Universiteit Leiden

Meta-overfitting and Overtuning of Small Language Models Binary Classification Task Fine-tuning

Laith Agbaria (s4036328)

Supervisors:
Andreas Paraskeva, Sietse Schröder & Jan N. van Rijn

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 11, 2026

Abstract

Hyperparameter optimization is essential for fine-tuning pre-trained language models, yet it often relies on noisy validation performance that could diverge from true generalization performance on unseen data. This thesis empirically investigates the prevalence and magnitude of meta-overfitting, which is the estimate difference between validation and test performance, and overtuning, which is the phenomena where perceived improvements on validation performance lead to worse test performance of incumbents.

We conducted a total of 480 experiments by running 30 independent iterations for 16 configurations of tasks, models, and samplers, with each iteration consisting of 30 consecutive optimization trials. In the experiments we evaluated four pre-trained small language models (two decoders and two encoders) on two binary classification tasks, utilizing both a non-adaptive random search and an adaptive Tree-structured Parzen Estimator Bayesian optimization as hyperparameter configuration samplers within a fine-tuning hyperparameter search space. By logging both the validation and test accuracy performances throughout the optimization trials, we collected data for trend visualization by generating plots using aggregated means and standard deviations.

Our results show that both meta-overfitting and overtuning appear in all experimental configurations, with their magnitude being largely affected by the task selection while consistently increasing as the optimization trials progress for both tasks. Hyperparameter configuration sampling methods appear to affect the magnitudes of both meta-overfitting and overtuning, while the number of model parameters appear to affect only the magnitude of overtuning. Model architecture does not appear to affect either meta-overfitting or overtuning on its own, however when considered with parameter count, there appears to be a relation with the magnitudes of meta-overfitting.

With this research and the results within, we highlight the importance of accounting for both meta-overfitting and overtuning in hyperparameter optimization for fine-tuning workflows, providing insights that could be accounted for in the creation of more reliable machine learning models.

Contents

1 Introduction

Automated machine learning workflows aim to make the training of models an accessible and streamlined process [3], which is useful in reducing the need for manual trial-and-error while also increasing the efficiency and reproducibility during hyperparameter optimization [13]. Automated hyperparameter optimization formalizes the task of searching for a configuration that optimizes an evaluation metric within a pre-determined search space [6], this is useful in cases where the search space is extremely vast or experimental reproducibility is important [11] as well as in the training of foundational models.

With transformer-based language model use becoming widespread as foundational models [7], it is becoming more important to develop the academic understanding of fine-tuning of pre-trained models [12]. Running hyperparameter optimization studies requires many training trials to evaluate the performance of various configurations [13], which is computationally expensive when it comes to large language models that sometimes contain billions of parameters which need to be calculated and updated. Small language models are useful as computationally cheap and efficient tools for understanding the behaviors of their larger counterparts [15], this is particularly useful in situations with limited resources such as early stages of research

As hyperparameter optimization studies are unable to access large scale test data, trained models are typically evaluated using a representative validation dataset that is obtained from the same distribution as the test dataset [13]. However, in the same way a model can overfit its weights to the training data, a hyperparameter optimization study is also able to overfit the configuration selection to the validation data [9]. This causes unintended effects such as over-promising on model performance, referred to as meta-overfitting [17], and the selection of a configuration with worse test performance due to continued search, known as overtuning [16]. These phenomena occur due to the optimization methods relying on the validation dataset, which could contain stochastic noise or hidden biases that distract the trained models from true general patterns.

While meta-overfitting and overtuning have been studied and documented in traditional machine learning setups, there appears to be an academic gap in the field when looking at language models, specifically focusing on automated hyperparameter optimization of fine-tuning for transformer-based models. Understanding how these two phenomena behave in this context, and how they are affected by common variables such as hyperparameter configuration sampling methods and model architecture, can be considered crucial for reliable machine learning practices.

The objective of this thesis to explore meta-overfitting and overtuning as a result of automated hyperparameter optimization, utilizing unseen test dataset performance during the fine-tuning of small language models on classification tasks. By strictly segregating training, validation, and test datasets during multiple fine-tuning tasks, we will analyze the divergence between validation accuracy performances and that of hidden test accuracy that is intended to represent real-world cases. Through an empirical investigation, we will clarify how different hyperparameter configuration sampling methods as well as model architectures and number of parameters contribute to missing optimal configurations and providing overly optimistic performance estimates.
To fulfill our research objective, this study addresses the following central question:

- To what extent does hyperparameter optimization lead to meta-overfitting and overtuning in the fine-tuning of small language models?

We investigate this overarching question through the exploration of three primary sub-questions:

a How do meta-overfitting and overtuning of incumbents evolve across fine-tuning evaluations?

b How do non-adaptive hyperparameter fine-tuning methods such as random search compare to sequential adaptive methods such as Bayesian optimization with regards to their resulting meta-overfitting and overtuning?

c How do different model architecture and number of parameters affect the susceptibility to meta-overfitting and overtuning from hyperparameter optimization methods?

After this introduction, we will give an academic review of relevant work in Section 2, which is then followed by an explanation of the experiments used to study and answer the research questions in Section 3. The results of the experiments, as well as the analysis and explanation, are shown and used to answer the research questions in Section 4, with a summary conclusion and further research suggestions in Section 5.

2 Background

In this thesis we study meta-overfitting and overtuning that arises from the automated hyperparameter optimization of transformer-based small language model fine-tuning, and in this section we will provide an academic overview for each of the mentioned concepts.

2.1 Language Model Fine-tuning Mechanics

Transformer architecture based models are now the leading models which perform exceptionally well on a wide variety of tasks [7], utilizing attention mechanisms to map long-range dependencies across text sequences to solve natural language problems [20].

Language models can typically be categorized using a combination of two architectural classes which depend on their attention routing; encoder models (e.g. BERT), which utilize bi-directional self-attention to construct contextual representations of text [10], and decoder models (e.g. GPT-2), which utilize causal masked self-attention to predict tokens sequentially [14]. These structural differences result in the encoder models achieving better performance in natural language understanding tasks that require utilizing the full context representations, such as sentiment classification and textual entailment [22], while decoder models outperform in tasks that require autoregressive token generation, such as open-ended questions and zero-shot instruction following [8].

Regardless of their specific architectures, these models all go through a stage of unsupervised learning on large and diverse unlabeled corpora to learn generalized semantic representations and syntax [12]. By building a broad representation, a pre-trained model is able to effectively adapted to numerous similar downstream tasks with less annotated data, allowing for improved performance and faster convergence. Additionally, small language models are emerging as computationally cheap and highly efficient substitutes to their larger counterparts in early stages of research and testing, achieving faster fine-tuning speeds due to the reduced number of parameters while retaining comparative accuracy performances [15, 7].

To improve a pre-trained model's performance on a specific downstream task, it is typically fine-tuned using an automated hyperparameter optimization workflow [13, 11], utilizing transfer learning and a task specific architectural head in updating the model weights through supervised learning on curated datasets.

2.2 Hyperparameter Optimization Frameworks

Hyperparameter optimization formalizes the task of searching for a hyperparameter configuration $\boldsymbol{\lambda}$ for a machine learning model, within a designated search space $\boldsymbol{\Lambda}$, which optimizes an evaluation metric $c(\boldsymbol{\lambda})$ [6]. This task could be effectively automated to allow for higher efficiency and more reliability than the manual human trial and error process, as an algorithm could work without biases and practically without pause between searches.

Within each automated hyperparameter optimization study [13], multiple configurations are sampled using a search algorithm, and each configuration trial then trains and validates the model on the provided dataset. After a configuration trial is completed or pruned, it is scored using the evaluation metric and compared with other completed trials, the best performing trial is then selected as the current incumbent. The various methods of choosing hyperparameter configurations each have their benefits and detriments, however this thesis only focuses on two prominent optimization algorithms, namely:

2.2.1 Random search

An unguided and non-adaptive optimization method that randomly samples hyperparameter configurations from a uniform distribution over the bounded search space [5], it can run multiple trials in parallel and is a simple algorithm suited for high dimensional search spaces.

Despite its simplicity, random search is an effective optimization method due to it not suffering from grid-allocation inefficiencies, which come with repeated costly explorations of an important hyperparameter value against less consequential ones while exploring continuous dimensions. However, because it treats each trial independently, it fails to exploit historical evaluation data to guide the search toward high-performance regions.

2.2.2 Bayesian optimization

A sequential adaptive optimization framework that treats the evaluations of hyperparameter configurations as a black-box optimization problem, constructing a probabilistic surrogate model to approximate validation performance across the search space based on historical trials [19, 2].

Tree-structured Parzen Estimators are an implementation of Bayesian optimization which estimate the probability of a hyperparameter configuration being higher than a past performance threshold, as opposed to the Gaussian process which estimates the performance of hyperparameter configurations. The mathematical change causes the computation complexity to grow linearly with the number of trials, which makes it efficient for single-objective optimization [4].

An acquisition function mathematically balances exploration of uncertain regions and exploitation of known high-performing regions, proposing optimal configurations based on the results of prior trials. However, Bayesian optimization's effectiveness can decline in high-dimensional, highly noisy, or irregular search spaces, where selecting or accurately approximating the objective function be challenging.

2.3 Meta-overfitting

Previous work by Cawley et al. (2010) [9] demonstrated how machine learning models could exhibit at a form of overfitting at the model-selection during hyperparameter optimization through selection

bias, in which the model validation performance continues to improve despite the test performance beginning to stagnate or even worsen.

This overfitting occurs because the hyperparameter optimization process uses validation as an optimization target to select the best configuration, which could unintentionally guide the selection towards validation specific noise or biases and cause it to be overly optimistic relative to its actual general performance. This divergence between validation and test performances in incumbents of sequential trials is referred to as meta-overfitting [17], and it contains two distinct mechanisms, i.e., selection-based meta-overfitting and adaptive meta-overfitting, which we will explain in distinct sections.

2.3.1 Selection-based meta-overfitting

This is driven by the volume of configurations evaluated, as when a hyperparameter optimization method tests a large number of distinct hyperparameter settings, it increases the statistical probability that a configuration will randomly have an inflated validation performance. This configuration amplifies the stochastic noise or biases within the validation set, making its performance less indicative on general unseen data. Selection-based overfitting can affect all hyperparameter sampling methods.

