On Resource Efficient Transfer Learning via End Task Aware Training

Lucio Mwinmaarong Dery

CMU-CS-24-136

July 2024

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Graham Neubig, Co-Chair
Ameet Talwalkar, Co-Chair
Zico Kolter
Luke Zettlemoyer (University of Washington & Meta)
Marc'Aurelio Ranzato (Google DeepMind)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by: SRI International under award number 21485; the Defence Science and Technology Agency under award number RPS1DST000EPO21000070; the National Science Foundation under award number IIS2046613; and DSO National Laboratories under award number DSOCO21021.. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Transfer Learning, Auxiliary Learning, Meta-Learning, Machine Learning Efficiency, Structured Pruning

For my family, friends and mentors

Abstract

Transfer learning is a machine learning (ML) paradigm where performance on a desired end task[1] is improved by exploiting "*knowledge*" from other tasks. The technique has become a critical workhorse driving many of the advances on the envelope of capabilities of machine learning models. The current formula is relatively simple – train a large model on large amounts of data from the transfer task(s); then apply the learned model either zero-shot or adapted to the desired downstream task(s).

This thesis recognizes that these powerful models are not developed in-vacuo but rather require non-trivial resources to train and deploy. As such, there are a wide range of salient problems and communities of researchers that the status-quo leaves behind. In the first part of this thesis, we will focus on the training time problem of data-efficient transfer learning. We will begin by making a case for exploiting advanced knowledge of the desired downstream task(s) – which is commonly the case in many ML settings – to inform different dimensions of transfer learning. We dub this end task aware transfer learning. Next, we will present a set of novel end task aware optimization algorithms that bias the learning trajectory towards data-efficient solutions with strong generalization on the end task. We will close this part by providing an automated approach to constructing and searching over task-relevant transfer objectives when only end task data is available and in limited amounts.

For the second section of this thesis, we will develop algorithms for compute and memory efficient transfer learning. Our goal will be to deliver a small and efficient yet performant task specific model for deployment seeded from a large, generalist model that has already been pre-trained on a transfer task (or set of tasks). Focusing on structured pruning as the technique for making models smaller, we will investigate pruning under two resource constrained settings: (1) limited task data, where we will exploit extra transfer tasks to learn pruning structures that, at the same task performance, lead to more compute and memory efficient models (2) settings of limited memory, where many of the classical pruning techniques break down because they require gradient-based optimization which can have prohibitive memory overhead.

This thesis concludes by presenting more avenues for future work on resource efficient transfer learning by building on our past work and suggesting novel branches of investigation.

[1] end task here may encompass an aggregated suite of tasks

Acknowledgments

In a multitude of ways, this thesis would not have been possible without the support, advice and encouragement of my advisors: Graham Neubig and Ameet Talwalkar. My PhD had a rough start. Due to mental health reasons, I had to take a leave of absence within the first two months of the program. I seriously questioned my ability to see the program through, and at one point, in a moment of despair, got my old job as a research engineer at Meta back. I distinctly remember a meeting I had with Graham and Ameet in December 2019; I planned on telling them that I was leaving the program permanently. However, this meeting ended up going the opposite direction, where I reaffirmed my commitment to the program. During that fateful meeting, I could feel the kindness that both my advisors had towards me, and I had renewed faith that they would lead me with care and understanding towards a successful and enriching PhD experience despite my fears about my mental health. I am lucky to have had Ameet and Graham as my advisors; I learned a lot from them about how to be a strong and creative researcher. Above all, they have been shining examples of what it means to be an incredibly smart and accomplished yet kind person and I hope I can mirror them throughout my career.

This thesis would not have been possible without the excellent feedback and advice from members of my thesis committee: Zico Kolter, Luke Zettlemoyer and Marc'Aurelio Ranzato. They have been integral parts of my PhD journey even outside of their duties as members of my committee. Zico was my first faculty contact at CMU and was instrumental in making me feel welcomed during my early days in the program. My time as a research engineer at Meta under Luke was basically a pre-doctoral program and a lot of what I learned from him were instrumental in the early days of my PhD. Working with Marc'Aurelio during my internship at Google DeepMind in 2022 was instrumental in shaping the latter parts of my PhD. I am excited to explore the next stage of my career learning from him and his team.

In producing the works featured in this thesis, I have had the good fortune of collaborating with – and thus receiving mentorship from – the following people: Yann Dauphin, David Grangier, Abram Friesen, Nando De Freitas, Yutian Chen, Awni Hannun, Aditi Raghunathan, Virginia Smith and Jean-Francois Kagey. Special thanks to Yann Dauphin for the many 1-1 mentorship meetings where I was lucky enough to tap into his incredible wisdom. I would also like to acknowledge the formative advice I have received from the following more seasoned researchers in the field of machine learning: Timnit Gebru, Moustapha Cisse, Navdeep Jaitly, Sammy Bengio, Natalie Schluter, Sara Hooker and Afshin Rostamizadeh. I have had the pleasure of mentoring a few junior researchers during my PhD. To Atharva Kulkarni, Steven Kolawole and Vashisth Tiwari, I want to say thank you for the privilege of being part of your academic journey. I have learned a lot about how to be a good guide from working with you.

My sanity as a PhD student has leaned heavily on the many groups and organizations I been apart of during my time at CMU. To the students in Graham and Ameet's group – thank you. I have learned a lot from you and I am proud to call such an incredible group of brilliant and kind people my friends and colleagues. To Paul Michel, Shruti Rihjwani, Patrick Fernandes, Amanda Bertsch, Simran Khanuja, Lindia Tjuatja, Shuyan Zhou, Anjali Kantharuban, Perez Ogayo, Emmy Liu, Frank F. Xu, Hao Zhu, Vijay Viswanathan, Zhiruo Wang and Zhengbao Jiang: Neulab go brrrrrrrr!. I will sorely miss the banter and food "wars" at Ameet's lab lunches with Mikhail Khodak, Jeremy Cohen, Junhong Shen, Gregory Plumb, Wayne Chi, Liam Li and Nari Johnson. To the "Wotchers" group: Mikhail Khodak, Shir Maimon, Mark Gillespie, Jalani Williams, Abhiram Kothapalli, Jalani Williams just to mention a few, thanks for the weekend watch parties which helped me fight the Monday scaries! Special shoutout to my CSD year mates – Asher Trockman, Justin Whitehouse, Praneeth Kacham, Long Pham – we did it, and we did it by being there for each other.

To my close friends, those I made here in Pittsburgh and before the program, this PhD is for you. You held me down when I was low, listened to my fears, wiped my tears and bought me shots of tequila to drown my sorrows. Laura Hubbard, Richard Mantey, Chala Fekadu Fufa, Adeshina Adesoji, Robel Mengistu, Rhodaline Benjamin-Addy, Efua Kumea Asibon, Job Nalianya, Pascal Odek, Allan Marube, Tamer Shabani, Nafia Chowdhury, Amaya Christie, Tolani Olarinre, Keith Mphuthi, Nahom Mossazghi, Ibrahim Kante, Selam Gano, Lydia Girmai, Uche Agwu, Victor Akinwande, Gaoussou Kebe and Jasmine Kwasa, I cannot thank you guys enough.

Finally, I would like to thank my family. My dad and mum have always advised me to follow my heart and my dreams. They have always been incredibly patient, loving and supportive and I owe them the world. To my sisters, Lester and Loreta – thank you for your love, smiles and grace; they have powered (and continue to) me through some of the darkest times in my journey.

Contents

List of Figures

List of Tables

Part I

Introduction and Preliminaries

Chapter 1

Introduction

Machine learning (ML) models are becoming increasingly more powerful, resulting in their widespread adoption across many task domains (Gururangan et al., 2020a; Liu et al., 2022), data modalities (Team et al., 2023; McKinzie et al., 2024) and end-user applications (Bommasani et al., 2021; Maslej et al., 2023). Arguably one of the key driving forces of this staggering pace of growth is transfer learning. In transfer learning, we seek to improve performance on a desired end task (or set of tasks) by leveraging *knowledge* from a different, *hopefully related* task (Bozinovski and Fulgosi, 1976; Pratt, 1992; Ruder et al., 2019). Many end-tasks we wish to solve have limited data or are too complex to directly specify or learn with a practical number of supervised samples. Transfer learning enables us to tackle such problems by not only providing proxy data but by also enabling efficient learning of complex tasks by exploiting their structural relationships with chosen transfer tasks (Thrun and Schwartz, 1994; Baxter, 2000).

Despite its successes, transfer learning in its modern realization can be prohibitively resource intensive. Take for instance the ubiquitous pretrain-then-adapt paradigm [1]. With this approach, increasingly larger models are first trained on increasing larger piles of data, with these models eventually being adapted to a wide swath of down-stream tasks (Liang et al., 2022) by finetuning (Devlin et al., 2018; Abnar et al., 2021), prompting (Brown et al., 2020a; Liu et al., 2023) or reinforcement learning from human feedback (RLHF) (Christiano et al., 2017). GPT-4 (Achiam et al., 2023), a popular model under this paradigm which is rumored to be over 1.7 trillion parameters in size [2], is estimated to have been trained on over 10 trillion tokens; a total of over $1e^{25}$ flops of compute ($\sim$ \$100M at the time). Though these colossal training costs are typically justified as amortized over many future end-tasks, the size of such models present a significant memory, latency, compute and energy burden upon deployment and thus beg the question of the true degree of resource savings.

This thesis is dedicated to the exploration of techniques for resource efficient transfer learning. We recognize that there not only exists a broad swath of ML practitioners who are resource constrained but also that there are many tasks that have built-in resource limits at both training and deployment time (e.g. tasks performed on edge devices tend to be memory bound). Even for institutions with the means to train and use large models, resource efficient transfer learning can bring significant financial savings and limit the strain placed on the environment through CO_2

[1] we will discuss different kinds of transfer learning in Chapter 2

[2] See here

emissions (Ligozat et al., 2022).

This thesis is concerned with three main resource dimensions: *data*, *compute* and *memory* at both training and deployment time. Our goal is to produce performant (encompassing tasks specific metrics like accuracy or F1) models whilst being aware of resource efficiency requirements at train and test times. A foundational insight we will exploit to achieve the above goal is that ML practitioners usually have some degree of *apriori* awareness of the end-task(s) that their models will be used for. This end task awareness, allows us to make informed design decisions that result in efficient yet strong models produced in a resource-conscious manner. Succinctly, this thesis is grounded in the following problem statement :

Given a particular end task $\mathbf{T}^*$*, how can we produce models that satisfy various performance criteria on* $\mathbf{T}^*$ *in a resource efficient manner by leveraging a set of transfer tasks* $\mathbf{T}_{\text{aux}}$.

The idea of end task aware transfer learning is in itself not new. Previous work has explored asymmetrical transfer in the settings of solving complex planning problems (Stone and Veloso, 1994), improving the performance of support vector machines (Wu and Dietterich, 2004) and constructing priors for bayesian linear regression (Raina et al., 2006). We are interested in expanding upon the existing literature and developing novel approaches that are tailored to the new, deep-learning dominated era (LeCun et al., 2015; Goodfellow et al., 2016). Unlike past work, we are not only focused on improving task metrics like accuracy or perplexity, we are also interested in achieving these improvements in a resource efficient manner. Below, we provide a high level overview of the different pieces of work featured in this thesis and how they relate to our defined goal.

1.1 Thesis Overview and Contributions

This thesis consist of three main parts. Figure 1.1 provides a pictorial summary of the works featured in this thesis.

`PART I` - Introduction and background Part I of this thesis (which includes this introduction) sets the stage for understanding transfer learning, end task awareness and how we thread these together when thinking about resource efficiency. In chapter 2 we provide some background on transfer learning and contextualize this thesis with respect to prior work.

`PART II` - Data efficient transfer learning We begin this part by making a case for end-task awareness in the transfer learning pipeline as a means of achieving the various forms of resource efficiency we are interested in. The contents of Chapter 3, which make this case, were published as Dery et al. (2021a) at ICLR 2022[3]. Continuing this part, we propose various approaches for improving data-efficiency not only with respect to the end task, $\mathbf{T}^*$, but also the set of auxiliary tasks $\mathbf{T}_{\text{aux}}$. To achieve more data-efficient transfer learning, we exploit end task

[3]https://openreview.net/forum?id=2bO2x8NAIMB

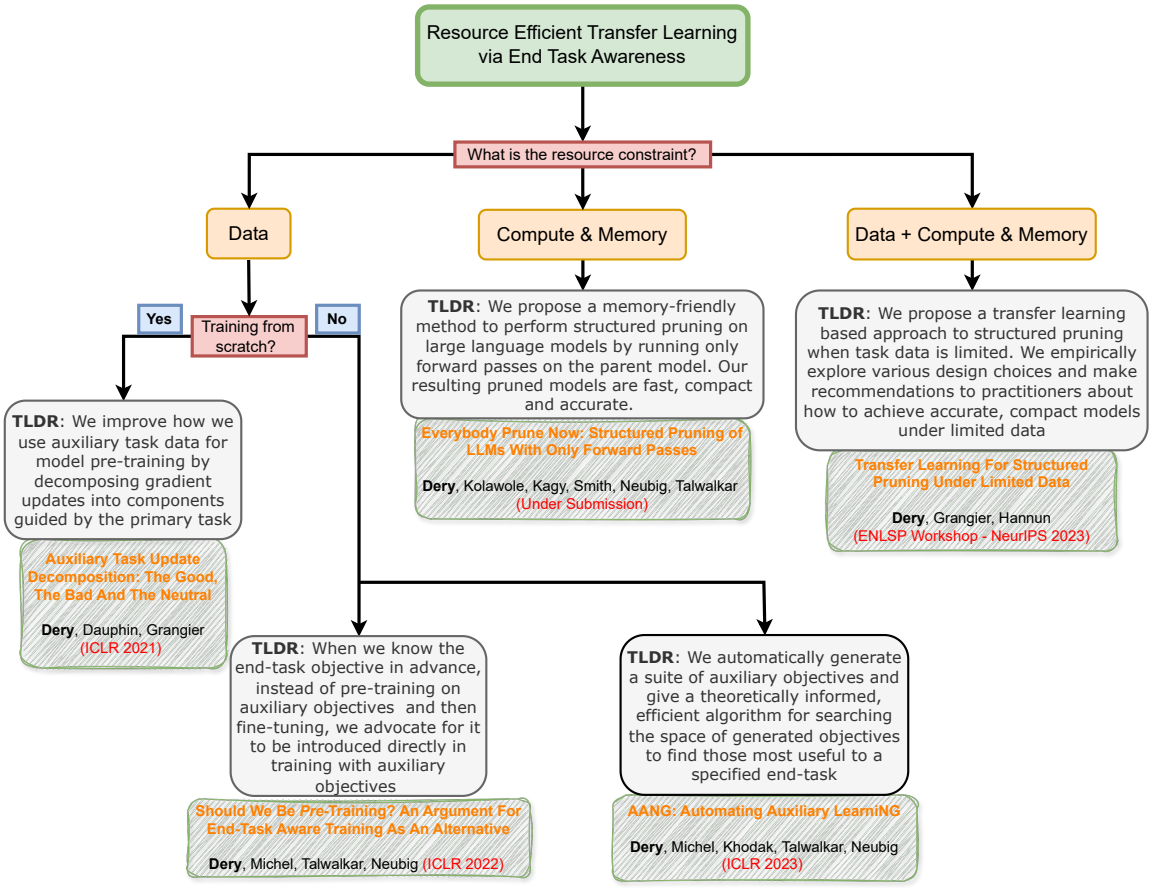


Figure 1.1: An overview of this thesis.

awareness both when considering how to construct the set of transfer tasks $\mathbf{T}_{\text{aux}}$ and how to optimize over this set to achieve improved performance on $\mathbf{T}^*$.

1. Optimizing over $\mathbf{T}_{\text{aux}}$ (Chapter 4): Tasks with smaller training sets often resort to pre-training or multitask learning to leverage data from other tasks. In this case, careful consideration is needed to select tasks and model parameterizations such that updates from the auxiliary tasks actually help the primary task. We seek to alleviate this burden by formulating a model-agnostic framework that performs fine-grained manipulation of the auxiliary task gradients. We propose to decompose auxiliary updates into directions which help, damage or leave the primary task loss unchanged. This allows weighting the update directions differently depending on their impact on the problem of interest. Our novel optimization algorithm allows us to improve data-efficiency by achieving higher performance at a fixed amount of end task data compared to strong baselines. This work was published as Dery et al. (2021b) at ICLR 2021[4].
2. Constructing $\mathbf{T}_{\text{aux}}$ (Chapter 5): Past work has generally assumed that the set $\mathbf{T}_{\text{aux}}$ is provided a-priori, and has focused on algorithms for using $\mathbf{T}_{\text{aux}}$ to improve $\mathbf{T}^*$. We present an approach for automatically generating a suite of auxiliary tasks. We achieve this by deconstructing existing objectives within a novel unified taxonomy, identifying connections between them, and generating new ones based on the uncovered structure. Our approach allows us to construct $\mathbf{T}_{\text{aux}}$ directly from end task data without having to introduce external data. This improves performance on $\mathbf{T}^*$ by using task data only, resulting in improved data-efficiency. This work was published as Dery et al. (2022) at ICLR 2023 [5].

`PART III` - Compute and memory efficient transfer learning As is the current status-quo, there exists a plethora of large models that have been pre-trained on massive datasets. The large size of these models prevent them from being applicable to tasks that are deployed in compute and memory constrained settings. For such tasks to benefit from transfer learning, these large models have to be pruned or compressed to desirably sizes. In lieu of compressing large models to smaller, task-agnostic models, we exploit end task awareness to achieve more drastic levels of compression. We concern ourselves not only with the compute and memory efficiency of the final target model, but also with the resource efficiency of the pruning process itself. This is different from most other end task aware compression methods that assume a resource rich setting during the pruning process itself.

1. Structured pruning of large pre-trained models when $\mathbf{T}^*$ is data-constrained (Chapter 6): While existing pruning algorithms can be efficient, the common practical setting where task-specific data is limited is yet to be addressed. To alleviate the data scarcity problem, we propose a structured pruning strategy that leverages transfer learning. Detailed analyses of simple transfer learning based remedies lead us to a simple, flexible formulation of what, how and when to transfer, resulting in pruned models with improved generalization over strong baselines under limited data for $\mathbf{T}^*$. This work was published as Dery et al. (2023) at ENLSP-III Workshop at NeurIPS 2023[6].

[4]https://openreview.net/forum?id=1GTma8HwlYp
[5]https://openreview.net/forum?id=vtVDI3w_BLL
[6]https://arxiv.org/pdf/2311.06382

2. Structured pruning of large pre-trained models when memory is constrained (Chapter 7): In this work, we consider the setting of pruning large pre-trained models under limited memory. This setting makes it infeasible to use existing structured pruning approaches that learn pruning variables via backward passes. We present a zeroth-order, bayesian-optimization inspired method for structured pruning that has significantly lower memory overhead. This allows a wider swath of practitioners to perform pruning of large pre-trained models on their own, memory-constrained hardware. This work is under submission at the time of handing in this thesis.

Chapter 2

Preliminaries and Background on Transfer Learning

In this chapter, we will paint the sub-field of transfer learning in broad strokes, with the goal of setting the stage for understanding the specific problem of interest that this thesis tackles. We will focus on transfer learning in the context of deep neural networks since they are currently the de facto model class used in machine learning. Throughout this section, we will underline categories to denote settings that we will focus on in this thesis.

2.1 Setup and Definitions

Consider a dataset $D = \{(x_i, y_i)_{i \in [m]}\}$ consisting of m data points sampled from some joint distribution $\mathcal{P}_{\text{data}} : \mathcal{X} \times \mathcal{Y} \rightarrow (0, 1) \mid \int \mathcal{P}_{\text{data}}(x, y)\text{dxdy} = 1$. In supervised machine learning y_i would correspond to a label whilst for unsupervised learning y_i would in general be some transformed version of x_i. As a concrete example, a dataset for product sentiment analysis would have x_i corresponding to a product review text whilst y_i is the customer sentiment expressed in the particular review. Informally, the goal of most of machine learning is to learn a model M_θ parameterized by θ, using D, that is able to map a given x_k to an appropriate $\tilde{y}_k$ such that $\mathcal{P}_{\text{data}}(x_k, \tilde{y}_k)$ is maximized.

We define a task as an objective function and dataset pair: $T = \{\mathcal{L}(\cdot), D\}$. The objective function $\mathcal{L}(y_i, M_\theta(x_i))$ evaluates how well a model prediction $M_\theta(x_i)$ fits the true label y_i, such as cross-entropy loss in the case of classification or ℓ_2 in regression. Note that the task dataset, D, is typically decomposed into the sets $(D^{\text{train}}, D^{\text{val}}, D^{\text{test}})$. D^{train} is the set of examples used for model training whilst D^{test} is used for final task evaluation. The validation set, D^{val}, is typically used for model selection but it is also frequently used in meta-learning (Schmidhuber, 1987; Naik and Mammone, 1992) to define the meta-objective – $\mathcal{L}^{val}$. How well a model performs on a task is usually measured by the loss on the test set (generalization error):

$$\mathcal{L}_{\text{test}} = \frac{1}{|D^{\text{test}}|} \sum_{(x_i, y_i) \in D^{\text{test}}} \mathcal{L}\big(y_i, M_\theta(x_i)\big)$$

2.2 Transfer Learning at a glance

Transfer learning typically involves learning a model, M_θ[1], not just from a single task but a set of tasks $\mathbf{T} = \{T_1, \ldots, T_\ell\}$. This set is either decided by the constraints of the practitioner setting, or selected for by the practitioner themselves based on an *a-priori* belief that this set of tasks can be mutually beneficial. Moving from a single task to learning from multiple has several advantages, including but not limited to:

1. providing proxy data. Some, or all of the tasks in $\mathbf{T}$ may be data starved. By pooling and learning from $\mathbf{T}$ instead of individually, transfer learning effectively expands the amount of data available for learning any individual task T_i (Baxter, 2000; Hutchinson et al., 2017; Khanuja et al., 2023).
2. regularizing / biasing training. Whilst there may be many models that achieve low loss on the training data of any target task T_i, there are other properties like generalization to unseen examples and robustness to worst case group error that we would like our final task model to possess. As shown in works like Caruana (1997a); Sener and Koltun (2018); Kulkarni et al. (2023), transfer learning can constrain/regularize the solution space during neural network training so that surfaced solution satisfies other desirable criteria.
3. improving learnability. Some task are hard to learn due to the twin difficulties of uncovering key features and discerning their complex relationship with the task output. Transfer learning makes it easier to learn such tasks when they are paired with other tasks where discovering said key features are easier. Ruder (2017) call this eavesdropping.

2.2.1 Broad categorizations of transfer learning problems

There are many ways to describe the landscape of transfer learning problems. One axis for categorizing problems is to consider the practitioner end-goal with respect to $\mathbf{T}$.

Symmetric / Multitasking Learning (MTL): In MTL, the practioner cares about delivering a model that performs well across an aggregation of all the tasks in $\mathbf{T}$: $\mathcal{L}_{\text{total}} = \sum_{\{T_i \in \mathbf{T}\}} \alpha_i \mathcal{L}_{\text{test}}^{T_i}$ is minimized under M_θ. $\{\alpha_i\}$ are either chosen by the practitioner or learned in some settings like in Distributionally Robust Optimization (Duchi and Namkoong, 2021). The objective is to obtain a single model that outperforms models trained in a single task fashion.

Asymmetric / Auxiliary Learning: In this setting, not all the tasks in $\mathbf{T}$ are created equal. Specifically, there is a proper, non-empty subset $\mathbf{T}^* \subset \mathbf{T}$, we call the target set, that the practitioner cares about and all other tasks in $\mathbf{T}_{\text{aux}} = \{\mathbf{T} \setminus \mathbf{T}^*\}$ are solely in service of achieving good performance on the target set. The metric of concern is therefore $\mathcal{L}_{\text{total}} = \sum_{\{T_j \in \mathbf{T}^*\}} \alpha_j \mathcal{L}_{\text{test}}^{T_j}$. $\{\alpha_i\}$ can be treated as discussed under MTL. This thesis will primarily focus on this asymmetrical learning setting. We will exploit advanced knowledge of $\mathbf{T}^*$ during learning to achieve resource efficiency in transfer learning. We will discuss this further in Chapter 3.

[1] maybe even a set of models but for simplicity, and based on what has most coverage in practice, we will focus on the single model case.

Another common way of categorizing transfer learning problems is based on the relationship between the input and output spaces of the tasks in $\mathbf{T}$ (Zhuang et al., 2020; Bao et al., 2023).

Homogeneous Transfer: When the input spaces and output spaces of the tasks satisfy $\{\mathcal{X}_1 = \ldots = \mathcal{X}_\ell\}$ and $\{\mathcal{Y}_1 = \ldots = \mathcal{Y}_\ell\}$ respectively, the transfer learning problem is classified as homogeneous. An example of such a transfer problem is an image digit classification problem where each task has images from different domains eg. CIFAR (Krizhevsky and Hinton, 2010) and MNIST (LeCun et al., 1995).

Heterogeneous Transfer: If $\exists\ (i,j) \in [\ell], s.t \quad i \neq j\ \&\&\ \mathcal{Y}_i \neq \mathcal{Y}_j$ (or similarly for any pair of input spaces), we refer to this as heterogeneous transfer. An example problem of this type would be sentiment classification using both audio and text reviews of the same product.

The problems we explore in this thesis will have homogeneous input spaces but flexible (both heterogeneous and homogeneous) output spaces. Note that in general, it is possible to convert a problem with heterogeneous input spaces into a homogeneous one by learning a shared, intermediate representation space — for example as is done in cross-modal transfer learning (Shen et al., 2023).

2.3 Design Choices during Transfer Learning

There are myriad design choices to consider when performing transfer learning. An overarching theme of this thesis is that certain design choices made during transfer learning can allow the practitioner to be *efficient* with respect to desired choices of resources. One way of making sure that the appropriate choice is made – which this thesis strongly advocates for – is by leveraging advanced knowledge of the desired end-task(s). Below, we discuss some of the options available to practitioners along a subset of relevant design dimensions.

2.3.1 What tasks are in T

As already mentioned, when $\mathbf{T}$ is not induced by the setting, selecting the tasks that make up $\mathbf{T}$, usually involves practitioner intuition. We will see in Chapters 4 and 5, that they can be wrong about which sets of tasks are mutually beneficial, and in such cases data-efficiency and generalization can be harmed. Unfortunately, there is no single, universally accepted notion of task relatedness that practitioners can mechanistically follow, given their desired objectives, to construct $\mathbf{T}$. We can however categorize attempts at defining task relatedness into two camps: axiomatic and empirical.

Empirical approaches as in (Zamir et al., 2018; Wang et al., 2018a) seek to build graphs of the relationships between classes of tasks by experimentally verifying whether task pairings (or orderings) result in positive, neutral or negative outcomes with respect to the performance of final model. Whilst these graphs enjoy the advantage of being re-usable across different transfer problem instances, they are computationally prohibitive to construct. Axiomatic approaches tend to propose a (typically) problem conducive definition of task relatedness based on formal mathematical measures. For example, an information-theoretic choice of measure of relatedness is the Kullback-Leibler divergence (or any other suitable choice of distance measure) $D_{\mathrm{KL}}(\mathcal{P}^{T_i}_{\mathrm{data}} \| \mathcal{P}^{T_j}_{\mathrm{data}})$ between the data generating distributions, whilst one optimization inspired

measure is $\|\nabla_\theta \mathcal{L}\left(D^{T_i}, M_\theta\right) - \nabla_\theta \mathcal{L}\left(D^{T_j}, M_\theta\right)\|$ which we will use in Chapter 5. When chosen carefully, such measures can be easy to compute and dynamic – capturing the evolving task relationships throughout model training.

2.3.2 The order of Transfer

Given the tasks in $\mathbf{T}$, practitioners need to decide on how to train M_θ: whether all together or in a chosen order. In **Sequential** training, tasks are trained in an order that is predetermined by the practitioner. The popular pretraining-then-finetune paradigm (Devlin et al., 2018; Abnar et al., 2021) is an example of this setting where we first train on the pre-training task and the resulting model is further adapted to the downstream task. Sequential training can be beneficial in the case of tasks that conflict when trained together. However, the more tasks in $\mathbf{T}$, the more choices there exist for task orderings (Zamir et al., 2018) which increases the depth of design choices.

For **joint training**, the model M_θ is trained on all the tasks in $\mathbf{T}$ in tandem, with with training biased towards the overall objective – the problem may be symmetric or asymmetric amongst the tasks. Joint training has the advantage of avoiding having to make task ordering decisions, and allowing task information to interact throughout training and not just sequentially which can yield more performant models (Chen et al., 2021; Kulkarni et al., 2023).

2.3.3 What to share or transfer

Practitioners have a choice of what information to share between the tasks in $\mathbf{T}$. Some of these choices include the following:

Features: As previously mentioned, some features are difficult to compute for certain tasks either due to limited data or complex relationships between said features and task outputs. Learning from multiple tasks is an avenue for sharing features between tasks (Peters et al., 2018a; Kumar et al., 2022). Joint training in particular can also serves to constraint feature spaces, allowing models to focus on those feature sets that enable good performance across several tasks and are thus more likely to be the truly causal set of task features instead spurious ones (Arjovsky et al., 2019; Rosenfeld et al., 2020; Huh et al., 2024).

Model weights: Learned model weights for one task can serve as a strong initialization for another task as common in the pretrain-then-finetune paradigm. Many works have shown that transferring model weights can result in improved generalization (Abnar et al., 2021), more robust solutions (Hendrycks et al., 2019) and data-efficiency (Gururangan et al., 2020a) with respect to downstream task.

Model architecture / structure: Works in the sub-fields of structured pruning (Xia et al., 2023) and neural architecture search (Gao et al., 2020, 2024) exploit transfer learning to discover good architectures for the practitioner end-task(s). Not only do these structured facilitate performant models, they also allow the discovery of model that meet inference time memory and latency constraints.

In this thesis, we will design transfer learning approaches that are targeted at sharing all these different modes.

Part II

On Data-Efficient Transfer Learning

Chapter 3

A Case For End Task Aware Transfer learning

3.1 Chapter Overview

In most settings of practical concern, machine learning practitioners know in advance what end-task they wish to boost with auxiliary tasks. However, widely used methods for leveraging auxiliary data like pre-training and its continued-pretraining variant are end-task agnostic: they rarely, if ever, exploit knowledge of the target task. Because of this, practitioners have to be careful with their choice of auxiliary tasks, the order in which they are trained on, and the early-stopping criteria for each pre-training stage so as to actually achieve good downstream end task performance (Zamir et al., 2018; Abnar et al., 2021). In the absence of principled criteria to make these difficult design choices, it is common to instead resort to the resource intensive heuristic of pre-training on as much data as possible on as large a model as possible.

We begin this part of the thesis by making a case for replacing end task agnostic continued training of pre-trained language models with end task aware training of said models. We argue that for sufficiently important end-tasks, the benefits of leveraging auxiliary data in a task-aware fashion can justify forgoing the traditional approach of obtaining generic, end-task agnostic representations as with (continued) pre-training. By the end of this chapter, we will demonstrate that multi-tasking the end-task and auxiliary objectives results in significantly better downstream task performance and data-efficiency than the widely-used task-agnostic continued pre-training paradigm of Gururangan et al. (2020a).

3.2 Introduction

The increasingly popular pre-training paradigm (Dai and Le, 2015; Devlin et al., 2018; Gururangan et al., 2020a) involves first training a *generalist* model on copious amounts of easy-to-obtain data, e.g. raw text data in NLP, and then using this model to initialize training on a wide swath of downstream tasks. Generalist models like BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019a), and GPT-3 (Brown et al., 2020a) have a strong appeal; a few institutions with significant resources incur the cost of training these large models whilst the rest of the research community

enjoys a significant performance improvement at minimal computational overhead. However, the advantages of initializing a downstream task from a generalist model are not guaranteed. Previous work has shown that the benefits of pre-training depend heavily on the degree of domain overlap between the end-task data and the massive, heterogenous data on which the generalist model was trained (Beltagy et al., 2019a; Gururangan et al., 2020a).

Notably, Gururangan et al. (2020a) have demonstrated the benefits of continued pre-training of generalist models using data that is similar to that of the end-task. Their approach is formalized into two classes: Domain Adaptive Pre-training (DAPT) and Task Adaptive Pretraining (TAPT) where further stages of pre-training of generalist models are conducted on domain- and task-specific data, respectively. DAPT and TAPT exploit the fact that *we often know the end-task beforehand*, and so we can make specific choices about our pre-training regimen to improve end-task performance.

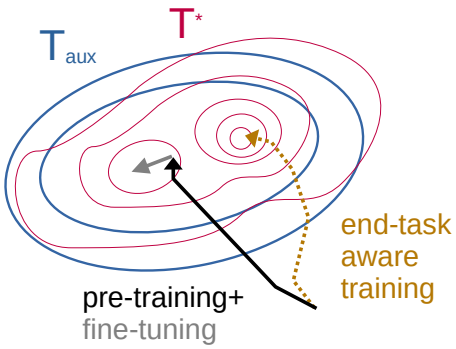


Figure 3.1: Pre-training trains on auxiliary task T_{aux} before fine-tuning on primary task T^*. End-task aware training optimizes both T_{aux} and T^* simultaneously and can find better minima since optimization is informed by the end-task.

However, in both pre-training for generalist models and continued pre-training, the training procedure itself does not explicitly incorporate the *end-task objective function*. Because of this, practitioners have to be careful with their choice of auxiliary tasks, the order in which they are trained on, and the early-stopping criteria for each pre-training stage so as to actually achieve good downstream end-task performance (Gururangan et al., 2020a; Dery et al., 2021b). In the absence of principled criteria to make these difficult design choices, it is common to instead resort to the computationally demanding heuristic of pre-training on as much data as possible for as long as possible.

In this Chapter, we raise the following question: "*In settings where we have a particular end-task in mind, should we be **pre**-training at all?*". We define ***pre***-training as any form of task-agnostic training that a model undergoes before it is finally fine-tuned on the end-task of interest. As a first milestone in addressing the larger question posed above, we explore the ubiquitous continued pre-training setting (Gururangan et al., 2020a; Aghajanyan et al., 2021). Specifically, our work questions the wisdom of having disjoint *further pre*-training then fine-tuning steps on a *generalist* model. In response, we advocate for an alternative approach in which we directly introduce the end-task objective of interest into the learning process. This results in a suite of end-task aware methods called TARTAN (end-**T**ask **A**wa**R**e **T**r**A**ini**N**g). Our formulations incorporate both unsupervised auxiliary objectives traditionally used in NLP pre-training (such as masked language modeling as in Devlin et al. (2018)) *and* the end-task objective, followed by an optional fine-tuning step on the end-task. We motivate TARTAN experimentally in the continued pre-training setting and based on this, we make the following contributions to the literature on leveraging auxiliary tasks and data:

- In lieu of standard end-task agnostic continued pre-training, we suggest introducing the

end-task objective into the training process via multi-task learning (Caruana, 1997a; Ruder, 2017). We call this procedure **M**ulti-**T**asking end-**T**ask **A**wa**R**e **T**r**A**ini**N**g(MT-TARTAN) (Section 3.4.1). MT-TARTAN is a simple yet surprisingly effective alternative to task-agnostic pre-training. In Section 3.6, we demonstrate that MT-TARTAN significantly improves performance and data efficiency over Gururangan et al. (2020a)'s results. It also obviates the need for fickle hyper-parameter tuning through direct optimization of validation performance.

- To allow more fine-grained control of the end-task over the auxiliary tasks, in Section 3.4.2, we present an online meta-learning algorithm that learns adaptive multi-task weights with the aim of improving final end-task performance. Our **META**-learning end-**T**ask **A**wa**R**e **T**r**A**ini**N**g(META-TARTAN) allows us to robustly modulate between multiple objectives and further improves performance over MT-TARTAN .
- A naive implementation of META-TARTAN based on first-order meta-learning analysis results in a sub-optimal algorithm that ignores all tasks except the end-task. We trace this problem to the use of a single model training head for computing both the end-task training loss and meta-objective (end-task validation loss). To guard against this pathological solution, we introduce a separate model head for computing the meta-objective. In Section 3.4.3, we justify this simple-to-implement fix and validate its practical efficacy in Section 3.6.

Our results suggest that TARTAN may be an attractive alternative to the continued pre-training paradigm, and further research into the place of *pre*-training in end-task aware settings is warranted.

3.3 Formalizing Pre-training and Continued Pre-training

We restate the formalism established in Chapter 2 for ease of access.

Consider a dataset $D = \{(x_i, y_i)_{i\in[m]}\}$ consisting of m labelled examples. We define a task as an objective function and dataset pair: $T = \{\mathcal{L}(\cdot), D\}$. M_θ is a model parameterized by θ. The objective function $\mathcal{L}(y_i, M_\theta(x_i))$ evaluates how well a model prediction $M_\theta(x_i)$ fits the true label y_i, such as cross-entropy loss in the case of classification. Note that the task dataset, D, is typically decomposed into the sets $(D^{\text{train}}, D^{\text{val}}, D^{\text{test}})$. D^{train} is the set of examples used for model training whilst D^{test} is used for final task evaluation. The validation set, D^{val}, is typically used for model selection but it is also frequently used in meta-learning to define the meta-objective – $\mathcal{L}^{val}$.

Given a specific end-task T^*, our aim is to improve performance on T^* (as measured by the model loss on $D^{\text{test}}_{T^*}$) by leveraging auxiliary tasks $\mathbf{T}_{\text{aux}} = \{T_1, \ldots, T_n\}$. Note that we do not particularly care about the performance of any of the tasks in $\mathbf{T}_{\text{aux}}$. We are willing to sacrifice performance on $\mathbf{T}_{\text{aux}}$ if it improves performance on T^*.

From the perspective of model architecture, there are several ways to leverage $\mathbf{T}_{\text{aux}}$. We focus on the simple but widely-used parameter sharing setting. Here, all tasks share a model body θ_{body} but each task T_i has its own head ϕ^i for prediction. We denote the head belonging to T^* as ϕ'. Thus $\theta = \left[\theta_{\text{body}}; \left(\phi^1, \ldots, \phi^n, \phi'\right)\right]$ and θ_{body} is reusable across new tasks.

3.3.1 Pre-training

Pre-training is when a model is first trained on $\mathbf{T}_{\text{aux}}$ before performing a final fine-tuning phase on T^*. The motivation behind pre-training is that learning $\mathbf{T}_{\text{aux}}$ first hopefully captures relevant information that can be utilized during training of T^*. This desire has led to the proliferation of generalist pre-trained models like BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019a) and GPT-3 (Brown et al., 2020a) that have been trained on copious amounts of data. Generalist models have been widely successful at improving downstream task performance when used as initilization.

