AANG: AUTOMATING AUXILIARY LEARNING

Lucio M. Dery1,∗ Paul Michel2 Mikhail Khodak1 Graham Neubig1 Ameet Talwalkar1,3

1 Carnegie Mellon University 2 ENS PSL University 3 Hewlett Packard Enterprise

ABSTRACT

Auxiliary objectives, supplementary learning signals that are introduced to help aid learning on data-starved or highly complex end-tasks, are commonplace in machine learning. 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. Intuition for how and when these objectives improve end-task performance has also had limited theoretical backing. In this work, 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 theoretically formalize widely-held intuitions about how auxiliary learning improves generalization on the end-task. This leads us to a principled and efficient algorithm for searching the space of generated objectives to find those most useful to a specified end-task. With natural language processing (NLP) as our domain of study, we demonstrate that our automated auxiliary learning pipeline leads to strong improvements over competitive baselines across continued training experiments on a pre-trained model on 5 NLP tasks.

1 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., 2019b) 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). Auxiliary objectives are constructed by hand-design and without much overarching structure, relying on the experience and intuition of a select group 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, this paper 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 (D) → Input Transformation (T) →

1 Correspondence to : ldery@andrew.cmu.edu
2 Code available at : https://github.com/ldery/Automating-Auxiliary-Learning

Figure 1: We present the decomposition of some auxiliary objectives in NLP within our framework.
Model Representation \((R) \rightarrow \text{Output} \ (O)\). For instance, in RL, a common auxiliary objective is to predict the environment’s forward dynamics \([\text{Agrawal et al.} \ 2016, \ \text{Hafer et al.} \ 2019]\). To construct this objective, the current task state-action pair \((D)\) is corrupted \((T)\) and then passed through the model to produce a latent representation \((R)\) which is finally used to predict the next state \((O)\). Similarly, in NLP, the XLNet \([\text{Yang et al.} \ 2019]\) objective—which performs language modeling on a randomly factorized permutation of the input—can be written within our taxonomy as \(\{D = \text{Out-of-Domain}, T = \text{No-op}, R = \text{Random-Factorized}, O = \text{Next Token}\}\). These two examples (along with others listed in Figure 1) fall within a class we term named objectives: objectives that have been previously proposed in the auxiliary learning literature.

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 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 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 \([\text{Hardt et al.} \ 2016, \ \text{Bousquet & Elisseeff} \ 2002]\), we derive end-task generalization error bounds that are dependent on the choice of auxiliary task. This contributes to existing theory \([\text{Saunshi et al.} \ 2020, \ \text{Xie et al.} \ 2021]\) 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 \(\text{Bousquet & Elisseeff} \ 2002\), which in theory improves generalization of the final model.

Guided by our theory, we introduce AANG (Automating Auxilliary LearnNG), 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 \([\text{Gururangan et al.} \ 2020, \ \text{Aghajanyan et al.} \ 2021, \ \text{Dery et al.} \ 2021b, \ \text{Zhang et al.} \ 2022]\), 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 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), and in our empirical evaluation (Section 6), our theoretical results (Section 4) and AANG itself are domain-agnostic.

---

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


2 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, 1997; 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., 2020) as well as end-task aware multitasking approaches (Lin et al., 2019; Dery et al., 2021a;b). Whilst auxiliary objectives may be meta-learned (Liu et al., 2019a; 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 paper.

This work bears many parallels to the area of neural architecture search (NAS) (Stanley & Miikkulainen, 2002; Zoph & 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 & 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., 2018), 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 & 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 & 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 & Lampert, 2018).

3 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, 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., 2019) or the task data itself (Gururangan et al., 2020). It may even include data outside the modality of the end-task.

2. Input-Transformation, 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., 2019).

3. Representation, 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., 2018). It could also depend on the representations of a randomly selected permutation of other tokens (Random Factorized) (Yang et al., 2019).

4. Output, 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 \( \{D \rightarrow T \rightarrow R \rightarrow O\} \) is expansive enough to cover a range of named auxiliary objectives of interest in NLP (Figure 1). For example, we can write any member of the GPT series (Radford et al., 2018; 2019; Brown et al., 2020) which perform left-to-right language modelling on out-of-domain data as \( \{D = \text{Out-of-Domain}, T = \text{No-op}, R = \text{Left-To-Right}, O = \text{Next Token}\} \). 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 1 to the unique set \( R = \{\text{Bi-directional, Left-To-Right, Right-To-Left, Random-Factorized}\} \).

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 2). In general, given an instance of our taxonomy, we can construct a space of objectives \( A = D \times T \times R \times O \) of size \( |A| \leq |D| \times |T| \times |R| \times |O| \). Consider new \( \text{Obj}_{1} \) from Figure 2. This previously unexplored objective can be obtained by combining the special masking operation from BERT (\( BERT-\text{Op} \)) 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 3).