2.3.2 Adaptive meta-overfitting

This mechanism is driven by the feedback loop inherent to adaptive hyperparameter sampling methods such as Bayesian optimization, because the samplers actively utilize validation performance feedback from past trials to propose future configurations, the method actively learns to exploit the stochastic noise or biases within the validation set. As successive hyperparameter suggestions become increasingly customized to the nuances of the validation set, the divergence between validation and test performance increases.

2.4 Overtuning

Aside from causing overly optimistic model performance expectations, overfitting at the hyperparameter level can also cause the selection of a hyperparameter configuration incumbent that performs worse on general unseen data compared to prior incumbents, which is referred to by Schneider et al. [16] as overtuning. This evaluation error is caused by the same validation noise that causes meta-overfitting, and it highlights the need to know the correct incumbent selection criteria and optimization stopping point.

Overtuning has been shown to have several factors which increase its probability and magnitude, which are:

- Dataset size: There is a negative relation between sample size and overtuning. Smaller datasets have a correspondingly smaller validation sets, leading to more statistical variance and unstable performance results, while larger datasets, or those employing cross-validation, would have more of a buffer that reduces stochasticity.
- Model flexibility: Highly flexible models, such as neural networks, demonstrate higher vulnerability to overtuning and with higher magnitudes, being particularly prominent in binary classification tasks.

- Configuration sampling method: Bayesian optimization methods, such as SMAC3, slightly increase the odds of non-zero overtuning, while random search substantially increases the magnitude.
- Number of optimization trials: There is a positive direct relation between number of sequential trials and the probability of overtuning, however this relation plateaus at higher iterations. The effect of number of trials on the magnitude of existing overtuning has not been academically established.
- Task metric: Classification tasks, specifically binary classification, are more susceptible to overtuning than regression tasks, with accuracy being more sensitive to overtuning than other methods such as log loss. While this is not studied in this thesis, it is still relevant to mention due to the use of accuracy on binary classification to study the other effects.

Recent work by Schröder et al. (2026) [18] tackles mitigation strategies for overtuning, however as incorporating them into this thesis would further expand the scope of study, we have resolved to continue without applying any of the studied strategies.

Both meta-overfitting and overtuning are directly or indirectly caused by the stochastic nature of validation performance, and are in a way the consequence of hyperparameter optimization's test-blind judgment.

2.5 Formal Notation

In order to rigorously quantify meta-overfitting and overtuning, we take the following adapted mathematical notations:

For the duration of the hyperparameter optimization we fix the datasets $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, and $\mathcal{D}_{test}$ with all datasets being mutually exclusive, such that:

$$\mathcal{D}_{train} \cap \mathcal{D}_{val} = \mathcal{D}_{val} \cap \mathcal{D}_{test} = \mathcal{D}_{test} \cap \mathcal{D}_{train} = \varnothing$$

which are all generated using the same distribution $\mathbb{P}_{xy}$ with n_{train}, n_{val}, and n_{test} observations respectively such that:

$$\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s} \sim \mathbb{P}_{xy} \subset \mathcal{X} \times \mathcal{Y} \quad \text{for } s \in \{train,\ val,\ test\} \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^d$ is the d-dimensional input feature space, and $\mathcal{Y} = \{1, \ldots, k\}$ is the categorical label space space for k-unique classes.

Referencing Bischl et al. (2023) [6] to define the training of a machine learning model, we take a machine learning algorithm (inducer) $\mathcal{I}$ which has the hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ and maps it to a model $\hat{f}$ or its associated parameter weights vector $\hat{\theta}$ trained on $\mathcal{D}_{train}$:

$$\mathcal{I}_{\mathcal{D}_{train}} : \boldsymbol{\Lambda} \to \mathcal{H}, \quad \boldsymbol{\lambda} \mapsto \hat{f}$$

where $\boldsymbol{\Lambda}$ is the hyperparameter configuration space, and $\mathcal{H}$ is the model (hypothesis) space.

From here we formally define a trained model as:

$$\hat{f} = \mathcal{I}_{\mathcal{D}_{train}}(\boldsymbol{\lambda}) \tag{2}$$

which we utilize in the formal definition of the accuracy metric function:

$$\text{Accuracy}(\hat{f}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathbf{1}[\hat{f}(x) = y] \tag{3}$$

and since $\mathcal{D}_{val}$ and $\mathcal{D}_{test}$ are fixed, we are able to define the following objective functions:

$$\text{Acc}_{val}(\boldsymbol{\lambda}) := \text{Accuracy}(\mathcal{I}_{\mathcal{D}_{train}}(\boldsymbol{\lambda}), \mathcal{D}_{val}) \tag{4}$$
$$\text{Acc}_{test}(\boldsymbol{\lambda}) := \text{Accuracy}(\mathcal{I}_{\mathcal{D}_{train}}(\boldsymbol{\lambda}), \mathcal{D}_{test}) \tag{5}$$

From the objective function $\text{Acc}_{test}(\boldsymbol{\lambda})$ in Equation (5), we define the hyperparameter optimization goal function to be finding the hyperparameter configuration which results in the maximal test accuracy performance, estimated using validation through Equation (4):

$$\boldsymbol{\lambda}^* := \arg\max_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \text{Acc}_{val}(\boldsymbol{\lambda}) \tag{6}$$

Given the sequential nature of optimization trial evaluations, we define the trajectory as the ordered incumbent sequence for T trials as $(\boldsymbol{\lambda}^*{}_1, \ldots, \boldsymbol{\lambda}^*{}_T)$, such that:

$$\boldsymbol{\lambda}^*{}_t := \arg\max_{\boldsymbol{\lambda}\in\{\boldsymbol{\lambda}^*{}_1,\ldots,\boldsymbol{\lambda}^*{}_t\}} \text{Acc}_{val}(\boldsymbol{\lambda}) \tag{7}$$

where t is a trial index in the range $1 \leq t \leq T$.

Using these definitions, we reference the work of Schröder et al. (2025) [17] to formulate our definition of meta-overfitting as:

$$\mathcal{M}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t) := \text{Acc}_{val}(\boldsymbol{\lambda}^*{}_t) - \text{Acc}_{test}(\boldsymbol{\lambda}^*{}_t) \tag{8}$$

to denote the difference between validation and test accuracy performance of the incumbent after observing t sequential trials. This is modified from the original difference between test and validation loss for each configuration that was used in the original work. It is limited to values between $[-1, 1]$ where a positive-valued output from this equation could be therefore interpreted as the extent to which the optimizer is overly optimistic about model performance, while a non-positive output would mean the optimizer is correctly estimating or underestimating the model performance.

Finally, we adapt the overtuning definition given by Schneider et al. (2025) [16] to obtain:

$$\mathcal{G}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t) := \max_{\boldsymbol{\lambda}^*{}_{t'}\in\{\boldsymbol{\lambda}^*{}_1,\ldots,\boldsymbol{\lambda}^*{}_t\}} \text{Acc}_{test}(\boldsymbol{\lambda}^*{}_{t'}) - \text{Acc}_{test}(\boldsymbol{\lambda}^*{}_t) \tag{9}$$

which describes the difference between highest observed incumbent test accuracy performance in t sequential trials and the test accuracy performance of the latest observed incumbent. Therefore the output of this equation is limited to $[0, 1]$, where a positive-valued output implies overtuning and stopping earlier or choosing the incumbent differently would have prevented the selection of a configuration with worse test performance.

3 Experiment setup

For this study, we utilized 10 NVIDIA RTX 4060 GPUs with each running its own instance of an experiment [1], repeated three times to obtain a total of 30 independent samples containing 16 unique hyperparameter optimization setups obtained from the two datasets, four models, and two hyperparameters configuration sampling methods.

The experiments iterate over tasks by loading their training dataset which is shuffled and split, then for each dataset we iterate over the cartesian product of models and samplers to create a hyperparameter optimization study, sampling from the configuration space with the goal of maximizing validation accuracy.

A log of evaluation history is made at the end of each trial, with all trial logs being saved for later processing at the end of each combination.

3.1 Design and Dynamics

For this thesis we developed an experiment framework using the HuggingFace "Transformers" library [23], integrated with Optuna [2] automated hyperparameter optimization tool. The experiments execute fully independent hyperparameter optimization studies for each combination of dataset, language model, and hyperparameter sampling method, with each routine running for $T = 30$ complete trials and no pruning. During each trial t, the model performance on validation and test progression is monitored by evaluations at set intervals, the number of which can be controlled and is set to five evaluations per epoch for the entirety of the study. Each experiment randomly generates then uses a seed that is used for all model and configuration optimizer (sampler) combinations using the same dataset, this is done to ensure fairness and reproducibility.

At each evaluation, the HuggingFace "Trainer" calculates and records both $\text{Acc}_{val}(\boldsymbol{\lambda})$ and $\text{Acc}_{test}(\boldsymbol{\lambda})$ of the not yet fully trained model for the given study task simultaneously in addition to the loss and trial data, this is to allow easier data handling and analysis after the completing of the experimentation. It is crucial to note that while Acc_{test} is recorded in the log history, the value is never exposed to the Optuna samplers or optimization logic due to it being intended as an isolated and strictly hidden metric. After a trial t is complete and the configuration $\boldsymbol{\lambda}_t$ is added to the trajectory, the incumbent is selected according to Equation (7) for the trajectory $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t)$ using the highest observed $\text{Acc}_{val}(\boldsymbol{\lambda})$ during training for each configuration.

After a study completes all its trials, we are able to measure the progression of meta-overfitting and overtuning by using each trial's best performing evaluation checkpoint and comparing it to prior trials in the same study. By using an aggregate mean and standard deviation of 30 repetitions for each study combination, we are able to plot the progress of both phenomenon's estimates over trials and compare their final results.

3.2 Models, Data, and Samplers

For this study we utilized four small language models obtained from HuggingFace due to their small storage cost, high computational efficiency, and ease of use. To examine the susceptibility of different model architectures to we select a pair of decoder and encoder small language models with about 32 million parameters each, then to examine the effects of parameter counts we select another pair of decoder and encoder models with fewer than 5 million parameters.

Model Identifier	Approx. Parameters	Architecture Type
`sshleifer/tiny-gpt2`	2.1 Million	Decoder
`google/bert_uncased_L-2_H-128_A-2`	4.4 Million	Encoder
`EleutherAI/pythia-31m`	31 Million	Decoder
`microsoft/MiniLM-L12-H384-uncased`	33 Million	Encoder

Table 1: Structural specifications of selected pre-trained Small Language Models.