We can formalize the pre-training procedure as follows:

$$\theta_0 = \text{argmin}_\theta \left(\sum_{T_i \in \mathbb{T}_{\text{aux}}} \mathcal{L}_{T_i}(\theta) \right) \tag{3.1}$$

In Equation 3.1, we seek a point θ_0 that achieves minimal loss on the tasks in $\mathbf{T}_{\text{aux}}$. We hope that θ_0 will be a good starting point for gradient descent on T^*. Let $g(\theta_0)$ represent the set of end-points of stochastic gradient descent on an initialization, θ_0. Stochastic gradient descent from the same initialization can produce different end-points due to differences in hyper-parameters like learning rate, batch size and order, as well as regularization strength. We can write the fine-tuning phase as:

$$\theta^* = \text{argmin}_{\{\theta \in g(\theta_0)\}} \mathcal{L}_{T^*}(\theta) \tag{3.2}$$

Note that pre-training is *end-task agnostic*: the pre-training Equation 3.1 occurs entirely before training on the end-task Equation 3.2, and does not explicitly incorporate the end-task objective, T^*. Since there is no awareness of the end-task during pre-training it is important to carefully choose $\mathbf{T}_{\text{aux}}$ so that pre-training actually results in improved performance on T^* (Wang et al., 2018a). For text data, past work has found left-to-right language modeling (Peters et al., 2017) and masked language modeling (MLM) (Devlin et al., 2018) to be good choices to include in $\mathbf{T}_{\text{aux}}$.

3.3.2 Continued Pre-training

Recent work (Beltagy et al., 2019a; Gururangan et al., 2020a; Lee et al., 2020) showed that downstream performance on T^* can be improved by further adapting generalist models via continued pre-training on a more relevant set of auxiliary tasks. This is equivalent to sequentially performing multiple steps of Equation 3.1, with different $\mathbf{T}_{\text{aux}}$, before finally performing Equation 3.2 on T^*.

Domain and Task Adaptive Pre-training Gururangan et al. (2020a) present Domain Adaptive Pre-Training (DAPT) and Task Adaptive Pre-Training (TAPT) as methods for continued pre-training. During DAPT, a generalist model is further pre-trained on an unsupervised objective with large amounts of data from the same domain as the end-task. TAPT also pre-trains with the same unsupervised objective as DAPT, but on the actual dataset of the end-task. Gururangan et al. (2020a) find that performance can be further improved by chaining objectives, DAPT first, followed by TAPT.

Though TAPT and DAPT do not directly incorporate the end-task objective during training, it still indirectly informs both the choice of pre-training data and the order in which the pre-training tasks are trained on. Below, we explore stronger versions of this influence.

3.4 End-Task Aware Training (TARTAN)

In this section, we argue for the end-task to be added directly into the training process to create explicit interactions between T^* and $\mathbf{T}_{\text{aux}}$.

3.4.1 End-task Aware Training via Multi-tasking (MT-TARTAN)

We propose to directly incorporate knowledge of the end-task by multi-tasking T^* together with $\mathbf{T}_{\text{aux}}$, before optionally fine-tuning on T^* exclusively. To this end, we introduce a set of task weights $\mathbf{w} = (w^*, w_1, \cdots, w_{|\mathbf{T}_{\text{aux}}|})$ satisfying $w^* + \sum_i w_i = 1$, to modulate between the different losses. Our new formulation is:

$$\theta_0 = \text{argmin}_\theta \ \mathcal{L}_{\text{total}}(\theta, \mathbf{w}) = \text{argmin}_\theta \left(w^* \mathcal{L}_{T^*}(\theta) + \sum_i w_i \mathcal{L}_{T_i}(\theta) \right) \tag{3.3}$$

Here, Equation 3.3 replaces Equation 3.1 and can be followed by the optional fine-tuning stage of Equation 3.2. Note that this formulation fixes the tasks weights $\mathbf{w}$ throughout the training process. We call this formulation End-task Aware Training via Multi-tasking (MT-TARTAN) because we introduce the end-task **directly** into the training procedure, and do so by **multi-tasking** it with $\mathbf{T}_{\text{aux}}$.

MT-TARTAN allows us to prioritize performance on T^* in several ways. First, we can weight the end-task higher than all the other auxiliary tasks. Also, during training, we can monitor $\mathcal{L}_{T^*}$ on the end-task validation set and early stop when it plateaus; even if the auxiliary tasks have not yet converged. This is not possible during standard pre-training because we do not train T^* and so it performs at random before we actually start fine-tuning. Early stopping on T^* can represent significant computational savings over end-task agnostic pre-training when the savings in data-efficiency supersede the extra overhead of end-task aware gradient descent steps.

3.4.2 End-task Aware Training via Meta-learning (META-TARTAN)

MT-TARTAN, DAPT and TAPT, all share the same drawback: they implicitly assume that the auxiliary tasks have static importance to the end-task over the lifetime of its training, either by being end-task agnostic (DAPT and TAPT) or by having static task weights (MT-TARTAN). With MT-TARTAN, an additional drawback noted by Wang et al. (2019a); Yu et al. (2020) is that multi-tasking can negatively impact task performance compared to isolated training. These shortcomings motivate the formulation of an adaptive algorithm that can mitigate the negative influence of some tasks whilst responding to the changing relevance of auxiliary tasks over the lifetime of end-task training.

As they stand, the pre-training equation pair (Equations 3.1, 3.2) and the MT-TARTAN pair (Equations 3.2, 3.3) are decoupled. The inner-level variables of the pre-training phase do not

depend on the outer-level variables of the fine-tuning phase. Thus the equation pairs are typically solved sequentially. We propose to tightly couple Equations 3.2 and 3.3 by formulating jointly learning $\mathbf{w}$ and θ_0 as a bi-level optimization problem. A bi-level formulation allows us to leverage meta-learning (Schmidhuber, 1995) techniques to learn adaptive task weights which capture variable auxiliary task importance whilst mitigating the contribution of harmful tasks. We propose a meta-learning algorithm in the mold of Model Agnostic Meta-Learning (MAML) (Finn et al., 2017a) to learn task weights. As a bi-level problem, this can be formulated as :

$$\theta^*, \mathbf{w}^* = \text{argmin}_{\{\theta \in g(\theta_0),\, \mathbf{w}\}}\, \mathcal{L}_{T^*}(\theta) \tag{3.4}$$

where

$$\theta_0 = \text{argmin}_\theta\ \mathcal{L}_{\text{total}}(\theta, \mathbf{w}) = \text{argmin}_\theta\ \left(w^* \mathcal{L}_{T^*}(\theta)\ + \sum_{T_i \in \mathbf{T}_{\text{aux}}} w_i \mathcal{L}_{T_i}(\theta) \right) \tag{3.5}$$

We want to jointly learn $\mathbf{w}$, with θ_0, such that taking a gradient descent step modulated by $\mathbf{w}$ leads to improvement in end-task generalization. We use performance on the end-task validation set ($D^{\text{val}}_{T^*}$) as a meta-objective to train $\mathbf{w}$. Performance on $D^{\text{val}}_{T^*}$ serves as a stand-in for end-task generalization performance whilst also naturally capturing the asymmetrical importance of T^*.

Our joint descent algorithm proceeds as follows. At each timestep t, we hold the task weights fixed and update θ_t based on $\nabla_\theta \mathcal{L}_{\text{total}}(\theta_t, \mathbf{w})$. We then proceed to update $\mathbf{w}$ via gradient descent on the end-task validation loss at θ_{t+1}. For this, we derive an approximation for $\nabla_w \mathcal{L}^{val}_{T^*}(\theta_{t+1}, \mathbf{w})$ below:

$$\begin{aligned}
\mathcal{L}^{val}_{T^*}(\theta_{t+1}(\mathbf{w})) &= \mathcal{L}^{val}_{T^*}\left(\theta_t - \beta\big(w^* \nabla \mathcal{L}_{T^*} + \sum_i w_i \nabla \mathcal{L}_{T_i} \big) \right) \\
&\approx \mathcal{L}^{val}_{T^*}(\theta_t) - \beta \left(w^* \nabla \mathcal{L}_{T^*} + \sum_i w_i \nabla \mathcal{L}_{T_i} \right)^T \nabla \mathcal{L}^{val}_{T^*}(\theta_t)
\end{aligned}$$

We can take the gradient of the above first-order approximation w.r.t an individual weight w_i. This tells us how to update w_i to improve the meta-objective.

$$\frac{\partial \mathcal{L}^{val}_{T^*}(\theta_{t+1}(\mathbf{w}))}{\partial w_i} \approx -\beta \big(\nabla \mathcal{L}_{T_i} \big)^T \big(\nabla \mathcal{L}^{val}_{T^*}(\theta_t) \big) = -\beta \big(\nabla \mathcal{L}_{T_i} \big)^T \big(\nabla \mathcal{L}^{val}_{T^*}(\big[\theta_{\text{body}}, \phi' \big]_t) \big) \tag{3.6}$$

In Equation 3.6, we explicitly specify $\big[\theta_{\text{body}}, \phi'\big]_t$ because computing losses on T^* depend on only these parameters. $\mathcal{L}_{T_i}$ depends solely on $\big[\theta_{\text{body}}, \phi^i\big]_t$ but we leave this out to avoid notation clutter.

Our analysis above is similar to that of Lin et al. (2019) with one key difference: we learn a weighting for the main task w^* too. This ability to directly modulate T^* allows us to capture the fact that at certain stages in training, auxiliary tasks may have greater impact on end-task generalization than the end-task's own training data. This choice also allows us to control for over-fitting and the influence of bad (mislabelled or noisy) training data.

3.4.3 Introducing a separate classification head for meta-learning

Observe that from Equation 3.6, updates for $w \neq w^*$ involve gradients computed from different model heads ϕ^i and ϕ' whilst for w^*, we are taking the dot product of gradients from the same end-task head ϕ'. As we will show empirically in Section 3.6.4, computing weight updates this way creates a strong bias towards the primary task, causing w^* to rail towards 1 whilst the other weights dampen to 0, which may be sub-optimal in the long run.

Intuitively, this short-horizon (greedy) (Wu et al., 2018) behavior makes sense: the quickest way to make short-term progress (improve $\mathcal{L}^{val}_{T^*}(\theta_{t+1})$) is to descend solely on T^*. More formally, the greedy approach arises because we derive $\nabla_{w_i}\mathcal{L}^{val}_{T^*}(\theta_{t+1})$ in Equation 3.6 as a proxy for the gradient at θ^*, the outer-loop end-point in Equation 3.4. Variations of this substitution are common in the meta-learning literature (Finn et al., 2017a; Liu et al., 2018a; Nichol et al., 2018) because it is computationally infeasible to train a model to convergence every time we wish to compute $\nabla_{w_i}\mathcal{L}^{val}_{T^*}(\theta^*)$.

To remedy the greedy solution, instead of estimating $\nabla_\theta\mathcal{L}_{T^*}$ and $\nabla_\theta\mathcal{L}^{val}_{T^*}$ from the same classification head (Equation 3.6), we introduce a special head ϕ^* for computing the meta-objective. Specifically, instead of trying to compute θ^*, we approximate it by fixing the body of the network θ_{body} and training the randomly initialized head ϕ^* to convergence on a subset of the end-task training data. We do this every time we wish to estimate $\nabla_{w_i}\mathcal{L}^{val}_{T^*}(\theta^*)$. Introducing ϕ^* eliminates the strong positive bias on w^* and enables us to compute a better proxy for the meta-gradient at θ^*:

$$\frac{\partial\mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))}{\partial w_i} \approx \left(\nabla_\theta\mathcal{L}_{T_i}\right)^T\left(\nabla_\theta\mathcal{L}^{val}_{T^*}([\theta_{\text{body}};\phi^*]_t)\right) \tag{3.7}$$

Equation 3.7 represents a simple-to-implement alternative to Equation 3.6. We provide a more detailed justification for Equation 3.7 in Appendix A.2. In Section 3.6.4, we empirically validate that the transition from Equation 3.6 to 3.7 improves performance whilst mitigating pathological solutions. Our approach of creating ϕ^* for approximating the meta-objective (downstream validation performance) is inspired by Metz et al. (2018), who use a similar technique to construct a meta-objective for evaluating the quality of unsupervised representations.

Please see Algorithm 3 in Appendix A.1 for details about META-TARTAN.

3.5 Experimental Setup

Setting[1] Though our algorithms and methodology can be directly applied to both continued pre-training (Section 3.3.2) and pre-training from scratch (Section 3.3.1) of *generalist* models, we focus on the former scenario. This is because the continued pre-training setting is more common amongst everyday practitioners as it is less computationally demanding. It thus lends itself more easily to exploration under a realistic computational budget. In Appendix A.4, we show that end-task aware training from scratch is viable by studying a simple computer vision setting. Concurrent work by Yao et al. (2021) shows that from-scratch end-task aware training for NLP problems is viable.

[1]Code is released at https://github.com/ldery/TARTAN

In keeping with previous work (Devlin et al., 2018; Gururangan et al., 2020a), we focus on $\mathbf{T}_{\text{aux}}$ as a set of MLM tasks on varied datasets. In the case of DAPT and our end-task aware variants of it, $\mathbf{T}_{\text{aux}}$ is an MLM task with data from the domain of the end-task. For TAPT, $\mathbf{T}_{\text{aux}}$ is an MLM task with data from the end-task itself. DAPT, TAPT and DAPT+TAPT (chained pre-training with DAPT followed by TAPT) will serve as our baseline continued pre-training approaches. We will compare these baselines to their end-task aware variants that use MT-TARTAN and META-TARTAN.

Datasets Our experiments focus on two domains: computer science (CS) papers and biomedical (BIOMED) papers. We follow Gururangan et al. (2020a) and build our CS and BIOMED domain data from the S2ORC dataset (Lo et al., 2019). We extract 1.49M full text articles to construct our CS corpus and 2.71M for our BIOMED corpus. Under both domains, our end-tasks are *low-resource* classification tasks. Using low-resource tasks allows us to explore a setting where pre-training can have a significant impact. Under the CS domain, we consider two tasks: ACL-ARC (Jurgens et al., 2018) and SCIERC (Luan et al., 2018). ACL-ARC is a 6-way citation intent classification task with 1688 labelled training examples. For SCIERC, the task is to classify the relations between entities in scientific articles. This task has 3219 labelled examples as training data. We choose CHEMPROT (Kringelum et al., 2016) as the classification task from the BIOMED domain. This task has 4169 labelled training examples and the goal is to classify chemical-protein interactions. More details of these datasets can be found in Table 2 of Gururangan et al. (2020a). Gururangan et al. (2020a) evaluate against all 3 tasks and their available code served as a basis on which we built MT-TARTAN and META-TARTAN.

Model Details We use a pre-trained RoBERTa$_{\text{base}}$ (Liu et al., 2019a) as the shared model base and implement each task as a separate multi-layer perceptron (MLP) head on top of this pre-trained base. As in Devlin et al. (2018), we pass the `[CLS]` token embedding from RoBERTa$_{\text{base}}$ to the MLP for classification.

Training Details For DAPT and TAPT, we download the available pre-trained model bases provided by Gururangan et al. (2020a). To train thier corresponding classification heads, we follow the experimental setup described in Appendix B of Gururangan et al. (2020a).

Performing end-task aware training introduces a few extra hyper-parameters. We fix the other hyper-parameters to those used in Gururangan et al. (2020a). MT-TARTAN and META-TARTAN introduce joint training of a classification head for the end-task T^*. We experiment with batch sizes of 128, 256 and 512 for training this head. We try out learning rates in the set $\{10^{-3}, 10^{-4}, 10^{-5}\}$ and drop out rates of $\{0.1, 0.3\}$. For META-TARTAN since we are now learning the task weights, $\mathbf{w}$, we test out task weight learning rates in $\{10^{-1}, 5 \times 10^{-2}, 3 \times 10^{-2}, 10^{-2}\}$. Note that for all MT-TARTAN experiments we use equalized task weights $\frac{1}{|\mathbf{T}_{\text{aux}}|+1}$. A small grid-search over a handful of weight configurations did not yield significant improvement over the uniform task weighting. We use the Adam optimizer (Kingma and Ba, 2014) for all experiments.

As mentioned in section 3.4.3, we train as separate meta-classification head, ϕ^*, to estimate the validation meta-gradients. To estimate ϕ^*, we use batch sizes of $\{16, 32\}$ samples from T^*'s train set. We regularize the meta-head with l_2 weight decay and set the decay constant to 0.1. We use a learning rate 10^{-3} to learn the meta-head. We stop training ϕ^* after 10 gradient descent steps.

3.6 Results and Discussion

In this section, we will discuss the results of comparing our models against DAPT and TAPT baselines.[2] Broadly, we demonstrate the effectiveness of end-task awareness as improving both performance and data-efficiency.

Domain	Task	RoBERTa	TAPT	MT-TARTAN	META-TARTAN
CS	ACL-ARC	$66.03_{3.55}$	$67.74_{3.68}$	$\mathbf{70.48}_{4.42}$	$70.08_{4.70}$
	SCIERC	$77.96_{2.96}$	$79.53_{1.93}$	$\mathbf{80.81}_{0.74}$	$\mathbf{81.48}_{0.82}$
BIOMED	CHEMPROT	$82.10_{0.98}$	$82.17_{0.65}$	$\mathbf{84.29}_{0.63}$	$\mathbf{84.49}_{0.50}$

Table 3.1: Comparison of our end-task aware approaches to RoBERTa and TAPT. All end-task aware approaches use TAPT as the auxiliary task. Reported results are test macro-F1, except for CHEMPROT, for which we report micro-F1, following Beltagy et al. (2019a). We average across 10 random seeds, with standard deviations as subscripts. Statistically significant performance (p-value from permutation test < 0.05), is boldfaced. See A.3,A.5 for more details about this table

3.6.1 End-task awareness improves over task-agnostic pre-training

Table 3.1 compares TAPT to its end-task aware variants. As in Gururangan et al. (2020a), we observe that performing task adaptive pre-training improves upon just fine-tuning RoBERTa. However, note that introducing the end-task by multi-tasking with the TAPT MLM objective leads to a significant improvement in performance. This improvement is consistent across the 3 tasks we evaluate against. We find that both MT-TARTAN and META-TARTAN achieve similar results in this setting.

3.6.2 End-task awareness improves data-efficiency

Gururangan et al. (2020a) train DAPT on large amounts of in-domain data to achieve results competitive with TAPT. They use 7.55 billion tokens for the BIOMED domain and 8.10 billion for the CS domain. This is on average over $10^4\times$ the size of the training data of our end-tasks of interest. The large amount of data required to train a competitive DAPT model represents a significant computational burden to the every-day practitioner. This begets the question: *are such large amounts of auxiliary data necessary for achieving good downstream performance?* To answer this, we train DAPT and its TARTAN version on variable amounts of data for both SCIERC and ACL-ARC tasks.

TARTAN is more data-efficient than DAPT In Figure 3.2, we focus on training on a small fraction of available domain data $n = \{10^0, 10^1\} \times |\text{Train}|$ for the DAPT auxiliary task. Full

[2]Our results are slightly different from those presented in Table 5 of Gururangan et al. (2020a) in terms of absolute values but the trends observed there still hold here. We attribute these differences to (1) minor implementation differences, and (2) averaging performance over ten seeds instead of five as used in the original paper in order to more strongly establish statistical significance. We observe slightly lower performance on ACL-ARC and SCIERC tasks due to these changes and higher performance on CHEMPROT.

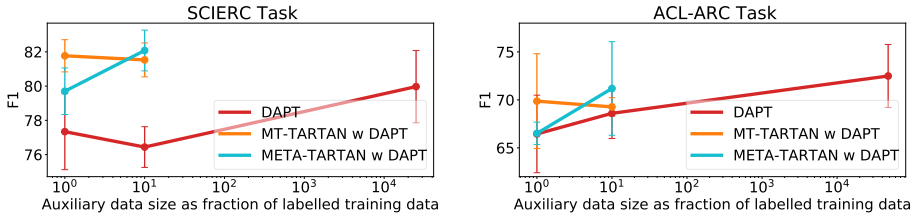


Figure 3.2: Compared to DAPT, TARTAN makes more efficient use of data. Large standard deviations are a result of the heterogeneity of the domain data used and the fact that our tasks are low-resource.

domain data is $n' \approx 10^4 \times |\text{Train}|$. This relatively low auxiliary data regime represents a realistic setting that is akin to those encountered by everyday practitioners who are likely to be computationally constrained. As can be seen in Figure 3.2, on the ACL-ARC task, META-TARTAN matches the performance of DAPT when the sizes of the domain data and end-task data are of the same order (10^0). At this data size, META-TARTAN supersedes DAPT on the SCIERC task. When trained on $10\times$ more auxiliary data, META-TARTAN supersedes DAPT in performance on both tasks. On the ACL-ARC task, META-TARTAN achieves $71.19_{4.88}$, which is close to DAPT's performance of $72.49_{3.28}$ using more than $10^3\times$ auxiliary data. These results indicate that end-task awareness can improve data-efficiency and in this case, improvements are on the order of $1000\times$.

Domain	Task	DAPT	DAPT+TAPT	MT-TARTAN	META-TARTAN
CS	ACL-ARC	$68.60_{2.62}$	$69.12_{5.76}$	$71.58_{1.65}$	$71.05_{2.37}$
	SCIERC	$76.44_{1.19}$	$77.62_{1.38}$	$\mathbf{81.02_{1.24}}$	$\mathbf{81.41_{1.70}}$
BIOMED	CHEMPROT	$80.76_{0.54}$	$78.22_{0.74}$	$\mathbf{83.77_{0.60}}$	$\mathbf{83.38_{0.89}}$

Table 3.2: We use $n = 10 \times |\text{Train}|$, a small fraction the full domain data which is $> 10^4 \times |\text{Train}|$. TARTAN methods are trained on both DAPT and TAPT. We average performance across 10 seeds. Statistically significant performance is boldfaced. See A.3, A.6 for more details about this table.

TARTAN is more data-efficient than DAPT+TAPT Table 3.2 compares DAPT and DAPT+TAPT (DAPT followed by TAPT) to *-TARTAN which multi-task DAPT, TAPT and the end-task. MT-TARTAN and META-TARTAN significantly outperform DAPT and DAPT+TAPT in 2 of the tasks whilst giving higher average performance in the ACL-ARC task. We thus conclude that **end-task awareness allows us to get a greater performance boost out of the same amount of data**.

We explore the data efficiency of TARTAN methods even further by comparing the relatively data-poor versions of MT-TARTAN and META-TARTAN above ($n = 10 \times |\text{Train}|$) to the DAPT and DAPT+TAPT variants trained on all the available domain data ($n' \approx 10^4 \times |\text{Train}|$). We can see from Table 3.3 that for the CS domain, our end-task aware variants come close to (ACL-ARC) and even supersede (SCIERC) the end-task agnostic variants though trained with $\approx 1000\times$ less data. For BIOMED domain (CHEMPROT task), increasing the amount of data drastically

improves the performance of end-task agnostic variants compared to MT-TARTAN and META-TARTAN trained on much less data. Zhang et al. (2020) show that different tasks exhibit sigmoid-like curves in terms of how much pre-training data is required to achieve good results before performance levels off. We contextualize Tables 3.2 and 3.3 within said work and posit that the CHEMPROT task intrinsically requires much more data (compared to our other tasks) before performance begins to improve appreciably.

Task	**DAPT**$_{full}$	**+TAPT**$_{full}$
ACL-ARC	$72.49_{3.28}$	$73.79_{1.75}$
SCIERC	$79.97_{2.11}$	$80.00_{1.08}$
CHEMPROT	$86.54_{1.05}$	$87.24_{0.81}$

Table 3.3: DAPT and DAPT+TAPT runs on all domain data available.

3.6.3 META-TARTAN more effectively utilizes out-of-distribution auxiliary data over MT-TARTAN

TARTAN	ACL-ARC	SCIERC	CHEMPROT
MT	$69.27_{0.96}$	$81.53_{0.99}$	$80.26_{3.79}$
META	$\mathbf{71.19}_{4.88}$	$\mathbf{82.08}_{1.19}$	$\mathbf{82.31}_{0.75}$

Table 3.4: All methods use only DAPT as auxiliary task. We use $n = 10 \times |\text{Train}|$. We report averages across 3 random seeds. Best average task performance is bolded.

We have seen that leveraging TAPT (Table 3.1 and 3.2) leads MT-TARTAN and META-TARTAN to perform similarly. The advantage of learning adaptive weights becomes pronounced in the DAPT only setting. Whilst TAPT uses the end-task's own training data for masked language modelling, DAPT uses heterogeneous domain data whose impact on the end-task performance is less clear. Notice from Table 3.4 that when required to rely solely on domain data for auxiliary tasking, META-TARTAN improves performance over MT-TARTAN. We attribute META-TARTAN's improvement over MT-TARTAN to its ability to more flexibly adapt to incoming data of variable utility to the end-task.

3.6.4 Task weighting strategies discovered by meta-learning

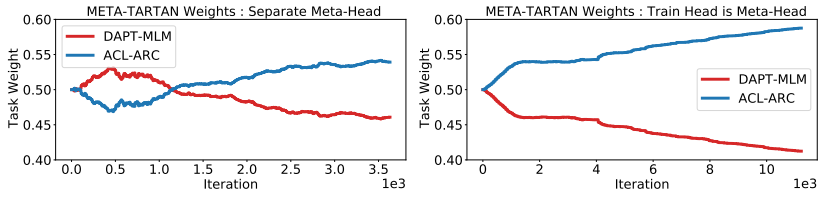


Figure 3.3: Having a separate classification head for computing meta-gradients is important. Using the same head as when training up-weights the end-task and under-utilizes auxiliary tasks.

To illustrate the importance of the separate classification head ϕ^* for computing the meta-signal for the task weights (described in Section 3.4.3), we run META-TARTAN experiments

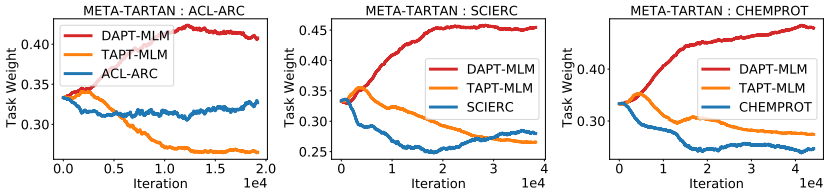


Figure 3.4: The meta-learned task weightings show similar trajectories across different end-tasks.

with ACL-ARC as the end-task and DAPT as the auxiliary task. We compare using either a separate (ϕ^*) or the same (ϕ') classification head for calculating the meta-gradient. Figure 3.3 plots the task weightings learned in each setting during training. We can clearly see that using a separate head counteracts the pathological solution of down-weighting all tasks that are not T^* and as a result, improves performance: **a delta of 1.7 F1 points in this case**. The strategy discovered by META-TARTAN presents an interesting contrast to classical pre-training: whilst the initial phase of classical pre-training involves solely the auxiliary task, early in training, META-TARTAN up-weights the auxiliary task but does not fully zero out the end-task. Later in training, we see leveling off of weights instead of railing the end-task to 1 as in classical pre-training.

Next, we plot a similar graph for using both DAPT and TAPT across our three tasks in Figure 3.4. From the figure, it is apparent that META-TARTAN discovers similar task-weighting strategies across different end-tasks. This suggests that the MLM objective and META-TARTAN 's strategy for learning task weights are generic enough to induce similar behaviours across tasks. In general, DAPT is significantly up-weighted compared to the end-task and TAPT. Note that the TAPT + ACL-ARC task weights (Figure 3.4) has the same approximate trajectory as ACL-ARC task weight in Figure 3.3. It seems important to assign high weight to the task data (Figure 3.3) but not necessarily all of it needs to go to the actual task loss (Figure 3.4). We hypothesize that the diversity in the domain data counteracts overfitting to the end-task data and results in DAPT being up-weighted.

3.7 Related Work

Multi-task learning can be traced back to seminal work by Caruana (1995), Caruana (1997a), and has since been the subject of a flourishing literature, recent surveys of which can be found in Ruder (2017) or Zhang and Yang (2021). In NLP, while initial work from Collobert and Weston (2008) already showed the benefits of multi-task learning, it has only recently become a central topic in the field, with the advent of multi-task benchmarks (Wang et al., 2018b; McCann et al., 2018).

Pre-training is where a machine learning model is first trained on a generic, data-rich task before being fine-tuned on an end-task. In NLP this practice dates back to the use of pre-trained word embeddings (Turian et al., 2010; Mikolov et al., 2013) and later pre-trained encoders (Kiros et al., 2015; Dai and Le, 2015). Peters et al. (2018b) and Howard and Ruder (2018) heralded a renaissance of pre-training before BERT (Devlin et al., 2018) and its many offshoots (Liu et al.,

2019a; Yang et al., 2019; Lewis et al., 2019) cemented it as the de facto standard for modern NLP.

Meta-learning dates back to early work from Schmidhuber (1995); Thrun (1998). More relevant to our work is gradient-based meta-learning for solving bi-level optimization problems, first popularized by Finn et al. (2017a) and followup work (Nichol et al., 2018; Rajeswaran et al., 2019) for few-shot learning. This method has transferred to a variety of applications such as architecture search (Liu et al., 2018a) and model poisoning (Kurita et al., 2020).

3.8 Conclusion

In this Chapter, we have advocated for a paradigm shift in the way we approach pre-training. We have motivated making pre-training more end-task aware when the end-task is known in advance. We introduced two novel end-task aware training algorithms: End-task Aware Training via Multi-tasking (MT-TARTAN) and End-task Aware Training via Meta-learning (META-TARTAN). In Section 3.6, we demonstrated the ability of our proposed algorithms to improve performance and data-efficiency over their end-task agnostic counterparts.

Beyond this thesis, the work in this Chapter suggests several promising directions for future work. Instead of learning coarse task level weights, can further performance improvements be achieved via finer-grained example level weighting as in Wang et al. (2020a)? Can meta-learning algorithms like META-TARTAN enable more effective utilization of previously discarded (Aroca-Ouellette and Rudzicz, 2020) pre-training auxiliary tasks like Next Sentence Prediction (NSP) (Devlin et al., 2018)? We hope this work spurs conversation around these questions and many more.

Chapter 4

End task aware training via gradient decomposition

4.1 Chapter Overview

In Chapter 3, we made a case for end task aware training when leveraging a set of auxiliary tasks. The algorithm we introduced, which we dubbed TARTAN, performs weighted multitask learning on both the primary and auxiliary tasks. We introduced META-TARTAN, a version of the TARTAN algorithm that used meta-learning to discern appropriate task weightings in order to mitigate the influence of tasks in $\mathbf{T}_{\text{aux}}$ that could negatively impact the primary task. Whilst effective, the TARTAN's approach of looking at the alignment of average gradient is coarse, applying the same weighting to the full task gradient (irrespective of disparate relations between gradients across different model components).

In this chapter, we present another algorithm for model-agnostic end-task aware transfer learning that performs fine-grained manipulation of the auxiliary task gradients. We propose to decompose auxiliary updates into directions which help, damage or leave the primary task loss unchanged. This allows weighting the update directions differently depending on their impact on the problem of interest. We present a novel and efficient algorithm for that purpose and show its advantage in practice. Our method leverages efficient automatic differentiation procedures and randomized singular value decomposition for scalability. We will empirically demonstrate in this chapter, that our gradient decomposition approach produces superior generalization performance over multitasking and PC-Grad (Yu et al., 2020) – as end task aware approaches – and representative end task agnostic ones.

4.2 Introduction

Multitask learning (Caruana, 1997b) and pretraining (Devlin et al., 2018; Caron et al., 2019) have transformed machine learning by allowing downstream tasks with small training sets to benefit from statistical regularities from data-rich related tasks (Collobert and Weston, 2008; Zhang et al., 2014; Liu et al., 2019a; Kornblith et al., 2019). Despite these advances, leveraging the mixing of tasks is still an art left to the practitioner. When one is interested in a *primary* task, it is

unclear how to select helpful auxiliary tasks, an appropriate parameter sharing architecture and a good way to filter out auxiliary data which might be detrimental to the primary tasks. Without careful choices, pre-training might hurt end-task performance (Gururangan et al., 2020b) or have limited impact (Raghu et al., 2019).

Prior work has examined these problems and proposed solutions, either to choose auxiliary tasks depending on their impact on the primary task (Du et al., 2018; Lin et al., 2019) or to equalize the impact of updates across tasks (Sener and Koltun, 2018; Chen et al., 2018; Hessel et al., 2019). Recently, several approaches (Sinha et al., 2018; Suteu and Guo, 2019; Yu et al., 2020) have been proposed that attempt to minimize interference between the updates across tasks. Our work builds on this direction, but unlike these previous approaches, we do not consider a symmetric view of multi-task learning in the sense that our goal is not to train a model performing well on all tasks. Instead, we focus on improving generalization for a single task, the primary task, and the other tasks, the auxiliary tasks are considered only through their impact on the problem of interest.

For that purpose, we introduce a framework which decomposes the gradient updates from the auxiliary tasks according to their impact on the primary task. We analyze the auxiliary task gradients in the subspace spanned by the primary task per-example gradients. This allows us to decompose auxiliary gradients into into three components : components that help, interfere or have no impact on the primary task according to the Taylor expansion of the expected primary loss. This decomposition allows us to re-weight each component differently prior to the update. Our framework enables us to treat each auxiliary update differently depending on its impact on the task of interest and it encompasses prior methods such as classical multitask learning (Caruana, 1997b) or more novel gradient surgery techniques (Yu et al., 2020). To achieve a tractable approach, we introduce an efficient, robust algorithm (ATTITTUD, Auxiliary Task Training with Influence from Target Task Update Direction) to estimate the subspace spanned by the primary task gradients in an online manner and decompose the auxiliary updates appropriately. As a result, we can integrate our approach with the stochastic training of large neural networks in various contexts.

The specific contribution of this Chapter are four-fold. To our knowledge, this Chapter proposes the first approach to adapt auxiliary gradients using a decomposition built from the span of the primary task Jacobian. In order to scale this approach to deep neural nets, we contribute a tractable and efficient algorithm called ATTITTUD that leverages insights from randomized linear algebra and automatic differentiation such as the R-operator (Pearlmutter, 1994). As our third contribution, we show that the fine-grained manipulation of the auxiliary task gradients under ATTITTUD, represents a unified framework that encompasses several previous approaches to asymmetrical task learning as special cases. Finally, we demonstrate the efficacy of our approach in both data-rich and data-starved primary tasks, over both images and textual data.

4.3 Related Work

Methods to leverage data outside of the task of interest have been popular in machine learning since the inception of multitask learning (Caruana, 1997b; Ruder, 2017; Vandenhende et al., 2020). These methods address multiple task simultaneously and have been successful in various

application domains (Collobert and Weston, 2008; Zhang et al., 2014; Misra et al., 2016). The optimization problem induced by multitask learning is difficult and solutions have been proposed for the various difficulties, including dealing with task gradients of different magnitude (Sener and Koltun, 2018; Chen et al., 2018; Hessel et al., 2019), or interfering with each others (Sinha et al., 2018; Suteu and Guo, 2019; Yu et al., 2020). The specific problem of interference has been studied extensively in the context of continual learning. Continual learning visits task in sequence and update interference is particularly problematic as it yields newer tasks to damage previously mastered tasks. In particular, a family of methods to project the gradient of the new tasks to be orthogonal to the gradient of the previous tasks has been proposed (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018; Farajtabar et al., 2019).

Different from many previous approaches, we are not interested in addressing multiple tasks per se. In our setting, only the *primary* task matters and the other *auxiliary* task have the sole role of improving generalization on the primary task. This is the setting considered by Du et al. (2018); Lin et al. (2019), who favor auxiliary tasks whose gradient directions are helpful to the primary task. Unlike these works that use coarse properties like the cosine similarity between averaged gradients, our approach allows fine-grained gradient manipulation within a subspace. Also, in our case, we do not distinguish between the different auxiliary tasks. Instead, we aim at correcting every auxiliary gradient in the same manner to improve the loss on the primary task. This type of gradient correction is related to Yu et al. (2020), which considers projecting multi-task gradients such that the directions of disagreement are removed. This method is actually a special case of our framework.

Our work also shares some similarities with data selection and domain adaptation approaches. In this case, the training data comes from a single task but its distribution is different from the validation/test distribution (Moore and Lewis, 2010; Axelrod et al., 2011; Ngiam et al., 2018). This classical problem has recently been addressed by sampling training points whose gradient aligns well with the expected validation gradient (Wang et al., 2020b,c). Instead of sampling individual points based on an estimated distribution of how helpful they will be to the primary task, our work avoids the use (and inherent challenges) of this reinforcement learning approach by operating on batch gradients of groups of points.

Our primary task/auxiliary task setting is also related to the pre-training then fine-tuning paradigm in which the auxiliary tasks are visited first (pre-training) to give an initialization for training on the primary task (fine-tuning). These methods have been very successful in settings where primary task data are rare. In particular, it is common to first rely on an unsupervised task over very large datasets prior to fine tuning over a supervised task (Devlin et al., 2018; Liu et al., 2019a; Kornblith et al., 2019; Yang et al., 2019; Song et al., 2019; Caron et al., 2018).

4.4 Auxiliary Task Update Decomposition

This section introduces a new method to improve generalization on a primary task T^* using training data from auxiliary tasks $\mathbf{T}_{\text{aux}} = \{T_1, \ldots, T_n\}$, where $\theta \in \mathbb{R}^D$ denote the parameters shared by all tasks. Our approach leverages gradient updates from the auxiliary tasks, but unlike the traditional approach, we decompose these gradients to maximize their usefulness to T^*. Precisely, we decompose the auxiliary task gradients into directions which decrease a first-order

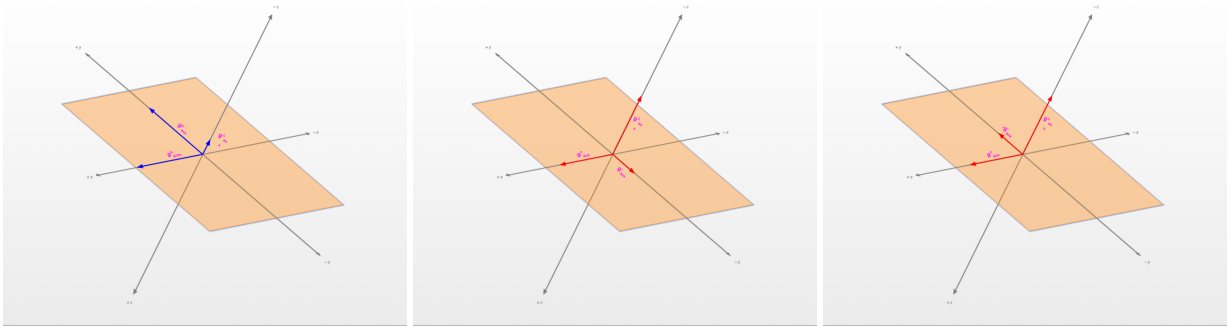


Figure 4.1: Example gradient manipulation in the 2-D $x - y$ plane with ATTITUD. ATTITUD can operate in any n-dimensional subspace. **Left**: Primary task gradient $\mathbf{g}_{prim}$ decomposed along the 3 Dimensions x, y and z. **Mid**: Decomposed Auxiliary task gradient $\mathbf{g}_{aux}$. We label the x component of $\mathbf{g}_{aux}$ as positive since it agrees (in direction) with the x component of $\mathbf{g}_{prim}$. Since the y component of $\mathbf{g}_{aux}$ is in the opposite direction as that of $\mathbf{g}_{prim}$, this is assigned a negative label. **Right**: Corresponds to $\tilde{\mathbf{g}}_{aux}$ obtained by applying $\boldsymbol{\eta}_{aux} = \big(1.0, 1.0, -1.0\big)$. We flip the conflicting gradient direction to agree with our primary task. This is just one configuration achievable under our framework.

approximation of the primary task loss, increase it or have no effect. This decomposition allows weighting these three directions differently when learning from the auxiliary tasks.