Our framework also allows us to reason about whole families of objectives, \( F \), by thinking in terms of design stages and choices. For example, given a particular end-task \( E \) with input text \( E_{D} \), 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 \( F_{D=E_{D}} \). \( F_{D=E_{D}} \) not only includes pre-existing TAPT Gururangan et al. (2020) but also unexplored objectives like task-data dependent variants of XLNET, ELMO etc. Auxiliary learning with \( F_{D=E_{D}} \) 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 \( \{T, R, O\} \). We can also reason about expanding particular stages to include new primitives. Any supervised loss can be added to the output stage, \( O \), 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 \( O \) to the end-task supervised output \( E_{O} \). This leads to \( F_{D=E_{D}}^{O=E_{O}} \) which is a subset of \( F_{D=E_{D}}^{O=E_{O}} \). \( F_{D=E_{D}}^{O=E_{O}} \) includes many objectives like predicting the end-task signal from corrupted input data. In Section 6 we will introduce a search space of objectives that leverages task augmentation.

4 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 3.

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 \( \|\nabla f_{E}(\theta) - \nabla f_{A}(\theta)\| \leq \Delta \forall \theta \).

Note that \( \theta \) represents all the parameters of the model. Smaller \( \Delta \) 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 \( \Delta \).

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., 2021a,b; 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

\(^{3}\text{Although this taxonomy is quite expansive, it obviously does not consider other elements of objective creation such as choice of model architecture, optimizer settings, etc.}\)
Algorithm 1 AANG

As a detailed inspection of the proof will show, we derive Equation 1 by appealing to algorithmic β
Here λ also like to set λe such that the bound on εgen is minimized. Whilst a closed form exists for the optimal weightings λe, (w^k), it depends on variables like {Δ^k}, {β^k_λ}, L that are hard to estimate.

5 End-task Aware Search of Structured Objective Spaces

Guided by Section 4 we build a practical method for exploring a set of objectives, A.

While the dynamic sampling setting described in Section 4 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 L_total = λ_e L_E + (1 − λ_e) L_K. Here, K is a chosen subset of A.

Second, as indicated in Section 4 given K, we can write the set as a single objective f_a = \sum_{k \in K} w^k f_a^k. By Prescription (P_1), we want to choose \{w^k\} such that f_e has a small Δ with the end-task f_e. We would also like to set λ_e such that the bound on εgen is minimized. While a closed form exists for the optimal weightings λ_e, (w^k), it depends on variables like {Δ^k}, {β^k_λ}, L that are hard to estimate.

\[\epsilon_{gen} \lesssim \left(\Delta\right)^{\frac{1}{1+\lambda}} \left(\frac{r e^{\gamma T}}{N^*}\right)^{1-\lambda} \text{ Where } \gamma = \frac{\lambda_{e}}{r_{e}}\]  

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

Proof. See Appendix E for full proof and Appendix F for more discussion

As a detailed inspection of the proof will show, we derive Equation 1 by appealing to algorithmic stability [Bousquet & Elisseeff, 2002] [Hardt et al., 2016] [Kuzborskij & Lampert, 2018] (Section 2). 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 1 surfaces the following prescriptions about learning with auxiliary tasks:

(P_1) Smaller Δ improves εgen. 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}\]. Since we usually have a fixed amount of task data N_e, we can increase N^* by adding more auxiliary data N_a.

\[\text{Algorithm 1 AANG}\]

\textbf{Input:} Search Space - \(\mathcal{A}\)
Factor vectors - \{\(W_{\text{All}}^w, W_{T}^w, W_{R}^w, W_{O}^w\}\)
End-task - \(E\), End-task weight - \(\lambda_e\)
Initial Model Params - \(\theta_0 \in \mathbb{R}^D\)

\textbf{repeat}
\quad Sample a batch of \(n\) objectives \(\mathcal{K}^n\) \(\text{Weighting of objectives in } \mathcal{K}^n\)
\quad Construct \(w_n\)
\quad for \(k = 1\) to \(n\) do
\quad \quad \(\hat{a} = \lfloor K^k\rfloor\) \(\text{stages}\)
\quad \quad \(w^k \propto \exp\left(W_{\text{All}}^w (d, t, r, o) + W_{T}^w + W_{R}^w + W_{O}^w\right)\)
\quad \quad \(w^k_n \leftarrow w^k\)
\quad end for
\quad Get losses from batches of data \(\hat{L}_A(\mathcal{K}^n, w_n) = \sum_{k=1}^{n} w^k L_k\)
\quad \(\hat{L}_{\text{total}} = \lambda_e L_E + (1 - \lambda_e) L_K\)
\quad Get gradients and update factors \(\theta_{t+1}, \{\nabla_w^w\}_{w_n} \leftarrow \text{META-TARTAN}(\theta_t, E, \hat{L}_{\text{total}})\)
\quad Update \(\{W_{\text{All}}^w, W_{T}^w, W_{R}^w, W_{O}^w\}\) using \(\nabla_w^w\)
\quad Update \(\lambda_e\) using \(\nabla_\lambda\)
\until done

\textbf{Return:} \(\theta_T\)
We focus on continued pre-training (Gururangan et al., 2020; 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., 2019b) as 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 (Qi et al., 2020; Du et al., 2020).