We utilized datasets two binary classification tasks for the fine-tuning, with the training data of each task training dataset being partitioned with shuffling according to the following splitting procedure to avoid data leakage:

- One third ($33.\overline{3}\%$) of all training data is split to form the test set ($\mathcal{D}_{test}$) intended for analysis only,
- One fifth (20%) of the remaining training data is split to form the validation set ($\mathcal{D}_{val}$) which Optuna utilizes to optimize hyperparameter configuration,
- All remaining training data is used as the training set $\mathcal{D}_{train}$ during fine-tuning.

The selected tasks for this study were the SST-2 sentiment binary sentiment classification of movie review sentences from the GLUE benchmark [22], and the BoolQ binary response reading comprehension task from SuperGLUE [21]. This selection comes from considerations about medium and small dataset sizes on the results, to observe the effects of meta-overfitting and overtuning in both cases.

Task	$\lvert\mathcal{D}_{train}\rvert$	$\lvert\mathcal{D}_{val}\rvert$	$\lvert\mathcal{D}_{test}\rvert$	Number of classes
GLUE - SST2	35919	8980	22450	2
SuperGLUE - BoolQ	5027	1257	3143	2

Table 2: Dataset specifications of selected fine-tuning tasks.

Finally, to navigate the configuration search space, we utilized adaptive and non-adaptive samplers from Optuna for hyperparameter optimization method to study their effects on both phenomena, random sampler for the non-adaptive optimization and tree-structured Parzen estimators for the adaptive optimization.

Sampler	Adaptive	Method
RandomSampler	False	Random search
TPESampler	True	Bayesian optimization

Table 3: Optuna samplers used for hyperparameter optimization

3.3 Search Space Configuration

The hyperparameter optimization configuration search space $\mathbf{\Lambda}$ ranges across six hyperparameters, that affect both the model learning and training costs. As this research has a focus on automated hyperparameter optimization, the search space intentionally spans wide ranges of value of common hyperparameters.

Hyperparameter	Type	Domain / Range	Scale / Options
Batch size	Categorical	{4, 8, 16, 32, 64}	Discrete options
Number of epoch	Integer	$[2, 5]$	Linear interval
Learning rate	Numeric	$[10^{-6}, 10^{-3}]$	Logarithmic scale
Scheduler	Categorical	{linear, cosine, constant, polynomial}	Discrete options
Warm-up ratio	Numeric	$[0, 10^{-1}]$	Linear interval
Weight decay	Numeric	$[10^{-6}, 10^{-2}]$	Logarithmic scale

Table 4: Formal hyperparameter search space configurations inside the Optuna objective loop.

The study evaluates each configuration trial based on the highest validation accuracy performance $\text{Acc}_{val}(\boldsymbol{\lambda})$, this value is then compared to prior trials to determine the incumbent with the best performing configuration.

4 Results

By taking the results of the 480 hyperparameter optimization studies, we generated the figures to analyze the extent meta-overfitting and overtuning across trial progression and setups. Utilizing these visualization we analyze and explain insights related to meta-overfitting and overtuning across two task datasets, relating them to the selection of hyperparameter sampling method, architecture type, and number of parameters.

Throughout the results section, meta-overfitting and overtuning will be measured on the best validation performance step for each trial, this is visually demonstrated in Figure 1 where meta-overfitting $\mathcal{M}$ is shown as the vertical (accuracy percentage) difference between the dashed horizontal lines and overtuning $\mathcal{G}$ is shown as the difference between the red horizontal lines.

4.1 Task Differences

While the core aims of our research do not include studying the effects of various tasks on meta-overfitting and overtuning, they do have a measurable influence [16] that must be understood before beginning the analysis of our studied factors.

We observe in Figure 2 the growth trajectories of meta-overfitting and overtuning for models trained on the tasks GLUE: SST-2 and SuperGLUE: BoolQ (Table 2), which show clear trends towards higher estimates across both tasks and phenomena. In Figure 2a we can observe the final percentage difference between validation and test accuracy performance (measured meta-overfitting $\mathcal{M}$) is equal to 0.2% in the case of SST-2 and just below 0.9% for BoolQ, while in Figure 2b the test accuracy performance percentage difference between the best and currently selected incumbents (measured overtuning $\mathcal{G}$) is approximately 0.04% for SST-2 and 0.28% for BoolQ.

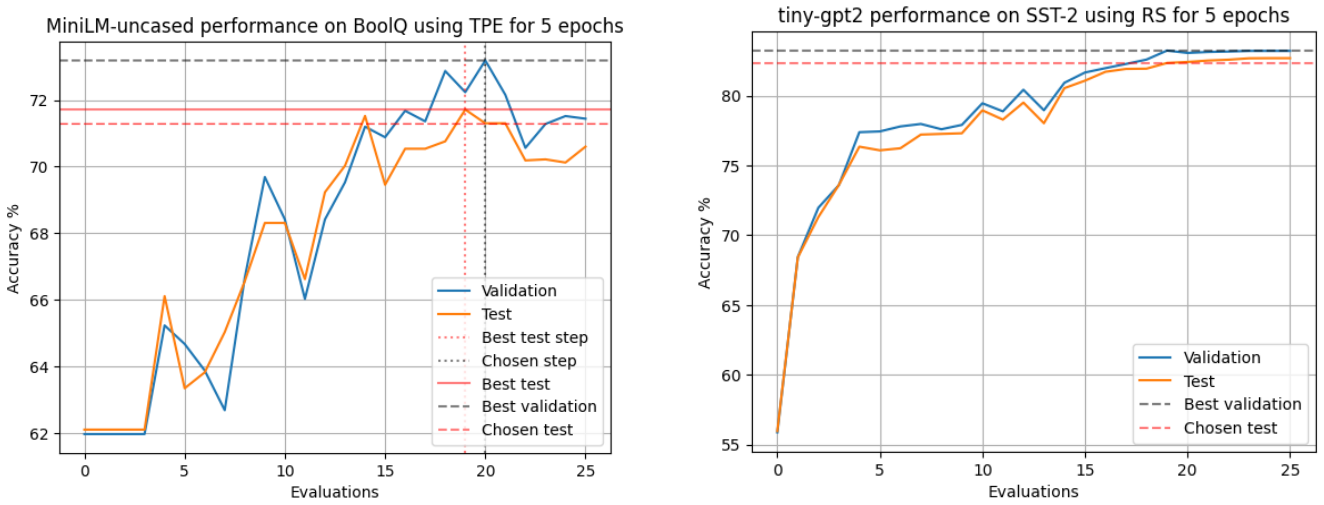
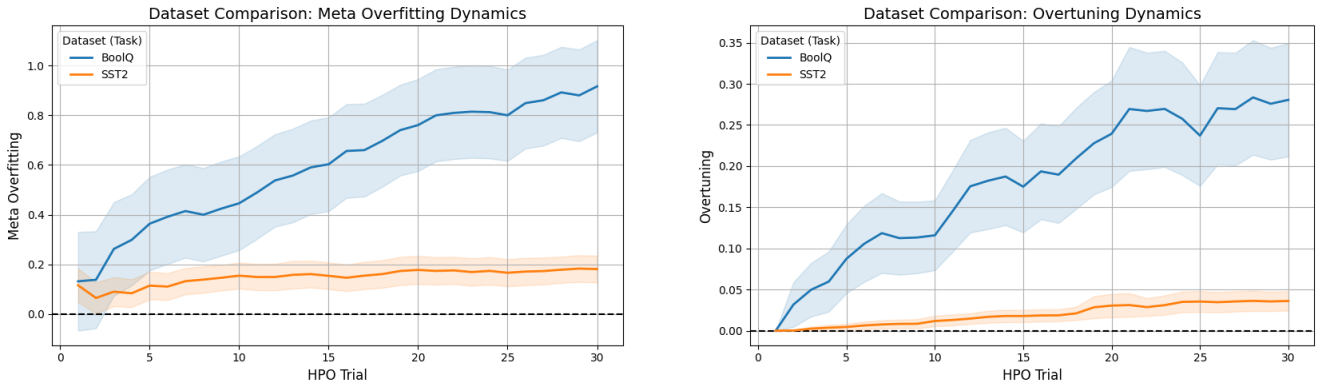


Figure 1: Example validation and test accuracy performance of the best performing trial with 5 epochs, used to visually highlight meta-overfitting (both) and overtuning (left).

The dataset of the BoolQ task had much fewer total data points, roughly 14% that of SST-2, which explains the observed the differences in growth trajectories and variance band sizes, as there is more possibility for noise and biases in either the validation dataset or the examination test dataset that would affect the calculations of performance and incumbent selection.



(a) Meta-overfitting

(b) Overtuning

Figure 2: Average progression of meta-overfitting $\mathcal{M}$ (left, 2a) and overtuning $\mathcal{G}$ (right, 2b) over 30 progressive trials with one standard deviation band, obtained from 240 hyperparameter optimization studies for each task dataset. Both figures show a clear progressive increase in estimate for both tasks, while also highlighting the effect of having a smaller dataset (BoolQ) on exaggerating both phenomena.

More detailed visualizations of task choice influence on meta-overfitting and overtuning is available in Appendix A, the individual results for the GLUE: SST-2 trials can be found in Appendix B and those for SuperGLUE: BoolQ in Appendix C.

4.2 Analysis and Explanation

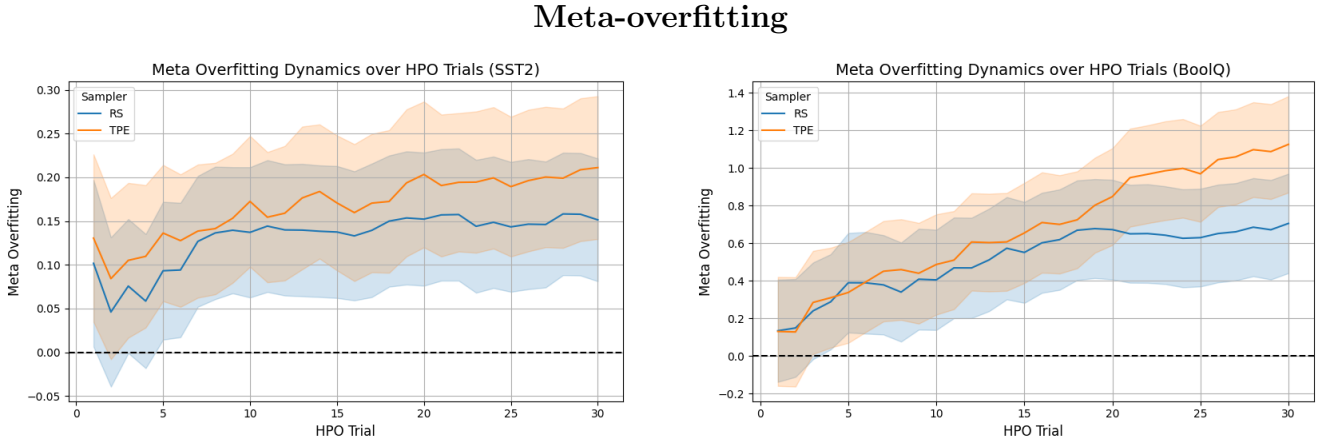
By establishing the need to separate the results based on dataset, we can begin analyzing the visual patterns that appear for both meta-overfitting and overtuning progression when separated into pairs for each of our studied factors. Beginning with a comparison of random search and TPE in Section 4.2.1, followed by examining the effects of model architecture in Section 4.2.2, and finally examining the influence of model parameter count in Section 4.2.3.