In order to decompose the auxiliary gradient, we must collect more fine-grained statistics about the primary task. At each training step, we collect the gradient of the loss with respect to θ for individual examples from the primary task, $\{\nabla_\theta \mathcal{L}_i^{\text{prim}}, \forall i\}$. The span of these vectors,

$$\mathcal{S} = \text{Span}\{\nabla_\theta \mathcal{L}_i^{\text{prim}}, \forall i\}$$

defines a subspace in which any linear combination of primary task gradients lies, including the gradient of the expected primary task loss, i.e. $\mathbf{g}_{prim} = \mathbb{E}(\nabla_\theta \mathcal{L}_i^{\text{prim}}) \in \mathcal{S}$. We denote the size of the subspace, $|\mathcal{S}| = K$. This is upper-bounded by the number of examples m, used to construct $\mathcal{S}$. If we define the orthogonal complement of $\mathcal{S}$ as $\mathcal{S}^\perp$, any vector $v \in \mathcal{S}^\perp$, is therefore orthogonal to $\mathbf{g}_{prim}$, i.e. $v \cdot \mathbf{g}_{prim} = 0$. This means that adding such a vector to the parameters has no impact on the expected primary task loss, according the order-1 Taylor expansion of $\mathcal{L}^{\text{prim}}$, i.e.

$$\mathcal{L}^{\text{prim}}(\theta + v) \simeq \mathcal{L}^{\text{prim}}(\theta) + v \cdot \mathbf{g}_{prim} = \mathcal{L}^{\text{prim}}(\theta).$$

We propose to project auxiliary task gradients onto $\mathcal{S}$ and $\mathcal{S}^\perp$. This allow to distinguish between the directions of the auxiliary task updates which impact the primary task loss and those which do not. If we denote the averaged auxiliary task gradient as $\mathbf{g}_{aux} = \mathbb{E}(\nabla_\theta \mathcal{L}_i^{\text{aux}})$, we can decompose this gradient as $\mathbf{g}_{aux} = \mathbf{g}_{aux}^{\pm} + \mathbf{g}_{aux}^{\perp}$. where $\mathbf{g}_{aux}^{\pm} \in \mathcal{S}$ is the portion of the gradient that lies in the span of the primary task example gradients and $\mathbf{g}_{aux}^{\perp} \in \mathcal{S}^\perp$ is the portion that lies outside of it. Since $\mathbf{g}_{aux}^{\perp} \in \mathcal{S}^\perp$, it is orthogonal to the average primary task gradient and parameter updates along the direction of $\mathbf{g}_{aux}^{\perp}$ are expected to have limited impact on the primary task loss. On the other hand, updates along the direction of $\mathbf{g}_{aux}^{\pm}$ can potentially improve or damage the averaged primary task loss. This component deserves a more careful treatment.

For that purpose, we introduce $\{u_i, i = 1, \ldots, K\}$ an orthonormal basis of $\mathcal{S}$. In this basis, we can measure if the components of $\mathbf{g}_{aux}^{\pm}$ agree or disagree with $\mathbf{g}_{prim}$. We say that the two gradients agree along u_i iif $\text{sign}(\mathbf{g}_{aux}^{\pm} \cdot u_i) = \text{sign}(\mathbf{g}_{prim} \cdot u_i)$. This means that we can decompose $\mathbf{g}_{aux}^{\pm} = \mathbf{g}_{aux}^{+} + \mathbf{g}_{aux}^{-}$ where $\mathbf{g}_{aux}^{+}$ refers to the projection of $\mathbf{g}_{aux}^{\pm}$ onto the basis vectors where $\mathbf{g}_{aux}^{\pm}$ and $\mathbf{g}_{prim}$ agree. By this decomposition, $\mathbf{g}_{aux}^{+}$ helps the primary task, $\mathbf{g}_{aux}^{+} \cdot \mathbf{g}_{prim} > 0$, while $\mathbf{g}_{aux}^{-}$ interfere with the primary task, $\mathbf{g}_{aux}^{-} \cdot \mathbf{g}_{prim} < 0$.

Guided by the primary task, we can therefore decompose the auxiliary task gradient as

$$\mathbf{g}_{aux} = \mathbf{g}_{aux}^{\perp} + \mathbf{g}_{aux}^{+} + \mathbf{g}_{aux}^{-} \tag{4.1}$$

which is described on Fig 4.1. Our approach proposes to re-weight differently the components of $\mathbf{g}_{aux}$, i.e.

$$\tilde{\mathbf{g}}_{aux} = \eta_{\perp}\mathbf{g}_{aux}^{\perp} + \eta_{+}\mathbf{g}_{aux}^{+} + \eta_{-}\mathbf{g}_{aux}^{-} \tag{4.2}$$

where $\boldsymbol{\eta}_{aux} = (\eta_{\perp}, \eta_{+}, \eta_{-})$ are hyper-parameters adjusting the auxiliary gradient according to the impact on the main task. If we also wish to include the primary task gradient in descent, as with multitasking, we can introduce $\boldsymbol{\eta}_{prim}$ as a scalar control variable to modulate its weighting.

A consequence of introducing $\boldsymbol{\eta}_{aux}$ is that specific configurations lead us to gradient updates that are guaranteed to do no harm to both tasks. This is captured by Theorem 1 below.

Theorem 1. *Let $\mathcal{L}^{\text{aux}}(\theta_t)$ and $\mathcal{L}^{\text{prim}}(\theta_t)$ represent the full batch losses of the auxiliary tasks and primary task respectively at step t. We assume the gradients of $\mathcal{L}^{\text{aux}}$ and $\mathcal{L}^{\text{prim}}$ are Lipschitz continuous with constant $L > 0$. Following the update rule : $\theta_{t+1} = \theta_t - \alpha \cdot \tilde{\mathbf{g}}_{aux}$, where $\alpha \leq \frac{1}{L}$ is the learning rate, we are guaranteed :*

$$\mathcal{L}^{\text{aux}}(\theta_{t+1}) \leq \mathcal{L}^{\text{aux}}(\theta_t)$$
$$\mathcal{L}^{\text{prim}}(\theta_{t+1}) \leq \mathcal{L}^{\text{prim}}(\theta_t)$$

If $\eta_{-} = 0$ and $\eta_{\perp}, \eta_{+} \geq 0$

Proof. See Appendix B.1 □

This theorem focuses on a single update and guarantees progress on both auxiliary and primary tasks. However, our asymmetric scenario is not interested in improving the auxiliary tasks per se and is amenable to more aggressive settings. Ideally we want gradient updates during pre-training with $\mathbf{T}_{\text{aux}}$ to not only do-no-harm to T^* when applied downstream but also to be along descent directions that are maximally beneficial to T^*. We can consider $\eta_{-} < 0$ as in Fig 4.1. Reversing the direction of $\mathbf{g}_{aux}^{-}$ by setting $\eta_{-} < 0$ preserves the descent guarantee on $\mathcal{L}_{prim}(\theta_{t+1})$ but no longer ensures descent on $\mathcal{L}_{aux}(\theta_{t+1})$. There are other interesting settings for our control parameters. One can recover the original gradient $\mathbf{g}_{aux}$ with $\eta_{\perp} = \eta_{-} = \eta_{+} = 1.0$. One can choose to drop gradients orthogonal to the primary task gradient span with $\eta_{\perp} = 0.0$, or ignore those which conflict with the main task by setting $\eta_{-} = 0.0$.

Relationships to other approaches Our framework is generic and encompasses other approaches as a particular case. One can train solely on the primary task by selecting $\boldsymbol{\eta}_{aux} = \big(0.0, 0.0, 0.0\big)$ and $\boldsymbol{\eta}_{prim} = 1.0$. Classical multitasking corresponds to $\boldsymbol{\eta}_{aux} = \big(1.0, 1.0, 1.0\big)$ and $\boldsymbol{\eta}_{prim} > 0.0$, while classical pre-training corresponds to performing a first phase with $\boldsymbol{\eta}_{aux} =$

$\big(1.0, 1.0, 1.0\big)$ and $\boldsymbol{\eta}_{prim} = 0.0$. Interestingly, our formulation introduces novel variants of pre-training, for instance, one can consider pre-training with only auxiliary gradients helpful to the primary task, $\boldsymbol{\eta}_{aux} = \big(0.0, 1.0, 0.0\big)$ and $\boldsymbol{\eta}_{prim} = 0.0$, followed by fine-tuning with $\boldsymbol{\eta}_{aux} = \big(0.0, 0.0, 0.0\big)$ and $\boldsymbol{\eta}_{prim} = 1.0$.

Our approach also instantiates PCGrad (Yu et al., 2020) as a particular case. This method was introduced to address the issue of conflicting gradients in multitask settings. PCGrad orthogonalizes the gradients of each task and removes conflicting gradients. To recover PCGrad under our approach, note that it is equivalent to a specific choice of our decomposition in the 1-D subspace spanned by the $\mathbf{g}_{prim}$. PCGrad then removes components of $\mathbf{g}_{aux}$ that conflict with $\mathbf{g}_{prim}$ which is equivalent to $\boldsymbol{\eta}_{aux} = \big(\alpha_{aux}, \alpha_{aux}, 0.0\big)$ and $\boldsymbol{\eta}_{aux} = \alpha_{prim}$.

4.5 Implementation

Equation 4.2 requires selecting a basis for the span of primary task gradients. Multiple choices are possible to define the basis $\{u_i\}$, to represent the span at each optimization time-step. This choice is important since the components of $\mathbf{g}^{\pm}_{aux}$ are labeled positive or negative depending on how they agree with the projection of the averaged primary task gradient onto the same basis. A natural choice is to select the basis as the singular vectors of the matrix of primary task per-example gradients $\mathbf{J}^* \in \mathbb{R}^{m \times D}$, also know as the Jacobian. To improve efficiency and prevent over-fitting on a few examples, we consider the span defined by the, $k < |\mathcal{S}|$, largest principal vectors of $\mathbf{J}^*$. Using the principal vectors as directions of descent instead of the mean induces a more robust algorithm since the mini-batch average gradient is susceptible to outliers and skew from replicated data-points. To the best of our knowledge, we are the first to propose using the singular vectors of $\mathbf{J}^*$ as directions of descent. We leave the theoretical implications of this algorithm to future work but note that its variance reduction properties may induce generalization benefits (Namkoong and Duchi, 2017).

We also consider alternative choices of bases as baselines, including the canonical parameter basis. This choice will examine the sign of every parameter update to verify whether it agrees with $\mathbf{g}_{prim}$. Whilst Theorem 1 holds irrespective of the choice of basis, its proof reveals that the amount of progress made on each loss depends on the choice of basis. Specifically, the reduction in $\mathcal{L}^{\text{prim}}(\theta_{t+1}), \mathcal{L}^{\text{aux}}(\theta_{t+1})$ after a gradient step along $\tilde{\mathbf{g}}_{aux}$ is proportional to the fraction of the norms of $\mathbf{g}_{prim}$ and $\mathbf{g}_{aux}$ captured by the subspace spanned by our choice of basis. To justify our use of the top singular values of $\mathbf{J}^*$, we evaluate this fraction for different choice of basis in our experiments (see Appendix B.3).

We are interested in applying our approach to the training of large neural networks and must consider a scalable algorithmic solution. As stochastic optimization is prevalent in this setting, we construct subspace S from a mini-batch of primary task data. Similarly, the expected gradients $\mathbf{g}_{prim}$ and $\mathbf{g}_{aux}$ are defined over a mini-batch. Instead of computing the singular value decomposition (SVD) of $\{\nabla_\theta \mathcal{L}^{\text{prim}}_i, \forall i\}$ exactly, we rely on a randomized approximation (Halko et al., 2011; Rokhlin et al., 2010; Nakatsukasa, 2017). This method does not require instantiating the vectors $\{\nabla_\theta \mathcal{L}^{\text{prim}}_i, \forall i\}$ and only needs a low dimensional projection onto a random subspace. This is advantageous for high dimensional cases, i.e. when the number of model parameters is large. In our case, this method also allows us to benefit from memory-efficient computation of

Jacobian Vector product using the R-operator (Pearlmutter, 1994) offered by automatic differentiation packages (Baydin et al., 2015) like Pytorch (Paszke et al., 2017). This means that we can compute SVD with a limited computational and memory burden, albeit without sacrificing approximation accuracy (Nakatsukasa, 2017). Additionally, we do not recompute the basis at every optimization step but at every n steps, which is efficient when training with small updates, e.g. when small learning rates and gradient clipping are used (Pascanu et al., 2013) (see Appendix B.3 for more details about n).

Algorithm 1: ATTITTUD : Construct Auxiliary Task Surrogate Gradient

Require : $\mathbf{g}_{aux}$, $\mathbf{J}^*$: Auxiliary task average gradient, primary task Jacobian

Require : $\boldsymbol{\eta}_{aux} = \left(\eta_\perp, \eta_+, \eta_-\right)$: Auxiliary task control parameters

Require : k : Size of subspace

$\mathbf{g}_{prim} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{J}^*_{i,:}$

$\mathbf{V} \leftarrow$ `randomized_lowrank_approx`$(\mathbf{J}^*, k)$

$\mathbf{p}_{prim},\ \mathbf{p}_{aux} = \mathbf{V}_t\left(\mathbf{g}_{prim}\right)^T,\ \mathbf{V}_t\left(\mathbf{g}_{aux}\right)^T$

`// ∘ is the hadamard product operator`

$\mathbf{p}^+_{aux}, \mathbf{p}^-_{aux} = \left(\mathbf{1}_{\left[\mathbf{p}_{prim} \circ \mathbf{p}_{aux} \geq 0\right]}\right) \circ \mathbf{p}_{aux},\ \left(\mathbf{1}_{\left[\mathbf{p}_{prim} \circ \mathbf{p}_{aux} < 0\right]}\right) \circ \mathbf{p}_{aux}$

`// Calculate the decomposition components`

$\mathbf{g}^+_{aux},\ \mathbf{g}^-_{aux} = \left(\mathbf{p}^+_{aux}\right)^T\mathbf{V},\ \left(\mathbf{p}^-_{aux}\right)^T\mathbf{V}$

`// Calculate the out of span component`

$\mathbf{g}^\perp_{aux} = \mathbf{g}_{aux} - \left(\mathbf{g}^+_{aux} + \mathbf{g}^-_{aux}\right)$

$\tilde{\mathbf{g}}_{aux} = \left(\eta_\perp \cdot \mathbf{g}^\perp_{aux}\right) + \left(\eta_+ \cdot \mathbf{g}^+_{aux}\right) + \left(\eta_- \cdot \mathbf{g}^-_{aux}\right)$

Return : $\tilde{\mathbf{g}}_{aux}$: Auxiliary task surrogate gradient

We study the impact of these choices in practice in Section 4.7. Putting it all together results in the ATTITTUD algorithm, Auxiliary Task Training with Influence from Target Task Update Direction, shown as Algorithm 1. The sub-procedure `randomized_lowrank_approx` is detailed in Appendix B.2 as Algorithm 4

4.6 Experimental Setup

We compare ATTITTUD with previous methods on a variety of tasks and domains. We rely on both text and image classification tasks to conduct our analysis. We also present ablation experiments to explain the impact of hyper-parameter selection. We make code for ATTITTUD and related experiments available on github. [1]

Text Classification. We apply our method on binary sentiment classification. We consider the Amazon Helpfulness (McAuley et al., 2015) and Imdb Movie Review (Maas et al., 2011) tasks. The Amazon Helpfulness task splits text reviews into 115k/5k/25k documents for train-validation-test split whilst the Imdb Review dataset has a 20k/5k/25k split. The Imdb Review task also has 50k unlabeled reviews as extra data which we utilize.

[1] Code available here https://github.com/ldery/ATTITTUD

For our models we build on top of Gururangan et al. (2020b)'s work where they introduce Task-Adaptive Pre-training (TAPT). TAPT further pre-trains a generic model, Roberta (Liu et al., 2019a), by performing Masked Language Modelling, MLM, (Devlin et al., 2018) on the task specific data (ignoring the labels) before doing supervised learning with the same data. We replicate Gururangan et al. (2020b)'s experimental setup and re-use their hyper-parameters for our experiments. We use the TAPT task as our auxiliary task. We extend TAPT to use our method by modifying the TAPT gradient with guidance from the supervised-learning task gradients. As baselines, we compare against TAPT and cross-TAPT: where we swap the masked language modelling pre-training data for the two tasks. Cross-TAPT is a setting where one uses out-of-distribution data for pre-training.

Image Classification. We apply our method to both high-resource and limited-data image classification tasks. We use the Cifar100 dataset (Krizhevsky et al., 2009) to explore the high-resource setting. We follow Rosenbaum et al. (2017) and treat each of the 20 super-classes / coarse labels of Cifar100 as a separate task. In our asymmetrical task setting, each of the 20 tasks is treated as a primary task, whilst the remaining 95 classes are grouped into a single auxiliary task. Thus, for each coarse label, we have an auxiliary 95-way classification task and a 5-way primary classification task. Moving forward, we refer to this setting as MultiCifar100.

We use a down-sampled version of Cifar10 (Krizhevsky et al., 2009) as a low-resource setting. Specifically, we rely on Cat-vs-Dog for the primary task and use the remaining 8 classes for the auxiliary task. Our auxiliary task is therefore an 8-way classification task where each class has 5,000 examples. We restrict the Cat and Dog classes to only 50 training examples from each class. We use the low-resource setting to compare against other methods and for our ablation study.

For these vision experiments, we use a WideResNet-22 architecture (Zagoruyko and Komodakis, 2016) with a depth of $k = 4$. We compare our method to 4 different baselines : no pre-training, vanilla pre-training, multitasking and PCGrad (Yu et al., 2020). Our architecture is more standard and allows gradient descent optimization unlike the routing network of Rosenbaum et al. (2017) and (Yu et al., 2020), which requires reinforcement learning for training.

Medical Imaging Transfer. We apply our method to cross-domain transfer for low-resource medical image classification. Specifically, we use 5k training examples from the ChexPert Dataset (Irvin et al., 2019) as our primary task and seek to identify 5 different thoracic pathologies: atelectasis, cardiomegaly, consolidation, edema and pleural effusion. This setup has been used in several cross-domain pretraining studies (Raghu et al., 2019; Jaiswal et al., 2019). Note that since we do not have access to the test set for this task, we use the validation set (231 images) as a proxy test set, and sample 100 images from the training data as a new validation set. We rely on generic photographs (Imagenet) as an auxiliary task (Deng et al., 2009). We use Tiny Imagenet Dataset (Le and Yang, 2015), a subset of Imagenet which consists of 500 examples each from 200 classes, instead of training on full Imagenet. All approaches are applied to the Resnet18 model (He et al., 2016) trained with Adam (Kingma and Ba, 2014).

For ease of interpretability in all our experiments, we select the auxiliary task control parameters $\boldsymbol{\eta}_{aux}$ within $\{(1.0, 1.0, -1.0), (1.0, 1.0, 0.0), (1.0, 0.0, -1.0), (1.0, 0.0, 0.0)\}$. For settings where we compare against multi-tasking, we select $\boldsymbol{\eta}_{prim}$ within a small subset of the settings that worked best with multitasking baseline experiments. These choices limit the overhead of hyper-parameter search but still allow us to show the empirical advantage of our method. In

	Imdb	Imdb + Amazon MLM	Amazon	Amazon + Imdb MLM
Roberta	95.4 ± 0.14	-	67.0 ± 0.50	-
TAPT	96.1 ± 0.11	95.1 ± 0.10	70.3 ± 0.87	67.8 ± 0.46
Ours	96.1 ± 0.09	$\mathbf{95.4 \pm 0.03}$	70.1 ± 1.13	$\mathbf{68.5 \pm 1.01}$

Table 4.1: Results on Text Classification measured by F1. Experiments are averaged over 5 runs.

all our experiments, we provide all methods with similar hyper-parameter search budgets, e.g. for Cifar10-Cat-vs-Dog, we ran a grid search with 16 configurations for regular pretraining, 16 configurations for PCGrad and 12 configurations for ATTITUD. More experimental details are available in Appendix B.3

4.7 Results and Discussion

Text Classification. Table 4.1 shows the results for text classification. When the same data is used both for the auxiliary task of MLM and the primary classification task, TAPT and ATTITTUD both bring a similar improvement over Roberta (Imdb, Amazon columns). For the Cross-TAPT setting where different data is used for the auxiliary task and the primary task (Imdb + Amazon MLM, Amazon + Imdb MLM columns), TAPT does not perform as well as ATTITTUD. This highlights the advantage of ATTITTUD when the auxiliary task data distribution differ from the primary task distribution.

Image Classification. Our results are presented in Table 4.2. Both for MultiCifar100 (high resource setting) and Cifar10-Cat-vs-Dog (low resource setting), ATTITUD shows a strong improvement over baselines. In general, we find that primary-task aware pre-training (Multitasking, PCGrad, Ours) is better than vanilla pre-training which also performs better than having no pre-training at all. For MultiCifar100, we find that using $\boldsymbol{\eta}_{aux} = (1.0, 1.0, -1.0), \boldsymbol{\eta}_{prim} = 0.1$ worked best for 11 out of the 20 Cifar100 super-classes tasks. Note that $\boldsymbol{\eta}_{aux} = (1.0, 1.0, -1.0)$ is an aggressive but novel configuration we introduce. Multitask learning and PCGrad produce better models on 6 and 3 tasks respectively. In the low-resource Cat-vs-Dog, setting ATTITUD produces a bigger boost in performance compared to baselines, with the best performing configuration being $\boldsymbol{\eta}_{aux} = (1.0, 0.0, 0.0), \boldsymbol{\eta}_{prim} = 0.01$. We posit that this configuration is successful because removal of the in-span components makes overfitting less likely. Applying the out-of-span components means the model learns features that do not harm the loss of the current mini-batch but could be useful later. Note that our best performing configurations are all novel and never an instantiation of PCGrad.

Medical Imaging Transfer. Table 4.3 shows our results on the ChexPert multi-label classification task. Per-pathology breakdowns are in Appendix B.3. Doing no pre-training at all performs worst. Our method outperforms using a pre-trained Resnet18 model over Imagenet. We apply the end-task-aware ATTITUD over 100k ImageNet images after the initial pretraining and we reach 83.3% AUC, an improvement over 81.4%.

Ablation Study. Our approach relies on the top-k singular vectors from *randomized_svd* to define the basis to identify the positive and negative component of the auxiliary task gradient,

Method	MultiCifar100	Cifar10-Cat-vs-Dogs
No-Pretraining	57.6	53.6 ± 2.26
Vanilla Pre-training	70.2	64.5 ± 1.26
PCGrad	75.6	64.2 ± 1.10
MT-TARTAN	75.5	65.3 ± 1.35
Ours	**76.1**	$\mathbf{67.1 \pm 1.31}$

Table 4.2: Average Accuracy on MultiCifar100 and Cat-vs-Dog Cifar10 tasks. Cat-vs-Dog experiments are averaged over 5 runs. We use MT-TARTAN over META-TARTAN because it is faster, and as we saw in Chapter 3, when the auxiliary tasks are in-distribution with respect to the primary tasks both versions of TARTAN perform similarly.

Method	Average AUC Across 5 Pathologies
No-Pretraining	78.3 ± 0.87
Pretrained-ResNet	81.4 ± 1.34
Pretrained-ResNet + Ours	$\mathbf{83.3 \pm 0.71}$

Table 4.3: Results on ChexPert-5k task measured by average AUC (Area Under Roc-Curve). All experiments are averaged over 5 runs.

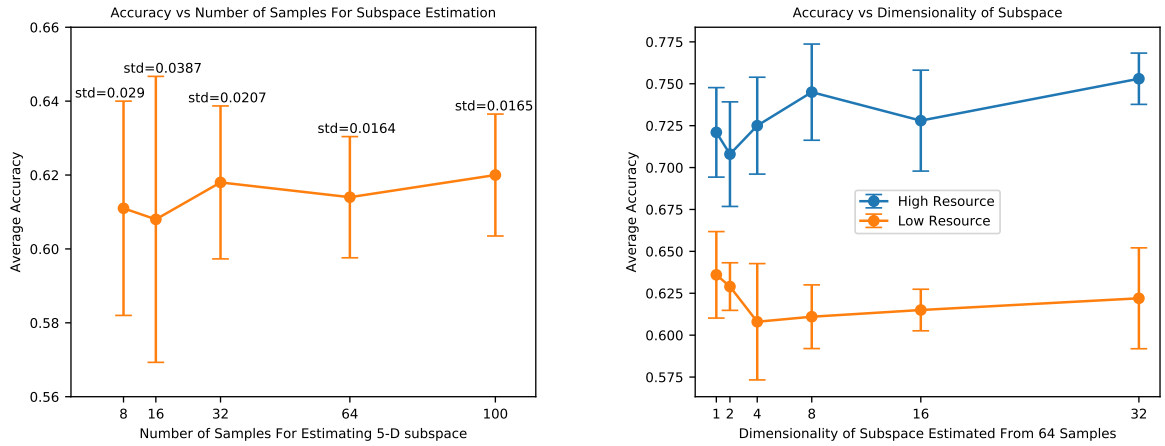


Figure 4.2: Averaged across 5 random initializations. **Left** We vary the number of samples used to estimate a 5-d subspace up to a maximimum of 100 (the total number of training examples in this low-resource setting). **Right.** We compare the effect of the dimensionality of the subspace in the low-resource (50 examples each for Cat, Dog classes) and high-resource (1000 examples each per class).

see Section 4.5. This method is more accurate than several alternatives; see Table 4.4. Namely, we compare our choice to *random*, the basis spanned by k randomly chosen orthogonal vectors in $\mathbb{R}^D$, *unit_avg_grad*, the basis spanned by the average primary task gradient, and *canonical*, the

Subspace	Canonical	Random	Unit_avg_grad	Randomized_SVD
Average Acc.	51.42 ± 2.09	58.72 ± 2.68	59.13 ± 2.08	$\mathbf{62.2 \pm 4.00}$

Table 4.4: Experiment conducted on Cat-vr-Dog Cifar10 dataset for different choices of subspace basis. We use $k = 5$ for Random and Randomized_SVD. This ablation uses a smaller hyper-parameter budget than Table 4.2

per-parameter basis. This ablation was performed under a more limited tuning budget (we cross-validated on configurations $(1, 1, 0)$ and $(1, 1, -1)$ only) than the full Cat-vs-Dog experiments from Table 4.2.

We also examine the number of samples to estimate the principal directions of the per-example primary task gradient. Larger sample sizes involve more computation but have limited benefit on average accuracy. Large sample sizes however reduce variance, as shown in Figure 4.2 (left). This is as expected since using more samples gives a higher fidelity estimate of the top-k singular vectors.

Another parameter of our algorithm is the size of our subspace, k. In general, we observe that in low-resource settings, it is better to operate on the auxiliary task gradient in a smaller dimensional subspace. The opposite holds for high-resource settings. This can be seen in Figure 4.2 (right). Whilst using a larger dimensional subspace captures a richer description of the $\mathbf{J}^*$, it also creates the risk of over-fitting especially in a limited data setting. This trade-off therefore has to be validated on a per-task basis.

4.8 Conclusions

We have proposed, in this Chapter, a new approach to training a model with additional help from an auxiliary task. Our method decomposes the gradients of the auxiliary task according to three directions, with positive, negative and neutral impact on the primary task. This decomposition allows a flexible re-weighting of the auxiliary task components and give rise to a family of training strategies, which encompasses novel and existing approaches. We leverage insights from randomized linear algebra and automatic differentiation to scale the approach to large deep networks. Experiments in multitasking, pretraining and domain transfer over vision and text classification task demonstrate that our work improves data-efficiency, allowing us to achieve higher accuries at fixed amounts of task data.

Chapter 5

Automated, end-task aware construction of transfer tasks

5.1 Chapter Overview

In the previous chapters, we made a case for end task aware transfer learning and provided principled algorithms for end task aware optimization. We assumed that the set of auxiliary tasks $\mathbf{T}_{\text{aux}}$ has been provided a-priori. However, this is not always guaranteed and whilst much work has been done to formulate useful auxiliary objectives, their construction is still an art which proceeds by slow and tedious hand-design.

In this chapter, we present an approach for automatically generating a suite of auxiliary objectives. We achieve this by deconstructing existing objectives within a novel unified taxonomy, identifying connections between them, and generating new ones based on the uncovered structure. Next, we present a principled and efficient algorithm for searching the space of generated objectives to find those most useful to a specified end task.[1].

5.2 Introduction

The auxiliary learning paradigm, where we augment a primary objective with extra learning signals to boost end-task performance, is a staple of many machine learning (ML) domains. In natural language processing (NLP), well known models like SpanBERT (Joshi et al., 2020) and RoBERTa (Liu et al., 2019a) are trained on masked language modelling (MLM) auxiliary objectives (Devlin et al., 2018) before fine-tuning on the end-task. And for speech processing and reinforcement learning (RL), Oord et al. (2018) introduced the popular contrastive predictive coding objective which achieved state of the art performance in many settings when multi-tasked with the end-task. Despite these successes and many more, research into devising such objectives has progressed in a very local, objective-by-objective manner (Raffel et al., 2019; Clark et al., 2020; Grill et al., 2020; Chen et al., 2020). Auxiliary objectives are constructed by hand-design and without much overarching structure, relying on the experience and intuition of a select group

[1] Code available at : https://github.com/ldery/Automating-Auxiliary-Learning.

of researchers versed at making appropriate design choices. Unfortunately, this status-quo not only creates a technical barrier of entry for exploring auxiliary objectives in new domains but also, by virtue of its incremental nature, limits the rate at which new objectives are discovered and investigated.

To address the above challenges, the work in this Chapter presents a framework for automatically generating and utilizing a large set of candidate auxiliary objectives. Our framework is seeded by the following key observation: leading auxiliary objectives across multiple domains can be viewed as making different design decisions within a 4 stage pipeline: **Input Data** $(\mathcal{D})$ $\rightarrow$ **Input Transformation** $(\mathcal{T})$ $\rightarrow$ **Model Representation** $(\mathcal{R})$ $\rightarrow$ **Output** $(\mathcal{O})$. For instance, in RL, a common auxiliary objective is to predict the environment's forward dynamics (Agrawal et al., 2016; Hafner et al., 2019). To construct this objective, the current task state-action pair $(\mathcal{D})$ is corrupted $(\mathcal{T})$ and then passed through the model to produce a latent representation $(\mathcal{R})$ which is finally used to predict the next state $(\mathcal{O})$. Similarly, in NLP, the XLNet (Yang et al., 2019) objective—which performs language modelling on a randomly factorized permutation of the input—can be written within our taxonomy as $\{\mathcal{D}$ = Out-of-Domain, $\mathcal{T}$ = No-op, $\mathcal{R}$ = Random-Factorized, $\mathcal{O}$ = Next Token$\}$. These two examples (along with others listed in Figure 5.1) fall within a class we term *named objectives*: objectives that have been previously proposed in the auxiliary learning literature.

Objective	Data ($\mathcal{D}$)	Transform ($\mathcal{T}$)	Representation ($\mathcal{R}$)	Output ($\mathcal{O}$)
BERT	Out-of-domain	BERT-Op	Bidirectional	Denoise Token
TAPT	Task data	BERT-Op	Bidirectional	Denoise Token
DAPT	In-domain	BERT-Op	Bidirectional	Denoise Token
ELMO	Out-of-domain	No-Op	Left-to-Right and Right-to-Left	Next Token
GPT	Out-of-domain	No-Op	Left-To-Right	Next Token
XLNet	Out-of-domain	No-Op	Random factorized	Next Token
Electra	Neural LM Data	Replace	Bidirectional	Real / Synthetic
...	...	...	...	...

Figure 5.1: We present the decomposition of some auxiliary objectives in NLP within our framework.

Decomposing named objectives within our taxonomy provides a unified view of the auxiliary learning landscape. From this vantage point, it becomes clear that there are many unexplored combinations of the various primitives used across named objectives. This presents a simple formula for automatically generating a large set of candidate objectives: take the cartesian product of the design decisions across given stages (Figure 5.2). Using this compositional process, not only can we reconstruct existing named objectives, we can also generate new combinations. This overcomes the tedium of implementing each objective independently since we can just reuse a small set of

Data ($\mathcal{D}$)		Transform ($\mathcal{T}$)		Representation ($\mathcal{R}$)		Output ($\mathcal{O}$)
Out-of-domain		No-Op		Bidirectional		Next Token
In-domain		Replace		Left-to-Right		Real / Synth
Task data	$\times$	Mask	$\times$	Right-to-Left	$\times$	Denoise Token
Neural LM Data		Noising embeds		Rand. factorized		TF-IDF
...		...		...		...

$\downarrow$

TAPT = {Task data $\rightarrow$ BERT-Op $\rightarrow$ Bidirectional $\rightarrow$ Denoise Token}
GPT = {Out-of-domain $\rightarrow$ No-Op $\rightarrow$ Left-to-Right $\rightarrow$ Next Token}
New-Obj$_1$ = {Task data $\rightarrow$ BERT-Op $\rightarrow$ Left-to-Right $\rightarrow$ Denoise Token}
New-Obj$_2$ = {In-domain $\rightarrow$ No-Op $\rightarrow$ Random Factorized $\rightarrow$ TF-IDF}
...

Figure 5.2: Our framework in the context of NLP. We decompose named objectives within our four staged taxonomy : $\{\mathcal{D}, \mathcal{T}, \mathcal{R}, \mathcal{O}\}$. By taking the cartesian product of choices across stages, we reproduce named objectives and discover new ones.

simple stage-wise primitives.

Generating a large set of objectives raises the natural question of how to efficiently select the most helpful ones for a given end task. Instead of leaving this to practitioner intuition, we develop principled guidelines to address this question by theoretically studying the impact of auxiliary learning on a particular end-task. Specifically, using arguments based on algorithmic stability (Hardt et al., 2016; Bousquet and Elisseeff, 2002), we derive end-task generalization error bounds that are dependent on the choice of auxiliary task. This contributes to existing theory (Saunshi et al., 2020; Xie et al., 2021) on how auxiliary learning impacts the end-task by suggesting a new candidate mechanism: auxiliary learning results in more stable optimization end-points in the sense of Bousquet and Elisseeff (2002), which in theory improves generalization of the final model.

Guided by our theory, we introduce AANG (**A**utomating **A**uxiliary Learni**NG**), an efficient, structure-aware algorithm for adaptively combining a set of related objectives to improve generalization on a specific end-task. AANG incorporates the following prescriptions from our theory: (i) auxiliary tasks that are more similar to the end-task are desirable. Given a set of objectives, AANG learns adaptive weights to bring the composite objective closer to the end-task; (ii) in general, more auxiliary data is better. AANG maximizes the effective amount of data used in training by using all the generated objectives instead of taking task-specific subsets.

To empirically validate our method for automatically generating and utilizing auxiliary objectives, we experiment on five NLP tasks. We do so in the widely-used setting of *continued pre-training* (Gururangan et al., 2020a; Aghajanyan et al., 2021; Dery et al., 2021a; Zhang et al., 2022), where a model trained with a single auxiliary objective on large-scale data is further trained on end-task related data. Without introducing any external data or architectural modifications, variants of AANG outperform strong and widely used baselines in 4 out of 5 tasks. AANG achieves an average improvement of $\mathbf{4.2}\%$ over standard fine-tuning of RoBERTa across our chosen tasks. We believe our results will spur further research into exploring automating auxiliary learning across a variety of settings. Notably, while we focus on NLP when discussing the space of auxiliary objectives (Section 5.4) and in our empirical evaluation (Section 5.7), our theoretical results (Section 5.5) and AANG itself are domain-agnostic[2].

5.3 Related Work

To properly scope this work, we define *auxiliary learning* as training a model on alternative objectives with the goal of improving performance on some primary end-task. Auxiliary learning is an instantiation of transfer learning (Caruana, 1997a; Baxter, 2000; Ruder et al., 2019). It covers the pretrain-then-finetune paradigm (Huh et al., 2016; Devlin et al., 2018; Schneider et al., 2019; Gururangan et al., 2020a) as well as end-task aware multitasking approaches (Lin et al., 2019; Dery et al., 2021b,a). Whilst auxiliary objectives may be meta-learned (Liu et al., 2019b; Navon et al., 2020), for simplicity – since incorporating these would require further complication of our design space – such objectives are out of the scope of this Chapter.

[2]Our ideas could be applied to domains like RL or computer vision (CV), where a similar dissection of existing objectives can be performed.

This work bears many parallels to the area of neural architecture search (NAS) (Stanley and Miikkulainen, 2002; Zoph and Le, 2016; Roberts et al., 2021). Whilst we seek to automate auxiliary learning, the objective of NAS is to automate the discovery of the right neural architecture given a specific end-task. Search spaces of candidate architectures are created by taking the cartesian product of architecture design choices across the depth of the network. The design of suitable architectural search spaces for a variety of settings has been an active area of research (Tan and Le, 2019; Howard et al., 2019; Dao et al., 2020; Roberts et al., 2021). To develop AANG, we borrow ideas from the NAS literature on efficient algorithms for sifting through spaces of architectures. Mirroring the popular differentiable NAS method DARTS Liu et al. (2018a), we perform a continuous relaxation over the search space of objectives, allowing for efficient search by gradient descent. We also use a factored approach to model relationships between objectives that share primitives. This is inspired by recent work on stochastic-relaxation weight sharing (Dong and Yang, 2019; Li et al., 2020).

As a theoretical contribution, this work derives an end-task aware generalization error bound for auxiliary learning. Our bound is built on that of Hardt et al. (2016), who derive generalization bounds for parametric models trained with stochastic gradient descent (SGD). To derive their bounds, they leverage the concept of algorithmic stability introduced by Bousquet and Elisseeff (2002). Informally, a randomized algorithm is *uniformly stable* if changing a single training data point in the given samples does not change its end-point *too much*. Said change is characterized as the average difference in predictions between the two learned models. Stability implies generalization in expectation (Hardt et al., 2016; Kuzborskij and Lampert, 2018).

5.4 Automatically Generating Auxiliary Objectives

To begin, we take a high-level view of the landscape of named objectives. Using running examples from NLP, we propose the following coarse structure for the sequence of choices made in the hand-design of auxiliary objectives:

1. **Data**, $\mathcal{D}$: Auxiliary objective pipelines begin with a choice of input data. Here, options can range from heterogeneous *out-of-domain* data (Radford et al., 2019), *in-domain* data with respect to the final end-task (Beltagy et al., 2019b) or the *task data* itself (Gururangan et al., 2020a). It may even include data outside the modality of the end-task.
2. **Input-Transformation**, $\mathcal{T}$: Many auxiliary objectives are self-supervised with respect to their input data. They corrupt or transform the input and then reconstruct it in whole or part. For example, input text tokens can be *masked*, *replaced* or *deleted*. Operations can also be aggregated as in *BERT-Op*: mask 80% of selected tokens and randomly replace 50% of the remaining Devlin et al. (2018); Liu et al. (2019a).
3. **Representation**, $\mathcal{R}$: After transformation, representations of the input data can be computed from a given model in different ways. A chosen token's representation can depend on only its left context (*Left-to-Right*) (Radford et al., 2018) or its right context (*Right-to-Left*) (Peters et al., 2018c). It could also depend on the representations of a randomly selected permutation of other tokens (*Random Factorized*) Yang et al. (2019).