Model Details and Datasets: We use a pre-trained RoBERTa$_{base}$ (Liu et al., 2019b) 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$_{base}$. For sequence generation objectives, we make a copy of the pre-trained output layer of RoBERTa$_{base}$ for each task. Table 4 in Appendix C 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., 2020; Dery et al., 2021b).

**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., 2019b):** We simply finetune a pre-trained RoBERTabase on the end-task.
2. **TAPT (Gururangan et al., 2020):** Continue training RoBERTabase 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., 2021b) 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. \( \{ \mathcal{I} = \text{End-task data}, \mathcal{T} = \text{No-op}, \mathcal{R} = \text{Left-To-Right}, \mathcal{O} = \text{Denoise Token} \} \).
2. **XLNET-style:** This is a denoising auxiliary objective that uses randomized masking for computing representations. \( \{ \mathcal{I} = \text{End-task data}, \mathcal{T} = \text{No-op}, \mathcal{R} = \text{Random-factorized}, \mathcal{O} = \text{Denoise Token} \} \).
3. **BERT-style / TAPT:** Denoising inputs corrupted via BERT-Op: 80% masking and 10% random replacement. \( \{ \mathcal{I} = \text{End-task data}, \mathcal{T} = \text{BERT-Op}, \mathcal{R} = \text{Bi-directional}, \mathcal{O} = \text{Denoise Token} \} \). Please note that this baseline is equivalent to META-TARTAN as introduced in Dery et al. (2021b).

Table 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 \( \{ \mathcal{I} = \text{End-task} \} \) in AANG-TD+ED, are instantiations of task augmentation as introduced in Section 3.

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

Table 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.

### Training Details
Please see Appendix D for more details about hyper-parameter configurations.

## 7 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.

### 7.1 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 2 shows that automatically generating auxiliary objectives from only task data and using them appropriately is productive.

**End-task awareness is key:** From Table 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 2 shows that multitraining 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 \( P_2 \) since we are increasing the effective amount of auxiliary data. We further posit that introducing more auxiliary objectives also serves to implicitly regularize the end-task during training.
Table 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$_{3.55}$ | 77.92$_{2.36}$ | 82.10$_{3.90}$ | 93.39$_{2.36}$ | 70.37$_{1.15}$ | 77.97 |
| TAPT   | 1        | 67.74$_{1.69}$ | 79.53$_{1.93}$ | 82.17$_{2.65}$ | 93.42$_{2.67}$ | 70.74$_{1.21}$ | 78.72 |
| [OURS] Static Multitask-TD | 24 | 69.68$_{2.81}$ | 83.37$_{2.56}$ | 83.42$_{3.36}$ | 97.90$_{1.73}$ | 71.02$_{1.43}$ | 81.07 |
| Yes    | X. GP-sty | 1        | 67.22$_{3.44}$ | 81.62$_{2.44}$ | 83.29$_{3.21}$ | 94.46$_{3.73}$ | 70.67$_{1.46}$ | 79.84 |
|        | Y. XLMER-sty | 1 | 69.76$_{4.42}$ | 81.81$_{4.42}$ | 83.35$_{3.31}$ | 94.61$_{3.62}$ | 71.18$_{1.58}$ | 80.51 |
|        | Z. BERT-style (Dery et al 2021b) | 1 | 70.08$_{4.70}$ | 81.48$_{6.82}$ | 84.49$_{5.60}$ | 96.84$_{1.72}$ | 72.70$_{1.60}$ | 81.12 |

| [OURS] AANG-{X+Y+Z} | 7 | 51.51$_{4.19}$ | 82.89$_{1.78}$ | 83.68$_{0.28}$ | 96.92$_{1.28}$ | 71.04$_{1.24}$ | 72.75$_{1.34}$ | 81.55 |
| [OURS] AANG-TD | 24 | 73.26$_{1.32}$ | 82.99$_{1.32}$ | 83.91$_{0.32}$ | 98.46$_{0.6}$ | 72.40$_{1.60}$ | 82.21 |

7.2 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 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. (2021b). However, AANG-TD+ED (73.70) does not significantly improve over AANG-TD (73.26) on the ACL-ARC task (Figure 3). This might seem at odds with Prescription (P9), 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 $\Delta$ as suggested by Prescription (P9). 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.

7.3 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 2?). Each sampled tasks weight is initialized to $\frac{1}{n}$ and this remains unchanged throughout training. This is the Static Multitask-TD baseline in Table 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 (P7)). 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 3 (Appendix A) 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 2 the fact that AANG-TD roughly matches the best single task performance (72.40$_{1.65}$ versus 72.77$_{1.60}$ for BERT-style) on the SE-2016-6 task suggests that it may be learning to mostly up-weight this task. Figure 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 4 also shows that AANG can adapt to overfitting. The vertical black lines indicate the point of best validation set performance. AANG responds to
Tasks with highest average weight during first 10% of training

| Task | Left-To-Right | None | Mask | Bert-op | None | Replace | Rand-Fact | None | Rand-Fact |
|------|----------------|------|------|---------|------|---------|-----------|------|-----------|
| O    | DENOISE        | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise |
| O    | SE-2016-6     | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 |

Tasks with highest average weight during later half of training

| Task | Left-To-Right | None | Mask | Bert-op | None | Replace | Rand-Fact | None | Rand-Fact |
|------|----------------|------|------|---------|------|---------|-----------|------|-----------|
| O    | DENOISE        | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise | Denoise |
| O    | SE-2016-6     | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 | SE-2016-6 |

Figure 4: Learned trajectories for AANG-TD for run instances of SE-2016-6 and SCIERC tasks.