4.2.1 Sampler comparison

The effects of the choice between adaptive and non-adaptive hyperparameter configuration sampling methods on meta-overfitting and overtuning have been documented and discussed previously by Schröder et al. (2025) [17] and Schneider et al. (2025) [16], with their results showing that adaptive methods lead to more meta-overfitting and more frequent, albeit lower in magnitude, overtuning.

We observe in the results of Figure 3 that meta-overfitting of models optimized using TPE and those using random search are quite similar in early trials, however in later trials when the black-box model has developed enough to navigate an approximate performance space we see a growing difference between both methods.

Models fine-tuned using the GLUE: SST-2 dataset (Figure 3a) experienced more similar meta-overfitting values across both sampling methods than when using the SuperGLUE: BoolQ dataset (Figure 3b), however meta-overfitting from random search in both tasks appears to stagnate in later trials while TPE continues to grow.

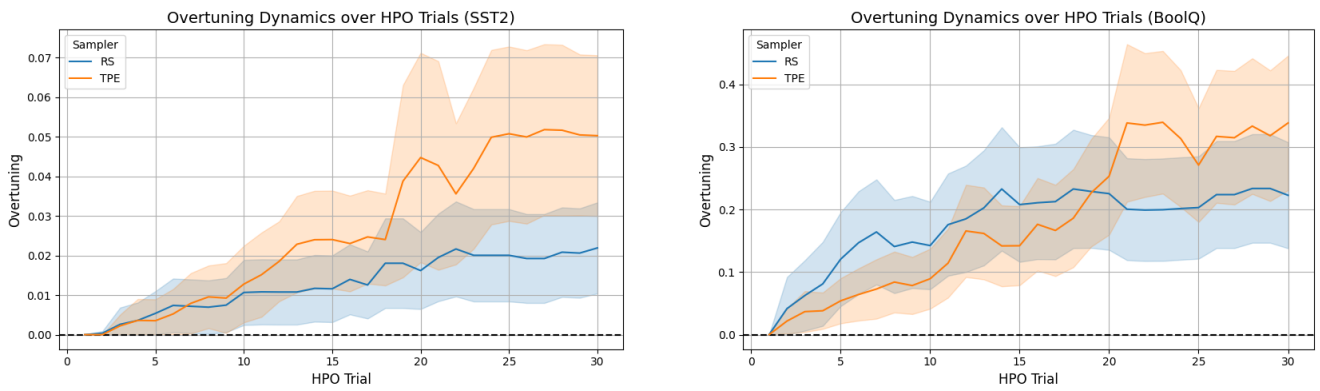


(a) GLUE: SST-2

(b) SuperGLUE: BoolQ

Figure 3: Average meta-overfitting ($\mathcal{M}$) progression over 30 hyperparameter optimization trials for Bayesian optimization (blue) and random search (orange), showing results of GLUE: SST-2 on the left and SuperGLUE: BoolQ on the right.

Similarly, Figure 4 shows optimization studies using TPE to sample hyperparameter configurations experience more overtuning in later trials, with overtuning from random search across both tasks beginning to plateau earlier and with a lower value. This appears to be inconsistent with the results of Schneider et al. (2025) [16], however it could be possibly be due to the vast differences between the experiment structures.



(a) GLUE: SST-2

(b) SuperGLUE: BoolQ

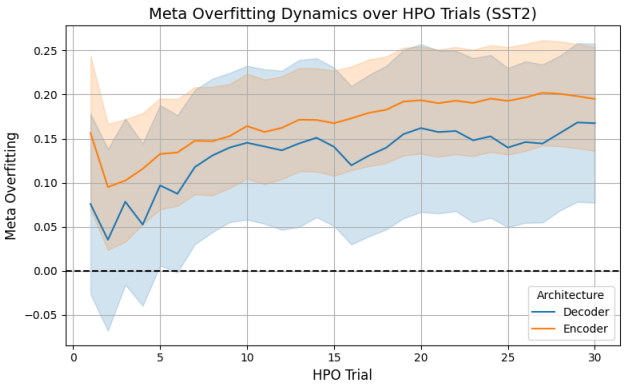
Figure 4: Average overtuning ($\mathcal{G}$) progression over 30 hyperparameter optimization trials for Bayesian optimization (blue) and random search (orange), showing results of GLUE: SST-2 on the left and SuperGLUE: BoolQ on the right.

4.2.2 Architecture effect

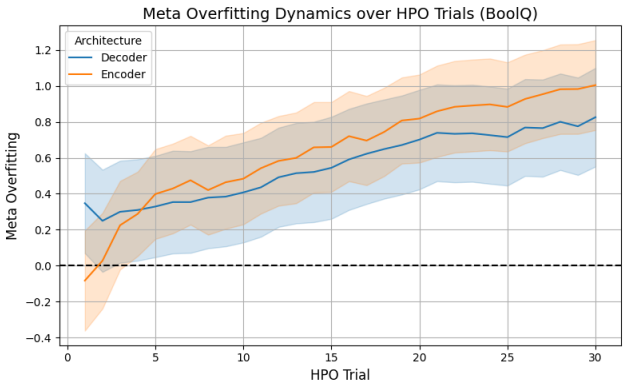
During the writing of this thesis, we were unable to find any prior work that investigates the differences in meta-overfitting and overtuning due to language model architecture, as such the following results could function as a baseline in future research.

Figures 5 and 6 each contain two sub-figures that show the trajectories of either meta-overfitting or overtuning, showing their growth throughout the optimization trials for both tasks. The trajectories in each sub-figure represent either decoder (blue) or encoder (orange) models, with encoder models having consistently higher average meta-overfitting and overtuning. The difference between the different architecture averages is always within a standard deviation, which makes it difficult to disregard the possibility of random chance when considering the experiment contains four models.

Meta-overfitting



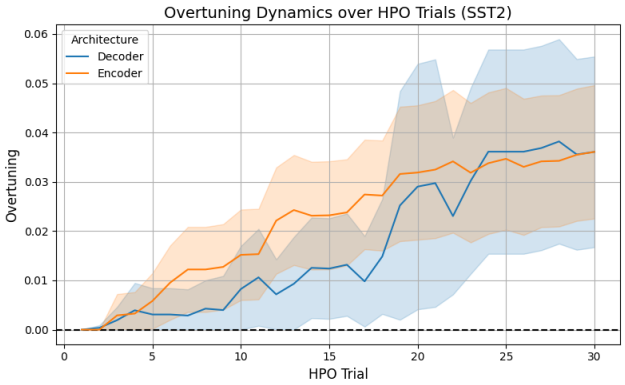
(a) Meta-overfitting on GLUE: SST-2



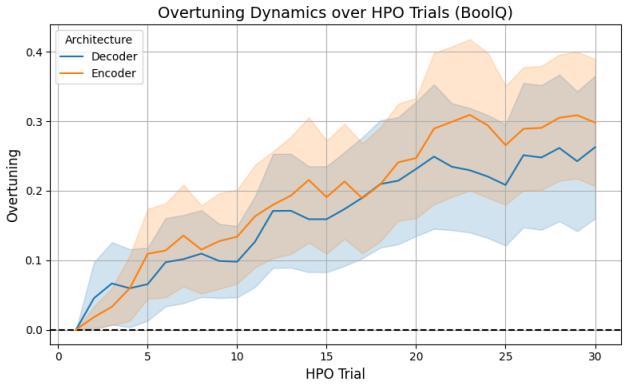
(b) Meta-overfitting on SuperGLUE: BoolQ

Figure 5: Average progression of meta-overfitting over trials depending on model architecture with a standard deviation band, for SST-2 (left) and BoolQ (right).

Overtuning



(a) Overtuning on GLUE: SST-2

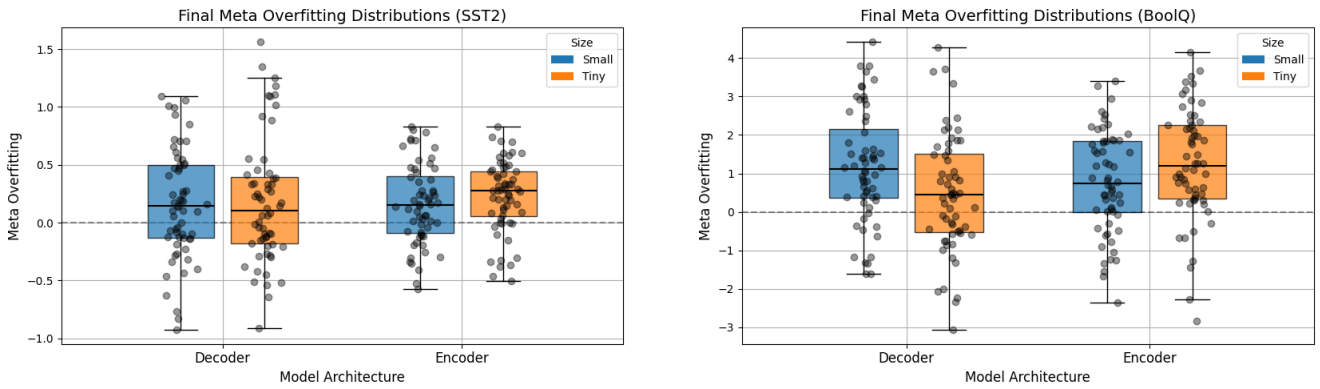


(b) Overtuning on SuperGLUE: BoolQ

Figure 6: Average progression of overfitting over trials depending on model architecture with a standard deviation band, for SST-2 (left) and BoolQ (right).