4. **Output**, $\mathcal{O}$: Finally, representations obtained from the previous stage are fed into a loss function producing a final output. The choice of output loss is usually coupled with the choice of transformation made in stage 2. Choices include but are not restricted to *denoising tokens*, *predicting the next token* or *predicting the TF-IDF* (Term Frequency-Inverse Document Frequency) of a token.

The above taxonomy $\{\mathcal{D} \rightarrow \mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{O}\}$ is expansive enough to cover a range of named auxiliary objectives of interest in NLP (Figure 5.1)[3]. For example, we can write any member of the GPT series (Radford et al., 2018, 2019; Brown et al., 2020b) which perform left-to-right language modelling on out-of-domain data as $\{\mathcal{D}$ = Out-of-Domain, $\mathcal{T}$ = No-op, $\mathcal{R}$ = Left-To-Right, $\mathcal{O}$ = Next Token$\}$.

We can summarize the pre-existing choices within each design stage to obtain a unique set of options. For example, we can reduce the set of model representation types used by the objectives enumerated in Figure 5.1 to the unique set $\mathcal{R} = \{$Bi-directional, Left-To-Right, Right-To-Left, Random-Factorized$\}$. Having summarized the list of primitives within each stage, a simple formula for generating a space of auxiliary objectives becomes apparent: take the cartesian product of the design choices at each stage (see Figure 5.2). In general, given an instance of our taxonomy, we can construct a space of objectives $\mathcal{A} = \mathcal{D} \times \mathcal{T} \times \mathcal{R} \times \mathcal{O}$ of size $|\mathcal{A}| \leq |\mathcal{D}| \times |\mathcal{T}| \times |\mathcal{R}| \times |\mathcal{O}|$. Consider $\mathrm{New\_Obj}_1$ from Figure 5.2. This previously unexplored objective can be obtained by combining the special masking operation from BERT (*BERT-Op*) with computing model representations based on left-to-right causal masking as in GPT. In fact, this objective proved one of the most useful ones in our experiments below (see Figure 5.5).

Our framework also allows us to reason about whole families of objectives, $\mathcal{F}$, by thinking in terms of design stages and choices. For example, given a particular end-task $\mathbf{E}$ with input text $\mathbf{E}_{\mathcal{D}}$, we can create a family of objectives based solely on task data by fixing to that option in our input data stage; we call this family $\mathcal{F}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$. $\mathcal{F}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$ not only includes pre-existing TAPT Gururangan et al. (2020a) but also unexplored objectives like task-data dependent variants of XLNET, ELMO etc. Auxiliary learning with $\mathcal{F}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$ can be seen as a relaxed form of data augmentation which we dub **task augmentation**. Whilst data augmentation requires applying transformations that preserve the data-point's label, task augmentation has no such restriction and thus offers greater flexibility in terms of specifying $\{\mathcal{T}, \mathcal{R}, \mathcal{O}\}$. We can also reason about expanding particular stages to include new primitives. Any supervised loss can be added to the output stage, $\mathcal{O}$, allowing us to potentially explore auxiliary objectives based on supervised signals like NER or POS tagging (Carreras et al., 2003; Charniak, 1997). A special example is setting $\mathcal{O}$ to the end-task supervised output $\mathbf{E}_{\mathcal{O}}$. This leads to $\mathcal{F}^{\mathcal{O}=\mathbf{E}_{\mathcal{O}}}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$ which is a subset of $\mathcal{F}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$. $\mathcal{F}^{\mathcal{O}=\mathbf{E}_{\mathcal{O}}}_{\mathcal{D}=\mathbf{E}_{\mathcal{D}}}$ includes many objectives like predicting the end-task signal from corrupted input data. In Section 5.7, we will introduce a search space of objectives that leverages task augmentation.

[3]Although this taxonomy is quite expansive, it obviously does not consider other elements of objective creation such as choice of model architecture, optimizer settings, etc.

5.5 The Impact of Auxiliary Learning on End-task Generalization

In this section, we relieve reliance on practitioner intuition by deriving a set of guiding principles on how to effectively utilize the automatically generated objectives from Section 5.4.

Auxiliary learning influences the end-task through both training and generalization error. Previous theory has largely focused on characterizing the impact on end-task training error. Liu et al. (2021), for example, show that end-task agnostic pre-training can create a performance gap in training error compared to training with the end-task alone. The size of this gap depends on how dissimilar the pre-training auxiliary objective is from the end-task. They introduce the following assumption (which we will borrow) to formalize their notion of task similarity:

Assumption A.1: Let f_e represent the end-task objective and f_a be the auxiliary objective. There exists $\Delta \geq 0$ such that $\|\nabla f_a(\theta) - \nabla f_e(\theta)\| \leq \Delta \ \forall \, \theta$.

Note that θ represents all the parameters of the model. Smaller Δ implies f_a is more similar to the primary task f_e. Liu et al. (2021) bound the end-task agnostic training error gap to be logarithmic in Δ.

Unlike training error, end-task generalization error has gone unstudied in the auxiliary learning setting. Bounding the generalization error not only adds to our theoretical understanding of the impact of auxiliary learning but also provides insights to guide algorithm design. To arrive at a bound, we adapt the technique of Hardt et al. (2016) who derive a generalization bound on training with **only the end-task** via stochastic gradient descent. We consider the end-task aware setting where the end-task is multi-tasked with the auxiliary objective. This setting has recently been shown to improve end-task performance over the pretrain-then-finetune paradigm (Dery et al., 2021b,a; Yao et al., 2021).

Auxiliary learning with Dynamic Sampling: We are given an auxiliary objective $f_a(\cdot; z) \in [0, 1]$ with N_a samples $S_a = (z_1, \ldots, z_{N_a})$ from the distribution $\mathcal{D}_a$. f_a can either be a single objective or a weighted linear combination of objectives : $f_a = \sum_k w^k f_a^k$. At any iteration of SGD, we sample a choice of the end-task function f_e or the auxiliary objective f_a according to the probabilities $\lambda_e, \lambda_a \in [0, 1] \mid \lambda_e + \lambda_a = 1$. Given the chosen objective, we sample a data-point and perform stochastic gradient descent based on the sampled data-point. We now present our bound in the setting described.

Theorem 2 (Auxiliary learning with Dynamic Sampling)**.** *Assume that $f_e(; z_e), f_a(; z_a) \in [0, 1]$ are both L-Lipschitz with β_e and β_a-smooth loss functions respectively. Consider that we have $N' = N_e + N_a$ total samples where f_e and f_a have N_e and N_a samples respectively. $r_e = \frac{N_e}{N'}$ is the fraction of the available data represented by the end-task. Suppose that we run stochastic gradient descent for T steps with monotonically non-increasing step sizes $\alpha_t \leq \frac{c}{t}$ by dynamically sampling the tasks according to λ_e and λ_a. Then, with respect to f_e, the generalization error is bounded by:*

$$\epsilon_{\text{gen}} \lessapprox (\Delta)^{\frac{1}{1+c\lambda^*\beta^*}} \left(\frac{\gamma T}{N'}\right)^{1-\frac{1}{c\lambda^*\beta^*+1}} \quad \textit{Where} \quad \gamma = \frac{\lambda_e}{r_e} \tag{5.1}$$

Here $\beta^* = \min\{\beta_e, \beta_a\}$ and λ^* is the weighting of the function with smaller smoothness.

Proof. See Appendix C.4 for full proof and Appendix C.5 for more discussion □

As a detailed inspection of the proof will show, we derive Equation 5.1 by appealing to algorithmic stability (Bousquet and Elisseeff, 2002; Hardt et al., 2016; Kuzborskij and Lampert, 2018) (Section 5.3). To our knowledge, ours is the first work to present an algorithmic stability view to formally explain how auxiliary learning influences end-task performance. Equation 5.1 surfaces the following prescriptions about learning with auxiliary tasks :

(P_1) Smaller Δ improves ϵ_{gen}. This implies that the more similar the auxiliary objective is to the end-task (under Assumption A.1), the lower the generalization error.

(P_2) Larger N' leads to smaller ϵ_{gen}[4]. Since we usually have a fixed amount of task data N_e, we can increase N' by adding more auxiliary data N_a.

5.6 End-task Aware Search of Structured Objective Spaces

Algorithm 2: AANG

Input: Search Space - $\mathcal{A}$
Factor vectors - $\{W^{\text{All}}, W^{\mathcal{I}}, W^{\mathcal{T}}, W^{\mathcal{R}}, W^{\mathcal{O}}\}$
End-task - $\mathbf{E}$, End-task weight - λ_e
Initial Model Params - $\theta_0 \in \mathbf{R}^D$
repeat
  Sample a batch of n objectives
  $\mathcal{K}^n \sim \mathcal{A}$
  Weighting of objectives in $\mathcal{K}^n$
  Construct $\mathbf{w}^n$
  for $k = 1$ **to** n **do**
    $(d,\ t,\ r,\ o) = [\mathcal{K}^n_k]$.stages
    $w^k \propto \exp\left(W^{\text{All}}_{(d,t,r,o)} + W^{\mathcal{I}}_d + W^{\mathcal{T}}_t + W^{\mathcal{R}}_r + W^{\mathcal{O}}_o\right)$
    $\mathbf{w}^n_k \leftarrow w^k$
  end for
  Get losses from batches of data
  $\hat{\mathcal{L}}_{\mathcal{A}}(\mathcal{K}^n, \mathbf{w}^n) = \sum_{k=1}^n w^k \mathcal{L}_k$
  $\mathcal{L}_{\text{total}} = \lambda_e \mathcal{L}_E + (1-\lambda_e)\hat{\mathcal{L}}_{\mathcal{A}}$
  Get gradients and update factors
  $\theta_{t+1}, \{\nabla_{\mathbf{w}^n, \lambda_e}\} \leftarrow \text{META-TARTAN}(\theta_t, E, \mathcal{L}_{\text{total}})$
  Update $\{W^{\text{All}}, W^{\mathcal{I}}, W^{\mathcal{T}}, W^{\mathcal{R}}, W^{\mathcal{O}}\}$ using $\nabla_{\mathbf{w}^n}$
  Update λ_e using ∇_{λ_e}
until done
Return : θ_T

Guided by Section 5.5, we build a practical method for exploring a set of objectives, $\mathcal{A}$.

Whilst the dynamic sampling setting described in Section 5.5 is amenable to theoretical consideration, we make a few practical changes to it. First, instead of performing alternating gradient descent by sampling f_a, f_e according to λ_e, λ_a, we instead use them as multitask weights and perform joint training. Joint training has been found to produce superior results compared to alternating optimization when leveraging auxiliary objectives (Aghajanyan et al., 2021). We perform gradient descent on the following total loss which interpolates between the end-task and the auxiliary loss $\mathcal{L}_{\text{total}} = \lambda_e \mathcal{L}_E + (1-\lambda_e)\mathcal{L}_{\mathcal{K}}$. Here, $\mathcal{K}$ is a chosen subset of $\mathcal{A}$.

Second, as indicated in Section 5.5, given $\mathcal{K}$, we can write the set as a single objective $f_a = \sum_{k \in \mathcal{K}} w^k f^k_a$. By Prescription ($P_1$), we want to choose $\{w^k\}$ such that f_a has a small Δ with the end-task f_e. We would also like to set λ_e such that the bound on ϵ_{gen} is minimized. Whilst a closed form exists for the optimal weightings $\lambda_e, \{w^k\}$, it depends on variables like $\{\Delta^k\}, \{\beta^k_a\}, L$ that are hard to estimate. We therefore propose to learn $\lambda_e, \{w^k\}$ in an online, data-driven way. To do this, we build on top of the META-TARTAN algorithm proposed by Dery et al. (2021a). META-TARTAN is a meta-learning algorithm that learns adaptive

[4]This holds at fixed γ which we achieve by adjusting λ_e to account for introducing more auxiliary data.

weights for different auxiliary tasks in a way that prioritizes end-task generalization. It learns $\{w^k\}$ by minimizing the loss on the end-task validation set: $\frac{\partial \mathcal{L}_{\mathbf{E}}^{val}}{\partial w^k} \approx -\left(\nabla_\theta \mathcal{L}_{f_a^k}\right)^T \left(\nabla_\theta \mathcal{L}_{\mathbf{E}}^{val}\right)$. This corresponds to learning $\{w^k\}$ such that $\left(\nabla_\theta f_a\right)^T \left(\nabla_\theta f_e\right)$ is maximized. This minimizes one of the terms that contributes to Δ and thus attempts to fulfil Prescription (P_1). We can similarly learn λ_e to minimize the end-task validation loss.

So far, we have introduced independent weights, $\{w^k\}$, for each objective. This is sufficient in the case of unrelated objectives. However, the objectives in $\mathcal{A}$ share an underlying structure. We recognize this by using a factored approach to model each w^k. We introduce a factor vector for each of the 4 stages introduced in Section 5.4: $W^{\mathcal{D}} \in \mathbf{R}^{|\mathcal{D}|}, W^{\mathcal{T}} \in \mathbf{R}^{|\mathcal{T}|}, W^{\mathcal{R}} \in \mathbf{R}^{|\mathcal{R}|}$ and $W^{\mathcal{O}} \in \mathbf{R}^{|\mathcal{O}|}$. This ties together the weights of objectives that share primitives in common. To capture the fact that an objective can be more than the sum of it parts, we also introduce an independent weight for each objective : $W^{\text{All}} \in \mathbf{R}^{|\mathcal{D}|\times|\mathcal{T}|\times|\mathcal{R}|\times|\mathcal{O}|}$. Consider the objective k which is generated by the composition of the operations $\{d \in \mathcal{D},\ t \in \mathcal{T},\ r \in \mathcal{R},\ o \in \mathcal{O}\}$, its weighting is computed as : $w^k \propto \exp\left(W^{\text{All}}_{(d,t,r,o)} + W^{\mathcal{I}}_d + W^{\mathcal{T}}_t + W^{\mathcal{R}}_r + W^{\mathcal{O}}_o\right)$. Our factored approach not only allows us to share information between objectives but it also allows us to analyze which stages and primitives are most important to a particular end-task after training is completed (Section 5.8).

Prescription (P_2) from Section 5.5, advocates for introducing as much auxiliary data as possible. As such, instead of fixing to a specific subset throughout training for a particular end-task, we propose to utilize all the objectives in $\mathcal{A}$. This also avoids the combinatorial explosion that comes with exploring subsets of $\mathcal{A}$ at a time. $|\mathcal{A}|$ can be large and descending on all of $\mathcal{A}$ at once can be computationally prohibitive. As an efficient work around, at each training step, we sample a subset of $\mathcal{A}$ for execution with META-TARTAN. Our samples are drawn from all of $\mathcal{A}$ so any objective can get used at any timestep. Because we model each w^k via a factored approach, even if an objective is not sampled its weight is implicitly updated. Our approach is reminiscent of stochastic-relaxation weight sharing (Pham et al., 2018; Dong and Yang, 2019; Li et al., 2020) where sampled architectural primitives result in updates to shared model weights which can be used by other primitives that are not sampled.

We coalesce all the ideas we have introduced so far into Algorithm 2 which we dub AANG (**A**utomated **A**uxiliary Learni**NG**). At a high-level, given an end-task $\mathbf{E}$:

1. We generate a space of auxiliary objectives $\mathcal{A}$ by leveraging the taxonomy discussed in Section 5.4. $\mathcal{A}$ may contain auxiliary tasks that can improve our performance on $\mathbf{E}$.
2. We leverage MAML-style (Finn et al., 2017b) meta-learning to adaptively weight the objectives in $\mathcal{A}$ based on measuring each objective's influence on $\mathbf{E}$'s validation set loss.
3. We make our algorithm scalable by sub-sampling the tasks $\mathcal{A}$. By exploiting the underlying structure of the objectives in $\mathcal{A}$ via a factored approach to modeling task weights, we reduce the impact of the inexact sub-sampling.

5.7 Experimental Setting

Our exploration of auxiliary learning has made the following transitions from the status-quo: manual to automated, single task to multitask, end-task agnostic to end-task aware. In this sec-

tion, we set up experiments to validate these deviations from the standard.

We focus on continued pre-training (Gururangan et al., 2020a; Aghajanyan et al., 2021). In this setting, we perform further auxiliary learning on an already pre-trained model. We favor this setting over pre-training from scratch (Liu et al., 2019a; Yang et al., 2019) not only because it is a more computationally feasible arena for experimentation but also because it is more relevant to modern ML systems where building upon pre-trained models is the norm (Qiu et al., 2020; Du et al., 2020).

Model Details and Datasets: We use a pre-trained RoBERTa$_{\text{base}}$ (Liu et al., 2019a) as the shared model base. We implement each auxiliary objective as a separate head on top of this shared base. For classification based objectives, the output head is a 2-layer multi-layer perceptron (MLP) that receives representations for the special classification token `[CLS]` (Devlin et al., 2018) from RoBERTa$_{\text{base}}$. For sequence generation objectives, we make a copy of the pre-trained output layer of RoBERTa$_{\text{base}}$ for each task. Table D.1 in Appendix C.2 provides details of the 5 datasets used. All datasets are low-resource classification tasks. Not only are these datasets more amenable to meta-learning from a computational standpoint, but low-resource tasks also benefit the most from auxiliary learning. We also choose these tasks because they feature in previous work which we use as baselines (Gururangan et al., 2020a; Dery et al., 2021a)

Table 5.1: AANG-TD (task data) has 24 objectives and is based on only end-task data. AANG-TD+ED (task data + external data) has 40 objectives and uses both end-task and in-domain data.

	$\mathcal{I}$	$\mathcal{T}$	$\mathcal{R}$	$\mathcal{O}$
TD	End-task	*BERT-op*	Bi-directional	Denoise Token
		Mask	Left-to-Right	End-task
TD+ED	End-task	Replace	Right-to-Left	
	In-Domain data	No-op	Random-Factorized	

Baselines and Search Spaces: The following methods are end-task agnostic baselines. By end-task agnostic, we mean that these do not multitask with the end-task. Finetuning on the end-task occurs *after* training on the auxiliary objective.

1. **RoBERTa (Liu et al., 2019a):** We simply finetune a pre-trained RoBERTa$_{\text{base}}$ on the end-task.
2. **TAPT (Gururangan et al., 2020a):** Continue training RoBERTa$_{\text{base}}$ on masked language modelling on end-task data itself before finetuning on the end-task.

The following named objectives are end-task aware baselines that use META-TARTAN (Dery et al., 2021a) but utilize only 1 auxiliary task. Each auxiliary objective is multi-tasked with the end-task.

1. **GPT-style:** We perform end-task aware training with a denoising auxiliary objective based on left-to-right causal masking for computing representations. $\{\mathcal{I}$ = End-task data, $\mathcal{T}$ = No-op, $\mathcal{R}$ = Left-To-Right, $\mathcal{O}$ = Denoise Token $\}$.

2. **XLNET-style:** This is a denoising auxiliary objective that uses randomized masking for computing representations. $\{\mathcal{I}$ = End-task data, $\mathcal{T}$ = No-op, $\mathcal{R}$ = Random-factorized, $\mathcal{O}$ = Denoise Token$\}$.
3. **BERT-style / TAPT:** Denoising inputs corrupted via *BERT-Op*: 80% masking and 10% random replacement. $\{\mathcal{I}$ = End-task data, $\mathcal{T}$ = *BERT-Op*, $\mathcal{R}$ = Bi-directional, $\mathcal{O}$ = Denoise Token$\}$. Please note that this baseline is equivalent to META-TARTAN as introduced in Dery et al. (2021a).

Table 5.1 details the search spaces that we evaluate against the above baselines. This is by no means the most encompassing search space but we leave more expansive space design to future work. Please note that all tasks within AANG-TD, and those with $\{\mathcal{I} =$ End-task$\}$ in AANG-TD+ED, are instantiations of task augmentation as introduced in Section 5.4.

Training Details : Please see Appendix C.3 for more details about hyper-parameter configurations.

5.8 Results and Discussion

In this section, we experimentally validate our case for automating the creation of auxiliary objectives and using them in an end-task aware multitask fashion.

5.8.1 Going a Long Way Without External Data

We first consider the setting where we rely solely on end-task data (task augmentation), and work with the AANG-TD search space. This search space has 24 objectives. Table 5.2 shows that automatically generating auxiliary objectives from only task data and using them appropriately is productive.

End-task awareness is key: From Table 5.2, methods that are end-task aware result in over 1.12% average improvement over those that are end-task agnostic even under the most generous comparison (GPT-style 79.84% vs task-agnostic TAPT 78.72%). Knowing the end-task means that at each iteration, AANG can make informed gradient updates by adapting task weights so the resulting auxiliary task better aligns with the end-task (Prescription (P_1)). Amongst the single task objectives, BERT-style performs best. We posit that this is because RoBERTa was trained from scratch on a similar objective and so this objective represents minimal shift in training distributions.

Adaptive multi-task auxiliary learning improves performance: We compare single-task end-task aware auxiliary learning to its multitask variant. Table 5.2 shows that multitasking our 3 different types of language modelling tasks results in improved average performance over using the tasks individually (81.12% for the BERT-style and 81.55% for combining the three single task objectives). We get our best performance when we multitask 24 auxiliary objectives automatically generated with our framework using AANG-TD. Boosting the number of objectives from 3 to 24 resulted in a 0.66% improvement in average performance across tasks. This is in line with Prescription (P_2) from Section 5.5 since we are increasing the effective amount of aux-

Table 5.2: Our framework and AANG on tasks **using only task data**. Without using any external data, we are able to get significant average performance improvement over baselines. Superscripts are p-values from paired t-tests (best multitask versus best single-task).

Task Adaptive	Method	#	CS		BIOMED	NEWS	STANCE	
			ACL-ARC	SCIERC	CHEMPROT	H.PARTISAN	SE-2016-6	AVG
No	RoBERTa	1	$66.03_{3.55}$	$77.96_{2.96}$	$82.10_{0.98}$	$93.39_{2.26}$	$70.37_{1.51}$	77.97
	TAPT	1	$67.74_{3.68}$	$79.53_{1.93}$	$82.17_{0.65}$	$93.42_{2.87}$	$70.74_{1.21}$	78.72
	[OURS] Static Multitask-TD	24	$69.60_{3.80}$	$\mathbf{83.37}_{0.58}$	$83.42_{0.26}$	$97.95_{0.73}$	$71.02_{0.43}$	81.07
Yes	X. GPT-style	1	$67.22_{0.44}$	$81.62_{0.84}$	$83.29_{1.21}$	$96.41_{0.73}$	$70.67_{1.46}$	79.84
	Y. XLNET-style	1	$69.76_{2.42}$	$81.81_{0.42}$	$83.39_{0.31}$	$96.41_{1.92}$	$71.18_{0.58}$	80.51
	Z. BERT-style (Dery et al., 2021a)	1	$70.08_{4.70}$	$81.48_{0.82}$	$\mathbf{84.49}_{0.50}^{(0.09)}$	$96.84_{1.72}$	$72.70_{0.60}$	81.12
	[OURS] AANG-[X+Y+Z]	3	$71.51_{3.19}$	$82.89_{0.78}$	$83.68_{0.45}$	$96.92_{1.26}$	$\mathbf{72.75}_{0.82}^{(0.94)}$	81.55
	[OURS] AANG-TD	24	$\mathbf{73.26}_{1.32}^{(0.28)}$	$\mathbf{82.98}_{1.52}^{(0.27)}$	$83.91_{0.32}$	$\mathbf{98.46}_{0.0}^{(0.14)}$	$72.46_{1.65}$	**82.21**

iliary data. We further posit that introducing more auxiliary objectives also serves to implicitly regularize the end-task during training.

5.8.2 Introducing External Data

For the ACL-ARC task, we experiment with introducing auxiliary tasks based on external data. AANG-TD+ED has 40 tasks, 16 of which are based on domain data. We introduce CS domain data (from the S2ORC dataset (Lo et al., 2019)) that is $n = 10\times$ the size of the task data. From Figure 5.3 we see that AANG-TD+ED makes better use of domain-data than doing end-task aware training using only BERT-style objective with task (TAPT) and domain-data (DAPT) jointly as in Dery et al. (2021a). However, AANG-TD+ED (73.70) does not significantly improve over AANG-TD (73.26) on the ACL-ARC task (Figure 5.3). This might seem at odds with Prescription (P_2) since the TD+ED search space introduces more data. However, note that the AANG search algorithm is approximate and as such, with a larger search space, it can be harder to find composite tasks with a small Δ as suggested by Prescription (P_1). We posit that we need more external data than $n = 10\times$ in order to see marked improvements to offset our inexact search of the space of composite functions. However, such scales are outside our computational budget.

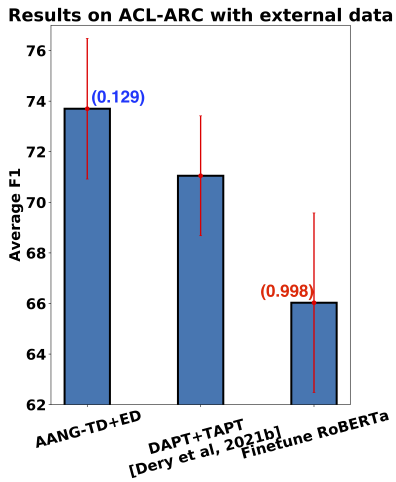


Figure 5.3: AANG effectively leverages out-of-task data. P-values (in brackets) are comparisons to (Dery et al., 2021a)

5.8.3 Why does AANG Work ?

To better understand why our auxiliary learning pipeline improves end-task performance, we perform multiple ablations under AANG-TD.

Static versus Dynamic Weighting: We ablate the impact of using static task weights throughout training, as against adaptive task weights. Just as with AANG, we sub-sample n tasks from the search space at every iteration (n is cross-validated exactly as AANG is – Table C.1). Each sampled tasks weight is initialized to $\frac{1}{n}$ and this remains unchanged throughout training. This is the Static Multitask-TD baseline in Table5.2. AANG-TD improves upon the static multitask baseline by over 1.1% on average. With adaptive weighting, AANG down-weights objectives that are harmful to the end-task whilst up-weighting relevant ones (Prescription (P_1)). However, using static weightings is more compute friendly since we do not have to calculate task-weight meta-gradients. This compute-vs-performance trade-off is left for practitioners to resolve based on their available resources.

Impact of number of sampled objectives: Due to computational constraints, AANG sub-samples the set of generated objectives. Whilst this sampling can result in approximation error when inferring task weightings, it can also introduce stochasticity which can help regularize the learned model. From Table C.1 (Appendix C.1) we find that for some tasks (ACL-ARC and SCIERC) sampling a larger number of tasks helps. SE-2016-6 and CHEMPROT on the other hand benefit from smaller number of sampled tasks. Our recommendation is that the number of sampled tasks be cross-validated on a per-task basis.

Learned task weight trajectories: AANG learns interesting trajectories for weighting design stage primitives. From Table 5.2, the fact that AANG-TD roughly matches the best single task performance ($72.46_{1.65}$ versus $72.70_{0.60}$ for BERT-style) on the SE-2016-6 task suggests that it may be learning to mostly up-weight this task. Figure 5.4 provides evidence of this. For the SE-2016-6 task (row 1), composing the highest weighted primitive from each stage [BERT $\circ$ None $\circ$ DENOISE] results in BERT-style, the best single task objective. Figure 5.4 also shows that AANG can adapt to overfitting. The vertical black lines indicate the point of best validation set performance. AANG responds to over-fitting by down-weighting objectives based on the output loss being over-fit to. Thus, after several iterations, the objective that dominates when the validation performance is at its highest (black vertical line) gets down-weighted in response to it becoming saturated.

What tasks are important and when they are important? We study which tasks are most highly weighted early in training (first 10% of learning trajectory) and later in training (last 50%). We aggregate statistics across 3 datasets. Note that early in training, objectives based on the self-supervised output $\mathcal{O} = \{\text{DENOISE}\}$ are highly weighted but later, objectives based on supervised signal, $\mathcal{O} = \{\text{Task}\}$ play a larger role. AANG rediscovers the common practice of training on self-supervised objectives before introducing supervised ones. It is also interesting to note that many newly generated objectives (outside of the 3 named single task baselines in Table 5.2) such as simple input reconstruction were discovered to have relevant impact on the end-tasks. This means AANG can automatically surface new, previously unexplored objectives relevant to the end-task.

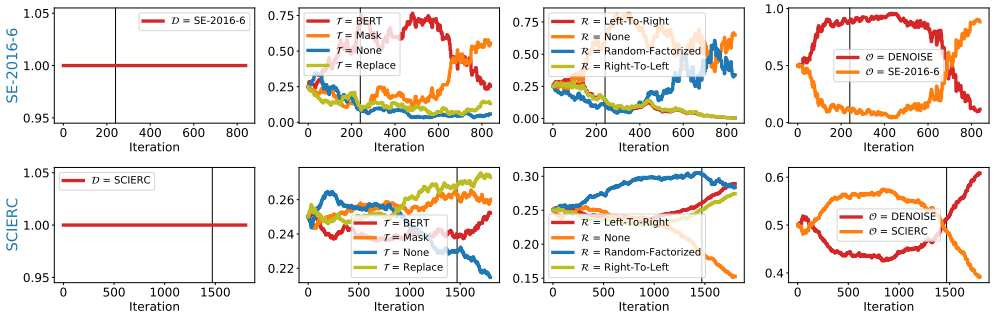


Figure 5.4: Learned trajectories for AANG-TD for run instances of SE-2016-6 and SCIERC tasks.

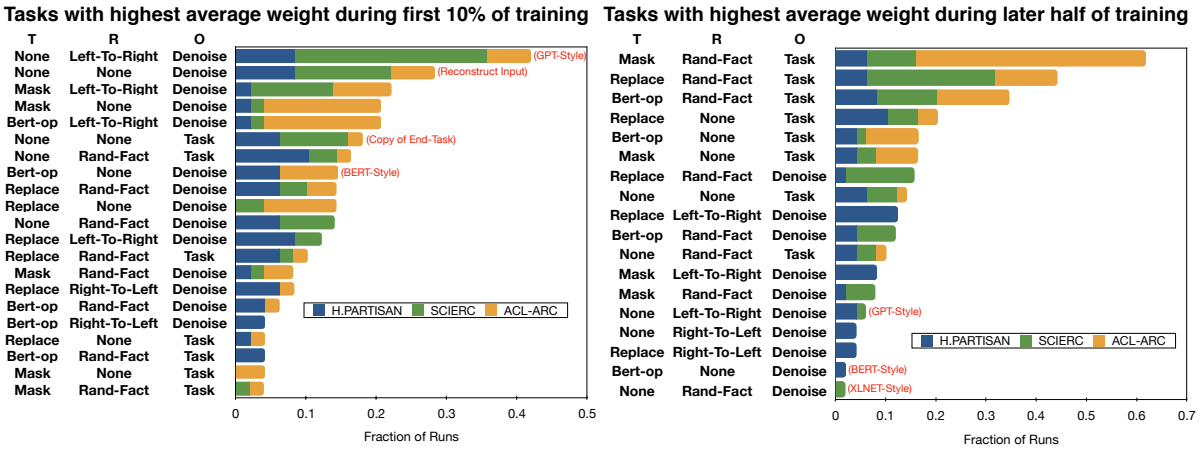


Figure 5.5: Top ranked objectives (averaged weight) early in training (left) and later in training (right)

5.9 Limitations and Conclusion

Our work has some limitations that we leave for future work. First, because AANG relies on meta-learning, it presents extra compute burden over simple multitasking. This is because, we have to independently compute meta-gradients for each auxiliary task thus requiring $\mathcal{O}(n)$ forward-backward operations for n sampled tasks compared to $\mathcal{O}(1)$ for static multitasking. In Table 5.2, we show that our static Multitask-TD method outperforms all other non-task-adaptive methods by $\approx 2.4\%$ and is thus a viable alternative when runtime is a signficant constraint. Secondly, AANG as presented is an approximate algorithm – primarily due to sub-sampling the space of tasks. Thus as mentioned in Section 5.8.2, we do not get as much gain as desired when our search space becomes larger. We leave finding an efficient exact search algorithm for future exploration.

This chapter presents a procedure for automating the creation of auxiliary objectives. We showed, theoretically, how auxiliary learning impacts end-task generalization. This resulted in prescriptions that informed the design of AANG, an algorithm to search the space of gener-

ated objectives in an end-task aware multitask fashion. Our experiments show that AANG is a promising first step in automating auxiliary learning.

Part III

On Memory and Compute Efficient Transfer Learning

Chapter 6

Structured Pruning of Pre-trained Models Under Limited Data

6.1 Chapter Overview

In Part II, we focused on how to perform data-efficient transfer learning by leveraging end task awareness. In this chapter and the following, we will focus on delivering small, task-specific, compute and memory efficient models by pruning and adapting large pre-trained models.

Specifically, in this chapter, we will explore the problem of structured pruning under limited target task data. While existing pruning algorithms can be efficient, the common practical setting where task-specific data is limited is yet to be addressed. To alleviate the data scarcity problem, we propose a structured pruning strategy that leverages transfer learning. Detailed analyses of simple transfer learning based remedies lead us to a simple, flexible formulation of what, how and when to transfer, resulting in pruned models with improved generalization over strong baselines.

6.2 Introduction

Large pre-trained language models have been successfully applied to a wide variety of application scenarios (Bommasani et al., 2021; Anil et al., 2023). However, not all applications can justify the cost of running such large models. E.g. an interactive, offline spellchecker for a phone has strong memory limits compared to a server-side chat model (Dettmers et al., 2022). Even server-side, the benefit/cost of large models depends on the application. This situation motivates research into structured model pruning algorithms.

Structured pruning algorithms generate smaller, faster and yet reasonably accurate sub-models from large pre-trained ones by removing components (beyond individual parameters) like convolutional channels, attention heads and whole layers. Several works over the years (Wang et al., 2019b; Sanh et al., 2020; Xia et al., 2022) have been proposed to perform task-specific structured pruning. Unfortunately, to the best of our knowledge, all existing algorithms have been developed without consideration for the amount of training data available for the target task. Thus, as Figure 6.1 shows that, even state-of-the-art methods like CoFi (Xia et al., 2022), do not gracefully handle scenarios with limited training data. We argue that the data-limited structured pruning

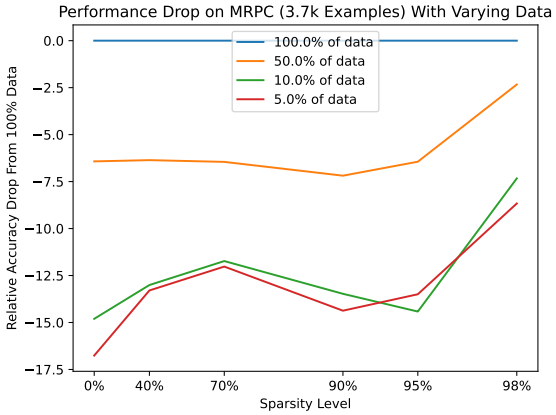


Figure 6.1: Accuracy degradation of CoFi (Xia et al., 2022) vs training data sizes. Sparsity level refers to the fraction of removed weights (excluding embeddings). Accuracy at 50% data is stable across sparsity levels (except for 98% sparsity) while more data-limited regimes (10%–5%) exhibit stronger sensitivity to the sparsity level.

setting is important since limited compute for inference and data scarcity for training tend to co-occur often in practice Ahia et al. (2021). A popular remedy to the limited data problem at fixed model size, is to leverage transfer learning (Caruana, 1997a; Erhan et al., 2010; Dery et al., 2022) by introducing external data or extra tasks. In this work, we investigate transfer learning based remedies for structured pruning under limited data. Structured pruning algorithms need to jointly learn both model weights and structural variables (which layers, attention heads, etc. to prune) for the final size-reduced model Wang et al. (2019b); Xia et al. (2022). This added complexity makes deploying transfer learning in the structured pruning setting non-trivial and raises several questions. Do we only perform transfer learning for model weights or do we include structural variables too? How do we learn structural variables for the target task in a way that benefits from the presence of a transfer task? When is it best to introduce transfer learning so as to produce the most accurate pruned target model?

This work aims to provide answers to the questions above. We propose a simple modification to existing structured pruning algorithms to allow for effective transfer of both structural variables and model parameters. Overall, our analyses allow us to provide prescriptions to researchers about what, how and when to transfer during structured pruning. Our effort results in significant improvements in generalization performance even at compression ratios as high as $50\times$.

6.3 Background

Unstructured Pruning approaches sparsify models by zeroing out individual components of weight matrices (Frankle and Carbin, 2018; Sanh et al., 2020). The resulting sparse matrices reduce the memory overhead of the model but run-time gains cannot be realized unless on specialized hardware (Liu et al., 2018b; Ma et al., 2021). Over the years, many criteria for choosing which parameters to remove have been explored. Some approaches like magnitude pruning Han et al. (2015a) and Wanda Sun et al. (2023b) prune parameters based on either their magnitudes

or the magnitude of their product with previous layer activations respectively. Other approaches like Frankle and Carbin (2018); Sanh et al. (2020) use information about about much parameters have changed since initialization whilst others learn unstructured masks based using gradient descent (Ramanujan et al., 2020). Ahia et al. (2021) introduce the term *the low-resource double-bind* for the challenge of compressing models in data limited regimes. Unlike us, they study magnitude pruning, which as mentioned, does not ordinarily lead to run-time gains. They also do not propose a remedy for the limited-data problem, which we do in this paper.

Structured Pruning algorithms remove whole components from pre-trained models such as attention heads (Michel et al., 2019; Voita et al., 2019), whole layers (Fan et al., 2019) or intermediate dimensions of fully connected layers (Wang et al., 2019b) in order to produce faster, memory efficient sub-models without overly sacrificing downstream accuracy. Unlike unstructured pruning, there is no need for specialized hardware in order to realize the run-time speedups from compression. These approaches require optimizing over structural variables (to decide which model components to prune) and model weights (to adapt the final model to the disruption that results from removing whole components). Joint optimizations like these mean more variables to learn, resulting in the need for mode end-task data points. To the best of our knowledge, we are the first consider the challenge of structured pruning under limited data.

Other model compression approaches Quantization methods (Polino et al., 2018; Dettmers et al., 2022) reduce model size by reducing the number of bits required to represent each weight. These methods are generally complementary to pruning approaches but only achieve maximum size reductions on the order of 2-4$\times$ before substantial model performance degradation. We are interested in achieving extreme compressions to the order of 50$\times$ reduction without significant loss in performance. Distilling directly to a target task has been shown to be a data-hungry process (Jiao et al., 2019), often requiring a general distillation step (on abundant external data) to be able to achieve competitive performance with approaches modern structured pruning methods like CoFi Xia et al. (2022).

Multitask Transfer Learning (Caruana, 1997a) is a common recipe for improving a models average performance on a desired end-task. When the end-task is data-limited, auxiliary tasks can be multi-tasked with the end-task (Dery et al., 2021b,a) to serve as proxy data. Previous work at the intersection of pruning and multitasking have only studied how to prune multi-task models (Garg et al., 2023; Yang et al., 2023). Unlike these, our starting point is not a multitask model but a generalist pre-trained model like BERT (Devlin et al., 2018). Our work is interested in using multitasking in as much as it improves generalization of the pruned model with respect to the data-starved end-task only.