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 O = {DENOISE} are highly weighted but later, objectives based on supervised signal, O = {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 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.

8 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 O(n) forward-backward operations for n sampled tasks compared to O(1) for static multitasking. In Table 2, we show that our static Multitask-TD method outperforms all other non-task-adaptive methods by ≈ 2.4% and is thus a viable alternative when runtime is a significant constraint. Secondly, AANG as presented is an approximate algorithm – primarily due to sub-sampling the space of tasks. Thus as mentioned in Section 7.2, 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 paper 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 generated objectives in an end-task aware multitask fashion. Our experiments show that AANG is a promising first step in automating auxiliary learning.
9 ACKNOWLEDGEMENTS

This work was supported in part by DSO National Laboratories, an ENS-CFM Data Science Chair, DARPA FA875017C0141, the National Science Foundation grants IIS1705121, IIS1838017, IIS2046613 and IIS-2112471, an Amazon Web Services Award, a Facebook Faculty Research Award, funding from Booz Allen Hamilton Inc., and a Block Center Grant. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of these funding agencies. We are grateful for helpful feedback from Uri Alon, Patrick Fernandes, Joon Sik Kim, Han Guo, Victor Akinwande and Clara Na.

REFERENCES

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.

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.

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

Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. CoRR, abs/1903.10676, 2019. URL http://arxiv.org/abs/1903.10676.

Olivier Bousquet and André Elisseeff. Stability and generalization. The Journal of Machine Learning Research, 2:499–526, 2002.

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, 2020. URL https://arxiv.org/abs/2005.14165.

Xavier Carreras, Lluís Márquez, and Lluís Padró. A simple named entity extractor using adaboost. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003, pp. 152–155, 2003.

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

Eugene Charniak. Statistical techniques for natural language parsing. AI magazine, 18(4):33–33, 1997.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pp. 1597–1607. PMLR, 2020.

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.

Tri Dao, Nimit S Sohoni, Albert Gu, Matthew Eichhorn, Amit Blonder, Megan Lesczczynski, Atri Rudra, and Christopher Ré. Kaleidoscope: An efficient, learnable representation for all structured linear maps. arXiv preprint arXiv:2012.14966, 2020.

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

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, 2021b.
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.

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, pp. 1761–1770, 2019.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

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.

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, 2020.

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, pp. 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, pp. 1225–1234. PMLR, 2016.

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, pp. 1314–1324, 2019.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? arXiv preprint arXiv:1608.08614, 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.

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.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. SemEval-2019 task 4: Hyperpartisan news detection. In Proceedings of the 13th International Workshop on Semantic Evaluation, pp. 829–839, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2145. URL https://aclanthology.org/S19-2145.

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

Ilja Kuzborskij and Christoph Lampert. Data-dependent stability of stochastic gradient descent. In International Conference on Machine Learning, pp. 2815–2824. PMLR, 2018.

Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. arXiv preprint arXiv:2004.07802, 2020.

Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. Advances in neural information processing systems, 32, 2019.
Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Shikun Liu, Andrew J Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *arXiv preprint arXiv:1901.08933*, 2019a.

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*, 2019b.

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](https://arxiv.org/abs/2111.12292).

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL [https://arxiv.org/abs/1711.05101](https://arxiv.org/abs/1711.05101).

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.

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)*, pp. 31–41, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1003. URL [https://aclanthology.org/S16-1003](https://aclanthology.org/S16-1003).

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *arXiv preprint arXiv:2007.02693*, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 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, 2018. URL [http://arxiv.org/abs/1802.05365](http://arxiv.org/abs/1802.05365).

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104. PMLR, 2018.

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*, pp. 1–26, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 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.

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.

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.

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*, pp. 15–18, 2019.
Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648*, 2020.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

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.

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.

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.

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.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
A  MORE ABLATION TABLES

Table 3: Varying number of sampled objectives per-iteration.

| Task       | $\frac{3}{2}$ tasks | $\frac{6}{2}$ tasks |
|------------|----------------------|---------------------|
| ACL-ARC    | 72.11, 12            | 73.26, 32           |
| SCIERC     | 82.35, 17             | 82.98, 52           |
| SE-2016-6  | 72.46, 65            | 72.46, 90           |
| CHEMPROT   | 83.91, 32            | 83.99, 98           |
| H.PARTISAN | 98.46, 0              | 97.95, 73           |

B  DISCUSSION OF META-TARTAN (DERY ET AL., 2021B)

META-TARTAN (Dery et al., 2021b) is a MAML style (Finn et al., 2017) meta-learning algorithm that learns to adaptively weight a given set of tasks based on their influence on the end-task validation performance. META-TARTAN achieves this by formulating the following bi-level optimization problem:

$$\theta^*, w^* = \underset{\theta \in g(\theta_0)}{\arg\min} \{ \theta \in \mathcal{L}_E(\theta) \}$$

where

$$\theta_0 = \underset{\theta}{\arg\min} \mathcal{L}_{total}(\theta, w) = \underset{\theta}{\arg\min} \left( \mathcal{L}_E(\theta) + \sum_{T_i \in A} w_i \mathcal{L}_{T_i}(\theta) \right)$$

Note that $E$ is the end-task and $A$ is the set of auxiliary tasks.