During the analysis of our results, we noticed that the combined effects of model architecture and number of parameters showed a possible relation to meta-overfitting. To investigate this possible effect, we generated a box plot for the meta-overfitting of each fine-tuning task (Figure 7), grouping trial meta-overfitting values by architecture then sub-grouping the distributions of parameter count.

The generated Figure 7 supports this suspected relation between meta-overfitting and the combination of model architecture and parameter count, with decoder models showing less meta-overfitting with more parameters while encoder models show more meta-overfitting with more paramters.



(a) SST-2 (b) BoolQ

Figure 7: Box plot of per trial meta-overfitting ($\mathcal{M}$) separated in each sub-figure by model architecture (decoder on left, and encoder on right) and number of parameters ($\sim$ 32M in blue, and $>$ 5M in orange).

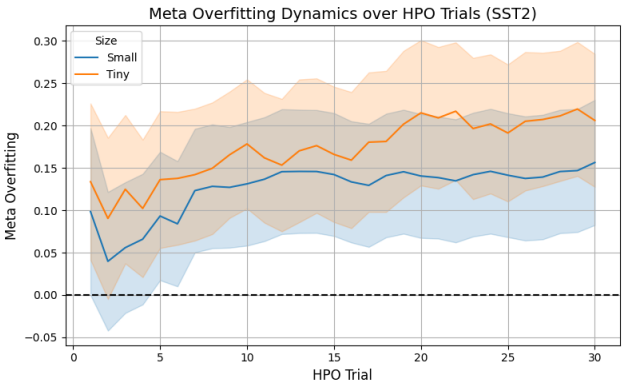
4.2.3 Parameter count influence

Similarly to model architecture, there appears to be no published works that study the influence of language model parameter count on either meta-overfitting or overtuning, however Schneider et al. (2025) [16] do mention that model flexibility influences the probability and magnitude of overtuning and as such models with more parameters (i.e. higher flexibility) should demonstrate this relation.

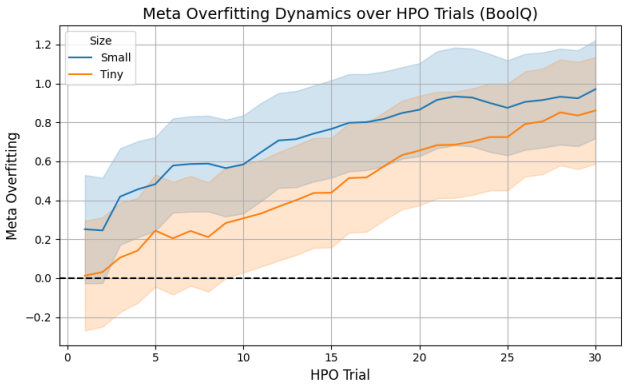
Looking at the influence of just model parameter count in meta-overfitting shown in Figure 8, we observe no signs of a relation. Small models ($\sim$ 32M parameters) and tiny models ($>$ 5M parameters) appear to have nearly identical growth trends across both tasks, and neither having consistently more meta-overfitting. We take this as evidence that the number of model parameters does not have a noticeable effect on the magnitude of meta-overfitting, meaning that meta-overfitting is more affected by training influences or more significant model differences.

On the other hand, Figure 9 does show visual evidence that the number of model parameter does influence overtuning, with tiny models having more exaggerated overtuning with further possible growth had the trials continued as the values had not yet plateaued after 30 trials on the SuperGLUE: BoolQ task.

Meta-overfitting



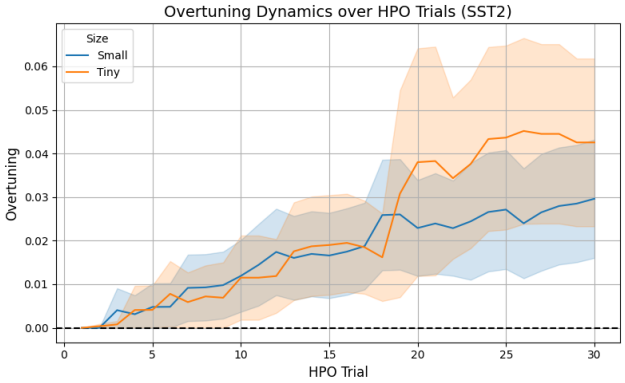
(a) SST-2



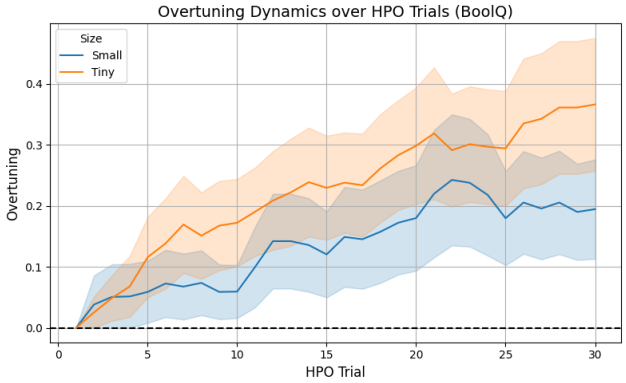
(b) BoolQ

Figure 8: Average meta-overfitting ($\mathcal{M}$) progression over 30 hyperparameter optimization trials for small models under $\sim$ 32M (blue) and tiny models $>$ 5M (orange), showing results of GLUE: SST-2 on the left and SuperGLUE: BoolQ on the right.

Overtuning



(a) SST-2



(b) BoolQ

Figure 9: Average overtuning ($\mathcal{G}$) progression over 30 hyperparameter optimization trials for small models under $\sim$ 32M (blue) and tiny models $>$ 5M (orange), showing results of GLUE: SST-2 on the left and SuperGLUE: BoolQ on the right.

5 Conclusion and Further Research

In this thesis we tackled an academic gap in the field of hyperparameter optimization meta-overfitting and overtuning, focusing on the automated fine-tuning of transformer-based small language models. We provided empirical insights regarding the trajectories of both phenomena over trials, as well as the effects of hyperparameter sampling method, model architecture, and number of parameters.

Our results found that both phenomena appeared in all combinations of tasks, samplers, and models, with the trajectories of meta-overfitting continuing to grow till the 30th trial across both examined classification tasks, while overtuning showed signs of reaching a plateau in later trials. Additionally, the task with fewer data points showed significantly higher values for both

meta-overfitting and overtuning, matching the results of Schneider et al. (2025) [16].

As for our investigations into the effects of different samplers, architectures, and parameter counts, we found that the hyperparameter configuration sampling method resulted in visual evidence of influencing the magnitude of meta-overfitting, as did the combination of model architecture and parameter count. For overtuning we found that the sampling method was once again visually evident as having an effect, with the addition of parameter count.

Through our empirical exploration we combine the fields of hyperparameter overfitting and language model research, creating a framework for future studies as well as baselines that could be directly used. We also publish the code and resulting detailed logs as artifacts for public use.

Further Research

Our experiments were limited by physical computational and time restraints, as running the experiments took approximately two days. These constraints have reduced the number of models available for use and limited the task possibilities to those with smaller training datasets. Additionally, the coding complexity limited us to using classification tasks with accuracy being the most suitable metric for binary classification, which prior work has shown to be more vulnerable to overtuning [16].

We attempted to add statistical testing to this thesis with the aim of strengthening the evidence of the claims, however the base assumptions required for ANOVA testing are not met in our study, the identification and execution of a suitable statistical test remains a direction for future work.

Further research can be done by incorporating the results of fine-tuning more models within the same setup, allowing for more concrete proof of the observed outcomes. Another possible modification to the original experiment that we consider worth further investigation is increasing the number of trials in each hyperparameter optimization study, as this will allow clear observations of the behavior of meta-overfitting over longer studies, as well as highlight the observed effect of parameter count on overtuning.

Alternatively, a repetition of the same experiment with different tasks or metrics could prove to be of high interest, as that would allow for comparisons usable in making generalizations. Finally, expanding the work by including cross-validation, which is shown to mitigate both meta-overfitting and overtuning, or a selection of the overtuning mitigation strategies discussed by Schröder et al. (2026) [18] would ground the results to more common fine-tuning practices.

References

[1] Laith Agbaria. Meta-overfitting and overtuning in small language models. https://github.com/L-Agbaria/meta-overfitting_overtuning, 2026.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. ACM, 2019.

[3] Mitra Baratchi, Can Wang, Steffen Limmer, Jan N. van Rijn, Holger H. Hoos, Thomas Bäck, and Markus Olhofer. Automated machine learning: past, present and future. *Artificial Intelligence Review*, 57(5):122, 2024.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, pages 2546–2554, 2011.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Leraning Research*, 13:281–305, 2012.

[6] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowedge Discovery*, 13(2), 2023.

[7] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[8] 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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[9] Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[11] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305, 2020.

[12] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[13] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning - Methods, Systems, Challenges*. Springer, 2019.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical Report 8, OpenAI, 2019.

[15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[16] Lennart Schneider, Bernd Bischl, and Matthias Feurer. Overtuning in hyperparameter optimization. In *Proceedings of the International Conference on Automated Machine Learning (AutoML 2025), 8-11 September 2025, Cornell Tech, New York, NY, USA*, volume 293, pages 17/1–43. PMLR, 2025.

[17] Sietse Schröder, Mitra Baratchi, and Jan N. van Rijn. Overfitting in combined algorithm selection and hyperparameter optimization. In *Advances in Intelligent Data Analysis XXIII - 23rd International Symposium on Intelligent Data Analysis, IDA 2025, Konstanz, Germany, May 7-9, 2025, Proceedings*, volume 15669, pages 181–194. Springer, 2025.

[18] Sietse Schröder, Zubin Zellmann, Matthias Feurer, Mitra Baratchi, Jan N. van Rijn, and Bernd Bischl. A large-scale study of overtuning mitigation strategies in hyperparameter optimization. In *Proceedings of the Fifth International Conference on Automated Machine Learning*. PMLR, 2026. Forthcoming.

[19] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2960–2968, 2012.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[21] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275, 2019.

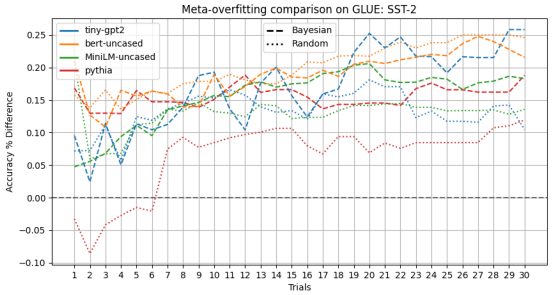
[22] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding.