6.4 Methodology

The goal of this paper is to improve the generalization of pruned models when the end-task is data-limited without sacrificing memory and run-time gains. We assume that we are given a

structured pruning algorithm that jointly learns structural/masking variables (which we denote as $\{\mathbf{z}^k_{\text{target}}\}$) and their corresponding parameters $\{\theta^k_{\text{target}}\}$ for the target end-task. k indexes the set of K structural variables being explored. We are primarily concerned with how to incorporate a transfer task by learning $\left[\{\mathbf{z}^k_{\text{transfer}}\}, \{\theta^k_{\text{transfer}}\}\right]$ such that we enjoy improved generalization with the target task's final model. Since our focus is on the limited-data problem, we care less about a specific structured pruning algorithm and more about how to adapt any appropriate algorithm in the data starved setting. We therefore focus on building on top of a state-of-the-art structured pruning algorithm, CoFi Xia et al. (2022) which we take as a representative algorithm. Whilst we describe CoFi below to provide sufficient background, for the rest of the paper, we will abstract away the details of the pruning algorithm and focus on the specifics of adapting transfer learning to this setting.

6.4.1 CoFi

CoFi (**Co**arse- and **Fi**ne-grained Pruning) is a mixed resolution structured pruning algorithm. Previous algorithms to prune transformer models Vaswani et al. (2017) have focused on removing high level units like whole layers (Fan et al., 2019) or finer grained modules like attention heads Voita et al. (2019) and dimensions of fully connected layers (Wang et al., 2019b) but not both types. CoFi introduces variables that account for pruning at multiple levels of granularity.

Coarse Grain: Each transformer layer consists of a multi-headed attention component that feeds into a fully connected two-layer non-linear perceptron Vaswani et al. (2017). CoFi introduces variables sets $\{z^i_{\text{MHA}}\}_{i\in[N]}$ and $\{z^i_{\text{FFN}}\}_{i\in[N]}$ for each of the model N layers. z^i_{MHA} denotes the probability that the **whole** attention component of the ith layer is removed whilst z^i_{FFN} is similarly defined for the fully connected component of the specified layer. CoFi also removes whole columns of the residual stream: $z^\ell \in \mathbb{R}^d \rightarrow \widehat{z}^\ell \in \mathbb{R}^{\hat{d}} \quad \forall \ell \in [N]$. For a BERT model, $d = 768$ is typically reduced to $d \approx 750$. Xia et al. (2022) find that though relatively few columns are dropped, including columns as structural variables is important for producnig performant compressed models.

Fine Grain: Given a particular layer i, CoFi prunes subsets of the attention heads available. The variables $\{z^i_{j,\text{head}}\}_{[j\in n_h]}$ represent the jth attention head in the i layer which has n_h total attention heads. A similar set of variables is defined for the fully connected units within a layer : $\{z^i_{j,\text{fc}}\}_{[j\in n_f]}$ where the ith fully connected layer has n_f units.

For the jth attention head of the ith layer, the likelihood that this head is left unpruned is proportional to $z^i_{\text{MHA}} \cdot z^i_{j,\text{head}}$. This allows the algorithm to make coupled fine and coarse grained decisions that lead to improved results. We collectively represent $\{\mathbf{z}\}$ as the set of all structural variables that are learned by CoFi. For a model with parameters θ, $\{\mathbf{z}\}$ are learned by applying the reparameterisation trick on the hard concrete distribution (Louizos et al., 2018) and minimizing a joint loss wrt $\{\mathbf{z}, \theta\}$ that includes

1. distance from target size. CoFi follows Wang et al. (2019b) and adds a lagrangian term that penalizes deviations from the target sparsity.
2. target task loss. Practitioners ultimately want a pruned model that generalizes well on their end-task. CoFi jointly optimizes the target task loss along with the pruning objective in order to produce performant pruned models.

3. a distillation objective on the original large model. Following Sanh et al. (2020), CoFi jointly performs distillation and structured pruning by introducing a layer-wise distillation objective.

With these high level details in mind, we proceed to present our simple, transfer learning based modification to CoFi that leads to improved results in data-limited settings.

6.4.2 Transfer Learning for Structured Pruning under limited data

Given a target task $\mathbf{T}$ with limited training data, we want to improve the final model generated by CoFi through leveraging additional training data from an auxiliary task $\mathbf{A}$[1]. Let $\{\mathbf{z_T}, \theta\}$ be the initial set of all structural variables and model parameters for the target task and $\{\widehat{\mathbf{z}}_\mathbf{T}, \widehat{\theta}\}^\gamma$ be final output of CoFi at a chosen sparsity level γ. $\widehat{\mathbf{z}}$ are binary variables $\widehat{z}_i \in \{0, 1\}$ which indicate whether component i is dropped/masked out (0) or is retained (1). We would like a procedure that leverages the auxiliary task (with its own set of variables $\{\mathbf{z_A}\}$ such that the generalization performance of the pruned model when using using data from $\mathbf{A}$ and $\mathbf{T}$ jointly improves upon using only data from $\mathbf{T}$.

There are several design questions that arise in this setting when thinking about how to effectively utilize $\mathbf{A}$. In the following sections, we discuss some of these pertinent questions and propose some reasonable choices which we will later experimentally validate.

What criteria do we use to select the auxiliary task A ?

The choice of auxiliary task, $\mathbf{A}$, is an important design decision that must be considered carefully. A poor choice could result in poor generalization performance (with respect to $\mathbf{T}$) of the pruned model instead of being helpful. To this end, inspired by existing literature, we propose two criteria for evaluating what auxiliary task to leverage:

(1) resourcedness: Previous work on transfer learning for learning model parameters has demonstrated the benefits of leveraging large pools of data (which may possibly be unrelated to the eventual end-task) for pre-traing (Anil et al., 2023) or multi-tasking (Dery et al., 2021a). We therefore have a strong prior that using data-rich auxiliary tasks might be helpful for also learning structural parameters even if they are unrelated to the end-task.

(2) task-similarity Both theoretical (Baxter, 2000; Maurer et al., 2016; Dery et al., 2022) and empirical works (Gururangan et al., 2020a; Dery et al., 2022) have shown that transfer learning works best when the auxiliary task is similar or related to the end-task. As a proxy for similarity, we consider auxiliary tasks that are from the same *domain* as the end-task.

When should we introduce A?

Structured pruning approaches like CoFi usually perform a two stage process. In the first stage, they generate a pruned model at the desired sparsity level; this involves learning both $\{\widehat{\mathbf{z}}, \widehat{\theta}\}^\gamma_\mathbf{T}$. In

[1] we use $\mathbf{A}$ to denote the auxiliary task instead of the usual $\mathbf{T}_{\text{aux}}$ to avoid cluttered notation

the second stage, pruned model is then fine-tuned on the end-task by updating only $\widehat{\theta}^{\gamma}_{\mathbf{T}}$ keeping $\widehat{\mathbf{z}}^{\gamma}_{\mathbf{T}}$ fixed. The auxiliary task can be introduced in either or both of these stages. We explore following choices:

Prune$(A) \rightarrow$ **FT**(T)**:** We do structural pruning to learn both the weights and structure for a small model using ***only*** the transfer task, $\mathbf{A}$: $\{\widehat{\mathbf{z}}, \widehat{\theta}\}^{\gamma}_{\mathbf{A}}$. We then fine-tune (FT) the pruned model on the target task (T) only to obtain $\widehat{\theta}^{\gamma}_{\mathbf{T}}$. Here, the target task is used only in the final fine-tuning stage and is not involved in learning the pruned model structure.

Prune$(T) \rightarrow$ **FT**(A, T)**:** We learn both the weights and structure for a small model using the target task: $\{\widehat{\mathbf{z}}, \widehat{\theta}\}^{\gamma}_{\mathbf{T}}$. We share the pruned model parameters $\widehat{\theta}^{\gamma}$ and fine-tune on both (T) and (A). Here, the auxiliary task is used only in the final fine-tuning stage and is not involved in learning the pruned model structure.

Prune$(A, T) \rightarrow$ **FT**(T)**:** We learn both the weights and structure for a small model using both the transfer and end-task: $\{\widehat{\mathbf{z}}_{\mathbf{T}}, \widehat{\mathbf{z}}_{\mathbf{A}}, \widehat{\theta}\}^{\gamma}$ are learned jointly (we will explore how in Section 6.4.2). We then fine-tune (FT) the pruned model weights on the target task (T) only.

Prune$(A, T) \rightarrow$ **FT**(T, A)**:** We learn both the weights and structure for a small model using both the transfer and end-task: $\{\widehat{\mathbf{z}}_{\mathbf{T}}, \widehat{\mathbf{z}}_{\mathbf{A}}, \widehat{\theta}\}^{\gamma}$ are learned jointly (we will explore how in Section 6.4.2). We then fine-tune (FT) the pruned model weights on **both** the target and auxiliary tasks.

How do we incorporate A when optimizing for $\{\widehat{\mathbf{z}}_{\mathbf{T}}, \widehat{\theta}\}^{\gamma}$

When using the auxiliary task directly during pruning, there is the question of what the best way to jointly optimize $\{\widehat{\mathbf{z}}_{\mathbf{T}}, \widehat{\mathbf{z}}_{\mathbf{A}}, \widehat{\theta}\}^{\gamma}$ such that we achieve improved generalization for the final pruned model with respect to $\mathbf{T}$. Note that we are assuming that the model parameter weights θ are shared between the two tasks but the structural variables are separate. This is because there are many more model parameters than structural variables $\|\theta\|_0 \gg \|\mathbf{z}_{\mathbf{T}}\|_0 + \|\mathbf{z}_{\mathbf{A}}\|_0$. And so introducing separate model weights for the auxiliary task presents a much more significant modelling overhead than introducing new structural variables.

We can explore different strategies for sharing variables across the two tasks such that the $\mathbf{T}$ benefits from $\mathbf{A}$.

Single mask multi-task learns a single set of structural $\{\mathbf{z}\}$ and model $\{\theta\}$ parameters that are shared between both tasks. This choice tightly couples the two tasks. Whilst this allows maximal sharing of information between the target and transfer task, poor choices of transfer tasks could cause this to perform worse than no transfer at all.

Multi-mask multi-task learns distinct structural parameters $\{\mathbf{z}\}_{\mathbf{T}}$ and $\{\mathbf{z}\}_{\mathbf{A}}$ for each task but a single set of model parameters $\{\theta\}$ is shared between both tasks. There is no transfer of structural information and only the shared model parameters provide a coupling of the two tasks.

Our δ-Formulation aims to leverage strength from both alternatives. We propose this method where both tasks share a base set of structural variables $\{\mathbf{z}\}_{\text{base}}$ but also have task specific addends such that: $\{\mathbf{z}\}_{\mathbf{T}} = \{\mathbf{z}_{\text{base}} + \delta_{\mathbf{T}}\}$ and $\{\mathbf{z}\}_{\mathbf{A}} = \{\mathbf{z}_{\text{base}} + \delta_{\mathbf{A}}\}$. We regularize δ_* to encourage sharing between tasks via $\mathbf{z}_{\text{base}}$ whilst maintaining flexibility for task-specific modelling.

6.5 Experimental Setup

Our experimental framework is introduced to investigate the questions posed in the previous section.

Datasets We consider 3 pairs of tasks. One pair of classification tasks are from the computer science domain tasks – SCIIE (Luan et al., 2018) and ACL-ARC (Jurgens et al., 2018) with 3.2k and 3.7k training samples respectively. The second pair of tasks are biomedical domain tasks - RCT (Dernoncourt and Lee, 2017) (we artificially create a low-resource version of this task with 10k training samples) and CHEMPROT (Kringelum et al., 2016) which has 4.2k training samples. We use GLUE (Wang et al., 2018b) tasks for our last pair: STSB and MRPC are sentence similarity and paraphrase detection tasks with 7k and 3.7k train examples respectively. For the GLUE tasks, we follow previous work (Jiao et al., 2019; Wang et al., 2019b; Xia et al., 2022) and report results on the validation set. For Non-GLUE tasks, we report test set results. Please see Appendix D.1 for more details about the tasks we investigate.

Model Details Since we use CoFi (Xia et al., 2022) as our representative structured pruning algorithm, we use the same model configuration. We use the $\text{BERT}_{\text{base}}$ (Devlin et al., 2018) which has ~ 110M parameters. We explore pruned model sparsities in the set $\{40\%, 70\%, 90\%, 95\%, 98\%\}$. $\gamma\%$ sparsity means that the model has been reduced to $(100-\gamma)\% \times 110$M parameters. Similar to Sanh et al. (2020) we also freeze the model embedding weights. See Appendix D.2 for details about training as well as hyper-parameter values.

Training details We mostly follow the training recipe from CoFi with a few minor changes. CoFi assumes that *one starts pruning after finetuning the full parent model on the target task* and so introduces a distillation loss as part of the pruning objective. In our case, we start directly from the pre-trained model without first fine-tuning on the target task. This is because of the risk of over-fitting due to the smaller target task size. Due to this, we find that the distillation based losses from the original CoFi paper are unnecessary and we did not see significant performance differences with or without them. When multitasking, we explore a small set of weighting hyper-parameters $\{(1.0, 1.0)\ ,\ (1.0, 2.0)\ ,\ (2.0, 1.0)\}$ for any losses relating to the target and auxiliary tasks respectively. Table D.2 has details of the hyper-parameters we cross validate against for all our experiments.

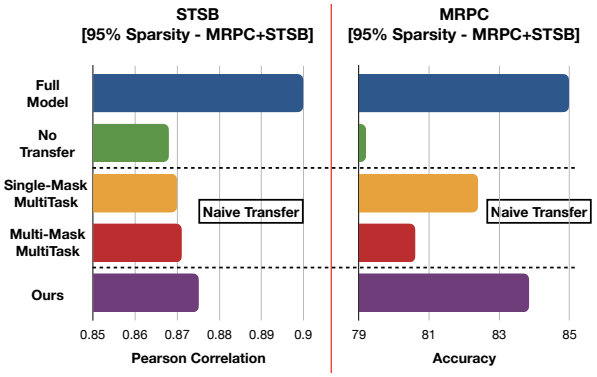


Figure 6.2: STSB and MRPC performance at 95% sparsity. Our proposed δ-Formulation outperforms all other methods on both tasks.

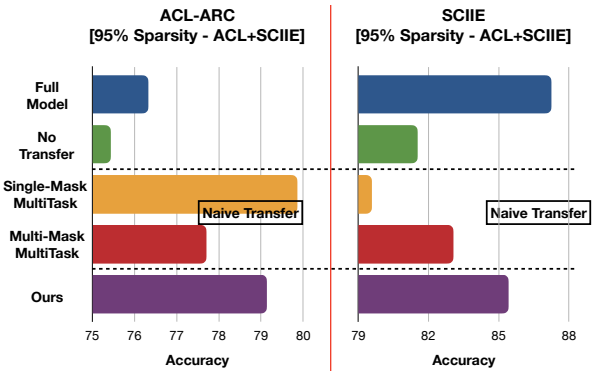


Figure 6.3: SCIIE and ACL_ARC performance at 95% sparsity. Our δ-Formulation produces the best average performance across the two tasks when either is used as the auxiliary task for the other.

6.6 Empirical Recommendations for practitioners

In Section 6.4.2, we posed different design questions around how to perform transfer learning for structured pruning under limited data and presented different options for resolving said questions. In this section, we proceed to perform a sequence of experiments to validate which choices lead to superior end-task generalization after pruning, so we can make principled recommendations to practitioners.

6.6.1 How should you transfer?

In Section 6.4.2, we introduced various approaches for coupling the auxiliary task with the target task during structured pruning. Figures 6.2, 6.3 and 6.4 , show experimental results after implementing various options with different pairs of datasets. Across all dataset pairs, our δ-Formulation produces the best performance when averaged across the task pair.

For the SCIIE task, tightly coupling its structural variable with those of ACL_ARC (as an auxiliary task) under the single-mask multi-task approach can negatively impact performance

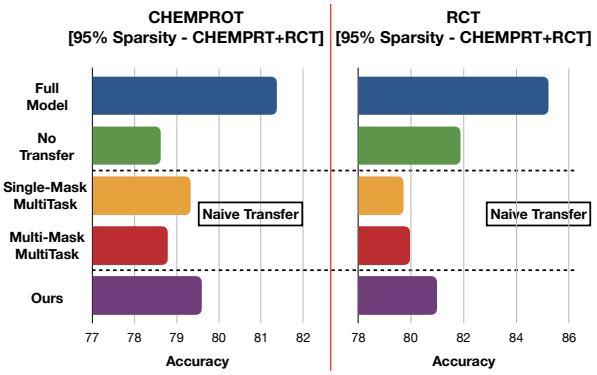


Figure 6.4: RCT and Chemprot performance at 95% sparsity. We see negative transfer from Chemprot to RCT across all transfer methods. Our proposed approach suffers least from performance degradation.

compared to not doing transfer learning at all (Figure 6.3). Our δ-Formulation ensures that SCIIE actually benefits introducing transfer learning by outperforming the multi-mask multi-task approach that fully decouples the structural variables. For the ACL_ARC task, our formulation recovers close to the best performance (single mask multi-task). Note that in principle, our formulation can mimic the single-mask multitask setting by using a high enough regularization on the δ offsets but we used a default l_2-regularization strength of $1e^{-2}$ for all experiments to exhibit robustness of our method. It is interesting to note that for the ACL_ARC task, all transfer learning approaches at 95% sparsity outperform training the full model on task data only. Note from Table D.1 that ACL_ARC is our smallest dataset. We posit that training the full, large model on this task leads to overfitting, resulting in poor generalization compared to leveraging transfer-learning at a reduced model size.

Figure 6.4 presents an interesting scenario where Chemprot benefits from transfer but RCT does not. Whilst this could be due to the fact that we perform limited hyper-parameter tuning (mainly to exhibit the robustness of our method and to reflect compute constrained settings), it is encouraging to see that the δ-Formulation for coupling structural masks significantly helped dampen the impact of negative transfer in the case of RCT as the target task.

6.6.2 What should you transfer?

So far, we have discussed using the auxiliary task when learning both the structural variables and parameters of the pruned model. In this section, we investigate if transferring both is needed. We perform the following ablation at 95% sparsity to determine what is most important to transfer. For this, we assume that the *the target task is not used during pruning* but is only introduced during the final fine-tuning of the smaller, pruned model.

Weights Only: We learn model weights and structural mask for the auxiliary task only. We then generate a random structural mask at the appropriate sparsity level (95%) and extract the model weights corresponding to this mask from the model trained on the transfer task. We then fine-tune this smaller, pruned model on the target task.

Masks Only: We learn model weights and structural mask for the auxiliary/transfer task only.

Table 6.1: Transferring structure, weights or both on STSB and MRPC? It is most beneficial to transfer both the learned weights and structural variables (masks)

	Metric	No Transfer	Weights Only	Structure Only	Both
STSB $\rightarrow$ MRPC	Accuracy %	79.2	68.4 ($\downarrow$)	76.96 ($\downarrow$)	79.7 ($\uparrow$)
MRPC $\rightarrow$ STSB	Pearson C.	0.868	0.23 ($\downarrow$)	0.8527 ($\downarrow$)	0.871 ($\uparrow$)

Table 6.2: When to introduce each task on MRPC and STSB? We find that it is optimal to prune with both the auxiliary and target task jointly.

	Metric	No Transfer	Prune(A) $\rightarrow$FT(T)	Prune(T) $\rightarrow$FT(A, T)	Prune(T, A) $\rightarrow$FT(T)	Prune(T, A) $\rightarrow$FT(A, T)
STSB $\rightarrow$ MRPC	Accuracy %	79.2	79.7 ($\uparrow$)	83.09 ($\uparrow$)	83.82 ($\uparrow$)	84.56 ($\uparrow$)
MRPC $\rightarrow$ STSB	Pearson C.	0.868	0.871 ($\uparrow$)	0.861 ($\downarrow$)	0.8751 ($\uparrow$)	0.872 ($\uparrow$)

We then reset the model weights to the pre-trained (not-yet-finetuned) state. Given the learned mask from the transfer task, and the untuned model weights, we then fine-tune this pruned model on the target task.

Masks and Weights: We use the transfer task to learn both the model weights and structural mask. We take weights and masks of this small model and fine-tune it on the target task.

Table 6.1 shows the results of this ablation. For both STSB $\rightarrow$ MRPC and MRPC $\rightarrow$ STSB, we see that if we are only introducing the target task in the fine-tuning stage, it is beneficial to transfer both the weights and structure that are learned from the auxiliary task.

6.6.3 When should you transfer?

Table 6.2 shows results for the different choices presented in Section 6.4.2 relating to when to introduce the transfer task. These experiments are also conducted at a target sparsity of 95%.

We obtain the best performance with the Prune$(A, T) \rightarrow$ FT(T) and Prune$(A, T) \rightarrow$ FT(A, T) approaches. This matches intuition because we expect an appropriately chosen auxiliary task to be helpful in terms of learning both structure and parameters of the final pruned model. Thus, introducing it in the first (pruning) stage mitigates the challenge that is exacerbated by learning a larger set of variables from limited data.

6.6.4 How should we choose the transfer task?

Table 6.3 contains experimental results highlighting our investigation of different variables that can impact the quality of a transfer task.

We get the best improvements when we use a high resource auxiliary task from the same domain as the target task. As mentioned in Section 6.4.2, we use domain as a proxy for task

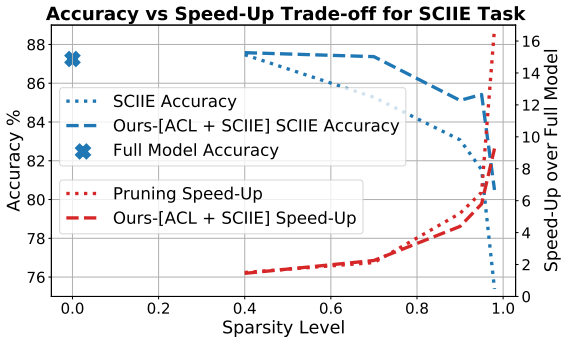


Figure 6.5: Accuracy vs speed tradeoff on SCIIE. Our δ-formulation, gives accuracy boosts on SCIIE at varying levels of compression.

relatedness. We see that even using a high resource task that is out-of-domain with respect to the end-task (RCT) can improve generalization over not introducing a transfer task at all (85.29 versus 79.2 for MRPC) and (0.873 versus 0.863 for STSB).

Table 6.3: Selecting the auxiliary task. A high-resource, in-domain task leads to the best result. For all experiments, best results from hyper-parameter search are reported. All models (except Full BERT) are pruned to 95% sparsity.

Target	Full BERT	No Transfer	Domain	Resourced-ness	Transfer Task	Performance
MRPC	83.48	79.2	In-Domain	High (364k)	QQP	**85.78**
			In-Domain	Low (7k)	STSB	83.82
			Out-of-Domain	High (180k)	RCT	85.29
STSB	0.901	0.868	In-Domain	High (364k)	QQP	**0.877**
			In-Domain	Low (3.7k)	MRPC	0.875
			Out-of-Domain	High (180k)	RCT	0.873

6.6.5 Does the learned structured sparsity translate to hardware speedups?

So far, we have only discussed the impact of transfer learning on generalization with respect to the end-task. However, when generating pruned models, we not only care about their generalization but also the degree of speedup that is achieved at the target sparsity.

Taking SCIIE as our primary task and ACL-ARC as the transfer task, we explore the accuracy-speedup tradeoff that is induced by leveraging transfer learning for structured pruning. We vary the degree of compression from 40% sparsity to 98%. To benchmark speed, we use the wall-clock time required to perform inference on the full SCIIE dataset through the model using at a batch-size of 128. All experiments were conducted on NVIDIA V100 GPUs. Figure 6.5 summarises our findings. For SCIIE+ACL, we fix the task weighting to the best performing configuration from our 95% sparsity experiments, the rest of the hyper-parameters are cross-validated from

values in Table D.2. At 50× compression (95%) sparsity, we are able to obtain a $\sim 5\%$ boost in accuracy over not using a transfer task, whilst achieving a $\sim 10\times$ speedup in inference. Transfer leraning enables a more graceful degradation in accuracy (dashed blue line) whilst still finding pruned models with comparable speed-ups.

Another view of Figure 6.5 is to consider the model size required for a threshold level of accuracy for deployment. At a threshold of 84% accuracy, whilst naive pruning results would produce a model at 80% sparsity, we are able to produce one at 95% sparsity! This is a $\sim 1.2\times$ memory saving and $\sim 2.8\times$ inference speedup.

6.6.6 What are the structural differences between a pruned model using transfer learning and without?

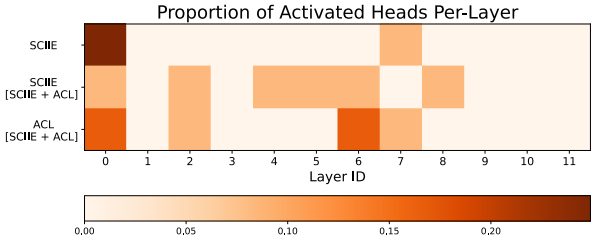


Figure 6.6: Structural visualization at 98% sparsity. Qualitatively, using a transfer task changes the pruned model structure significantly. The ACL transfer task in this case induces the learned SCIIE structure to be more diffuse across the layers of the model.

At extreme sparsity levels, the differences in speedup from learning a pruned model with and without transfer learning (Figure 6.5) suggest that the models discovered have different structures.

Figures 6.6 and 6.7 show the fraction of attention heads and MLP intermediate dimensions respectively, that are preserved across each layer with respect to the original $\text{BERT}_{\text{base}}$ model. Pruning with the target task alone results most of the preserved parameters coming from earlier in the network. With an auxiliary task however, the pattern of preserved modules is more diffuse across layers. We posit that this results from the the two tasks being multi-tasked preferring different layers thus resulting in a more diffuse distribution of preserved modules as a compromise in order to perform reasonably well on both tasks.

6.7 Conclusion

As coined in Ahia et al. (2021), the *low-resource double bind* describes the challenge of producing compressed models to serve compute-starved (memory and latency limits) tasks under a setting where these tasks also have limited data for pruning. In this work, we have explored adapting transfer learning, which has traditionally been leveraged only for learning model weights, to robustly prune models when the target task is data-limited.

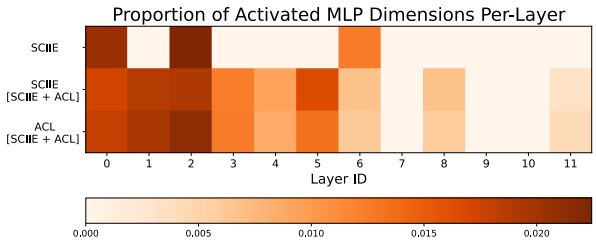


Figure 6.7: Structural visualization at 98% sparsity. Qualitatively, using a transfer task changes the pruned model structure significantly. The ACL transfer task in this case induces the learned SCIIE structure to be more diffuse across the layers of the model.

We have provided practitioners with recommendations on how to choose a transfer task, when and how to incorporate it into the pruning optimization procedure and what elements to transfer from the auxiliary task to the target. Equipped with this knowledge, we plan to explore the problem of structured pruning under limited target data for larger scale models.

Chapter 7

Structured Pruning of Large Pre-trained Models Under Limited Inference Compute and Memory

7.1 Chapter Overview

In the previous chapter, we considered the problem of task specific pruning of large pre-trained models when the target task is data-limited. Data constraints are not the only type of bottlenecks that everyday ML practitioners face. Given the burgeoning size of the current generation of large pre-trained models, many practitioners barely have the compute/memory to run inference on these models. Whilst structured pruning might seem like an obvious remedy, when the size of the current generation of pre-trained model is paired against the available hardware resources (of the everyday practioner), standard approaches to structured pruning become infeasible. This is because these methods perform backward passes through the large model to learn structural variables — a step not possible below certain GPU memory thresholds.

In this Chapter, we will propose an approach for memory-efficient pruning of large pre-trained models. Our approach will be usable in settings where we have enough memory to run inference / forward passes on the large model but not enough memory to perform backward passes on it. The inability to perform backward passes in this setting rules out most existing structured pruning algorithms.

7.2 Introduction

As large language models (LLMs) (OpenAI et al., 2023; Touvron et al., 2023; Team et al., 2023) continue to grow in size, the gap between models that achieve state-of-the-art performance and those that every-day machine learning (ML) practitioners can feasibly run on their available hardware continues to widen (Bender et al., 2021; Vivek, 2023; Samsi et al., 2023). With the goal of democratizing access to these powerful models, previous research has proposed approaches such as pruning (Xia et al., 2022; Sun et al., 2023a), distillation (Hinton et al., 2015; Gu et al., 2023) and quantization (Xiao et al., 2023) to create smaller models from larger pre-trained ones.

Unfortunately, so far, these methods have fallen short of the goal of truly democratizing access to LLMs. In the process of producing a smaller model out of an LLM, methods such as distillation and structured pruning are inaccessible to the everyday practitioner due to their prohibitive resource consumption at training time. Specifically, pure distillation-based techniques require running LLMs to generate large amounts of teacher data (Jiao et al., 2019; Hsieh et al., 2023) whilst existing gradient based structured pruning approaches like LLM-Pruner (Ma et al., 2023) and LoRAPrune (Zhang et al., 2023) require several times more memory than is needed to run inference on the model being pruned. Though unstructured pruning (Frantar and Alistarh, 2023; Sun et al., 2023a) and quantization are less restrictive at training time, the models they produce are not faster except in the presence of specialized hardware for the former (Mishra et al., 2021), whilst the latter can actually slow down inference due to added overhead (Dettmers et al., 2022). This limits the usefulness of these options for practitioners who are concerned with latency-critical applications. Table 7.1 summarizes the landscape of existing methods and their limitations.

Table 7.1: Landscape of resource consumption (memory and compute) of different model compression methods at training time and the inference time resource consumption of the models they deliver. ✗ $\rightarrow$ a prohibitive cost to the lay practitioner whilst ✓ $\rightarrow$ a viable option with respect to that resource.

Regime	**Resource**	**Approaches**				
		Quantization (Mixed Precision)	Distillation	Unstructured Pruning	Gradient-Based Structured Pruning	Bonsai (Ours)
Train	Memory	✓	✓	✓	✗	✓
	Compute	✓	✗	✓	✓	✓
Inference	Memory	✓	✓	✓	✓	✓
	Compute	✗	✓	✗	✓	✓

We aim to empower ML practitioners to compress LLMs *by themselves* using their available resources whilst still producing accurate yet fast and compact models. To this end, we propose a novel memory-friendly structured pruning method. We observe that the significant memory overhead of prior structured pruning methods chiefly comes from having to perform gradient-based optimization: a backward pass requires $\gtrapprox 2\times$ (Bridger, 2023) the memory consumption of a forward pass, while popular stateful optimizers like AdamW (Loshchilov and Hutter, 2017a) need $\gtrapprox 3\times$. To capture the widest range of memory budgets available to practitioners, we focus on developing an approach for the following concrete setting:

The practitioner only has enough memory on their hardware to run inference on the model to be pruned.[1]

The above setting is evergreen. As state-of-the-art models become more compute intensive over time, the generational gap in hardware available to the lay practitioner versus the most resource endowed institutions is expected to persist or possibly widen.

[1]We assume we can run a forward pass with a batch size of at least 1.

In light of the proposed setting, we present Bonsai, a structured pruning approach that only requires forward passes through the parent model and is capable of delivering fast, compact, and accurate pruned models under the memory limitations that are typical of consumer hardware[2]. To decide which modules (attention head, rows in feedforward projection, etc.) of the LLM to prune in the absence of gradient information, Bonsai estimates module importances perturbatively by generating sub-models and evaluating their performance by just running inference. We make this approach tractable by contributing multiple techniques.

First, we treat the problem of inferring each module's importance from the performance of generated sub-models as an underdetermined regression problem. This enables us to estimate the importance of a large number of modules by exploring a manageable number of random sub-models. This is unlike past perturbative approaches, which prohibitively require roughly as many sub-models as there are modules to select from Ancona et al. (2020), making them intractable for LLMs. Next, instead of instantiating sub-models by dropping modules with equal likelihood (Kang et al., 2023), we use informative priors derived from work on unstructured pruning (Han et al., 2015b; Sun et al., 2023a). We thus obtain better estimates of module relevance with fewer evaluated sub-models. Finally, unlike past gradient-free approaches that greedily make pruning decisions layer-by-layer (Dekhovich et al., 2021; Nova et al., 2023; Sun et al., 2023a), Bonsai takes a holistic view to preserve the accuracy of the pruned model: modules across layers are removed and evaluated together and relevance scores are computed globally to make pruning decisions.

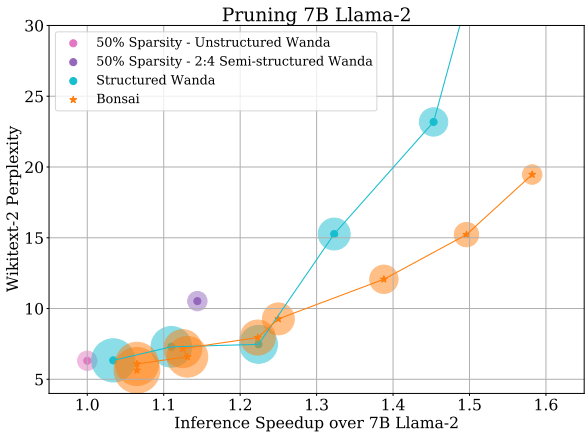


Figure 7.1: Perplexity versus inference speed-up of pruned models for methods that use only forward passes on the parent model. **At any given target perplexity, Bonsai produces the fastest model, resulting in improved latency and throughput**. Circle sizes $\propto$ model's memory footprint.

We conduct several experiments that demonstrate Bonsai's efficacy. We start by pitching Bonsai against a forward-only structured pruning version of Wanda (Sun et al., 2023a) – we show that our approach significantly outperforms this baseline. Next, we demonstrate that when the resulting pruned model small enough to be fine-tuned on available hardware, Bonsai achieves comparable performance to 2:4 semi-structured sparsity with Wanda (Sun et al., 2023a) – also finetuned – but is $2\times$ faster (an overall inference speed-up of $1.58\times$ compared to the parent model). Even when compared to SoTA gradient-based structured pruning methods like LLM-Pruner and LoRAPrune, Bonsai outperforms these methods on 4 out of 6 evaluation settings in our experiments. We also use Bonsai to prune the $\sim$3B Phi-2 (Li et al., 2023) model to a $\sim$1.8B

[2]We only ever run inference (i.e., forward passes) on the large parent model. If, however, the generated child model is small enough that that it can be fine-tuned with backward passes after pruning, we do so.

model that performs competitively against other sub-2B parameter model of the same class on the Huggingface Open LLM leaderboard. Based on these strong results and its usability under real-world memory constraints, we view Bonsai as a significant contribution to unlocking the power of LLMs for a broader spectrum of practitioners facing diverse hardware constraints.

7.3 Methodology

We will develop a structured pruning method that exclusively does inference on the parent model.

7.3.1 Background on Pruning, Problem Definition and Notation Setup

We are given an LLM, $\mathbf{M}_\theta$, parameterized by $\theta \in \mathbb{R}^D$. Also provided is U, a utility function that evaluates a model's performance on a target task. We are interested in pruning $\mathbf{M}_\theta$ to produce a smaller and faster but still performant (with respect to U) model under the constraint that we only have enough memory to run inference on $\mathbf{M}_\theta$. Even though we assume we can run $\mathbf{M}_\theta$ on available hardware, pruning can be critical for achieving latency targets, reducing compute burden, or making the model small enough to adapt to new (out-of-domain) tasks by gradient-based fine-tuning.

Unstructured pruning approaches compress $\mathbf{M}_\theta$ by removing individual parameters θ_j from the model. This results in the updated model consisting of sparsified weight matrices with a smaller memory footprint. Unfortunately, the updated model does not enjoy inference speedups except when specialized hardware is available and thus poses a compute burden during inference. Whilst semi-structured variants – those that remove parameters in patterns like 2:4 or 4:8 (Mishra et al., 2021) – achieve some speedup, these are modest compared to those achieved with structured pruning.

Structured pruning takes a more modular view of the units to be removed from $\mathbf{M}_\theta$. Take that $\mathbf{M}_\theta$ is made up of modules $\mathbf{m} = \{m_i\}_{i\in[N]}$ each with corresponding parameter counts $\mathbf{s} = \{s_i\}_{i\in[N]}$ such that $\sum_i s_i = D$. For a transformer model, $\mathbf{m}$ could consist of attention heads, dimensions of fully connected layers, or even whole layers. For simplicity, we assume that $\mathbf{m}$ is made up of non-overlapping modules. Structured pruning compresses $\mathbf{M}_\theta$ by finding accurate sub-models defined by subsets of $\mathbf{m}$: provided with $\bar{\mathbf{m}} \subseteq \mathbf{m}$, we can construct an updated model $\mathbf{M}_{|\bar{\mathbf{m}}}$ that is produced by dropping the modules not in $\bar{\mathbf{m}}$ from $\mathbf{M}$. Thus, given a sparsity target p, structured pruning can be cast as the following combinatorial optimization problem:

$$\mathbf{m}^* = \text{argmax}_{\bar{\mathbf{m}}\in\mathcal{F}_p} \quad U\left(\mathbf{M}_{|\bar{\mathbf{m}}}\right) \quad \text{where} \quad \mathcal{F}_p = \left\{\bar{\mathbf{m}} \subseteq \mathbf{m} \;\middle|\; \Big(\sum_{[j:m_j\in\bar{\mathbf{m}}]} s_j\Big) \leq (1-p)D\right\} \tag{7.1}$$

$\mathcal{F}_p$ consists of all sub-models that meet the sparsity threshold. Note that, in general, not only does $\mathbf{M}_{|\mathbf{m}^*}$ have a smaller memory footprint than $\mathbf{M}$, it is also faster to run inference on it since it has fewer modules. Many structured pruning methods attempt to solve Equation 7.1 by gradient-guided optimization (or search) over the space of sub-models. However, since we are interested in the memory-constrained setting where computing gradients is not feasible, these methods cannot be used.

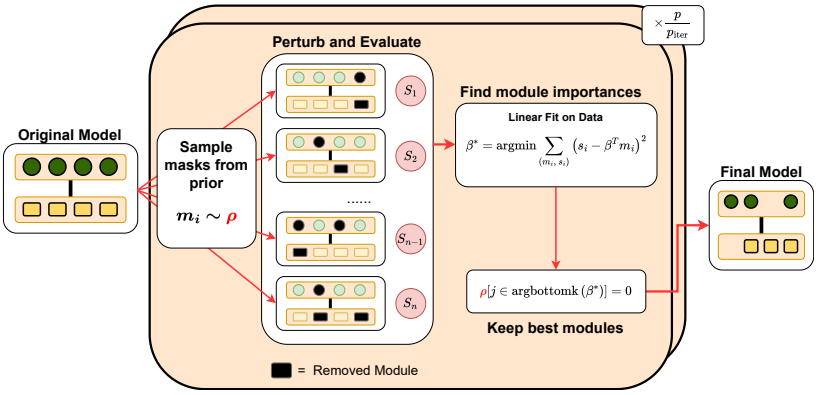


Figure 7.2: Overview of Bonsai. Perturbations to the original model are generated according to an intelligent prior. These perturbed models are evaluated on downstream task and the collected data is used to build a simple linear model of module importances. Bonsai iteratively drops modules of least importance until the target model size is reached.