Since the above bi-level problem is difficult to solve directly, Dery et al. (2021a) relax the problem and into an alternating optimization problem where task weights are updated based on 1-step improvement to the validation performance of the end-task:

$$\partial \mathcal{L}_{val}^E(\theta_t+1(w)) \approx -\beta (\nabla \mathcal{L}_{T_i}(\theta_t))^T (\nabla \mathcal{L}_{val}^E(\theta_t))$$

To prevent the above relaxation from finding the trivial solution of just upweighting solely the end-task, Dery et al. (2021b) introduce a special dev-head which they use for estimating the meta-gradient:

$$\partial \mathcal{L}_{val}^E(\theta^*(w)) \approx -\beta (\nabla \mathcal{L}_{T_i}(\theta^*))^T (\nabla \mathcal{L}_{val}^E([\theta_{body}; \phi^*]_t))$$

Where $\phi^*_t$ is the special dev-head and $\theta_{body}$ is the body of the model. For even more details about META-TARTAN, please see Section 3 of Dery et al. (2021b).

Though we leverage MET-TARTAN, compared to Dery et al. (2021b), we make three distinct contributions to the field of auxiliary learning. We list them below

1. **Novel Problem Formulation**: As far as we are aware of, we are the first to formulate the problem of automated auxiliary learning. Specifically, we presented an approach for automatically constructing a suite of auxiliary objectives based on existing objectives. Please note that Dery et al. (2021b) perform auxiliary learning with only the DAPT/TAPT variants of the BERT objective. They effectively assume that the search space of objectives (the 2 they explore) is given before-hand. Our approach automatically creates the search space.

2. **Theoretical Novelty**: To the best of our knowledge, we are the first work to provide an exploration of why auxiliary learning improves primary task performance via algorithmic stability. Dery et al. (2021b) in introducing META-TARTAN do not attempt to give a theoretical characterization of why the algorithm improves end-task performance.

3. **Algorithm Improvements to META-TARTAN**: Please note that META-TARAN as presented in Dery et al. (2021b) was used with only 2 auxiliary tasks. When scaling to more tasks, using META-TARTAN naively becomes computationally prohibitive. Specifically, on a search space of $N$ tasks, META-TARTAN requires $O(N)$ order computation per step.
We improve upon this by introducing the task sub-sampling of \((k \ll N)\) which reduces the compute overhead to \(O(k)\). To account for the impact of sub-sampling as an approximation, we introduced the factorised modelling of task weights which allows sharing of information between auxiliary tasks that might themselves be related.

### C DATASET DETAILS

Table 4: Specifications of datasets used to evaluate our methods.

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

### D 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 & Hutter, 2017], with weight decay of 0.01 for all experiments.

Table 5: AANG-TD specific Hyper-parameters

| Hyper-parameter | Values | Description |
|-----------------|--------|-------------|
| aux\_lr         | 1.0, 0.1 | Learning rate for factor vectors - \(\{W^A_{\text{All}}, W^Z, W^T, W^R, W^O\}\) |
| sapt\_lr        | 0.1, 0.01 | Learning rate for primary task weighting \(\lambda_e\) |
| nconf\_subsamp  | 3, 6   | Number of sub-sampled auxiliary tasks. |
| learning rate   | 1e-3, 1e-4 | Learning rate used for further training of RoBERTa\_base |
| aux\_bsz        | 256    | Batch size of for auxiliary objectives |

Table 6: AANG-TD+ED specific Hyper-parameters

| Hyper-parameter | Values | Description |
|-----------------|--------|-------------|
| aux\_lr         | 1.0, 0.5, 0.1 | Learning rate for factor vectors - \(\{W^A_{\text{All}}, W^Z, W^T, W^R, W^O\}\) |
| sapt\_lr        | 0.1    | Learning rate for primary task weighting \(\lambda_e\) |
| nconf\_subsamp  | 6, 12, 24 | Number of sub-sampled auxiliary tasks. |
| learning rate   | 1e-4   | Learning rate used for further training of RoBERTa\_base |
| aux\_bsz        | 1024   | Batch size of for auxiliary objectives |

Table 7: META-TARTAN Hyper-parameters for single task auxiliary tasks

| Hyper-parameter | Values | Description |
|-----------------|--------|-------------|
| sapt\_lr        | 1.0, 0.1, 0.01 | Learning rate for primary task weighting \(\lambda_e\) |
| learning rate   | 1e-3, 1e-4, 5e-5 | Learning rate used for further training of RoBERTa\_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., 2021b] when available. We use the hyper-parameters specified for TAPT in [Gururangan et al., 2020] 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.