In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

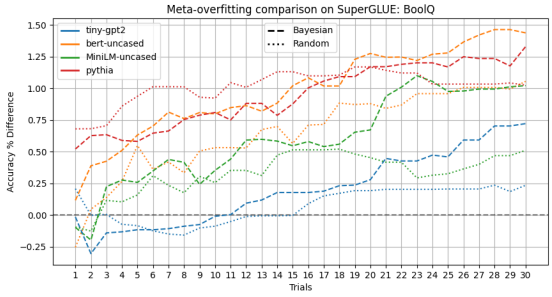
[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

A Expanding on task differences

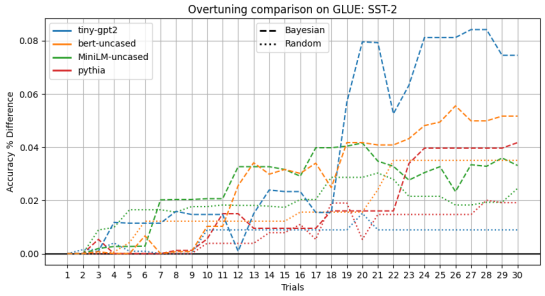
Figure 10 shows the average meta-overfitting ($\mathcal{M}$) and overtuning ($\mathcal{G}$) trajectories of every model-sampler combination over 30 fine-tuning trials in hyperparameter optimization studies on GLUE: SST-2 and SuperGLUE: BoolQ, with the values for the smaller BoolQ dataset being noticeably larger in both phenomena with both tasks showing growth in meta-overfitting and overtuning.



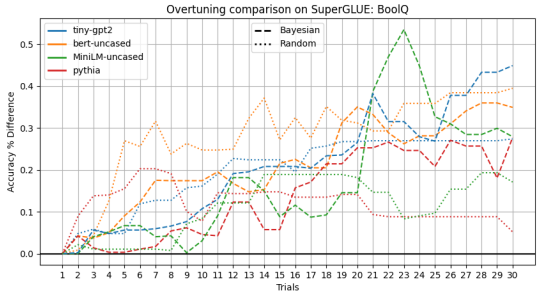
(a) Meta-overfitting on GLUE: SST-2



(b) Meta-overfitting on SuperGLUE: BoolQ



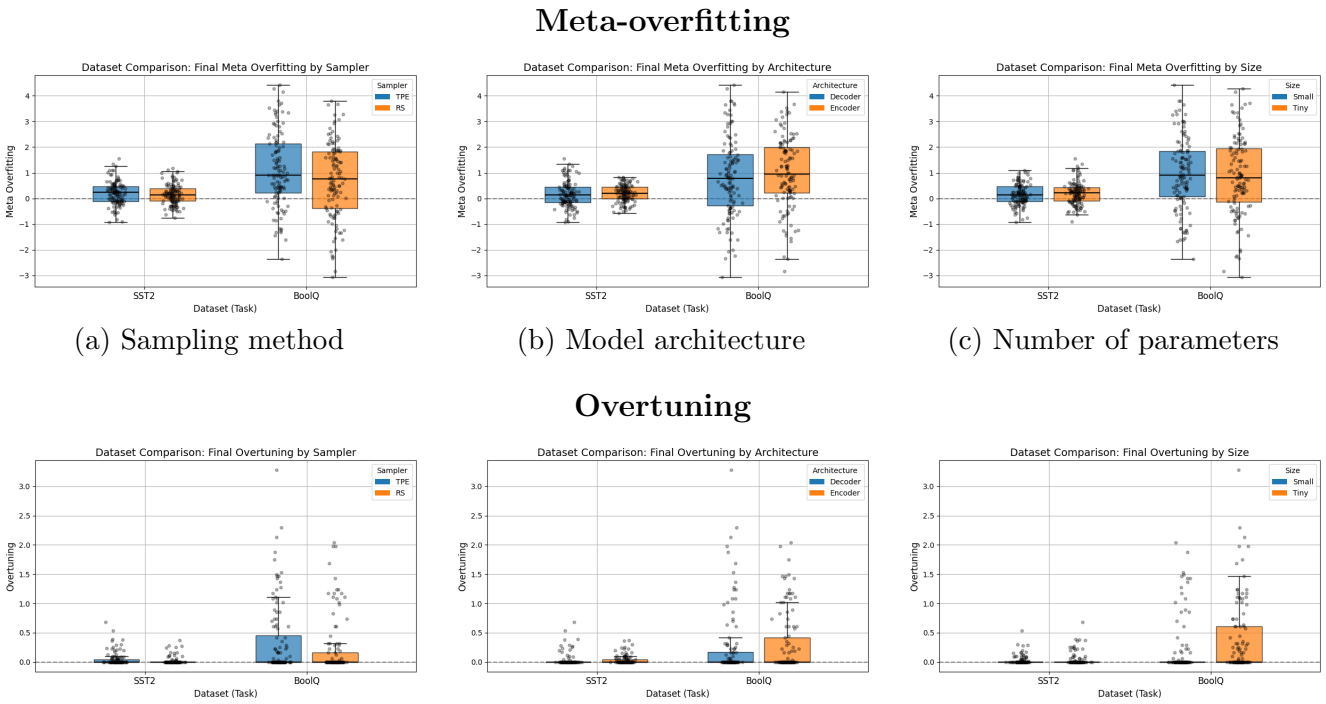
(c) Overtuning on GLUE: SST-2



(d) Overtuning on SuperGLUE: BoolQ

Figure 10: **Top row**: Average progression of meta-overfitting of various model/sampler combinations over 30 trials from 30 hyperparameter optimization studies on SST-2 (left) and BoolQ (right).
Bottom row: Average progression of overtuning of the same trials.

By grouping each of our studied effects into box plot distributions we obtain Figure 11, in which we can observe the relation of sampling method on meta-overfitting and overtuning (Figures 11a and 11d) as Bayesian optimization shows more of each phenomena. We can also observe a large difference in overtuning for parameter counts for BoolQ in Figure 11f, which is the primary cause for the main effect statistically significant result. Finally, Figures 11b and 11e show signs of a possible influence from model architecture which does not appear as statistically significant in any of the ANOVA summaries, with encoder models having higher averages and denser distributions on both meta-overfitting and overtuning compared to decoder models.



(a) Sampling method (b) Model architecture (c) Number of parameters

(d) Sampling method (e) Model architecture (f) Number of parameters

Figure 11: Box plots show the calculated of meta-overfitting and overtuning when isolating based on sampling method (left, 11a and 11d respectively), model architecture (center, 11b and 11e), and number of model parameters (right, 11c and 11f).

B Individual results of GLUE: SST-2

The progression of meta-overfitting ($\mathcal{M}$, orange) and overtuning ($\mathcal{G}$, blue) (Eq (8) and (9)) for each model fine-tuned on GLUE: SST-2 are shown in Figure 12, with the mean trajectories for Bayesian optimization and random search being split in both phenomena and bands surrounding these trajectories being one standard deviation obtained from 30 repetitions.

Visually, decoder models (tiny-gpt2 and pythia) show more differences between the sampling methods on both phenomena, while encoder models have less deviations in both sampling methods.

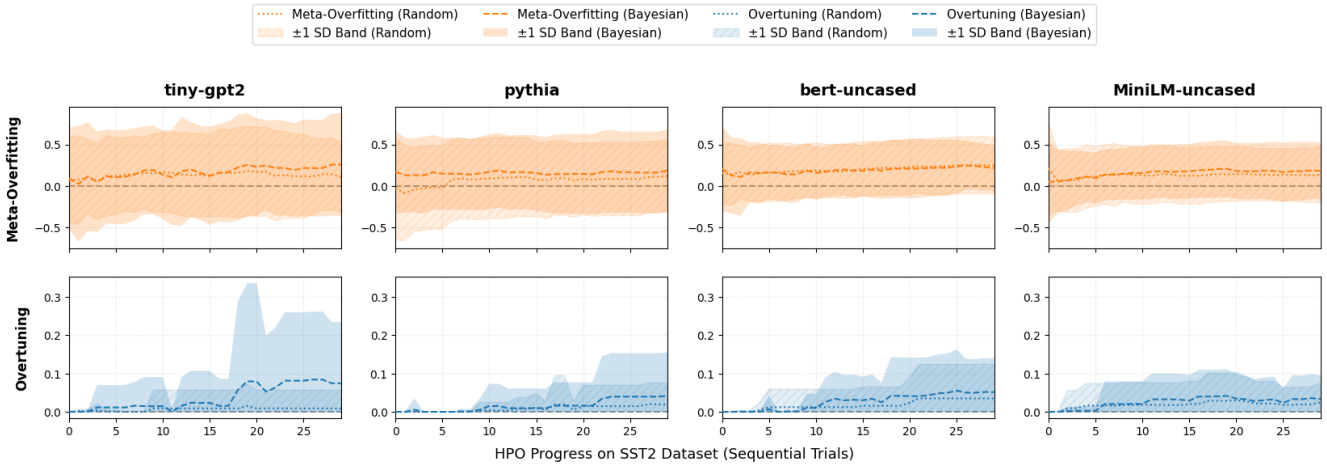
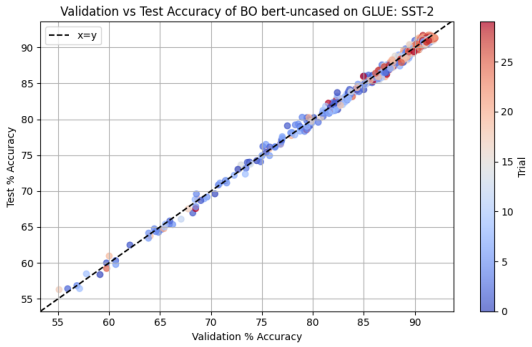


Figure 12: Progression of average meta-overfitting and overtuning for individual models fine-tuning on SST-2, with a standard deviation band obtained from 30 iterations.