7.3.2 Estimating module relevance with only forward passes

We have motivated the setting of pruning a model that is so large (relative to the amount of memory available to the practitioner), such that we can only run forward passes through it. This means that we have to solve Equation 7.1 by relying on only evaluations of U, as opposed to gradient-based optimization with respect to U. A brute force approach would enumerate all members of $\mathcal{F}_p$, find their corresponding performances and pick the best one. Unfortunately, this would be computationally infeasible since $|\mathcal{F}_p|$ is combinatorial in the size of the model.[3].

We propose a computationally tractable approach where we first perform a small number, n, of evaluations, where $n \ll |\mathcal{F}_p|$, to gather data for estimating the relevance of each module in $\mathbf{M}$ with respect to the metric U. Upon carrying out this estimation, we can greedily choose the member of $\mathcal{F}_p$ that has the highest total module relevance. Specifically, let us assume that we have estimated $\beta = \{\beta_i\}_{i \in [N]}$ to be the relevance of each of the N modules. We can generate an approximate solution to Equation 7.1 as:

$$\mathbf{m}^* \approx \mathbf{m}^{\text{approx}} = \text{argmax}_{\bar{\mathbf{m}} \in \mathcal{F}_p} \sum_{j \in \bar{\mathbf{m}}} \beta_j \tag{7.2}$$

Equation 7.2 is straightforward to solve as it simply requires sorting β_js and greedily selecting the top modules until the parameter constraint is met (this approach may slightly overshoot the sparsity constraint but since $s_i \ll (1-p)D \; \forall \; i$ for our settings of interest, the difference is not significant).

Estimating β: To obtain estimates of the module relevance scores $\beta \in \mathbb{R}^N$, we propose to generate and evaluate $n \ll |\mathcal{F}_p|$ sub-models, and construct a dataset of the sampled sub-models and their corresponding evaluated performances: $\mathbb{D} = \{\bar{\mathbf{m}}_k, U_k\}_{k \in [n]}$ where $U_k = U(\mathbf{M}_{|\bar{\mathbf{m}}_k})$.

[3]The typical fully connected (FC) sub-layer in an LLM has size $> 10^4$ units. Pruning just 1 of these FCs to 0.5 sparsity in this way requires evaluating over $\approx \binom{10^4}{10^3}$ subsets

Given $\mathbb{D}$, we treat the problem of estimating β as an under-specified regression problem :

$$\hat{\beta} = \text{argmin}_{\beta \in \mathbb{R}^N} \quad \frac{1}{n} \sum_{(\bar{\mathbf{m}}_k, U_k) \in \mathbb{D}} \left(U_k - \beta^T \alpha_{\bar{\mathbf{m}}_k}\right)^2 + \gamma \|\beta\| \tag{7.3}$$

where $(\alpha_{\bar{\mathbf{m}}_k})_i = \mathbf{1}[i \in \bar{\mathbf{m}}_k]$, is the binary vector that has 0 at indices with modules dropped. Implementing a sub-model $\bar{\mathbf{m}}_k$ as a binary mask $\alpha_{\mathbf{m}_k}$ is key to practically realizing our approach. We never actually instantiate sub-models as this would be prohibitively expensive. Instead, we create them *virtually* by zeroing out the outputs of the parts to be pruned so they have no effect on the model output.

7.3.3 Selecting sub-models for evaluation

An as yet unexplored design choice is how to choose the n candidate sub-models for evaluation. A naive approach would be to sample uniformly from the space of all feasible sub-models. However, we can show that this is sub-optimal. Take m_i as a module that is critical for good performance under evaluation with U. Since $n < N$, it means that some modules may never be "turned on" in the list of n chosen sub-models. If m_i happens to be one of these masked-out modules, the resulting estimate of $\hat{\beta}_i = 0$ would in turn result in $\mathbf{m}^{\text{approx}}$ being a poor estimate for $\mathbf{m}^*$. We therefore need to make a more informed choice to ensure that our estimates of β are accurate and useful.

Given a module m_i, we can set the likelihood of it being present in any of the n sampled sub-models to be proportional to some measure ρ_i which captures our prior belief that m_i is a useful module. To define ρ_i, we can turn to metrics from the pruning literature. Specifically, we set ρ_i to be a module-level analogue of any of the pruning metrics like Wanda (Sun et al., 2023a) or activation magnitude. As an illustration, consider a 1 hidden layer (of dimension d) network we wish to prune. If activation magnitude is the metric for our prior, we would calculate the activation vector for the d hidden dimensions – $\hat{\mathbf{a}}$ – averaged across multiple samples. The likelihood that the ith column of the matrix $W \in R^{d \times d}$ is left unpruned would be: $\rho_i \propto \hat{\mathbf{a}}_i = \frac{1}{B} \sum_b \left|\sigma\left(\left(W^T[i,:]\right) x_b\right)\right|$ where σ is the nonlinearity. Sampling sub-models according to ρ means we are more likely to explore models that already have a good performance U. The priors ρ_i can be computed via forward passes through the unmodified model $\mathbf{M}_\theta$ which allows us to respect memory constraints. Appendix E.6 has the list of priors we explore.

To enhance the efficiency of our method, instead of considering all active modules for pruning, in each layer we consider the bottom $2p$ fraction of modules[4] ranked according to our prior ρ. Thus, the top $1 - 2p$ fraction of modules remain fixed whilst the rest are perturbed and their relevance scores are computed and compared for pruning. This helps reduce the space of possible sub-models being considered for evaluation. For the bottom $2p$ fraction being compared, whenever we generate a mask $\alpha_{\mathbf{m}_k}$ with sparsity level p, we also generate its complement $\alpha^c_{\mathbf{m}_k}$, which is obtained by flipping the values in $\alpha_{\mathbf{m}_k}$ (except for the fixed $1 - 2p$ fraction of entries). (Covert and Lee, 2020) show that this technique can help reduce the variance of the estimator obtained from regression with binary inputs.

[4] $2p$ is arbitrary. Practitioners can run hyper-parameter searches to find more optimal values for their settings.

7.3.4 Iterated Pruning

Previous work on gradient-based pruning (Anwar et al., 2017; Frankle and Carbin, 2018) have shown that taking an iterated approach to pruning yields improved results over pruning directly to the target sparsity p. Similarly, we define an updated pruning fraction $p_{\text{iter}} < p$ and perform $\text{iter} = \lceil \frac{p}{p_{\text{iter}}} \rceil$ steps where we explore $n_{\text{iter}} = \lceil \frac{n}{\text{iter}} \rceil$ sub-models at a time. At the beginning of each iteration, we re-estimate the priors ρ for the unpruned modules and use them to select the n_{iter} sub-models to evaluate.

We combine the recipes developed in Sections 7.3.2, 7.3.3 and 7.3.4 together to produce Bonsai[5], our gradient-free structural pruning algorithm (Figure 7.2). Algorithm 5 specifies Bonsai in detail.

7.4 Experimental Details and Main Results

Table 7.2: Reported memory consumption of different methods. The minimum amount of memory required to run a Llama-7B model at half precision (FP16) is 14Gb. Running a forward pass with a batch size of 1 using the default model sequence length of 4096 uses around 20Gb of memory.

	Forward Only		Gradient-Based			
Llama-2-7B (Model Only)	Forward with bsz=1 (Bonsai Min.)	Bonsai (Faster)	LoRA Prune (Zhang et al., 2023)	Compresso (Guo et al., 2023)	LLM-Pruner (Ma et al., 2023)	ShearedLlaMA (Xia et al., 2023)
14GB	20GB	A6000 (48GB)	1 A100 (80GB)	4 V100 (128GB)	2 A100 (160GB)	8 A100 (640GB)

In all Bonsai experiments, we prune (1) the heads in the self-attention layers, and (2) the dimensions of the fully connected layers. In general, we will consider pruning LLMs of size $\sim 7B$ parameters. Since the primary goal of this paper is to provide a method for memory constrained practitioners, in Table 7.2, we give a brief tour of the amount of memory required to prune a Llama-2-7B model by different approaches. As can be seen, for models of our size range of interest, we can run Bonsai on any device with ≈ 20Gb of memory if we restrict our batch size to 1.

To obtain lower variance estimates of the score of any sampled structure, we average over a batch of 32 data-points. A constrained practitioner with only 20Gb of memory would do this by running 32 forward passes with batch-size of 1. For us, this would dramatically slow down our experimentation and so we elect (though it is not required by Bonsai) to run all experiments on 1 A6000 GPU which has 48Gb of memory – allowing us fit batches of size 4-6 to speed up experiments.

[5]Structural pruning is a canonical way of giving a bonsai tree its shape, hence the name.

7.4.1 Bonsai is competitive with other forward pass-only, structured pruning methods

We focus our first set of experiments on comparing structured pruning methods that can be run without gradient-based optimization[6]. We prune the LLaMA-2 7B model (Touvron et al., 2023) to 50% sparsity and evaluate on the Wikitext-2 (Merity et al., 2016) validation dataset. Our module importance signal for pruning, U, is the language modeling performance on the training set. When measuring speedups, we consider *end-to-end latency* of running inference on `model.sequence_length` chunks of the Wikitext-2 validation set. See Table E.3 (Appendix E.1.4) for details about the hyper-parameters used for this experiment.

Figure 7.1 shows the results of our experiments. Compared to a structured version of Wanda (Sun et al., 2023a), at any fixed desired speedup over the parent model, Bonsai produces a child model with the lowest perplexity.

7.4.2 Introducing Post-Pruning Adaptation (PPA)

After pruning the parent model, depending on the sparsity level p achieved, it is possible to obtain a pruned model on which we can run full fine-tuning or a parameter-efficient fine-tuning method like LoRA (Hu et al., 2021) with the available hardware memory. In this case, we can fine-tune the pruned model on the downstream task in order to recover more performance relative to that of the parent model.

Like many past works (Sanh et al., 2020; Xia et al., 2022), we can combine pruning with distillation by incorporating a distillation loss in the training objective during fine-tuning of the pruned model. Let $\mathcal{L}_{\text{task}}$ be the loss function over the task data and $\mathcal{L}_{\text{distill}}$ be the distillation objective. We optimize the following post-pruning objective: $\mathcal{L}^{\text{post}-\text{prune}} = \mathcal{L}_{\text{task}} + \lambda\mathcal{L}_{\text{distill}}$. Using i to index the task data, we have:
$\mathcal{L}_{\text{distill}} = \sum_i D_{\text{KL}}\big(\text{logits}^i\left(\mathbf{M}_{|\mathbf{m}^{\text{approx}}}\right) \parallel \text{logits}^i\left(\mathbf{M}\right)\big)$, where λ is a scalar weighting that can cross validated. Note, distillation can be performed without significant memory overhead by apriori caching the logits from the parent model $\mathbf{M}$ instead of hosting the model in memory during fine-tuning. In the sections ahead, we will apply PPA **after** pruning the parent model **if the child model is small enough to allow for this**.

Bonsai is competitive with semi-structured pruning methods

We compare Bonsai to the semi-structured variant of Wanda (Sun et al., 2023a). In general, structured pruning methods under-perform semi-structured pruning methods, but compensate for this in speedup.

Before fine-tuning, the Wanda 2:4 model is more accurate (10.52ppl vs 19.47ppl) but slower (1.14× vs 1.58×) than the model from Bonsai. Since the Bonsai child model is small enough, we can perform PPA on it, resulting in improved accuracy (8.89ppl) with unchanged speedup.

[6]Near submission, we learned of FLAP (An et al., 2024), a recent forward pass only method put out a little over a month before the release of Bonsai. Please see Appendix E.1.2 for a set of early-stage experimental comparisons.

Table 7.3: Wikitext-2 perplexity at 50% sparsity of LLaMA-2 7B with end-to-end latency speedups.

Model	∼Size	Fine-tune	PPL	Speedup
LlaMA-2	7B	✗	5.11	1×
Phi-2	3B	✓	8.69	1.24×
Wanda 2:4	3B	✗	10.52	1.14×
+ PPA		✓	8.34	0.75×
Bonsai	3B	✗	19.47	**1.58**×
+ PPA		✓	8.89	**1.58**×

Finetuning the semi-sparse Wanda 2:4 model is unfortunately less straightforward. It would require similar memory resources as needed to finetune the parent model[7] but we are in the setting where we do not have enough memory for this. We therefore have to use a parameter efficient fine-tuning method like LoRA (Hu et al., 2021) instead. While the performance gap can be bridged by LoRA finetuning (10.52 → 8.34), the adapted semi-structured model experiences a drastic slowdown (0.75×), since the learned low-rank matrices cannot be merged with the original sparsified ones without reverting back to dense computation. Thus, LoRA fine-tuned Wanda 2:4 is twice slower ($\sim 0.48\times$) than the model from Bonsai and similarly accurate.

In a memory-constrained setting, practitioners could opt for a pre-existing model of the target size instead of pruning a larger model. We compare the Bonsai-pruned model to Phi-2 (Li et al., 2023), a strong representative pre-existing model of similar size. As can be seen in Table 7.3, Bonsai is able to generate a model that is as accurate (0.2 difference in ppl) yet significantly faster (1.58× vs. 1.24× speedup), thus making it a competitive option to consider even if a model already exists at the target size.

Bonsai is competitive with gradient based structured pruning

We compare Bonsai to the following gradient-based structured pruning approaches: LLM-Pruner (Ma et al., 2023) and LoRA-Prune Zhang et al. (2023). We use the reported results from (Zhang et al., 2023) since none of these methods are actually runnable in our memory-constrained setting (Table 7.2). We choose to compare to these over Sheared LlaMA (Xia et al., 2023) since they have much lower memory requirements (Table 7.2). We prune the LLaMA-1 7B model (Touvron et al., 2023) to 50% sparsity since these approaches report their results for the LLaMA-1 model only. We compare these methods on Wikitext-2 and also on six tasks from the Eleuther LLM Evaluation Harness benchmark (Gao et al., 2023). The pruning signal used for the Wikitext-2 task is the same as the above experiments. For the Eleuther Harness tasks, we use language

[7]though the child tensors are sparse, the resulting gradients and cached intermediate tensors can be dense and have the same dimensions as those of the parent model (say $M \times M$). Since Bonsai does structured pruning, the actual tensor sizes are shrunk (say $N \times N \mid N < M$) which reduces memory during backward passes.

Table 7.4: LLaMA-1 (50% sparsity) after post-pruning adaptation. † are results as reported by (Zhang et al., 2023).

Method	Wikitext-2 ↓	BoolQ	HellaSwag	WinoGrande	ARC-e	ARC-c	Average ↑
LLaMA1-7B (Touvron et al., 2023)	5.68	75.05	56.92	69.93	75.34	41.89	63.83
LLM-Pruner† (Ma et al., 2023)	16.41	60.28	47.06	53.43	45.96	29.18	47.18
LoRAPrune† (Zhang et al., 2023)	11.60	61.88	**47.86**	55.01	45.13	**31.62**	48.30
Bonsai + PPA	**10.92**	**67.22**	43.09	**61.64**	**54.92**	26.28	**50.63**

Table 7.5: Phi-2 pruned to 35% sparsity and fine-tune the pruned model on small amount of the C4. We achieve strong performance compared to Phi-1.5 (trained from scratch). Since Sheared LlaMA has values absent, its MC Average would be misleading and we refrain from adding it.

Model	Size	Generation	Multiple Choice (MC)					
		GSM8k (5-shot)	ARC-c (25-shot)	Winogrande (5-shot)	Hellaswag (10-shot)	Truthful-QA (0-shot)	MMLU (5-shot)	MC Average ↑
Phi-2 (Li et al., 2023)	2.7B	54.81	61.09	74.35	75.11	44.47	58.11	62.63
Phi-1.5 (Li et al., 2023)	1.5B	12.43	52.9	72.22	63.79	40.89	43.89	54.74
Sheared LlaMA (Xia et al., 2023)	1.3B	Not Reported	33.5	57.9	60.7	Not Reported	25.7	*
Bonsai (w PPA)	1.8B	6.37	47.44	68.35	65.09	42.20	40.53	52.72
+ Reasoning Tuning	1.8B	27.67	45.56	68.82	64.51	42.58	40.97	52.49

modeling performance on the C4 (Raffel et al., 2020) dataset as pruning signal. We also perform parameter-efficient finetuning on our pruned model using 30K 512-length sequences from this corpus. Bonsai and LoRAPrune use similar amounts of the C4 dataset for Table 7.4 (30K vs 20K samples, respectively) whilst LLM-Pruner is trained on instruction tuned data with nearly twice the amount of unique samples (50K). Find more details in Appendix E.1.5.

As evinced by Table 7.4, Bonsai outperforms gradient-based methods even though it exclusively uses forward passes in the pruning stage. We attribute the superior performance of Bonsai to the fact that its pruning decisions are informed by directly exploring the space of sub-models whilst the other approaches resort to inaccurate proxies of module relevance in order to reduce the memory overhead of a fully gradient-based optimization approach (though not by enough to be runnable in our setting).

Bonsai can produce compressed models with strong zero-shot abilities

Considerable amounts of compute and data, beyond what is feasible for lay ML practitioners, are needed to train LLMs with strong zero-shot capabilities (OpenAI et al., 2023; Team et al., 2023). In this section, we demonstrate that Bonsai can empower everyday practitioners to produce strong and compact models with competitive zero-shot abilities by simply pruning bigger models on their available hardware.

We use Bonsai to prune a ≈3B Phi-2 model to ≈1.8B (35% sparsity). Bonsai hyper-parameters in this experiment are in Appendix E.1.6. Since it is small, the 1.8B pruned model can be fully

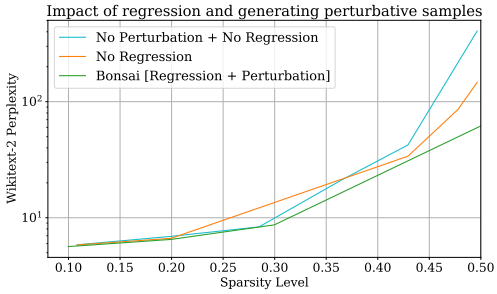


Figure 7.3: LLaMA-2 7B @ 50% sparsity (No PPA). Pertubation and regression components are both needed to make Bonsai effective. Experiment details in Appendix E.2.

	ns = 50	ns = 200	ns = 1000
PPL (↓)	NaN	61.63	22.09

Table 7.6: Wikitext-2 perplexity of LLaMA-2 7B @ 50% sparsity (No PPA). We vary the number of perturbative evaluations. Details in Appendix E.7.

fine-tuned on 1 A6000 GPU over 100k sequences of 2,048 tokens from the C4 dataset. As can be seen from Table 7.5, our pruned model achieves strong zero-shot performance on the Hugging Face OpenLLM leaderboard (Gao et al., 2023) compared to Phi-1.5, a smaller version in the Phi series of models that was trained from scratch.

Interestingly, one exception to the general trend of Bonsai's strong performance is the GSM-8K dataset, which is a mathematical reasoning dataset that requires generation of long reasoning chains. In our experiments, Bonsai prunes with respect to language modeling likelihood, as opposed to reasoning accuracy, and we posit that this mismatch may have contributed to our model's underperformance on GSM8K. We attempt to remedy the drop in reasoning ability by including the **GSM8K training data** during post-pruning finetuning (resulting in a total of 108K fine-tuning samples). This boosts our model's performance on the GSM8K with almost no degradation of performance on the other tasks.

7.5 Analysis

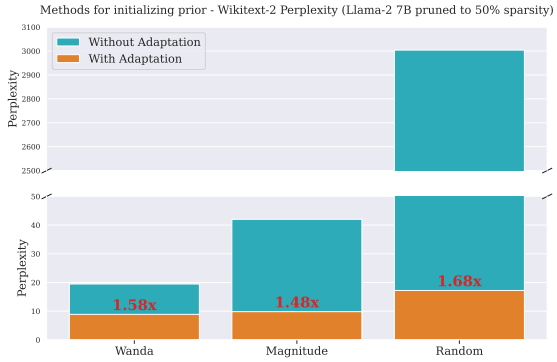
We conduct ablative experiments to understand the impact of the ingredients from Section 7.3.

Do we need both perturbative and regressive components of Bonsai? Figure 7.3 shows that both components are key to obtaining a good pruned model. Removing the estimation of module importances via regression leads to a degradation in performance (61.6 ppl $\rightarrow$ 146.6 ppl). Further degradation (146.6 ppl $\rightarrow$ 405.7 ppl) is encountered if we do not perform perturbative evaluations on the parent model but simply prune according to the prior ρ as computed from the unperturbed parent model.

Table 7.7: Impact of PPA on LLaMA-2 7B @ 50% sparsity. Details in Appendix E.5.

Method	Wikitext-2 PPL
No Post-Pruning Adaptation	19.47
Post-Pruning Finetuning	10.39
+ Distillation	8.89

Figure 7.4: LLaMA-2 7B pruned to 50% sparsity. See Appendix E.6 for experiment details and definitions of Wanda and Activation Magnitude priors.



What is a reasonable number of perturbative samples? We investigate the number of perturbative samples required to obtain good regression estimates of module importance based on Equation 7.3. Our results are shown in Table 7.6. As expected, performance improves as we increase the number of sub-models explored. We note that the number of samples being explored, ns, is significantly less than the number of candidate modules at each iteration ($N \approx 70k$). Nevertheless, Bonsai is able to deliver a performant pruned model because of the recipes developed in Section 7.3.

How much performance is recovered by post-pruning adaptation? During iterative pruning, Bonsai damages the parent model by removing modules but does not perform intermittent retraining to recover lost performance since even intermediate models may be too large for fine-tuning. Even so, as Table 7.7 shows, the final model produced by Bonsai has reasonable performance *without fine-tuning*. We attribute this to the redundancy of modules with respect to the target downstream tasks and Bonsai's ability to identify good candidates for pruning. If the pruned model is small enough in size, we may perform post pruning adaptation to recover more performance, as can be seen from Table 7.7.

What is the impact of the choice of metric for the prior ρ? We investigate three different choices of metrics for defining ρ. Figure 7.4 shows that using the module-level analogue of Wanda Sun et al. (2023a) yields the best performance, both before and after post-pruning adaptation. This indicates that Wanda is a strong signal for efficiently estimating the importance of model units.

Chapter 8

Conclusion

Historically, the primary objective of transfer learning has been to improve the performance (accuracy, generalization error) of a model on the desired end-task(s). This thesis has advocated for a refocusing of the lens towards a broader set of objectives with an eye towards efficiency. Specifically, we have looked into accounting for data, compute and memory efficiency when designing transfer learning techniques.

In the first part of this thesis, we made a case for looking beyond only the generalization error of the final model as the ultimate goal of transfer learning. We motivated this from three perspectives:

- the task itself. The setting of a task can itself place fundamental resource limitations in designing transfer learning algorithms. Latency critical tasks place a limit on model size whilst tasks that live on the edge (mobile devices, internet of things) have strict memory and compute consumption budgets.
- the individual ML practitioner. Democratizing machine learning entails making it possible for less resource endowed practitioners to build ML models for their specific needs. By designing resource-aware algorithms, we broaden the scope of people who can leverage transfer learning for their unique set of problems. All the work in this thesis is done with resources that are typical of the everyday ML practitioner, and we output models that are meant to be easily accessible.
- large corporations. Even in the case of resource rich large corporations, being resource conscious can not only lead to monetary savings but also free up resources for an expanded range of applications. There is also the issue of the environmental impact of exorbitant resource consumption – being resource aware enables corporations to build ML models that are more friendly towards the environment.

The beginning of the second part of this thesis highlighted the primary tool we would use to achieve resource efficiency in transfer learning: end task awareness. We showed that knowing something about the intended downstream applications of the model to be trained allows us to make informed design choices during tranfer learning that prioritize certain resources without sacrificing the model's end-task performance. In the rest of Part II, we presented three techniques: **TARTAN, ATTITUD and AANG** for improving data-efficiency in transfer learning.

Part III of this thesis moved beyond data-efficiency to include compute and memory effi-

ciency. Informed by knowledge of the end-task, this part focused on delivering compact yet performant versions of large pre-trained models by pruning these models to smaller sizes via modified structured pruning techniques. To make our methods widely usable, we focused on pruning these models in settings that are realistic with respect to the everyday practitioner. This included data-starved end-tasks (Chapter 6) and memory constraints during the pruning process itself (Chapter 7).

8.1 Future work

The works presented in this thesis were developed within the rapidly evolving landscape of deep learning. With the advent of scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), there has been even more focus on task agnostic approaches to transfer learning (specifically the pre-train then adapt setting) with a particular emphasis on scaling up models in terms of data and compute. Whilst our own resource limitations prevent us from conducting large scale studies to drive home the power of end-task awareness, it is heartening to repeatedly see larger scale work like Abnar et al. (2021); Isik et al. (2024) that show that (1) the one size fits all approach of pre-training without knowledge of the end-task leads to disparities in task-based outcomes (2) misalignment between pre-training and downstream tasks can result in worse scaling exponents leading to poor data-efficiency. We have also seen a ballooning in the sizes of model ML models (partially a result of task agnosticity of the pretrain-then-adapt paradigm) that has widened the gap between models that everyday ML practitioners can feasibly train and deploy in their settings of interest. These issues, within the context of this thesis present several avenues for future directions:

Exploring anchor tasks during pre-training: One way of marrying the resource efficiency of end-task aware approaches, with the current zeitgeist of delivering generalist models is to introduce a suite of *anchor* tasks during pre-training. These tasks would be constructed based on practitioner knowledge of dominant usage patterns on generalist models. Already, the existence of large scale benchmarks like the Eleuther Language Model Evaluation Harness (Gao et al., 2023) indicates that the ML community does have a non-trivial grasp of what a broad set of capabilities a general model should be descent at. Research in this direction could involve experimenting with various anchor tasks to determine their impact on model performance and data efficiency across different downstream tasks. As well as exploring strategies for incorporating these anchor tasks into large scale optimization pipelines without bottle-necking them.

Synthetic data *and* synthetic tasks: There has been a recent flurry of research into synethetic data (Lu et al., 2023; Gunasekar et al., 2023; Maini et al., 2024; Bauer et al., 2024) as a means of making training of subsequent LLM generations more data-efficient. Chapter 5 of this thesis showed that we can go a long way with only task data by introducing synthetic tasks. Marrying both synthetic data with synthetic task generation is appealing as an avenue for future work, not only as a means to even higher degrees of data-efficiency, but also towards building more LLMs with robust internal representations (Huh et al., 2024)

Modularity and continual learning: Thinking more deeply about modularity and continual learning when building deep learning models presents another exciting direction. Modularity (Pfeiffer et al., 2023; Douillard et al., 2024) allows for the development of models with interchangeable components that can be updated or replaced independently, facilitating easier adaptation to new tasks and environments and ultimately both data and memory efficiency. Continual learning (Hadsell et al., 2020; Mehta, 2023), on the other hand, enables models to incrementally learn from new data without forgetting previously acquired knowledge. This can be particularly useful in dynamic settings where data evolves over time. Research in this area could focus on designing modular architectures and developing algorithms that support continual learning, ensuring that models remain relevant and effective over prolonged periods whilst effectively reusing model components and thus amortizing resource consumption over many tasks.

Adaptive inference from the ground up: Finally, in order to make outsized strides in compute and memory efficiency with respect to deployed ML models, we have to rethink adaptive inference from the ground up. Adaptive inference techniques that dynamically adjust computational resources based on the complexity of the input can lead to more efficient and scalable models. They can also be seen as an approach to sharing *knowledge* across different model size families within the same training run. Future work could investigate various adaptive inference strategies, such as conditional computation (Raposo et al., 2024) and neural architecture search (Gao et al., 2020), to create models that are both computationally efficient and capable of maintaining high performance across a range of tasks.

Appendix A

TARTAN (Chapter 3) Appendix

A.1 Algorithm for META-TARTAN

A.2 Justifying the introduction of a Meta-Head

Proof. To arrive at Equation 3.7 we start with the closed form solution for $\nabla_{w_i}\mathcal{L}^{val}_{T^*}(\theta^*)$ and then introduce approximations in order to produce Equation 3.7. First, note that :

$$\frac{\partial \mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))}{\partial w_i} = \Big(\nabla_\theta \mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))\Big)^T \Big(\nabla_{w_i}\theta^*(\mathbf{w})\Big) \quad \texttt{[Chain rule]} \tag{A.1}$$

To get $\nabla_{w_i}\theta^*(\mathbf{w})$ we invoke the Cauchy Implicit Function Theorem (IFT) as with Lorraine et al. (2020); Navon et al. (2020); Liao et al. (2018):

$$\begin{aligned}
\nabla_{w_i}\theta^*(\mathbf{w}) &= \Big[\nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\Big]^{-1}\Big[\nabla_{w_i}\nabla_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\Big] \quad \texttt{[IFT]} \\
&= \Big[\nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\Big]^{-1}\Big[\nabla_{w_i}\nabla_\theta \Big(w^*\mathcal{L}_{T^*}(\theta^*(\mathbf{w})) + \sum_{T_i\in\mathbb{T}_{\text{aux}}} w_i \mathcal{L}_{T_i}(\theta^*(\mathbf{w}))\Big)\Big] \\
&= \Big[\nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\Big]^{-1}\Big[\nabla_\theta \mathcal{L}_{T_i}(\theta^*(\mathbf{w}))\Big] \quad \texttt{[Only terms with } w_i \texttt{ survive]}
\end{aligned}$$

Bringing it all together, we get :

$$\frac{\partial \mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))}{\partial w_i} = \Big(\nabla_\theta \mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))\Big)^T \Big(\Big[\nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\Big]^{-1}\Big[\nabla_\theta \mathcal{L}_{T_i}(\theta^*(\mathbf{w}))\Big]\Big) \tag{A.2}$$

□

Computing $\nabla_{w_i}\mathcal{L}^{val}_{T^*}(\theta^*)$ from Equation A.2 is computationally unwieldy since we would not only have to optimize θ to convergence for every step of w_i but we would also have to invert the Hessian of a typically large model. Our middle ground between Equations A.2 and 3.6 (Equation 3.7) makes use of the following approximations:

- We approximate the inverse Hessian with the identity. This approximation is not new; we follow previous work like Lorraine et al. (2020)(Table 3) who explore the use of this

Algorithm 3: End-task Aware Training via Meta-learning (META-TARTAN)

Require: $T^*, \mathbf{T}_{\text{aux}}$: End-task, Set of auxiliary pre-training tasks
Require: η, β_1, β_2: Step size hyper-parameters
Initialize :
Pre-trained RoBERTa as shared network body, θ_{body}
Task weightings: $w^*, w_i = \frac{1}{|\mathbf{T}_{\text{aux}}|+1}$
Randomly initialize :
end-task head as ϕ'
meta head for end-task as ϕ^*
task head, ϕ^i, for each $T_i \in \mathbf{T}_{\text{aux}}$
while *not done* **do**
$B^*_{\text{tr}} \sim T^*_{\text{train}}$ `// Sample a batch from end-task`
$g^*_\theta, g^*_\phi \leftarrow \left[\nabla_\theta, \nabla_{\phi'}\right]\left(\mathcal{L}_{T^*}(\theta, \phi', B^*_{\text{tr}})\right)$ `// Get end-task grads`
$g^i_\theta, g^i_\phi \leftarrow \left[\nabla_\theta, \nabla_{\phi^i}\right]\left(\mathcal{L}_{T_i}(\theta, \phi^i, B_i)\right)$ `// Get task grads.`
$\forall i \in [n], \;\; B_i \sim T_i$
`// Learn a new meta head`
$\phi^* \leftarrow$ `estimate_meta_head`$(B^*_{\text{tr}}, \beta_2, \theta, \phi^*)$ `//` $\;B^*_{\text{tr}} \sim T^*_{\text{train}}$
$g^*_{meta} \leftarrow \nabla_\theta \mathcal{L}_{T^*}(\theta, \phi^*, B^*_{\text{val}})$ `//` $\;B^*_{\text{val}} \sim T^*_{\text{val}}$
`// Update task weightings`
$w^* \leftarrow w^* + \eta \cos(g^*_{meta}, g^*_\theta)$
$w_i \leftarrow w_i + \eta \cos(g^*_{meta}, g^i_\theta)$
`// Update task parameters`
$\alpha^*, \alpha_1, \ldots, \alpha_{|\mathbf{T}_{\text{aux}}|} = \text{softmax}(w^*, w_1, \ldots, w_{|\mathbf{T}_{\text{aux}}|})$
Update $\theta_{\text{body}} \leftarrow \theta_{\text{body}} - \beta_1\left(\alpha^* g^*_\theta + \sum_i \alpha_i g^i_\theta\right)$
Update $\left(\phi_i \leftarrow \phi_i - \beta_2 g^i_\phi\right), \left(\phi' \leftarrow \phi' - \beta_2 g^*_\phi\right)$
end
Result : θ, ϕ'

approximation because of computational efficiency.

$$\left[\nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\right]^{-1} = \lim_{i \to \infty} \sum_{j=0}^{i} \left(\mathbf{I} - \nabla^2_\theta \mathcal{L}_{\text{total}}(\theta^*(\mathbf{w}))\right)^j \approx \mathbf{I}$$

We are assuming the contribution of terms with $i > 0$ are negligible.

- Instead of training the whole network to convergence, at each time-step, we fix the body of the network and train a special head ϕ^* to convergence on a small batch of end-task training data. We then use $[\theta_{\text{body}}; \phi^*]$ as a proxy for θ^*. This is a computationally feasible work-around to training all of θ to convergence to get a single step gradient estimate. Especially in the continued pre-training setting where a pre-trained *generalist* model like BERT is used as θ_{body}, this approximation is reasonable. To our knowledge, we are the

first to suggest this approximation.

$$\nabla_\theta \mathcal{L}^{val}_{T^*}(\theta^*) \rightarrow \nabla_\theta \mathcal{L}^{val}_{T^*}([\theta_{\text{body}}; \phi^*])$$

- Above, we have approximated $\theta^* = [\theta_{\text{body}}; \phi^*]$. Since ϕ^* is only used to evaluate end-task (T^*) validation data, it means θ remains unchanged with respect to the training data for task T_i. Thus $\nabla_\theta \mathcal{L}_{T_i}([\theta_{\text{body}}; \left(\phi^*, \ldots, \phi^i\right)]) = \nabla_\theta \mathcal{L}_{T_i}([\theta_{\text{body}}; \phi^i]) = \nabla_\theta \mathcal{L}_{T_i}(\theta)$

Bringing it all together, we get Equation 3.7, repeated here:

$$\frac{\partial \mathcal{L}^{val}_{T^*}(\theta^*(\mathbf{w}))}{\partial w_i} \approx \left(\nabla_\theta \mathcal{L}_{T_i}\right)^T \left(\nabla_\theta \mathcal{L}^{val}_{T^*}([\theta_{\text{body}}; \phi^*]_t)\right)$$

A.3 Calculating p-values from Permutation Test

We used the permutation test (Good, 2005; Dror et al., 2018) to test for statistical significance. For each test, we generate 10000 permutations to calculate significance level. This is sufficient to converge to a stable p-value without being a computational burden. We chose this over the common student t-test because :

1. We have only 10 runs per algorithm and permutation tests are more robust at low sample size
2. Permutation test is assumption free. Student t-tests assume that the samples are normally distributed
3. Permutation test is robust to variance in the samples, so even though error-bars can overlap, we still establish significant differences in the samples. Variance in our results is expected due to small dataset sizes of end-tasks.

A.4 Vision Experiments

We validate that the gains from end-task Aware Training are not siloed to only learning from text. We conduct an experiment comparing end-task aware training on images to its end-task agnostic variant. We use the Cifar100 dataset (Krizhevsky et al., 2009). We use the Medium-Sized

Method	Medium-Sized Mammals
Regular (Task-Agnostic) Pre-training	$46.7_{2.2}$
MT-TARTAN	$51.3_{1.2}$
META-TARTAN	$\mathbf{52.3_{3.8}}$

Table A.1: We report averages across 3 random seeds. Best average task accuracy is bolded.

Mammals superclass (one of the 20 coarse labels) as our main task whilst the other 19 super classes are used as auxiliary data. Our primary task is thus a 5-way classification task of images different types of medium-sized mammals whilst whilst the remaining 95 classes are grouped into a single auxiliary task.

As can be seen from Table A.1, being end-task aware improves over task agnostic pre-training. We find that, again, when our auxiliary task consist of solely domain data and no task data, META-TARTAN performs better than MT-TARTAN (as measured by averaged performance).

A.5 Full TAPT Table with Significance levels

We repeat Table 3.1 and provide details about levels of statistical signifance.

Task	**TAPT**	**MT-TARTAN**	$p-$values	**META-TARTAN**	$p-$values
ACL-ARC	$67.74_{3.68}$	$\mathbf{70.48}_{4.42}$	0.040	$70.08_{4.70}$	0.069
SCIERC	$79.53_{1.93}$	$\mathbf{80.81}_{0.74}$	0.038	$\mathbf{81.48}_{0.82}$	0.005
CHEMPROT	$82.17_{0.065}$	$\mathbf{84.29}_{0.63}$	0.000	$\mathbf{84.49}_{0.50}$	0.000

Table A.2: Significance levels as computed from the permutation test. All $p-$values are relative to the TAPT column. Statistically significant performance(p-value from permutation test < 0.05), is boldfaced

Task	**TAPT**	**META-TARTAN**	$p-$values
HYPER-PARTISAN	$93.39_{2.26}$	$\mathbf{96.84_{1.72}}$	0.003

Table A.3: Additional results for HYPERPARTISAN task. This is a binary, partisanship classification task with 515 labeled training examples.

A.6 Full DAPT/DAPT+TAPT Table

We repeat Table 3.3 and provide details about levels of statistical signifance.

Task	**DAPT**	**DAPT+TAPT**	**MT-TARTAN**	p-values	**META-TARTAN**	p-values
ACL-ARC	$68.60_{2.62}$	$69.12_{5.76}$	$71.58_{1.65}$	0.110	$71.05_{2.37}$	0.174
SCIERC	$76.44_{1.19}$	$77.62_{1.38}$	$\mathbf{81.02_{1.24}}$	0.000	$\mathbf{81.41_{1.70}}$	0.000
CHEMPROT	$80.76_{0.54}$	$78.22_{0.74}$	$\mathbf{83.77_{0.60}}$	0.000	$\mathbf{83.38_{0.89}}$	0.000

Table A.4: Duplicate of Table 3.2. Significance levels as computed from the permutation test. All $p-$values are relative to $\max\big(\text{DAPT}, \text{DAPT} + \text{TAPT}\big)$. Statistically significant performance(p-value from permutation test < 0.05), is boldfaced

A.7 FAQ

1. *What settings are TARTAN algorithms designed for?*
   TARTAN algorithms specialize auxiliary objectives to a particular end-task. This comes at a risk of losing the generic representations afforded by *generalist* pre-trained models. Thus if a practitioner has a sufficiently important end-task where obtaining improved end-task performance is paramount over generic representations, then TARTAN is a viable option.
2. *When do we get computational savings from META-TARTAN?*
   MT-TARTAN does not add any extra overhead compared to pre-train then fine-tune approaches. META-TARTAN however, adds extra overhead per gradient descent step due to computing meta-gradients. However, as shown in Section 3.6 we are able to get several orders of magnitude improvement in data-efficiency from applying the method. In general, for the tasks we experimented with, we find that the savings in data-efficiency superseded the extra per-timestep meta-learning overhead.
3. *When should we use META-TARTAN over MT-TARTAN?*

   In +TAPT settings (Tables 3.1, 3.3), we observe that META-TARTAN and MT-TARTAN perform similarly. We attribute this to the strength of TAPT-MLM objective. We were pleasantly surprised that the two methods performed comparatively in this setting but in hindsight, we appreciate the insight that went into designing TAPT-MLM as an objective which makes it a strong baseline. In other settings with less carefully designed auxiliary objectives and data (which can potentially be detrimental to the end-task) we expect META-TARTAN to perform better. Section 5.3 provides evidence of this.