E GENERALIZATION ERROR BOUND FOR END-TASK AWARE TRAINING

E.1 Definitions

Definition E.1. A function, \( f : \Omega \rightarrow \mathbb{R} \) is \( L \)-Lipschitz if \( \forall u, v \in \text{dom}(f) \):

\[
\| f(u) - f(v) \| \leq L \| u - v \|
\]

Note that \( L \)-Lipschitz implies bounded gradients.

\[
\| \nabla f(w) \| \leq L \quad \forall w
\]

Definition E.2. A function, \( f : \Omega \rightarrow \mathbb{R} \) is \( \beta \)-smooth if \( \forall u, v \in \Omega \):

\[
\| \nabla f(u) - \nabla f(v) \| \leq \beta \| u - v \|
\]

Definition E.3. An update rule, \( G \) is \( \sigma \)-bounded if :

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

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

Definition E.4. Population Risk

\[
R[w] = \mathbb{E}_{z \sim D_e} f_e(w; z)
\]

Definition E.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 E.6. Generalization Error \( \epsilon_{gen}(A, N_e) \)

\[
\epsilon_{gen}(A, N_e) = \mathbb{E}_{S,A} \left[ R_S[A(S)] - R[A(S)] \right]
\]

Definition E.7. Uniform Stability

A randomized algorithm \( A \) is \( \epsilon \)-uniformly stable if for all data sets \( S, S' \in Z, \|S\| = \|S'\| = N_e \) such that \( S \) and \( S' \) differ in at most one example, we have

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

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

E.2 Relevant Theorems

Theorem E.1 (Uniform Stability implies Generalization in expectation). Let Algorithm \( A \) be \( \epsilon \)-uniformly stable. Then,

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

For full proof see Theorem 2.2 of [Hardt et al., 2016].
**Theorem E.2** (Stochastic Gradient Method is stable). Assume that \( f_c(: ; z) \in [0, 1] \) is an \( L \)-Lipschitz and \( \beta_c \)-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:

\[
\epsilon_{sgm} \leq \frac{1 + \frac{1}{q}}{N_e - 1} \left( 2cL^2 \right)^{\frac{1}{q+1}} \frac{T^{q+1}}{c\beta_c + 1}
\]

where \( q = \beta_c c \).

We can simplify this to only terms involving \( T \) and \( N_e \)

\[
\epsilon_{sgm} \leq \frac{T^{1 - \frac{1}{c\beta_c + 1}}}{N_e} \tag{6}
\]

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

\[ \square \]

### E.3 Growth Functions

**Lemma E.3** (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 \( \Omega \) given that \( f : \Omega \to \mathbb{R} \) and define

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

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:

\[
\delta_0 = 0 \\
\delta_{t+1} \leq \min \left\{ (1 + \alpha \lambda_1 \beta_1) \delta_t + \alpha \lambda_2 (\Delta + 2L), \frac{1}{\delta_t + 2\sigma_t} \right\} \quad G_{f_t} = G'_{f_t} \\
G_{f_t}, G'_{f_t} \text{ are } \sigma \text{-bounded}
\]

Note that \( \mathcal{P} \) 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 \( \beta_1, \beta_2 \) respectively.
Proof. The second bound on $\delta_t$ is taken directly from Lemma 2.5 of [Hardt et al. (2016)]. We now derive the first-half of the first bound:

$$
\delta_{t+1} = E_{f_1 \ldots f_{t+1} \sim P_{\lambda}} [\|w_{t+1} - w_{t+1}'\|] \\
= E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|G_{f_1}(w_t) - G'_{f_1}(w_t')\| + \lambda_2 \|G_{f_2}(w_t) - G'_{f_2}(w_t')\| \right] \\
= E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|w_t - \alpha \nabla f_1(w_t) - w_t' + \alpha \nabla f_1(w_t')\| + \lambda_2 \|w_t - \alpha \nabla f_2(w_t) - w_t' + \alpha \nabla f_2(w_t')\| \right] \\
\leq E_{f_1 \ldots f_{t} \sim P_{\lambda}} [\|w_t - w_t'\|] + \alpha E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|\nabla f_1(w_t') - \nabla f_1(w_t)\| + \lambda_2 \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \right] \\
\text{(Triangle Inequality used for above step)} \\
= \delta_t + \alpha E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|\nabla f_1(w_t') - \nabla f_1(w_t)\| + \lambda_2 \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \right] \\
\text{(Without Loss of Generality, let $\beta_1 \leq \beta_2$)} \\
\leq \delta_t + \alpha E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \beta_1 \|w_t - w_t'\| + \lambda_2 \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \right] \quad \text{(Smoothness)} \\
= \delta_t + \alpha \lambda_1 \beta_1 \delta_t + \alpha \lambda_2 \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \\
\text{(Triangle Inequality)} \\
= (1 + \alpha \lambda_1 \beta_1) \delta_t + \alpha \lambda_2 \left( \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \right) \quad \text{(Triangle Inequality)} \\
\leq (1 + \alpha \lambda_1 \beta_1) \delta_t + \alpha \lambda_2 \left( \Delta + \|\nabla f_1(w_t') - \nabla f_2(w_t)\| \right) \quad \text{Using Assumption A.1} \\
\leq (1 + \alpha \lambda_1 \beta_1) \delta_t + \alpha \lambda_2 \left( \Delta + \|\nabla f_1(w_t')\| + \|\nabla f_2(w_t)\| \right) \quad \text{Triangle Inequality} \\
\leq (1 + \alpha \lambda_1 \beta_1) \delta_t + \alpha \lambda_2 (\Delta + 2L) \\
\text{L-Lipschitz function}
$$