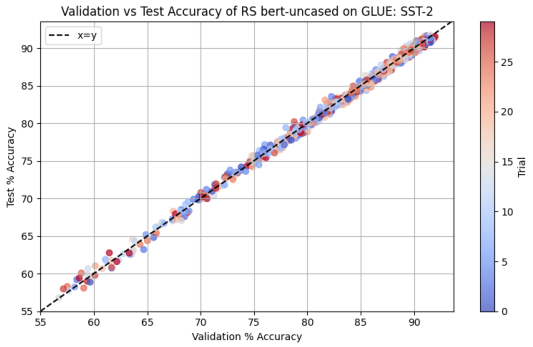
The validation and test accuracy performance of progressive GLUE: SST-2 fine-tuning hyperparameter optimization trials are plotted in Figure 13, with each model being displayed in a row and split by sampling method into two adjacent sub-figures. Each dot represents the accuracy performance of a given trial with the x-axis representing performance on the validation dataset, and the y-axis representing performance on the test dataset. Additionally, the color of each dot indicates how far along the optimization study the trial is, ranging from blue to red to represent start to end.

The larger models (pythia and MiniLM-uncased) show clear signs of being stuck with low performance values, which we suspect might be do to the range of hyperparameter configuration space. Obtaining a hyperparameter configuration that is not conducive to learning actual patterns in the data could be causing the model to either consistently head towards less optimal weights, or learn at such a low rate that the learning time is insufficient to have a noticeable improvement.

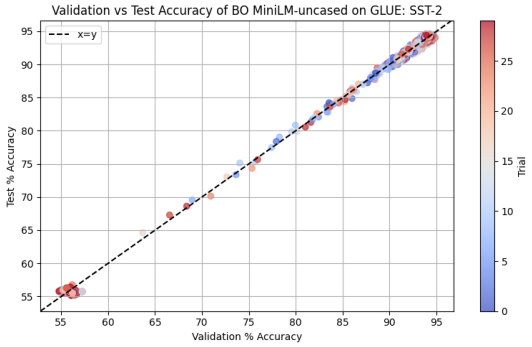
The decoder models performed worse on this binary classification task, with the best performances being around the 84% to 92% range compared to the 92% to 96% range of encoders. Bayesian optimization methods show higher concentrations of later trials near the best performance estimates, while later trials are spread randomly across the performance ranges. Finally, comparing the right-most trials (best validation) to the x=y baseline provides intuitive understanding on why overfitting at the hyperparameter level could occur, which is especially clear when looking at Bayesian optimization.



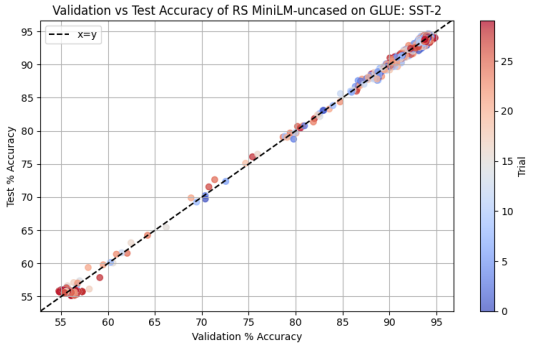
(a) bert-uncased (Bayesian)



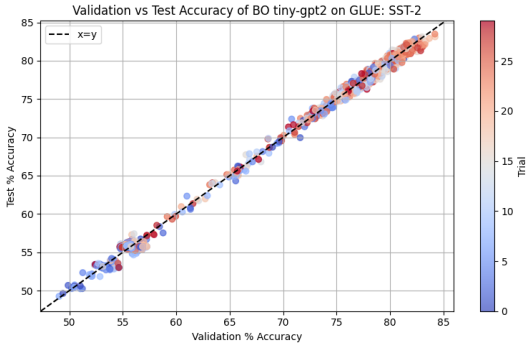
(b) bert-uncased (Random)



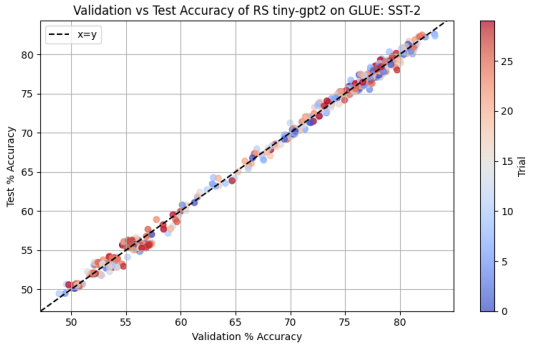
(c) MiniLM-uncased (Bayesian)



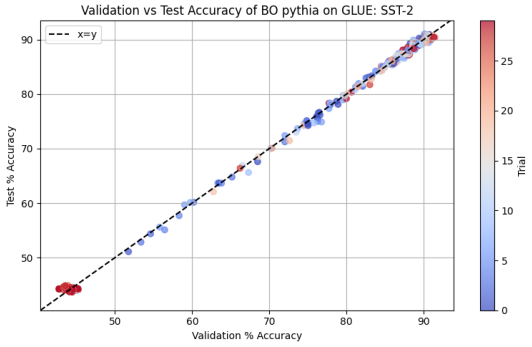
(d) MiniLM-uncased (Random)



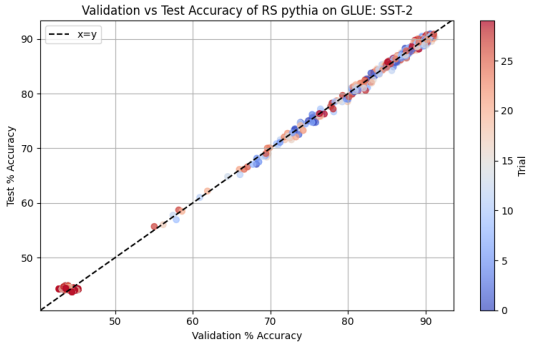
(e) tiny-gpt2 (Bayesian)



(f) tiny-gpt2 (Random)



(g) Pythia (Bayesian)



(h) Pythia (Random)

Figure 13: Accuracy performance results of each trial for all SST-2 fine-tuning studies, separated by model and sampling method with each colored dot representing a trial.

C Individual results of SuperGLUE: BoolQ

The progression of meta-overfitting ($\mathcal{M}$, orange) and overtuning ($\mathcal{G}$, blue) (Eq (8) and (9)) for each model fine-tuned on SuperGLUE: BoolQ are shown in Figure 14, with the mean trajectories for Bayesian optimization and random search being split in both phenomena and bands surrounding these trajectories being one standard deviation obtained from 30 repetitions.

Visually, the larger models (pythia and MiniLM-uncased) experienced less variance and overtuning than the smaller models, while all models show increased meta-overfitting and overtuning by the final trials when using Bayesian optimization regardless of size.

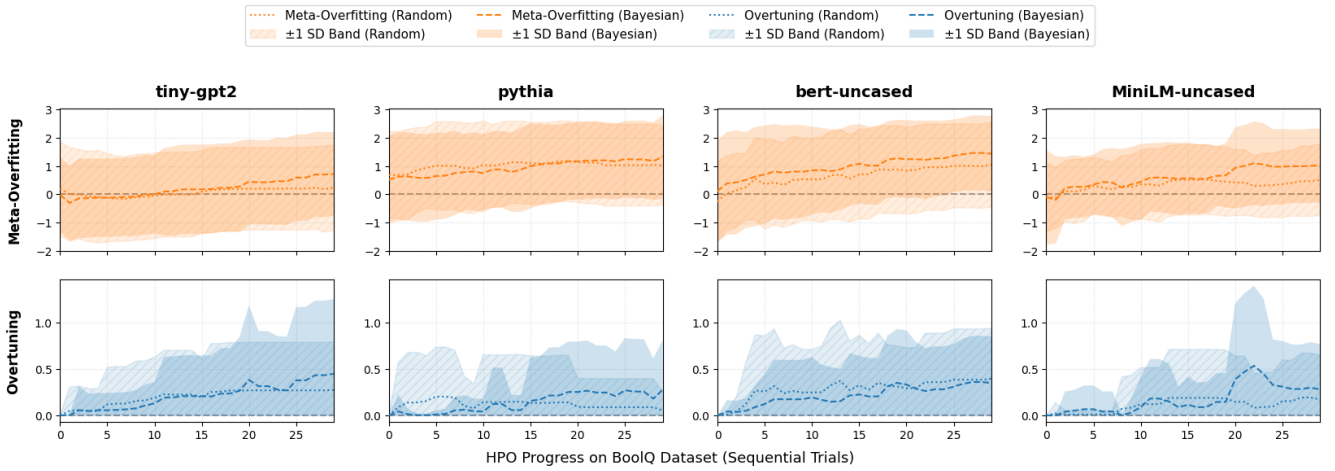
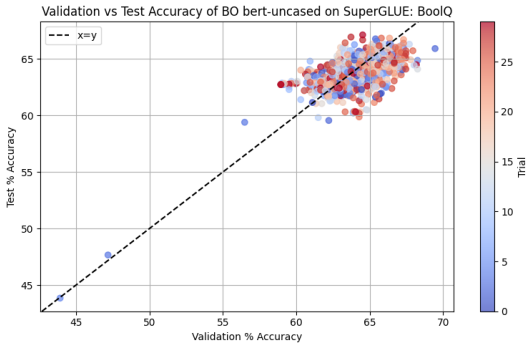


Figure 14: Progression of average meta-overfitting and overtuning for individual models fine-tuning on BoolQ, with a standard deviation band obtained from 30 iterations.

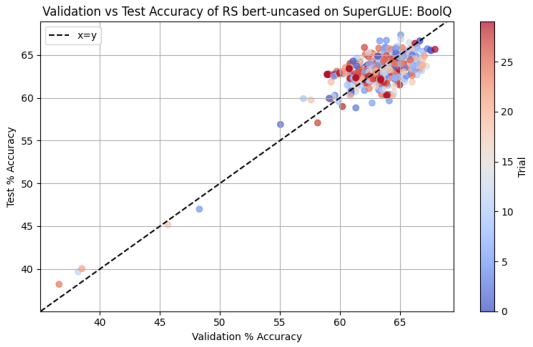
The validation and test accuracy performance of progressive SuperGLUE: BoolQ fine-tuning hyperparameter optimization trials are plotted in Figure 15, with each model being displayed in a row and split by sampling method into two adjacent sub-figures. Each dot represents the accuracy performance of a given trial with the x-axis representing performance on the validation dataset, and the y-axis representing performance on the test dataset. Additionally, the color of each dot indicates how far along the optimization study the trial is, ranging from blue to red to represent start to end.

As the BoolQ dataset does not have many data points that could be used for training, most models experienced a phenomena where they were sometimes unable to learn and improve their performance with certain hyperparameter configurations. This resulted in the performances to center around two values, with an exception of the bert-uncased model which did not appear to encounter this worse performing local minimum.