Appendix B

ATTITUD (Chapter 4) Appendix

B.1 Proof of Theorem 1

Theorem 3. *Let $\mathcal{L}^{\text{aux}}(\theta_t)$ and $\mathcal{L}^{\text{prim}}(\theta_t)$ represent the full batch losses of the auxiliary tasks and primary task respectively at step t. We assume the gradients of $\mathcal{L}^{\text{aux}}$ and $\mathcal{L}^{\text{prim}}$ are Lipschitz continuous with constant $L > 0$. Following the update rule : $\theta_{t+1} = \theta_t - \alpha \cdot \tilde{\mathbf{g}}_{aux}$, where $\alpha \leq \frac{1}{L}$ is the learning rate, we are guaranteed :*

$$\mathcal{L}^{\text{aux}}(\theta_{t+1}) \leq \mathcal{L}^{\text{aux}}(\theta_t)$$
$$\mathcal{L}^{\text{prim}}(\theta_{t+1}) \leq \mathcal{L}^{\text{prim}}(\theta_t)$$

If $\eta_- = 0$ and $\eta_\perp, \eta_+ \geq 0$

Proof. Let $\mathbf{V}_t \in \mathbf{R}^{K \times D}$ be the orthonormal matrix whose rows span the per-example primary task gradients $\mathbf{J}^*$ at timestep t. The projections of the average primary task gradient $\mathbf{g}_{prim} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{J}^*_{i,:}$ and average auxiliary task gradient $\mathbf{g}_{aux}$ at iteration t are :

$$\mathbf{p}_{prim} = \mathbf{V}_t\big(\mathbf{g}_{prim}\big)^T$$
$$\mathbf{p}_{aux} = \mathbf{V}_t\big(\mathbf{g}_{aux}\big)^T$$

$\mathbf{p}_{prim}$ and $\mathbf{p}_{aux}$ will agree on some directions (same sign on those components). We use the operator $[x]_+$ to mark these directions of agreement. This operator preserves components that agree and sets those that disagree to zero. As an example given $\mathbf{p}_{prim} = [1, 1, -1]$ and $\mathbf{p}_{aux} = [1, 3, 10]$, $[\mathbf{p}_{prim}]_+ = [1, 1, 0]$ and $[\mathbf{p}_{aux}]_+ = [1, 3, 0]$. For directions that disagree (different signs of the respective components), we introduce the operator $[x]_-$. In the above example $[\mathbf{p}_{prim}]_- = [0, 0, -1]$ and $[\mathbf{p}_{aux}]_- = [0, 0, 10]$. Note that our operators are defined by comparing two vectors x_1 and x_2, Our operators have the following properties by definition :

$$x = [x]_- + [x]_+$$

and

$$[x]_+ \perp [x]_-, [x_1]_\pm \perp [x_2]_\mp$$

From Equation 4.2 :

$$\tilde{\mathbf{g}}_{aux} = \eta_+ \mathbf{g}^+_{aux} + \eta_- \mathbf{g}^-_{aux} + \eta_\perp \mathbf{g}^\perp_{aux}$$

We can re-write this in terms of $[x]_\pm$ as :

$$\tilde{\mathbf{g}}_{aux} = \eta_+[\mathbf{p}_{aux}]_+ + \eta_-[\mathbf{p}_{aux}]_- + \eta_\perp\big(\mathbf{g}_{aux} - \mathbf{p}_{aux}\big)$$

We now proceed to show the effect of the gradient descent update below on $\mathcal{L}^{\text{aux}}(\theta_{t+1})$ and $\mathcal{L}^{\text{prim}}(\theta_{t+1})$.

$$\theta_{t+1} = \theta_t - \alpha \cdot \tilde{\mathbf{g}}_{aux} \tag{B.1}$$

How does this update affect the loss on the primary task loss $\mathcal{L}^{\text{prim}}(\theta_{t+1})$?
$\mathcal{L}^{\text{prim}}(\theta_{t+1}) = \mathcal{L}^{\text{aux}}(\theta_t - \alpha \cdot \tilde{\mathbf{g}}_{aux})$

$$\begin{aligned}
&\approx \mathcal{L}^{\text{prim}}(\theta_t) - \alpha\big(\tilde{\mathbf{g}}_{aux}\big)^T\mathbf{g}_{prim} \;\; \textit{(First order Taylor Expansion)} \\
&= \mathcal{L}^{\text{prim}}(\theta_t) - \alpha\Bigg(\eta_+[\mathbf{p}_{aux}]_+ + \eta_-[\mathbf{p}_{aux}]_- + \eta_\perp\mathbf{g}^\perp_{aux}\Bigg)^T\mathbf{g}_{prim} \\
&= \mathcal{L}^{\text{prim}}(\theta_t) - \alpha\Bigg(\eta_+[\mathbf{p}_{aux}]_+ + \eta_-[\mathbf{p}_{aux}]_- + \eta_\perp\mathbf{g}^\perp_{aux}\Bigg)^T\Big([\mathbf{p}_{prim}]_+ + [\mathbf{p}_{prim}]_-\Big) \\
&= \mathcal{L}^{\text{prim}}(\theta_t) - \alpha\Bigg(\eta_+\Big([\mathbf{p}_{aux}]_+^T[\mathbf{p}_{prim}]_+ + [\mathbf{p}_{aux}]_+^T[\mathbf{p}_{prim}]_-\Big) + \eta_-\Big([\mathbf{p}_{aux}]_-^T[\mathbf{p}_{prim}]_+ + [\mathbf{p}_{aux}]_-^T[\mathbf{p}_{prim}]_-\Big)\Bigg) \\
&= \mathcal{L}^{\text{prim}}(\theta_t) - \alpha\Bigg(\eta_+[\mathbf{p}_{aux}]_+^T[\mathbf{p}_{prim}]_+ + \eta_-[\mathbf{p}_{aux}]_-^T[\mathbf{p}_{prim}]_-\Bigg) \\
&\leq \mathcal{L}^{\text{prim}}(\theta_t) \;\; \textit{(if } \eta_- \leq 0,\ \eta_\perp, \eta_+ \geq 0\textit{)}
\end{aligned}$$

Note that in going from line 3 to 4 in the proof above, we use the fact that $\big(\mathbf{g}^\perp_{aux}\big)^T\mathbf{g}_{prim} = 0$ since $\mathbf{g}^\perp_{aux}$ lies outside the subspace and $\mathbf{g}_{prim}$ lies inside it. For the last step of the proof, we use the observations below :

$$\begin{aligned}
&[\mathbf{p}_{aux}]_+[\mathbf{p}_{prim}]_+ \geq 0 \;\; \text{since these directions agree in sign} \\
&[\mathbf{p}_{aux}]_-[\mathbf{p}_{prim}]_- \leq 0 \;\; \text{since these directions disagree in sign} \\
&[\mathbf{p}_{aux}]_+[\mathbf{p}_{prim}]_- = 0 \;\; \text{by the property of the } [x]_\pm \text{ operator} \\
&[\mathbf{p}_{aux}]_-[\mathbf{p}_{prim}]_+ = 0 \;\; \text{same motivation as above}
\end{aligned}$$

How does Equation B.1 affect the auxiliary task loss $\mathcal{L}^{\text{aux}}(\theta_{t+1})$?

$$\begin{aligned}
\mathcal{L}^{\text{aux}}(\theta_{t+1}) &= \mathcal{L}^{\text{aux}}(\theta_t - \alpha \cdot \tilde{\mathbf{g}}_{aux}) \\
&\approx \mathcal{L}^{\text{aux}}(\theta_t) - \alpha\big(\tilde{\mathbf{g}}_{aux}\big)^T\mathbf{g}_{aux} \;\; \textit{(First order Taylor Expansion)} \\
&= \mathcal{L}^{\text{aux}}(\theta_t) - \alpha\big(\eta_\perp\mathbf{g}^\perp_{aux} + \eta_+\mathbf{g}^+_{aux} + \eta_-\mathbf{g}^-_{aux}\big)^T\big(\mathbf{g}^\perp_{aux} + \mathbf{g}^+_{aux} + \mathbf{g}^-_{aux}\big) \\
&= \mathcal{L}^{\text{aux}}(\theta_t) - \alpha\big(\eta_\perp\|\mathbf{g}^\perp_{aux}\|^2 + \eta_+\|\mathbf{g}^+_{aux}\|^2 + \eta_-\|\mathbf{g}^-_{aux}\|^2\big) \;\; \textit{(Cross terms cancel due to orthogonality)} \\
&\leq \mathcal{L}^{\text{aux}}(\theta_t) \;\; \textit{(If } \eta_-, \eta_\perp, \eta_+ \geq 0\textit{)}
\end{aligned}$$

Thus, choosing $\eta_- = 0$ ensures that we are minimizing both $\mathcal{L}^{\text{aux}}(\theta_t)$ and $\mathcal{L}^{\text{prim}}(\theta_t)$. We can combine this with the constraint on $\alpha \leq \frac{1}{L}$ to derive convergence guarantees after some T steps as in optimization literature. □

B.2 Randomized Matrix Theory

Algorithm 4: `randomized_lowrank_approx` : Construct low rank approximation

Require : $\mathbf{J} \in \mathbf{R}^{m \times D}$: Input Matrix
Require : k : Rank of subspace
$\Pi \sim \mathcal{N}(0, I) \in \mathbf{R}^{k \times m}$
$\mathbf{C} = \Pi \mathbf{J}$
$\mathbf{V} \leftarrow \mathrm{Gram\_Schmidt}(\mathbf{C})$
Return : $\mathbf{V} \in \mathbf{R}^{k \times D}$: Low rank approximation of $\mathbf{J}$

The $\mathrm{Gram\_Schmidt}$ procedure orthogonalizes the rows of an input matrix.

B.3 More Experimental Details

Image Classification For MultiCifar100, unlike Rosenbaum et al. (2017); Yu et al. (2020) who use a 500-100 train-test split for examples under each fine-grained CIFAR 100 label, we include a validation set and therefore opt for a 400-100-100 train-validation-test split. We test on all 1000 test examples per class.

For Cat-vs-Dog, we use 100 examples from the training set as validation and test on all 1000 test examples per-class.

For Image Classification experiments, we perform pre-training with a learning rate of 1e-4 for all experiments and finetuning learning rate of 5e-4. These values were selected after coarse hyper-parameter search. In both pre-training and finetuning settings, we decay the learning rate by 0.5 if the validation loss has not improved over 4 epochs, up till a minimum learning rate of 1e-5. we use the Adam Optimizer (Kingma and Ba, 2014) with $\beta = (0.9, 0.999)$. We clip all gradient norms to 1.0 before performing gradient descent. We cross-validated dropout rates within the set {0.05, 0.1, 0.2, 0.3} for both pre-training and finetuning steps. We cross validate η_{prim} based on the relative sizes of primary and auxilary task datasets. All experiments are averaged over 5 random seeds. For all our Vision experiments, we either recompute our subspace basis every $n = 5$ or $n = 10$ iterations. We find that n is not as important as the other hyper-parameters, with the two choices showing similar performance when the other hyper-parameters (learning rate and gradient norm clipping) are fixed to reasonable values.

Due to the fact that Yue et al (PCGrad) treat all tasks symmetrically, which is different from our primary-auxiliary setting, we introduced an extra parameter, α_{prim}, for PCGrad to account for weighting the primary task. We cross validated values of $\alpha_{prim} \in \{0.1, 0.05, 0.01, 0.001\}$.

Medical Imaging Transfer Table 4.4 presents a more detailed breakdown of the ChexPert task. For 50k examples from Imagenet, our best performing configuration was $\eta_{aux} = (1.0, 0.0, -1.0)$. We did not use the primary task gradient directly for pre-training so $\eta_{prim} = 0.0$ for all cases. For ATTITUD, we use the same learning rates as in the Image classification setup above. For the No-Pretraining and Vanilla pretraining we cross-validated the learning rates for both finetuning

and pre-training from the set {1e-3, 1e-4}. We cross-validated the same list of dropout values above.

Method	No-Pretraining	Pretrain w Imgnet	Pretrained + Ours (50k)	Pretrained + Ours (100k)
Atelectasis	76.0 ± 1.82	79.0 ± 3.66	$\mathbf{81.6 \pm 1.38}$	$\mathbf{81.8 \pm 0.80}$
Cardiomegaly	74.9 ± 2.34	75.8 ± 4.04	78.0 ± 2.13	$\mathbf{80.7 \pm 1.79}$
Consolidation	83.2 ± 2.26	$\mathbf{85.3 \pm 1.86}$	$\mathbf{85.6 \pm 2.32}$	84.9 ± 1.36
Edema	79.5 ± 1.27	82.6 ± 0.76	$\mathbf{85.2 \pm 1.23}$	84.7 ± 1.78
P. Effusion	77.9 ± 1.88	$\mathbf{84.4 \pm 0.75}$	83.4 ± 1.80	$\mathbf{84.3 \pm 0.65}$

Table B.1: Results on ChexPert-5k tasks measured by average AUC (Area Under Roc-Curve)

Text Classification For our NLP experiments, we tried limiting the number of layers we applied ATTITUD to. We achieved good performance without applying ATTITUD to the word embedding layers (these were updated with untouched auxiliary task gradients). We cross-validated $\eta_{prim} = \{0.01, 0.05, 0.0025\}$. For all our NLP experiments, we either recompute our subspace basis every $n = 1$ or $n = 4$ times

For all experiments involving ATTITUD, We cross-validate the following choices of the subspace size $k \in \{5, 10, 20\}$ from $\mathbf{J}^* \in \mathbf{R}^{m \times D}$ using $m \in \{32, 64\}$. We recompute the subspace every 10 steps for vision experiments and every 4 steps for NLP experiments. We run all experiments for a maximum of 150 pretraining epochs and 500 finetuning epochs. We performed early stopping for all experiments if no improvement after 10 consecutive epochs.

Ablation of Fraction of Norm within Subspace The left pane of Figure B.1 reinforces our intuition and confirms that our choice of the top-k singular vectors (*randomized_svd*) gives the best accuracy as averaged across 5 seeds. *random* is the basis spanned by k randomly chosen orthogonal vectors in $\mathbf{R}^D$, *unit_avg_grad* is the basis spanned by the average primary task gradient whilst *canonical* uses the per-parameter basis. Note that $k = 5$ for *random* and *randomized_svd* whilst for *unit_avg_grad* and *canonical*, $k = 1$ and $k = D$ respectively. We use the fraction of the norm of sample gradients within a subspace as indicators of how *semantically* meaningful that choice of subspace is. We expect that a *semantically* meaningful choice of basis will achieve better generalization performance because it captures the essential parts of the gradient with $k \ll D$. *canonical* trivially captures all the norm of the sampled gradient vectors but because $k = D$, it generalizes poorly. Notice that only small fractions of the norms of sample primary and auxiliary task average gradients lie in the subspace for *random* and *unit_avg_grad*, whilst significant fractions lie in *randomized_svd*.

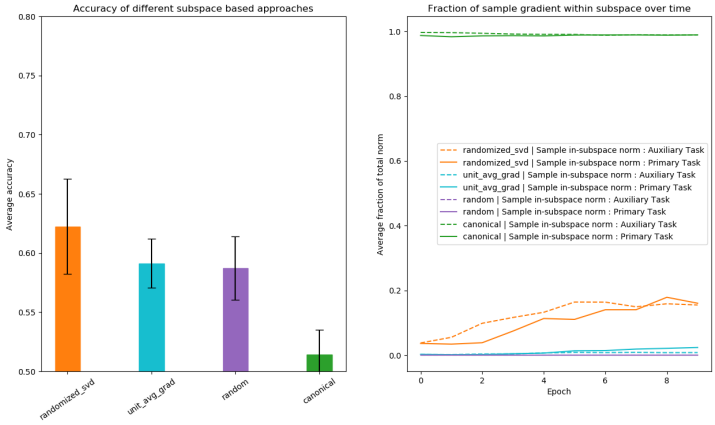


Figure B.1: Experiment conducted on Cat-vr-Dog Cifar10 dataset. **Left** Averaged accuracy across 5 seeds of different choices of basis. Our choice, randomized_svd performs best. **Right** We look at the fraction of the norm of $\mathbf{g}_{aux}$ within each subspace (dashed line). We also do so for a randomly sampled mini-batch of the primary task (solid line).

Appendix C

AANG (Chapter 5) Appendix

C.1 More Ablation Tables

Table C.1: Varying number of sampled objectives per-iteration.

Task	$\frac{3}{24}$ tasks	$\frac{6}{24}$ tasks
ACL-ARC	$72.11_{2.12}$	$\mathbf{73.26}_{1.32}$
SCIERC	$82.35_{1.76}$	$\mathbf{82.98}_{1.52}$
SE-2016-6	$\mathbf{72.46}_{1.65}$	$\mathbf{72.46}_{0.90}$
CHEMPROT	$\mathbf{83.91}_{0.32}$	$83.69_{0.98}$
H.PARTISAN	$\mathbf{98.46}_{0.0}$	$97.95_{0.73}$

C.2 Dataset Details

Table C.2: Specifications of datasets used to evaluate our methods.

Domain	Task	Label Type	Train Size	Dev Size	Test Size	Classes	Metric
BIOMED	CHEMPROT Kringelum et al. (2016)	relation classification	4169	2427	3469	13	Accuracy
CS	SCIERC Luan et al. (2018)	relation classification	3219	455	974	7	F1
STANCE	SE-2016-6 Mohammad et al. (2016)	stance detection	2497	417	1249	3	Accuracy
CS	ACL-ARC Jurgens et al. (2018)	citation intent	1688	114	139	6	F1
NEWS	H.PARTISAN Kiesel et al. (2019)	partisanship	515	65	65	2	Accuracy

C.3 More Training Details

We run each hyper-parameter configuration across 3 seeds $\{0, 1, 2\}$. We use a batch size of 128 for all end-tasks tasks except H.PARTISAN where we use a batch size of 64. The auxiliary task batch-size, `aux_bsz`, is shared across all the n sub-sampled auxiliary objectives according to the objective's weight.

We use the AdamW optimizer (Loshchilov and Hutter, 2017b), with weight decay of 0.01 for all experiments.

Table C.3: AANG-TD specific Hyper-parameters

Hyper-parameter	Values	Description
aux_lr	1.0, 0.1	Learning rate for factor vectors - $\{W^{\text{All}}, W^{\mathcal{I}}, W^{\mathcal{T}}, W^{\mathcal{R}}, W^{\mathcal{O}}\}$
sopt_lr	0.1, 0.01	Learning rate for primary task weighting λ_e
nconf_subsamp	3, 6	Number of sub-sampled auxiliary tasks.
learning rate	1e-3, 1e-4	Learning rate used for further training of $\text{RoBERTa}_{\text{base}}$
aux_bsz	256	Batch size of for auxiliary objectives

Table C.4: AANG-TD+ED specific Hyper-parameters

Hyper-parameter	Values	Description
aux_lr	1.0, 0.5, 0.1	Learning rate for factor vectors - $\{W^{\text{All}}, W^{\mathcal{I}}, W^{\mathcal{T}}, W^{\mathcal{R}}, W^{\mathcal{O}}\}$
sopt_lr	0.1	Learning rate for primary task weighting λ_e
nconf_subsamp	6, 12, 24	Number of sub-sampled auxiliary tasks.
learning rate	1e-4	Learning rate used for further training of $\text{RoBERTa}_{\text{base}}$
aux_bsz	1024	Batch size of for auxiliary objectives

Table C.5: META-TARTAN Hyper-parameters for single task auxiliary tasks

Hyper-parameter	Values	Description
sopt_lr	1.0, 0.1, 0.01	Learning rate for primary task weighting λ_e
learning rate	1e-3, 1e-4, 5e-5	Learning rate used for further training of $\text{RoBERTa}_{\text{base}}$

META-TARTAN introduces a dev-head which is trained sporadically during training for estimating the meta-gradients. We use the following hyper-parameters for training this dev-head : we sample 32 examples (8 examples in the case of H.PARTISAN) and perform full batch gradient descent with a learning rate of 1e-2 for 10 iterations. The dev-head is trained with the AdamW optimizer with weight decay set to 0.1.

We copy the end-task agnostic baseline results from (Dery et al., 2021a) when available. We use the hyper-parameters specified for TAPT in Gururangan et al. (2020a) to train for the SE-2016-6 task.

All models were trained on one of two types of gpus: NVIDIA A100 or NVIDIA A6000. All models fit within a single gpu. We used gradient accumulation to expand the effective batch sizes used for our experiments.

C.4 Generalization Error Bound for End-task Aware Training

C.4.1 Definitions

Definition C.4.1. *A function, $f : \Omega \to \mathbb{R}$ is L-Lipschitz if $\forall u, v \in \mathrm{dom}(f)$:*

$$\|f(u) - f(v)\| \leq L\|u - v\|$$

Note that L-Lipschitz implies bounded gradients.

$$\|\nabla f(w)\| \leq L \quad \forall w$$

Definition C.4.2. *A function, $f : \Omega \to \mathbb{R}$ is β-smooth if $\forall u, v \in \Omega$:*

$$\|\nabla f(u) - \nabla f(v)\| \leq \beta\|u - v\|$$

Definition C.4.3. *An update rule, G is σ-bounded if :*

$$\sup_{w\in\Omega} \ \|w - G(w)\| \leq \sigma$$

Consider the following general setting. There is an unknown distribution $\mathcal{D}_e$ over examples from some space $\mathcal{Z}$. We receive a sample $S = (z_1, \ldots, z_{N_e})$ of N_e examples drawn i.i.d. from $\mathcal{D}_e$. Our goal is to find a model w, that parameterizes the function f_e, with small population risk defined as:

Definition C.4.4. ***Population Risk***

$$R[w] = \mathbf{E}_{z\sim\mathcal{D}_e} f_e(w; z)$$

Definition C.4.5. ***Empirical Risk***

Since we have a finite number of samples, we can only compute the empirical risk which is :

$$R_S[w] = \frac{1}{N_e}\sum_i f_e(w; z_i),$$

Let A be a potentially randomized algorithm (such as Stochastic Gradient Descent) that is a function of the S such that $w = A(S)$.

Definition C.4.6. ***Generalization Error*** $\epsilon_{gen}(A, N_e)$

$$\epsilon_{gen}(A, N_e) = \mathbf{E}_{S,A}\big[R_S[A(S)] - R[A(S)]\big]$$

Definition C.4.7. ***Uniform Stability***

A randomized algorithm A is ϵ-uniformly stable if for all data sets $S, S' \in \mathcal{Z},\ |S| = |S'| = N_e$ such that S and S' differ in at most one example, we have

$$\sup_z \ \mathbf{E}_A\big[f_e(A(S); z) - f_e(A(S'); z)\big] \leq \epsilon$$

Here, the expectation is taken only over the internal randomness of A. We will denote by $\epsilon_{\text{stab}}(A, N_e)$ *the infimum over all* ϵ *for which the above holds.*

C.4.2 Relevant Theorems

Theorem 4 (Uniform Stability implies Generalization in expectation)**.** *Let Algorithm A be* ϵ*-uniformly stable. Then,*

$$\epsilon_{gen}(A, N_e) = \left| \mathbf{E}_{S,A}\big[R_S[A(S)] - R[A(S)]\big] \right| \leq \epsilon_{stab}(A, N_e)$$

For full proof see Theorem 2.2 of Hardt et al. (2016).

Theorem 5 (Stochastic Gradient Method is stable)**.** *Assume that* $f_e(;z) \in [0,1]$ *is an L-Lipschitz and* β_e*-smooth loss function for every* z*. Suppose that we run SGM for* T *steps with monotonically non-increasing step sizes* $\alpha_t \leq \frac{c}{t}$*. Then, SGM has uniform stability with :*

$$\epsilon_{sgm} \leq \frac{1+\frac{1}{q}}{N_e - 1}\big(2cL^2\big)^{\frac{1}{q+1}} T^{\frac{q}{q+1}}$$
$$\text{where } \; q = \beta_e c$$

We can simplify this to only terms involving T *and* N_e

$$\epsilon_{sgm} \lessapprox \frac{T^{1-\frac{1}{c\beta_e+1}}}{N_e} \tag{C.1}$$

Proof. For the full proof, see *Theorem 3.12* of Hardt et al. (2016)

□

C.4.3 Growth Functions

Lemma 6 (Growth Recursion Under Dynamic Sampling)**.** *We consider the Stochastic Gradient update rule* $G : \Omega \to \Omega$ *:*

$$G_f(w) = w - \alpha \nabla f(w)$$

Fix an arbitrary sequence of updates $G_{f_1}, \ldots, G_{f_T}$ *and another* $G'_{f_1}, \ldots, G'_{f_T}$*. Let* $w_0 = w'_0$ *be a starting point in* Ω *given that* $f : \Omega \to \mathbb{R}$ *and define*

$$\delta_t = \mathbb{E}_{f_1 \ldots f_t \sim \mathcal{P}_\lambda}\big[\|w_t - w'_t\|\big]$$

where w_t, w'_t *are defined recursively through :*

$$w_t = G_{f_t}(w_{t-1}) \quad w'_t = G'_{f_t}(w'_{t-1}) \quad t \geq 0$$

Then we have the recurrence relation :

$$\delta_0 = 0$$

$$\delta_{t+1} \leq \begin{cases} \min\big\{\big(1+\alpha\lambda_1\beta_1\big)\delta_t + \alpha\lambda_2\big(\Delta + 2L\big),\ \big(1+\alpha\big(\lambda_1\beta_1 + \lambda_2\beta_2\big)\big)\delta_t\big\} & G_{f_t} = G'_{f_t} \\ \delta_t + 2\sigma_t & G_{f_t}, G'_{f_t} \text{ are } \sigma\text{-bounded} \end{cases}$$

Note that $\mathcal{P}_f$ *is a distribution over the support* $\{f^1, f^2\}$ *according to probabilities* $\{\lambda_1, \lambda_2 \mid \lambda_1 + \lambda_2 = 1\}$*.* $\{f_1, f_2\}$ *have smoothness* β_1, β_2 *respectively.*

Proof. The second bound on δ_t is taken directly from Lemma 2.5 of Hardt et al. (2016). We now derive the first-half of the first bound

$$\begin{aligned}
\delta_{t+1} &= \mathbb{E}_{f_1 \dots f_{t+1} \sim \mathcal{P}_\lambda}\big[\|w_{t+1} - w'_{t+1}\|\big] \\
&= \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big[\lambda_1 \|G_{f^1}(w_t) - G'_{f^1}(w'_t)\| + \lambda_2 \|G_{f^2}(w_t) - G'_{f^2}(w'_t)\|\Big] \\
&= \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big[\lambda_1 \|w_t - \alpha \nabla f^1(w_t) - w'_t + \alpha \nabla f^1(w'_t)\| + \lambda_2 \|w_t - \alpha \nabla f^2(w_t) - w'_t + \alpha \nabla f^2(w'_t)\|\Big] \\
&\leq \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\big[\|w_t - w'_t\|\big] + \alpha \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big(\lambda_1 \|\nabla f^1(w'_t) - \nabla f^1(w_t)\| + \lambda_2 \|\nabla f^2(w'_t) - \nabla f^2(w_t)\|\Big) \\
&\quad \text{(Triangle Inequality used for above step)} \\
&= \delta_t + \alpha \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big(\lambda_1 \|\nabla f^1(w'_t) - \nabla f^1(w_t)\| + \lambda_2 \|\nabla f^2(w'_t) - \nabla f^2(w_t)\|\Big) \\
&\quad \text{(Without Loss of Generality, let } \beta_1 \leq \beta_2\text{)} \\
&\leq \delta_t + \alpha \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big[\lambda_1 \beta_1 \|w_t - w'_t\| + \lambda_2 \|\nabla f^2(w'_t) - \nabla f^2(w_t)\|\Big] \quad \text{(Smoothness)} \\
&= \delta_t + \alpha \lambda_1 \beta_1 \delta_t + \alpha \lambda_2 \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\Big[\|\nabla f^2(w'_t) - \nabla f^2(w_t)\|\Big] \quad \text{(Triangle Inequality)} \\
&= \big(1 + \alpha \lambda_1 \beta_1\big)\delta_t + \alpha \lambda_2 \Big\|\nabla f^2(w'_t) - \nabla f^1(w'_t) + \nabla f^1(w'_t) - \nabla f^2(w_t)\Big\| \quad \text{(add zero)} \\
&\leq \big(1 + \alpha \lambda_1 \beta_1\big)\delta_t + \alpha \lambda_2 \Big(\|\nabla f^2(w'_t) - \nabla f^1(w'_t)\| + \|\nabla f^1(w'_t) - \nabla f^2(w_t)\|\Big) \quad \text{(Triangle Inequality)} \\
&\leq \big(1 + \alpha \lambda_1 \beta_1\big)\delta_t + \alpha \lambda_2 \Big(\Delta + \|\nabla f_1(w'_t) - \nabla f_2(w_t)\|\Big) \quad \text{Using Assumption A.1} \\
&\leq \big(1 + \alpha \lambda_1 \beta_1\big)\delta_t + \alpha \lambda_2 \Big(\Delta + \|\nabla f_1(w'_t)\| + \|\nabla f_2(w_t)\|\Big) \quad \text{Triangle Inequality} \\
&\leq \big(1 + \alpha \lambda_1 \beta_1\big)\delta_t + \alpha \lambda_2 \big(\Delta + 2L\big) \quad L\text{-Lipschitz function}
\end{aligned}$$

To obtain the second half of the first bound :

$$\begin{aligned}
\delta_{t+1} &= \mathbb{E}_{f_1 \dots f_{t+1} \sim \mathcal{P}_\lambda}\left[\|w_{t+1} - w'_{t+1}\|\right] \\
&= \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\lambda_1 \|G_{f^1}(w_t) - G'_{f^1}(w'_t)\| + \lambda_2 \|G_{f^2}(w_t) - G'_{f^2}(w'_t)\|\right] \\
&= \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\lambda_1 \|w_t - \alpha \nabla f^1(w_t) - w'_t + \alpha \nabla f^1(w'_t)\| + \lambda_2 \|w_t - \alpha \nabla f^2(w_t) - w'_t + \alpha \nabla f^2(w'_t)\|\right] \\
&\leq \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\|w_t - w'_t\|\right] + \alpha \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left(\lambda_1 \|\nabla f^1(w'_t) - \nabla f^1(w_t)\| + \lambda_2 \|\nabla f^2(w'_t) - \nabla f^2(w_t)\|\right)
\end{aligned}$$

(Triangle Inequality used for above step)

$$\begin{aligned}
&\leq \delta_t + \alpha \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\lambda_1 \beta_1 \|w_t - w'_t\| + \lambda_2 \beta_2 \|w_t - w'_t\|\right] \quad \text{(Smoothness)} \\
&= \delta_t + \alpha \lambda_1 \beta_1 \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\|w_t - w'_t\|\right] + \alpha \lambda_2 \beta_2 \mathbb{E}_{f_1 \dots f_t \sim \mathcal{P}_\lambda}\left[\|w_t - w'_t\|\right] \\
&= \delta_t + \alpha(\lambda_1 \beta_1 + \lambda_2 \beta_2)\delta_t \\
&= (1 + \alpha(\lambda_1 \beta_1 + \lambda_2 \beta_2))\delta_t
\end{aligned}$$

□

C.4.4 Stability of Dynamic Sampling

We repeat the description of our Auxiliary Learning with Dynamic Sampling Setting here for ease of access.

Setting : We are given an auxiliary objective $f_a(\cdot; z) \in [0, 1]$ with N_a samples $S_a = (z_1, \dots, z_{N_a})$ from the distribution $\mathcal{D}_a$. At any iteration of SGD, we sample a choice of either the end-task function f_e or the auxiliary objective f_a according to the probabilities λ_e, $\lambda_a \mid \lambda_e + \lambda_a = 1$. Given the chosen objective, we sample a data-point and perform stochastic gradient descent (SGD) based on the sampled data-point.

An equivalent way to instantiate this procedure to create S_A by drawing $N' = N_e + N_a$ total samples from the end-task and auxiliary task according to $\mathcal{P}_\lambda$. S'_A is then created by replacing 1 end-task sample in S_A. At each step, a sample is drawn from a distribution : $z_i, z'_i \sim P_{S_A}, P_{S'_A}$ and a gradient step is taken on the function corresponding to the set the sample was drawn from.

Lemma 7 (Stability of dynamic sampling)**.** *We denote the outputs of T steps of SGM on S_A and S'_A with the dynamically sampled functions, as w_T and w'_T respectively. Then, for every $z_e \in Z_e$ and every $t_0 > 0$, under both the random update rule and the random permutation rule, we have :*

$$\mathbb{E}\big|f_e(w_T; z) - f_e(w'_T; z)\big| \leq \frac{\gamma t_0}{N'} \sup_{w, z_e} f_e(w; z_e) + L\mathbb{E}[\delta_T | \delta_{t_0} = 0]$$

Where $N' = N_e + N_a$ and $\gamma = \frac{\lambda_e \cdot N'}{N_e} = \frac{\lambda_e}{\lambda^r}$.

Proof. Let $\mathcal{E} = \mathbf{1}[\delta_{t_0} = 0]$ denote the event that $\delta_{t_0} = 0$. We have

$$\begin{aligned}
\mathbb{E}\big|f_e(w_T; z) - f_e(w'_T; z)\big| &= P\{\mathcal{E}\}\mathbb{E}\big[\big|f_e(w_T; z) - f_e(w'_T; z)\big||\mathcal{E}\big] \\
&\quad + P\{\mathcal{E}^c\}\mathbb{E}\big[\big|f_e(w_T; z) - f_e(w'_T; z)\big||\mathcal{E}^c\big] \\
&\leq \mathbb{E}\big[\big|f_e(w_T; z) - f_e(w'_T; z)\big||\mathcal{E}\big] + P\{\mathcal{E}^c\} \cdot \sup_{w, z_e} f_e(w; z_e) \\
&\quad \text{because } f_e \text{ is non-negative} \\
&\leq L\mathbb{E}\big[\|w_T - w'_T\||\mathcal{E}\big] + P\{\mathcal{E}^c\} \cdot \sup_{w, z_e} f_e(w; z_e) \\
&\quad \text{because } f_e \text{ is } L\text{-Lipschitz}
\end{aligned} \tag{C.2}$$

We now proceed to bound $P\{\mathcal{E}^c\}$. Let $i_* \in [N']$ denote the position in which S_A, S'_A differ and consider the random variable I assuming the index of the first time step in which SGM uses the example $z_e^{i_*}$. Note that when $I > t_0$, then we must have that $\delta_{t_0} = 0$ since the two samples are identical up until this point.

$$P\{\mathcal{E}^c\} = P\{\delta_0 \neq 0\} \leq P\{I \leq t_0\}$$

Using the selection rule specified above (sample either f_e, f_a according to the probabilities λ_e, λ_a and then sample uniformly from the selected task data) we have that :

$$P\{I \leq t_0\} = \sum_{t=1}^{t_0} P\{I = t_0\} = \sum_{t=1}^{t_0} \big(\lambda_e \cdot \frac{1}{N_e}\big) = \frac{\lambda_e t_0}{N_e} = \frac{\gamma t_0}{N'}$$

□

Theorem 8 (Stability Bound on Dynamic Sampling)**.** *Assume that $f_e(; z_e), f_a(; z_a) \in [0, 1]$ are L-Lipschitz and β_e and β_a-smooth loss functions. Consider that we have $N' = N_e + N_a$ total samples where f_e and f_a have N_e and N_a samples respectively. Suppose that we run SGM for T steps with monotonically non-increasing step sizes $\alpha_t \leq \frac{c}{t}$ by dynamically sampling the tasks according to λ_e and λ_a. Then, with respect to f_e, SGM has uniform stability with :*

$$\epsilon_{\text{stab}} \leq \left(1 + \frac{1}{c\bar{\beta}}\right)\left(\frac{2\gamma L^2 c}{N' - \gamma} + \rho L c\right)^{\frac{1}{c\beta+1}}\left(\frac{\gamma T}{N'}\right)^{\frac{c\bar{\beta}}{1+c\beta}}$$

$$\textit{Where} \quad \gamma = \frac{\lambda_e N'}{N_e}$$

Given that $\beta^* = \min\{\beta_e, \beta_a\}$ and λ^* is the corresponding weighting of the function with smaller smoothness.

Depending on which one gives a tighter bound the pair $(\bar{\beta}, \rho)$ can be :

$$(\bar{\beta}, \rho)_1 = (\lambda^*\beta^*,\ (1 - \lambda^*)\big(\Delta + 2L\big))$$

or

$$(\bar{\beta}, \rho)_2 = (\lambda_e\beta_e + \lambda_a\beta_a,\ 0)$$

When $(\bar{\beta}, \rho)_1$ gives the tighter bound, we can simplify to :

$$\epsilon_{\text{gen}} \lessapprox \big(\Delta\big)^{\frac{1}{1+c\lambda^*\beta^*}}\left(\frac{\gamma T}{N'}\right)^{1 - \frac{1}{c\lambda^*\beta^*+1}}$$

As presented in Section 5.5.

Proof. Let S_A, S'_A be two sample of size $N' = N_e + N_a$ as described in lemma 7. Consider the gradient updates $G_{f_1}, \ldots, G_{f_T}$ and $G'_{f_1}, \ldots, G'_{f_T}$ induced by running SGM on samples S_A and S'_A respectively. Let w_T and w'_T denote the corresponding outputs of SGM. By lemma 7 we have :

$$\mathbb{E}\big|f_e(w_T; z) - f_e(w'_T; z)\big| \leq \frac{\gamma t_0}{N'} \sup_{w, z_e} f_e(w; z_e) + L\mathbb{E}[\delta_T | \delta_{t_0} = 0] \tag{C.3}$$

Let $\Psi_T = \mathbb{E}[\delta_T | \delta_{t_0} = 0]$. We will bound Ψ_T as function of t_0 and then minimize for t_0. Note the following :

- At any step t, with probability $\big(1 - \frac{\gamma}{N'}\big)$, the sample selected is the same in both S_A and S'_A. In this case $G_{f_t} = G'_{f_t}$ and we use the corresponding expansivity rule from lemma 7. This gives :

  $$\delta_{t+1} \leq \min\Big\{\big(1 + \alpha_t \lambda^* \beta^*\big)\delta_t + \alpha_t(1 - \lambda^*)\big(\Delta + 2L\big),\ \big(1 + \alpha_t\big(\lambda_e \beta_e + \lambda_a \beta_a\big)\big)\delta_t\Big\}$$

  Where $\beta^* = \min\{\beta_e, \beta_a\}$ and λ^* is the corresponding weighting of the function with smaller smoothness. To avoid deriving the bound independently for each case, we perform a variable substituation that captures the two cases :

  $$\delta_{t+1} \leq \big(1 + \alpha_t \bar{\beta}\big)\delta_t + \alpha_t \rho$$

  $\bar{\beta} = \big\{\lambda^* \beta^*,\ \lambda_e \beta_e + \lambda_a \beta_a\big\}$ and $\rho = \big\{(1 - \lambda^*)\big(\Delta + 2L\big), 0\big\}$. We can present the final bound in terns of these variables which can be substituted depending on the minimizer.
- With probability $\frac{\gamma}{N'}$ the selected example is different. Note that in this case, we know that we are evaluating the end-task function f_e. We use that both G_{f_t} and G'_{f_t} are $(\sigma_t = \alpha_t L)$-bounded according to lemma 6 since f_e is L-Lipschitz.