To obtain the second half of the first bound:

$$
\delta_{t+1} = E_{f_1 \ldots f_{t+1} \sim P_{\lambda}} [\|w_{t+1} - w_{t+1}'\|] \\
= E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|G_{f_1}(w_t) - G'_{f_1}(w_t')\| + \lambda_2 \|G_{f_2}(w_t) - G'_{f_2}(w_t')\| \right] \\
= E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|w_t - \alpha \nabla f_1(w_t) - w_t' + \alpha \nabla f_1(w_t')\| + \lambda_2 \|w_t - \alpha \nabla f_2(w_t) - w_t' + \alpha \nabla f_2(w_t')\| \right] \\
\leq E_{f_1 \ldots f_{t} \sim P_{\lambda}} [\|w_t - w_t'\|] + \alpha E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \|\nabla f_1(w_t') - \nabla f_1(w_t)\| + \lambda_2 \|\nabla f_2(w_t') - \nabla f_2(w_t)\| \right] \\
\text{(Triangle Inequality used for above step)} \\
\leq \delta_t + \alpha E_{f_1 \ldots f_{t} \sim P_{\lambda}} \left[ \lambda_1 \beta_1 \|w_t - w_t'\| + \lambda_2 \beta_2 \|w_t - w_t'\| \right] \quad \text{(Smoothness)} \\
= \delta_t + \alpha \lambda_1 \beta_1 E_{f_1 \ldots f_{t} \sim P_{\lambda}} [\|w_t - w_t'\|] + \alpha \lambda_2 \beta_2 E_{f_1 \ldots f_{t} \sim P_{\lambda}} [\|w_t - w_t'\|] \\
= \delta_t + \alpha (\lambda_1 \beta_1 + \lambda_2 \beta_2) \delta_t \\
= (1 + \alpha (\lambda_1 \beta_1 + \lambda_2 \beta_2)) \delta_t
$$

$\square$

### E.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 $S_a = (z_1, \ldots, z_{N_a})$ from the distribution $D_a$. 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 $\lambda_e, \lambda_a | \lambda_e + \lambda_a = 1$. 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 $P_A$. $S_A'$ 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 E.4 (Stability of dynamic sampling). We denote the outputs of $T$ steps of SGD 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:

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

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

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

$$
|E[f_e(w_T; z) - f_e(w_T'; z)]| = P\{E\} |E[f_e(w_T; z) - f_e(w_T'; z)]|E\} + P\{E'\} |E[f_e(w_T; z) - f_e(w_T'; z)]|E'\}
\leq |E[f_e(w_T; z) - f_e(w_T'; z)]|E\} + P\{E'\} \sup_{w, z_e} f_e(w; z_e)
$$

because $f_e$ is non-negative

$$
\leq LE[\|w_T - w_T'\|] + P\{E'\} \sup_{w, z_e} f_e(w; z_e)
$$

because $f_e$ is L-Lipschitz

We now proceed to bound $P\{E'\}$. 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 step time in which SGD uses the example $z_{i_*}^t$. 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.

$$
P\{E'\} = P\{\delta_0 \neq 0\} \leq P\{I \leq t_0\}
$$

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

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

Theorem E.5 (Stability Bound on Dynamic Sampling). Assume that $f_a(\cdot; z_e), f_a(\cdot; z_a) \in [0, 1]$ are L-Lipschitz and $\beta_e$ and $\beta_a$-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 SGD for $T$ steps with monotonically non-increasing step sizes $\alpha_t \leq \frac{\gamma}{T}$ by dynamically sampling the tasks according to $\lambda_e$ and $\lambda_a$. Then, with respect to $f_e$, SGD has uniform stability with:

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

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

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

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

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

or

$$
(\tilde{\beta}, \rho)_2 = (\lambda_e\beta_e + \lambda_a\beta_a, 0)
$$
When $(\bar{\beta}, \rho)_1$ gives the tighter bound, we can simplify to:

$$
\epsilon_{gen} \lesssim (\Delta)^{-\frac{1}{T+\lambda^*\beta^*}} \left( \frac{\gamma T}{N'} \right)^{1-\frac{1}{\alpha t + \bar{\beta} + \rho}}
$$

As presented in Section 4.

Proof. Let $S_A, S'_A$ be two samples of size $N' = N_e + N_a$ as described in lemma E.4. 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 E.4, we have:

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

(8)

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

- At any step $t$, with probability $(1 - \frac{\beta}{N'})$, 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 E.4. This gives:

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

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

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

$$
\bar{\beta} = \{ \lambda^* \beta^*, \lambda_e \beta_e + \lambda_a \beta_a \} \text{ and } \rho = \{(1 - \lambda^*) (\Delta + 2L), 0\}. \text{ We can present the final bound in terms of these variables which can be substituted depending on the minimizer.}
$$

- With probability $\frac{\beta}{N'}$ 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 E.3 since $f_e$ is $L$-Lipschitz.

