „Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation“. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 656–661.

Zellers, Rowan, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi (2019). „Hellaswag: Can a machine really finish your sentence?“ In: *arXiv preprint arXiv:1905.07830*.

Zhang, Jinghan, Shiqi Chen, Junteng Liu, and Junxian He (2023). „Composing parameter-efficient modules with arithmetic operations“. In: *arXiv preprint arXiv:2306.14870*.

Zhang, Qingru, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao (2023). „Adaptive budget allocation for parameter-efficient fine-tuning“. In: *arXiv preprint arXiv:2303.10512*.

Zhang, Renrui, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao (2023). „Llama-adapter: Efficient fine-tuning of language models with zero-init attention“. In: *arXiv preprint arXiv:2303.16199*.

Zhang, Xichen and Ali A Ghorbani (2020). „An overview of online fake news: Characterization, detection, and discussion“. In: *Information Processing & Management* 57.2, p. 102025.

Zhang, Yu and Qiang Yang (2021). „A survey on multi-task learning“. In: *IEEE Transactions on Knowledge and Data Engineering* 34.12, pp. 5586–5609.

Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, *et al.* (2023). „A survey of large language models“. In: *arXiv preprint arXiv:2303.18223*.

Zhao, Xiangyun, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu (2018). „A modulation module for multi-task learning with applications in image retrieval“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–416.

Zhao, Yan, Sheng Bin, Gengxin Sun, *et al.* (2022). „Research on information propagation model in social network based on BlockChain“. In: *Discrete Dynamics in Nature and Society* 2022.

Zhao, Zihao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh (2021). „Calibrate before use: Improving few-shot performance of language models“. In: *International Conference on Machine Learning*. PMLR, pp. 12697–12706.

Zhou, Kaiyang, Jingkang Yang, Chen Change Loy, and Ziwei Liu (2022). „Conditional prompt learning for vision-language models“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16816–16825.

Zhou, Wangchunshu, Canwen Xu, and Julian McAuley (2022). „Efficiently tuned parameters are task embeddings“. In: *arXiv preprint arXiv:2210.11705*.

Zhou, Yanqi, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, *et al.* (2022). „Mixture-of-experts with expert choice routing“. In: *Advances in Neural Information Processing Systems* 35, pp. 7103–7114.

Zhu, Beier, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang (2023). „Prompt-aligned gradient for prompt tuning“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15659–15669.

Zuboff, Shoshana (2023). „The age of surveillance capitalism“. In: *Social Theory Re-Wired*. Routledge, pp. 203–213.