Combining the above we have :

$$\begin{aligned}
\Psi_{t+1} &\leq \big(1 - \frac{\gamma}{N'}\big)\bigg(\big(1 + \alpha_t \bar{\beta}\big)\Psi_t + \alpha_t \rho\bigg) + \frac{\gamma}{N'}\big(\Psi_t + 2\alpha_t L\big) \\
&= \bigg(\frac{\gamma}{N'} + \big(1 - \frac{\gamma}{N'}\big)\big(1 + \alpha_t \bar{\beta}\big)\bigg)\Psi_t + \frac{2\gamma \alpha_t L}{N'} + \alpha_t\big(1 - \frac{\gamma}{N'}\big)\rho \\
&= \bigg(1 + \big(1 - \frac{\gamma}{N'}\big)\alpha_t \bar{\beta}\bigg)\Psi_t + \frac{\alpha_t\big(2\gamma L + (N' - \gamma)\rho\big)}{N'} \\
&\leq \bigg(1 + \big(1 - \frac{\gamma}{N'}\big)\frac{c}{t}\bar{\beta}\bigg)\Psi_t + \frac{c\big(2\gamma L + (N' - \gamma)\rho\big)}{tN'} \\
&\leq \exp\bigg(\big(1 - \frac{\gamma}{N'}\big)\frac{c}{t}\bar{\beta}\bigg)\Psi_t + \frac{c\big(2\gamma L + (N' - \gamma)\rho\big)}{tN'} \\
&\quad \text{We use } 1 + x \leq \exp(x)\ \forall x \\
&\leq \exp\bigg(\big(1 - \frac{\gamma}{N'}\big)\frac{c}{t}\bar{\beta}\bigg)\Psi_t + \frac{c\bar{\rho}}{tN'} \\
&\quad \text{Where } \bar{\rho} = \big(2\gamma L + (N' - \gamma)\rho\big)
\end{aligned} \tag{C.4}$$

We can unwind the recurrence until $\Psi_{t_0} = 0$.

$$\begin{aligned}
\Psi_T &\leq \sum_{t=t_0+1}^{T} \left(\prod_{k=t+1}^{T} \exp\left((1-\frac{\gamma}{N'})\frac{c\bar{\beta}}{k}\right)\right)\left(\frac{c\bar{\rho}}{tN'}\right) \\
&= \sum_{t=t_0+1}^{T} \left(\frac{c\bar{\rho}}{tN'}\right) \exp\left((1-\frac{\gamma}{N'})c\bar{\beta}\sum_{k=t+1}^{T}\frac{1}{k}\right) \\
&\leq \sum_{t=t_0+1}^{T} \left(\frac{c\bar{\rho}}{tN'}\right) \exp\left((1-\frac{\gamma}{N'})c\bar{\beta}\log\left(\frac{T}{t}\right)\right) \\
&= \frac{c\bar{\rho}T^{c\bar{\beta}(1-\frac{\gamma}{N'})}}{N'} \sum_{t=t_0+1}^{T} t^{-c\bar{\beta}(1-\frac{\gamma}{N'})-1} \qquad \text{(C.5)}
\end{aligned}$$

We can upper bound the sum over t with an integral + drop negative terms

$$\begin{aligned}
&\leq \frac{c\bar{\rho}}{N'c\bar{\beta}(1-\frac{\gamma}{N'})}\left(\frac{T}{t_0}\right)^{c\bar{\beta}(1-\frac{\gamma}{N'})} \\
&= \frac{\bar{\rho}}{\bar{\beta}(N'-\gamma)}\left(\frac{T}{t_0}\right)^{c\bar{\beta}(1-\frac{\gamma}{N'})} \\
&\leq \frac{\bar{\rho}}{\bar{\beta}(N'-\gamma)}\left(\frac{T}{t_0}\right)^{c\bar{\beta}}
\end{aligned}$$

Plugging this bound back into Equation C.3 and using the fact that $f_e \in [0,1]$:

$$\mathbb{E}\big|f_e(w_T;z) - f_e(w'_T;z)\big| \leq \frac{\gamma t_0}{N'} + \frac{L\bar{\rho}}{\bar{\beta}(N'-\gamma)}\left(\frac{T}{t_0}\right)^{c\bar{\beta}} \qquad \text{(C.6)}$$

We let $q^* = c\bar{\beta}$, we can minimize the R.H.S by setting :

$$t_0 = \left(\frac{N'Lc\bar{\rho}}{\gamma(N'-\gamma)}\right)^{\frac{1}{q^*+1}} T^{\frac{q^*}{q^*+1}}$$

Plugging this in gives us :

$$\begin{aligned}
\mathbb{E}\big|f_e(w_T;z) - f_e(w'_T;z)\big| &\leq \left(\frac{(1+\frac{1}{c\bar{\beta}})}{N'}\right)\left(\frac{N'Lc\big(2\gamma L + (N'-\gamma)\rho\big)}{(N'-\gamma)}\right)^{\frac{1}{c\bar{\beta}+1}}\big(\gamma T\big)^{\frac{c\bar{\beta}}{1+c\bar{\beta}}} \\
&= \left(1+\frac{1}{c\bar{\beta}}\right)\left(\frac{2\gamma L^2 c}{N'-\gamma} + \rho Lc\right)^{\frac{1}{c\bar{\beta}+1}}\left(\frac{\gamma T}{N'}\right)^{\frac{c\bar{\beta}}{1+c\bar{\beta}}} \qquad \text{(C.7)}
\end{aligned}$$

Recall that :

$$\bar{\beta} = \big\{\lambda^*\beta^*,\ \lambda_e\beta_e + \lambda_a\beta_a\big\}$$

$$\rho = \big\{(1-\lambda^*)\big(\Delta + 2L\big), 0\big\}$$

We can choose whichever of the pairs for $\bar{\beta}, \rho$ that minimizes the bound : □

C.5 Discussion of Generalization Error Bounds

C.5.1 What Does Theorem 8 Say.

We consider the setting where

$$\bar{\beta} = \lambda^* \beta^*$$
$$\rho = (1 - \lambda^*)\big(\Delta + 2L\big)$$

Assuming the ρ term dominates Equation C.7 in this setting is :

$$\begin{aligned}
\epsilon_{\text{gen}}^{\text{auxdyn}} \leq \epsilon_{\text{stab}}^{\text{auxdyn}}\big|_{(\bar{\beta},\rho)_1} &\lessapprox \sqrt[1+c\bar{\beta}]{(1-\lambda^*)(\Delta+2L)}\left(\frac{\gamma T}{N'}\right)^{\frac{c\bar{\beta}}{1+c\bar{\beta}}} \\
&\lessapprox \big(\Delta\big)^{\frac{1}{1+c\lambda^*\beta^*}}\left(\frac{\gamma T}{N'}\right)^{1-\frac{1}{c\lambda^*\beta^*+1}} \quad \text{This is Equation 5.1 from Section 5.5}
\end{aligned} \tag{C.8}$$

In going from the first line to the second we consider the setting where $\Delta \gg 2L$. This is a case where the auxiliary task is sufficiently different from the primary task. Some observations about this setting:

1. Smaller Δ implies auxiliary task is similar to main task and leads to improving the bound.
2. Dependence of the bound on N' is a bit more nuanced. **Note that increasing N' increases γ unless we reduce λ_e appropriately**. Remember that λ_e is the rate at which we sample the primary task. Thus, if we add more auxiliary data but still sample the primary task at the original rate, then we are effectively ignoring the extra auxiliary data.
3. It might be tempting to assume that we can get arbitrary improvements in this setting by setting $\lambda_e = 0$. However, **note that whilst this might reduce the generalization error**, it means that we are seeing none of the end-task which would result in large **increase in the training error**
4. Note that $(\bar{\beta} = \lambda^*\beta^* \leq \beta_e)$ always. So we get improvements on the dependence on T compared to Theorem 5.
5. We can optimize λ_e, λ_a to minimize $\epsilon_{\text{stab}}^{\text{auxdyn}}$.

Appendix D

Structured Pruning under Limited Task Data (Chapter 6) Appendix

D.1 Datasets

Table D.1 describes the tasks and datasets used in our experiments.

Table D.1: Specifications of datasets used to evaluate our methods.

Domain	Task	Task-Type	Train Size	Metric
BIOMED	CHEMPROT Kringelum et al. (2016)	Classification	4169	Accuracy
	RCT Dernoncourt and Lee (2017)	Classification	10K*	Accuracy
CS	SCIIE Luan et al. (2018)	Classification	3219	Accuracy
	ACL-ARC Jurgens et al. (2018)	Classification	1688	Accuracy
GLUE	STSB Wang et al. (2018b)	Sentence Similarity	7K	Pearson's Correlation
	MRPC Wang et al. (2018b)	Paraphrase Detection	3.7K	Accuracy

D.2 Training Details

We follow as closely as possible the hyper-parameters that are used in the original CoFi code base. Table D.2 reports CoFi-specific hyper-parameter settings.

Unlike the original CoFi, we turn off output prediction distillation for all experiments. ie – we do not distill the predictions from the pre-trained models since unlike in the original CoFi paper, we are not starting from a model that has already been fine-tuned on the target task but rather we are starting from the pre-trained model itself.

During pruning, we perform 10K gradient descent steps to learn both the structural and parameter variables of the model. We perform 20 epochs of post-pruning finetuning on the target task.

Table D.2: Hyper-parameter choices

Hyper-parameter	Values	Description
Task pair weightings	(1, 1), (1, 2), (2, 1)	Weightings applied to transfer task vs target task during training.
Model LR - Pruning	1e-4, 2e-5	Learning rate used for model parameters during pruning.
Model LR - Finetuning	1e-4, 2e-5	Learning rate used for finetuning pruned model.
Structure LR	0.1, 0.01	Learning rate used for learning structural parameters.
δ-l_2 Reg Weight	1e-2	Regularization weight used in δ-formulation.

Appendix E

Bonsai (Chapter 7) Appendix

E.1 Main Experiment Details

E.1.1 Full Algorithm

Algorithm 5: Bonsai

1: **Input:**
Model $[\mathbf{M}_\theta]$, sub-models per iteration $[n_{\text{iter}}]$
Sparsity per iteration $[p_{\text{iter}}]$, Target sparsity $[p]$
Module list $[\mathbf{m}]$
2: **for** $l = 1$ **to** $\lceil \frac{p}{p_{\text{iter}}} \rceil$ **do**
3: $\rho^l \leftarrow$ Calculate unstructured pruning metric for all modules in $\mathbf{m}$
4: $\bar{\rho}^l \leftarrow$ Fix the top $(1 - 2p_{\text{iter}})$ of ρ^l to ∞
5: Sample $\{\bar{\mathbf{m}}_i\}_{[n_{\text{iter}}]}$ sub-models according to $\bar{\rho}^l$
6: Run forward pass on each sub-model and compute U. Construct $\mathbb{D}^l = \{\bar{\mathbf{m}}_i, U_i\}_{[n_{\text{iter}}]}$
7: $\beta^l \leftarrow \text{Regress}\left(\mathbb{D}^l\right)$
8: $\{m_{\text{pruned}}\} \leftarrow$ sort β^l and drop the bottom k modules that make up p_{iter} fraction of the model.
9: $\mathbf{m} \leftarrow$ update module list to exclude $\{m_{\text{pruned}}\}$
10: **end for**
11: **Output:** Pruned model $\mathbf{M}_{|\mathbf{m}}$

A note about Line 5 in our algorithm. Depending on the task, we find that sampling $\bar{\mathbf{m}}_i$ and its complement $\bar{\mathbf{m}}_i^c$ helps reduce the variance of our regression estimate and leads to better results.

E.1.2 Comparison with FLAP (An et al., 2024)

Close to the time of submission, we found out about FLAP (An et al., 2024), a recently proposed forward pass only method that was put out a little over a month before the first release of this

Table E.1: Perplexity at 50% sparsity of LLaMA-{1,2} on Wikitext-2 datasets.

Dataset	Method	Sparsity	Llama-1	Llama-2
Wikitext-2	Base Model	0%	5.68	5.11
	Wanda-Structured	50%	226.23	147.04
	FLAP	50%	31.80	23.95
	Bonsai	50%	26.49	18.29

work. Table E.1 reflects results of preliminary experiments we run to compare to FLAP. We plan to include a broader set of result comparisons by the rebuttal period. But below, we discuss differences between our methods.

FLAP is much faster than Bonsai (< 1hr vs ≈ 24hrs to generate the above results). Though Bonsai is slower, one of its core strengths is that we can improve the quality of pruning by running for longer (more perturbations, more data-samples, slower iterative pruning) but at the cost of more time (a trade-off we note and leave to the discretion of the practitioner).

E.1.3 Hyper-parameters for all Bonsai regression during pruning

When using Bonsai, we estimate β from $\mathbb{D}^l$ by performing linear regression via gradient descent with Adam (Kingma and Ba, 2014). We cross-validate over the following set of hyper-parameters. Note that doing this cross-validation takes much less time than the time needed to construct the dataset $\mathbb{D}^l$. During cross validation, we choose the model whose predictions have

Table E.2: Bonsai hyper-parameters for regression. This applies to all experiments unless otherwise specified

γ(Regression Weight)	Learning rate	Batch Size	Epochs
{100, 0, 1e-4}	{100, 10, 1, 0.1}	{32, 64, 128}	50

the best Kendall rank correlation co-efficient (Kendall, 1948) with the target. We do this because we do not care about matching U_k exactly for each sub-model k; we rather care that our learned β predicts the correct rankings amounts sub-models, which would denote that β reasonably models relative module importances.

In general, we use ℓ_1-norm regularization on β for all experiments. For the Phi-2 experiment in Chapter 7, we find that ℓ_2-norm works better.

E.1.4 Forward Pass Only / Semi-structured pruning Experiments

Table E.3 show the Bonsai hyperparameters we used for the experiments in Section 7.4.

Table E.3: Bonsai hyper-params for forward only experiments

p_{iter}	$\text{ns}_{\text{sub-models}}$	ns_{data}	Metric for ρ
0.05	200	32 (per-iter)	Wanda

Table E.4: Bonsai fine-tuning HP for pruned LLaMA family models

LR	rank	LoRA-α	λ (Distill Weight)	LoRA Modules
1e-4	128	4×rank	0.01	All Modules

For Wanda(Sun et al., 2023a), we use the default hyper-parameters specified by the paper repo here for pruning. For fine-tuning, we use rank = 64. We apply LoRA to only the q_proj and v_proj matrices in each layer of the pruned LLaMA model – this is unlike with Bonsai where we fine-tune all modules. We cannot do same because since the Wanda model just produces sparse matrices, the matrices instantiated during the backward pass are the same sizes as the sparsified matrices and thus occupy more memory (compared to our approach that actually makes the matrices smaller in dimension instead of sparsifying). We are also unable to perform distillation on the Wanda models due to this reason. For fine-tuning the Phi-2 model on Wikitext-2, we use the same hyper-parameters as Bonsai in Table E.4.

E.1.5 Experiments comparing to Gradient based structured pruning

We compare to LoRA-Prune and LLM-Pruner. We take their performance results directly from the LoRA-Prune paper. Whilst we use 1 A6000 GPU (48G) for all experiments, LoRA-Prune uses A100 GPU (80G) for pruning LLaMA-1 7B.

All Bonsai hyper-parameters are the same as Appendix E.1.4 except for $\text{ns}_{\text{sub-models}}$ which we set to 1000.

E.1.6 Phi-2 pruning experiment details

For the Phi experiment in Chapter 7, All Bonsai hyper-parameters are the same as Appendix E.1.4 except for the following changes:

- $\text{ns}_{\text{sub-models}} = 2000$
- $p_{\text{iter}} = 0.35$. We thus perform 1-shot pruning directly to the target sparsity of 35%. We find that this seems to work best for the Phi-2 model. We posit that this might be because the Phi-2 models use LayerNorm(Ba et al., 2016) whilst the other models we explore, LLaMA and Mistral use RMSNorm.
- Due to its relatively small size, the 1.8B pruned model can be fully fine-tuned on a single A6000 GPU over 100k sequences of length 2,048 tokens from the C4 dataset instead of using LoRA.

E.2 Impact of regression and perturbation ablation details

For the experiment in E.8, All Bonsai hyper-parameters are the same as Appendix E.1.4 except $p_{\text{iter}} = 0.1$ to speed up pruning.

A simple alternative to Bonsai is to leverage the prior ρ, computed from the unperturbed parent model, and make pruning decisions exclusively according to this. This is the `No Perturbation`

`+ No Regression` baseline in Figure 7.3. This approach has quite poor performance. We can further improve this baseline by adding back perturbative aspect where we prune the parent model according to ρ' which is computed by aggregating the ρ metric computed over the **perturbed** models. Note that we use a Wanda based metric to define ρ for this experiment. Module level analogues of the unstructured pruning metrics we explore are defined in Appendix E.6.

Table E.5: Experiment on linear regression to estimate module importances. Wikitext-2 Perplexity. LLaMA-2 7B pruned to 50% sparsity

Linear Regression	Relative Speepdup	**w/o Post-Pruning Adaptation**	**w Post-Pruning Adaptation**
No	2.06	146.57	9.68
Yes	1.77	61.63	9.15

E.3 Varying the pruning fraction per-iteration

For the experiment in Section E.8, All Bonsai hyper-parameters are the same as Appendix E.1.4 except we vary p_{iter}.

Table E.6: Varying the fraction pruned at a time. Wikitext-2 Perplexity. LLaMA-2 7B pruned to 50% sparsity

Prune Frac	Relative Speepdup	**w/o Post-Pruning Adaptation**	**w Post-Pruning Adaptation**
0.05	1.58	19.47	8.89
0.1	1.77	61.63	9.15
0.20	1.67	209.44	9.57

E.4 Varying the number of calibration data points for pruning

All Bonsai hyper-parameters are the same as Appendix E.1.4 except we vary ns_{data} and $p_{\text{iter}} = 0.1$ to speed up pruning.

E.5 Post-pruning adaptation

For this experiment, All Bonsai hyper-parameters are the same as Appendix E.1.4.

Table E.7: How many data-points to consider during forward passes. Wikitext-2 Perplexity. Llama-2 7B pruned to 50% sparsity

ns_{data}	**w/o Adapt**	**w Adapt**
8	130.04	9.45
32	61.63	9.15

E.6 Impact of prior

For this experiment, All Bonsai hyper-parameters are the same as Appendix E.1.4 except we vary ρ

E.6.1 ρ is Activation Magnitude

MLP / Fully Connected Module: Let d be the intermediate dimension of the MLP to be pruned. Note that for all transformer models evaluate, the MLP components are 2 layer and thus have a single intermediate dimension. For any data-sample sequence b, we flatten model activation at this point $\mathbf{a} \in \mathbb{R}^{B \times S \times d} \rightarrow \mathbb{R}^{BS \times d}$ and then compute the following averaged activation magnitude :

$$\Big(\rho \in \mathbb{R}^d\Big) \propto \hat{\mathbf{a}} = \frac{1}{B} \sum_b \text{Mean}\Bigg(\Big|\mathbf{a}_b\Big|, \text{axis} = 0\Bigg) \tag{E.1}$$

Self-Attention Module: For any data-sample sequence b, the output of the self attention module before the final output projection is $\mathbf{a} \in \mathbb{R}^{B \times S \times d_h \times h}$ where h is the number of attention heads and d_h is the size of each head's output. We can flatten $\mathbf{a} \in \mathbb{R}^{B \times S \times d_h \times h} \rightarrow \mathbb{R}^{BSd_h \times h}$ and then use the same formula as Equation E.2 above to calculate ρ.

$$\Big(\rho \in \mathbb{R}^h\Big) \propto \hat{\mathbf{a}} \tag{E.2}$$

E.6.2 ρ is Wanda (Sun et al., 2023a)

MLP / Fully Connected Module: Let d be the intermediate dimension of the MLP to be pruned. Let $W \in \mathbb{R}^{d \times o}$ be the output projection matrix for the MLP. For any data-sample sequence b, we flatten model activation before the final output, $\mathbf{a} \in \mathbb{R}^{B \times S \times d} \rightarrow \mathbb{R}^{BS \times d}$ and then compute the following metric which is a module level analogue of Wanda:

$$\begin{aligned} \Big(\rho \in \mathbb{R}^d\Big) \propto \hat{\mathbf{a}} &= \frac{1}{o} \sum_o \mathbf{a}^o \\ \mathbf{a}^o &= \Big|W[:, o]\Big| \odot \text{RootMeanSquare}\Bigg(\mathbf{a}, \text{axis} = 0\Bigg) \end{aligned} \tag{E.3}$$

Self-Attention Module: Let $W \in \mathbb{R}^{d \times o}$ be the output projection matrix for the self-attention module. For any data-sample sequence b, the output of the self attention module before the final output projection is $\mathbf{a} \in \mathbb{R}^{B \times S \times d_h \times h}$ where h is the number of attention heads and d_h is the

size of each head's output. We can flatten $\mathbf{a} \in \mathbb{R}^{B \times S \times d_h \times h} \rightarrow \mathbb{R}^{BSd_h \times h}$ and then use the same formula as Equation E.2 above to calculate $\rho \in \mathbb{R}^h$.

E.7 How many perturbative samples are reasonable?

For this experiment, All Bonsai hyper-parameters are the same as Appendix E.1.4 except $p_{\text{iter}} = 0.1$ to speed up pruning.

Table E.8: Varying the number of sub-models generated. Wikitext-2 Perplexity. LLaMA-2 7B pruned to 50% sparsity

Num Samples	w/o Post-Pruning Adaptation	w Post-Pruning Adaptation
1000	22.09	9.25
200	61.63	9.15
50	NaN	9.24

Using $\text{ns}_{\text{sub}-\text{models}} = 50$ results in an model with NaN perplexity on the Wikitext validation set. We posit that this is because of the LLaMA models are half precision, and removing the wrong modules can result in activations going outside of the FP16 dynamic range for unique data-points. Note that we are able to recover good performance of the model after fine-tuning though (we do not observe NaNs with the Wikitext-2 training data). This indicates that Bonsai actually recovers good modules even using as few samples as 50 sub-models.

E.8 Mistral-7B Experiment Details

In addition to the primary experiments on the LLaMA and Phi-2 models, supplementary experiments were performed on the Mistral-7B Jiang et al. (2023) model in comparison with Wanda results on the stated model. We apply Bonsai with the same hardware and configuration settings as used for the LLaMA and Phi-2 experiments. We target different pruning fractions (0.05, 0.1, and 0.2) across different numbers of samples and masks per iteration to evaluate the method's performance under varying sparsity conditions.

The Mistral-7B model architecture differs from the LLaMA architecture in its use of group query attention and sliding window attention in lieu of the standard self-attention used in most transformer-based models like LLaMA Jiang et al. (2023). We factor these differences into consideration in the implementation of Bonsai for Mistral. For the experiments that produced the results below, all Bonsai hyper-parameters are the same as Appendix E.1.4.

Table E.9 presents the test perplexity results for Mistral-7B under different pruning methods. Considering the fully-structured sparsity nature of Bonsai, it achieves a test perplexity of 47.5 without post-pruning adaptation, with 1.66× inference speedup. After performing post-pruning adaptation on our pruned Mistral-7B, perplexity dropped drastically to 10.08. Note that the

reported results of Wanda-pruned Mistral-7B are not fine-tuned afterward; if they were, their results would be marginally better than Bonsai's results. However, as shown in Table 7.3 latency speedup would have dropped rapidly, while Bonsai stays the same at 1.66×.

Table E.9: Test perplexity of Mistral-7B model on Wikitext-2 across fully-structured Bonsai and semi-structured Wanda methods.

	Sparsity Level	**Method**	**Test PPL**
Original, unpruned Mistral-7B	N/A	N/A	5.245
Wanda	semi-structured 2-4	magnitude	13.81
		Wanda	12.38
		SparseGPT	10.46
Bonsai (w/o Adaptation)	structured 50%	magnitude	67.48
		Wanda	47.50
Bonsai (w/ Adaptation)	structured 50%	Wanda	10.08

We further investigate the pruning habits of Bonsai by examining the pruned layers of Mistral, as shown in Figure E.1. We notice a recurring theme: when an attention layer is significantly altered, it leads to compensation in the next layers within the sequence. This adaptive behavior, termed the "Hydra effect" by (McGrath et al., 2023), implies that the layers within a language model interact in a way that changes in one layer prompt adjustments in another. (McGrath et al., 2023) specifically mentioned that when one attention layer was removed from a language model, the model was still able to self-repair and produce similar outputs; but it did so by relying more heavily on other layers.

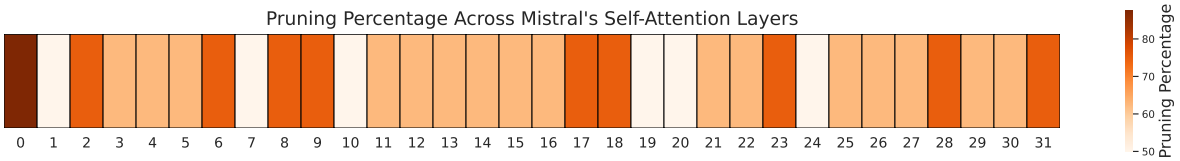


Figure E.1: Mistral's pruned attention layers. The heavily pruned layers are usually preceded by or sandwiched between lightly-pruned layers, exhibiting the self-repairing "Hydra effect" McGrath et al. (2023).

Should Bonsai prune iteratively? Table E.10 demonstrates the benefits of using Bonsai in an iterative fashion. Pruning slowly ($p_{\text{iter}} = 0.05$) yields the best results, but this comes at the cost of increasing the total time to prune the model. The performance gap between values of p_{iter} persists even after post-pruning adaptation, indicating that slower pruning allows for more accurate estimates of module importance.

Table E.10: Varying p_{iter}. Wikitext-2 perplexity of LLaMA-2 7B pruned to 50% sparsity.

	$p_{\text{iter}} = 0.05$	$p_{\text{iter}} = 0.1$	$p_{\text{iter}} = 0.2$
w/o Adapt	19.47	61.63	209.44
w Adapt	8.89	9.15	9.57

Bibliography

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023a.

Lucio M Dery, Paul Michel, Ameet Talwalkar, and Graham Neubig. Should we be pre-training? an argument for end-task aware training as an alternative. *arXiv preprint arXiv:2109.07437*, 2021a.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019a.

Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, Bohan Zhuang, et al. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

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

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the

opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, et al. Artificial intelligence index report 2023. *arXiv preprint arXiv:2310.03715*, 2023.

Stevo Bozinovski and Ante Fulgosi. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica*, volume 3, pages 121–126, 1976.

Lorien Y Pratt. Discriminability-based transfer between neural networks. *Advances in neural information processing systems*, 5, 1992.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18, 2019.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. *Advances in neural information processing systems*, 7, 1994.

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

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

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

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095*, 2021.

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

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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

Anne-Laure Ligozat, Julien Lefevre, Aurélie Bugeau, and Jacques Combaz. Unraveling the hidden environmental impacts of ai solutions for environment life cycle assessment of ai solutions. *Sustainability*, 14(9):5172, 2022.

Peter Stone and Manuela Veloso. Learning to solve complex planning problems: Finding useful auxiliary problems. In *Proceedings of the AAAI 1994 Fall Symposium on Planning and Learning*, pages 137–141, 1994.

Pengcheng Wu and Thomas G Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the twenty-first international conference on Machine learning*, page 110, 2004.

Rajat Raina, Andrew Y Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 713–720, 2006.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Lucio M Dery, Yann Dauphin, and David Grangier. Auxiliary task update decomposition: The good, the bad and the neutral. *arXiv preprint arXiv:2108.11346*, 2021b.

Lucio M Dery, Paul Michel, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Aang: Automating auxiliary learning. *arXiv preprint arXiv:2205.14082*, 2022.

Lucio Dery, David Grangier, and Awni Hannun. Transfer learning for structured pruning under limited task data. *arXiv preprint arXiv:2311.06382*, 2023.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Devang K Naik and Richard J Mammone. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 437–442. IEEE, 1992.

Maxwell L Hutchinson, Erin Antono, Brenna M Gibbons, Sean Paradiso, Julia Ling, and Bryce Meredig. Overcoming data scarcity with transfer learning. *arXiv preprint arXiv:1711.05099*, 2017.

Simran Khanuja, Srinivas Gowriraj, Lucio Dery, and Graham Neubig. Demux: Data-efficient multilingual learning. *arXiv preprint arXiv:2311.06379*, 2023.

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997a.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Atharva Kulkarni, Lucio M Dery, Amrith Setlur, Aditi Raghunathan, Ameet Talwalkar, and Graham Neubig. Multitask learning can improve worst-group outcomes. *Transactions on Machine Learning Research*, 2023.

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

John C Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization. *The Annals of Statistics*, 49(3):1378–1406, 2021.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong,

and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76, 2020.

Runxue Bao, Yiming Sun, Yuhe Gao, Jindong Wang, Qiang Yang, Haifeng Chen, Zhi-Hong Mao, Xing Xie, and Ye Ye. A survey on heterogeneous transfer learning. *arXiv preprint arXiv:2310.08459*, 2023.

Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2, 1995.

Junhong Shen, Liam Li, Lucio M Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine. In *International Conference on Machine Learning*, pages 31030–31056. PMLR, 2023.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*, 2018a.

Shuxiao Chen, Koby Crammer, Hangfeng He, Dan Roth, and Weijie J Su. Weighted training for cross-task learning. *arXiv preprint arXiv:2105.14095*, 2021.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. The risks of invariant risk minimization. *arXiv preprint arXiv:2010.05761*, 2020.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model ro-

bustness and uncertainty. In *International conference on machine learning*, pages 2712–2721. PMLR, 2019.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

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

Yuan Gao, Weizhong Zhang, Wenhan Luo, Lin Ma, Jin-Gang Yu, Gui-Song Xia, and Jiayi Ma. Aux-nas: Exploiting auxiliary labels with negligibly extra inference cost. *arXiv preprint arXiv:2405.05695*, 2024.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*, 2015.

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

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*, 2021.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

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

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

Jürgen Schmidhuber. On learning how to learn learning strategies. 1995.

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

Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization, 2018.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search.

arXiv preprint arXiv:1806.09055, 2018a.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. *arXiv preprint arXiv:1804.00222*, 2018.

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. Nlp from scratch without large-scale pretraining: A simple and efficient framework. *arXiv preprint arXiv:2111.04130*, 2021.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld. S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*, 2019.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406, 2018.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602*, 2018.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016, 2016.

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

Yian Zhang, Alex Warstadt, Haau-Sing Li, and Samuel R Bowman. When do you need billions of words of pretraining data? *arXiv preprint arXiv:2011.04946*, 2020.

Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems*, pages 657–664, 1995.

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

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, 2008.

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

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July

2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1040.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018b.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. *arXiv preprint arXiv:1909.04630*, 2019.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neubig. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*, pages 9983–9995. PMLR, 2020a.

Stéphane Aroca-Ouellette and Frank Rudzicz. On Losses for Modern Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4970–4981, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.403. URL https://www.aclweb.org/anthology/2020.emnlp-main.403.

Rich Caruana. Multitask learning. *Machine Learning*, 1997b.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2959–2968, 2019.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*. Springer, 2014.

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.00277. URL http://dx.doi.org/10.1109/CVPR.2019.00277.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020b. doi: 10.18653/v1/2020.acl-main.740. URL http://dx.doi.org/10.18653/v1/2020.acl-main.740.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in neural information processing systems*, pages 3347–3357, 2019.

Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *CoRR*, abs/1812.02224, 2018.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

Ayan Sinha, Zhao Chen, Vijay Badrinarayanan, and Andrew Rabinovich. Gradient adversarial training of neural networks. *arXiv preprint arXiv:1806.08028*, 2018.

Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Revisiting multi-task learning in the deep learning era, 2020.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv:1812.00420*, 2018.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. *arXiv preprint arXiv:1910.07104*, 2019.

Robert C Moore and William Lewis. Intelligent selection of language model training data. In *ACL*, 2010.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *EMNLP*, 2011.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V. Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *CVPR*, 2018.

Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neubig. Optimizing data usage via differentiable rewards. In *ACL*, 2020b.

Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. Learning a multi-domain curriculum for neural machine translation. In *ACL*, 2020c.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

Hongseok Namkoong and John C Duchi. Variance-based regularization with convex objectives. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2971–2980. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6890-variance-based-regularization-with-convex-objectives.pdf`.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, Jan 2010. ISSN 1095-7162. doi: 10.1137/080736417. URL `http://dx.doi.org/10.1137/080736417`.

Yuji Nakatsukasa. Accuracy of singular vectors obtained by projection-based svd methods. *BIT Numerical Mathematics*, 57(4):1137–1152, 2017.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent

neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

Amit Kumar Jaiswal, Prayag Tiwari, Sachin Kumar, Deepak Gupta, Ashish Khanna, and Joel JPC Rodrigues. Identifying pneumonia in chest x-rays: A deep learning approach. *Measurement*, 145:511–518, 2019.

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

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2015.

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

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena

Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648*, 2020.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Tong Zhang, Peng Gao, Hao Dong, Yin Zhuang, Guanqun Wang, Wei Zhang, and He Chen. Consecutive pretraining: A knowledge transfer learning strategy with relevant unlabeled data for remote sensing domain. *arXiv preprint arXiv:2207.03860*, 2022.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

Shikun Liu, Andrew J Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *arXiv preprint arXiv:1901.08933*, 2019b.

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *arXiv preprint arXiv:2007.02693*, 2020.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Nicholas Roberts, Mikhail Khodak, Tri Dao, Liam Li, Christopher Ré, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. *arXiv preprint arXiv:2103.15798*, 2021.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural

networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

Tri Dao, Nimit S Sohoni, Albert Gu, Matthew Eichhorn, Amit Blonder, Megan Leszczynski, Atri Rudra, and Christopher Ré. Kaleidoscope: An efficient, learnable representation for all structured linear maps. *arXiv preprint arXiv:2012.14966*, 2020.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. *arXiv preprint arXiv:2004.07802*, 2020.

Ilja Kuzborskij and Christoph Lampert. Data-dependent stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 2815–2824. PMLR, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676, 2019b. URL `http://arxiv.org/abs/1903.10676`.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018c. URL `http://arxiv.org/abs/1802.05365`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020b. URL `https://arxiv.org/abs/2005.14165`.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. A simple named entity extractor using adaboost. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 152–155, 2003.

Eugene Charniak. Statistical techniques for natural language parsing. *AI magazine*, 18(4):33–33, 1997.

Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Antoni B. Chan, and Rong Jin. Improved fine-tuning by leveraging pre-training data: Theory and practice. *CoRR*, abs/2111.12292, 2021. URL `https://arxiv.org/abs/2111.12292`.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017b. URL `https://arxiv.org/abs/1703.03400`.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26, 2020.

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

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019b.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

Orevaoghene Ahia, Julia Kreutzer, and Sara Hooker. The low-resource double bind: An empirical study of pruning for low-resource machine translation. *arXiv preprint arXiv:2110.03036*, 2021.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018b.

Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, et al. Non-structured dnn weight pruning—is it beneficial in any platform? *IEEE transactions on neural networks and learning systems*, 33 (9):4930–4944, 2021.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015a.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023b.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Siddhant Garg, Lijun Zhang, and Hui Guan. Structured pruning for multi-task deep neural networks, 2023.

Nakyeong Yang, Yunah Jang, Hwanhee Lee, Seohyeong Jung, and Kyomin Jung. Task-specific compression for multi-task language models using attribution-based pruning, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization, 2018.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*, 2017.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux,

Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the

dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

Skanda Vivek. The economics of large language models, Sep 2023. URL https://medium.com/emalpha/the-economics-of-large-language-models-2671985b621c.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Paul Bridger. Pytorch memory tuning, Jul 2023. URL https://paulbridger.com/posts/pytorch-memory-tuning/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017a.

Marco Ancona, Cengiz Öztireli, and Markus Gross. Shapley value as principled metric for structured network pruning. *arXiv preprint arXiv:2006.01795*, 2020.

Mintong Kang, Linyi Li, and Bo Li. Fashapley: Fast and approximated shapley based model pruning towards certifiably robust dnns. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 575–592. IEEE, 2023.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015b.

Aleksandr Dekhovich, David MJ Tax, Marcel HF Sluiter, and Miguel A Bessa. Neural network relief: a pruning algorithm based on neural activity. *arXiv preprint arXiv:2109.10795*, 2021.

Azade Nova, Hanjun Dai, and Dale Schuurmans. Gradient-free structured pruning with unlabeled data, 2023.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation via linear regression. *arXiv preprint arXiv:2012.01536*, 2020.

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13 (3):1–18, 2017.

Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance of large language models. *arXiv preprint arXiv:2402.04177*, 2024.

Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, and Wenqi Wei. Machine learning for synthetic data generation: a review. *arXiv preprint*

arXiv:2302.04062, 2023.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint arXiv:2401.16380*, 2024.

André Bauer, Simon Trapp, Michael Stenger, Robert Leppich, Samuel Kounev, Mark Leznik, Kyle Chard, and Ian Foster. Comprehensive exploration of synthetic data generation: A survey. *arXiv preprint arXiv:2401.02524*, 2024.

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

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Adhiguna Kuncoro, Yani Donchev, Rachita Chhaparia, Ionel Gog, Marc'Aurelio Ranzato, Jiajun Shen, and Arthur Szlam. Dipaco: Distributed path composition. *arXiv preprint arXiv:2403.10616*, 2024.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

Sanket Vaibhav Mehta. *Efficient Lifelong Learning in Deep Neural Networks: Optimizing Architecture, Training, and Data*. PhD thesis, University of Southern California, 2023.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pages 3082–3091. PMLR, 2018.

Phillip I Good. Permutation, parametric and bootstrap tests of hypotheses: a practical guide to resampling methods for testing hypotheses. 2005.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, 2018.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1003. URL `https://aclanthology.org/S16-1003`.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Cor-

ney, Benno Stein, and Martin Potthast. SemEval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2145. URL https://aclanthology.org/S19-2145.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017b. URL https://arxiv.org/abs/1711.05101.

Maurice George Kendall. Rank correlation methods. 1948.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.