Combining the above we have:

$$
\Psi_{t+1} \leq \left(1 - \frac{\gamma}{N'}\right) \left(1 + \alpha_t \bar{\beta}\right) \Psi_t + \alpha_t \rho + \frac{\gamma}{N'} \left(\Psi_t + 2\alpha_t L\right)
$$

$$
= \left(\frac{\gamma}{N'} + (1 - \frac{\gamma}{N'}) (1 + \alpha_t \bar{\beta})\right) \Psi_t + \frac{2\gamma \alpha_t L}{N'} + \alpha_t (1 - \frac{\gamma}{N'}) \rho
$$

$$
= \left(1 + (1 - \frac{\gamma}{N'}) \alpha_t \bar{\beta}\right) \Psi_t + \frac{\alpha_t (2\gamma L + (N' - \gamma) \rho)}{N'}
$$

$$
\leq \exp \left(1 - \frac{\gamma}{N'} \frac{c \bar{\beta}}{L}\right) \Psi_t + \frac{c (2\gamma L + (N' - \gamma) \rho)}{t N'}
$$

(9)

We use $1 + x \leq \exp(x) \forall x$

$$
\leq \exp \left(1 - \frac{\gamma}{N'} \frac{c \bar{\beta}}{L}\right) \Psi_t + \frac{c \rho}{t N'}
$$

Where $\bar{\rho} = (2\gamma L + (N' - \gamma) \rho)$
We can unwind the recurrence until $\Psi_{t_0} = 0$.

$$
\Psi_T \leq \sum_{t=t_0+1}^{T} \left( \prod_{k=t+1}^{T} \exp \left( (1 - \gamma \frac{c\beta_k}{N'}) \frac{c\rho}{tN'} \right) \right) \left( \frac{c\rho}{tN'} \right)
$$

$$
= \sum_{t=t_0+1}^{T} \left( \frac{c\rho}{tN'} \right) \exp \left( (1 - \gamma \frac{N'}{N'}) c\beta \sum_{k=t+1}^{T} \frac{1}{k} \right)
$$

$$
\leq \sum_{t=t_0+1}^{T} \left( \frac{c\rho}{tN'} \right) \exp \left( (1 - \gamma \frac{N'}{N'}) c\beta \log \left( \frac{T}{t} \right) \right)
$$

$$
= \frac{c\rho}{N'} c\beta \frac{(1 - \frac{1}{N'})}{\sum_{t=t_0+1}^{T} t} c\beta (1 - \frac{1}{N'})^{-1}
$$

Plugging this bound back into Equation 8 and using the fact that $f_e \in [0, 1]$:

$$
E \left| f_e(w_T; z) - f_e(w'_T; z) \right| \leq \gamma t_0 + \frac{L\rho}{\beta(N' - \gamma)} \left( \frac{T}{t_0} \right) c\beta
$$

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

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

Plugging this in gives us:

$$
E \left| f_e(w_T; z) - f_e(w'_T; z) \right| \leq \left( 1 + \frac{1}{c\beta} \right) \left( \frac{N'Lc(2\gamma L + (N' - \gamma)\rho)}{(N' - \gamma)} \right)^{\frac{1}{\gamma^{s^*} + 1}} \left( \frac{T}{t_0} \right)^{c\beta (\gamma T + 1)} + \frac{2\gamma L^2 c}{N' - \gamma} + \rho Lc \left( \frac{T}{N'} \right)^{\frac{1}{\gamma^{s^*} + 1}}
$$

(12)

Recall that:

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

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

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

F DISCUSSION OF GENERALIZATION ERROR BOUNDS

F.1 WHAT DOES THEOREM E.5 SAY.

We consider the setting where

$$
\tilde{\beta} = \lambda^* \beta^*
$$

$$
\rho = (1 - \lambda^*)(\Delta + 2L)
$$
Assuming the $\rho$ term dominates Equation (12) in this setting is:

$$
\epsilon_{\text{auxdyn}} \leq \epsilon_{\text{stab}} \Big|_{(\beta, \rho)} \lesssim 1 + e^{\beta} \sqrt{(1 - \lambda^*)/(\Delta + 2L)} \left( \frac{\gamma T}{N'} \right)^{\frac{\epsilon_{\beta}}{1 + e^{\beta}}}
$$

$$
\lesssim (\Delta)^{\frac{1}{\lambda^* - \lambda}} \left( \frac{\gamma T}{N'} \right)^{1 - \frac{1}{\lambda^* + 1}}
$$

This is Equation (1) from Section 4 (13)

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 $\Delta$ 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 $\gamma$ unless we reduce $\lambda_e$ appropriately. Remember that $\lambda_e$ 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 E.2.
5. We can optimize $\lambda_e, \lambda_a$ to minimize $\epsilon_{\text{stab}}$. 