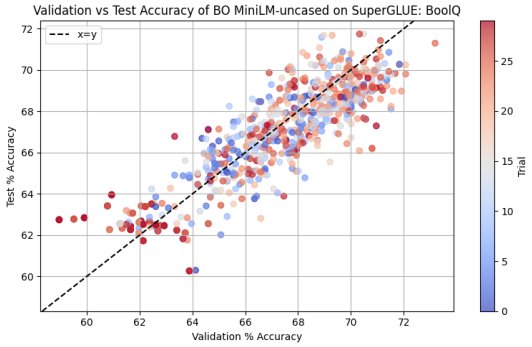
The decoder models performed worse on this binary classification task, with the best performances being around the 60% to 70% range compared to the 65% to 75% range of encoders. Bayesian optimization methods show higher concentrations of later trials near the best performance estimates, while later trials are spread randomly across the performance ranges. Finally, comparing the right-most trials (best validation) to the x=y baseline provides intuitive understanding on why overfitting at the hyperparameter level could occur.



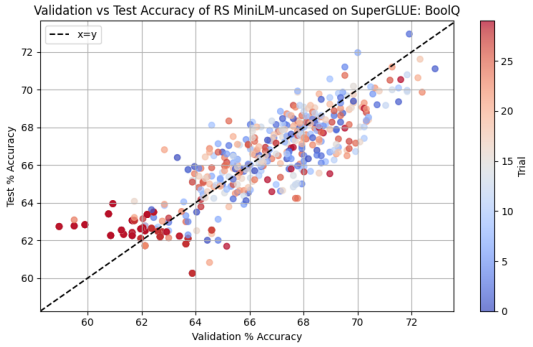
(a) bert-uncased (Bayesian)



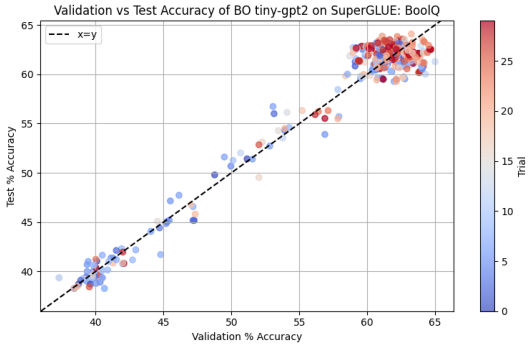
(b) bert-uncased (Random)



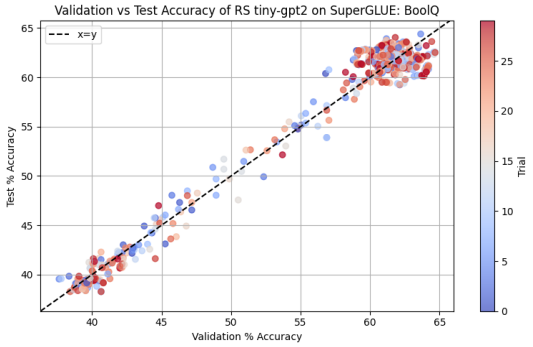
(c) MiniLM-uncased (Bayesian)



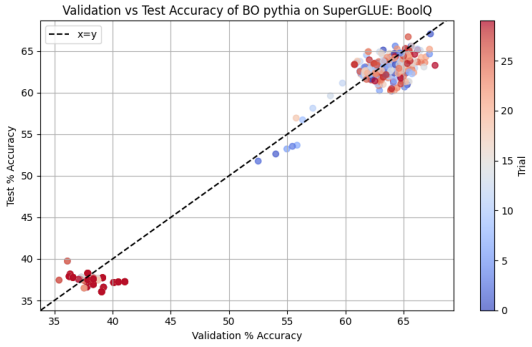
(d) MiniLM-uncased (Random)



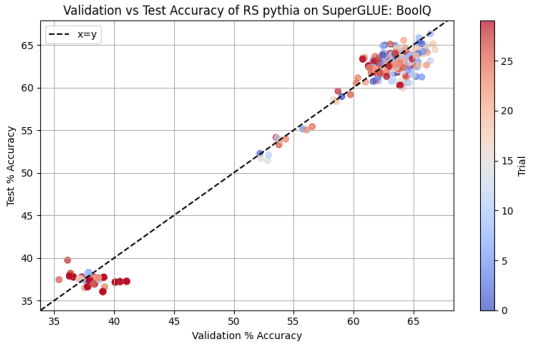
(e) tiny-gpt2 (Bayesian)



(f) tiny-gpt2 (Random)



(g) Pythia (Bayesian)



(h) Pythia (Random)

Figure 15: Accuracy performance results of each trial for all BoolQ fine-tuning studies, separated by model and sampling method with each colored dot representing a trial.

D ANOVA

The ANOVA (analysis of variance) statistical method that can be used to compare the average scores of multiple groups to see if any is significantly different from the others, which allows for testing of the multiple groups without inflating the chances of statistical errors.

Due to the skewed distributions of meta-overfitting and overtuning, using normal factorial ANOVA could result in false positives. However, there are a few ideas which could be explored with regards to the selection of the statistical method, the first being the fANOVA method and the second being running the experiments on more datasets to allow for the Wilcoxon Signed-Rank test multiple times and accounting for multiple testing.

The summary tables below show the results of both the four-way ANOVA results which would average the effects of both tasks and the task separated three-way ANOVA results, all tables show the degrees of freedom, sum of squares, F-value, p-value, and η_p^2. The results were originally only considered the results of p-value to determine the significance and η_p^2 to determine the effect size, a p-value equal to or less than .05 was taken as statistically significant and an $\eta_p^2 < .06$ being a small effect, $\eta_p^2 < .14$ is a medium effect, and $\eta_p^2 \geq .14$ being a large effect. Significant effects are highlighted in both bold and underline.

Table 5 shows that task choice had a significant effect on both meta-overfitting and overtuning (p-values under .001 for both phenomena) and the effect size was of a medium scale, these results highlight the importance of taking the underlying fine-tuning task into account when making any assumptions regarding the effects.

Hyperparameter configuration sampling method is shown to have a statistically significant effect of a small scale as a main effect to both meta-overfitting and overtuning, while model parameter count showed a significant effect of small scale on overtuning on its own and when combined with task choice. Model architecture didn't show a significant effect on either phenomena on its own, however the interaction with model parameter count did on a small scale for meta-overfitting even after adding the interaction with task. All other effects and interactions are statistically insignificant.

Tables 6 and 7 provide a summary of the ANOVA results for the separated GLUE: SST-2 and SuperGLUE: BoolQ performances respectively, which together highlight the observation that the studied effects had more significance in the smaller BoolQ task. This observation does not change our stated results from Table 5, as they are the number of data points from each task are equal and therefore the outcomes are averaged out to better match datasets which neither highlight nor inhibit the influence of our studied effects. This point is broadly addressed in Section 5 and Appendix A, which states the benefits of introducing different tasks and conducting the experiment with different models.

Table 5: Four-way ANOVA results for meta-overfitting and overtuning

		Meta-overfitting				**Overtuning**			
Source	df	SS	F	$\boldsymbol{p}$	η_p^2	SS	F	$\boldsymbol{p}$	η_p^2
Main Effects									
Task (T)	1	64.773	57.425	**.000**	.110	7.170	48.096	**.000**	.094
Sampler (S)	1	6.931	6.144	**.014**	.013	0.622	4.171	**.042**	.009
Architecture (A)	1	1.275	1.130	.288	.002	0.038	0.254	.614	.001
Parameter count (P)	1	0.107	0.095	.758	.000	1.024	6.867	**.009**	.015
2-Way Interactions									
T × S	1	3.912	3.468	.063	.007	0.228	1.532	.216	.003
T × A	1	0.686	0.608	.436	.001	0.038	0.254	.614	.001
T × P	1	0.759	0.673	.412	.001	0.757	5.081	**.025**	.011
S × A	1	0.020	0.018	.894	.000	0.300	2.015	.156	.004
S × P	1	0.007	0.006	.936	.000	0.044	0.293	.589	.001
A × P	1	11.210	9.938	**.002**	.021	0.016	0.109	.741	.000
3-Way Interactions									
T × S × A	1	0.177	0.157	.692	.000	0.141	0.946	.331	.002
T × S × P	1	0.004	0.004	.951	.000	0.122	0.816	.367	.002
T × A × P	1	9.667	8.570	**.004**	.018	0.021	0.142	.707	.000
S × A × P	1	0.419	0.372	.542	.001	0.036	0.243	.622	.001
4-Way Interaction									
T × S × A × P	1	0.041	0.037	.848	.000	0.009	0.060	.807	.000
Residual (Error)	464	523.372	–	–	–	69.171	–	–	–

Table 6: Three-way ANOVA results for meta-overfitting and overtuning from GLUE: SST-2

		Meta-overfitting				**Overtuning**			
Source	df	SS	F	$\boldsymbol{p}$	η_p^2	SS	F	$\boldsymbol{p}$	η_p^2
Main Effects									
Sampler (S)	1	0.214	1.167	.281	.005	0.048	5.641	**.018**	.024
Architecture (A)	1	0.045	0.247	.620	.001	0.000	0.000	1.000	.000
Parameter count (P)	1	0.148	0.806	.370	.003	0.010	1.170	.280	.005
2-Way Interactions									
S × A	1	0.158	0.860	.355	.004	0.015	1.737	.189	.007
S × P	1	0.000	0.001	.975	.000	0.010	1.144	.286	.005
A × P	1	0.029	0.155	.694	.001	0.000	0.019	.891	.000
3-Way Interactions									
S × A × P	1	0.098	0.536	.465	.002	0.005	0.538	.464	.002
Residual (Error)	232	42.616	–	–	–	1.985	–	–	–

Table 7: Three-way ANOVA results for meta-overfitting and overtuning from SuperGLUE: BoolQ

		Meta-overfitting				**Overtuning**			
Source	df	SS	F	$\boldsymbol{p}$	η_p^2	SS	F	$\boldsymbol{p}$	η_p^2
Main Effects									
Sampler (S)	1	10.628	5.129	**.024**	.022	0.802	2.769	.097	.012
Architecture (A)	1	1.915	0.924	.337	.002	0.038	0.254	.614	.001
Parameter count (P)	1	0.718	0.347	.557	.000	1.024	6.867	**.009**	.026
2-Way Interactions									
S × A	1	0.039	0.019	.891	.000	0.427	1.473	.226	.006
S × P	1	0.011	0.005	.941	.000	0.155	0.537	.464	.002
A × P	1	20.848	10.061	**.002**	.042	0.037	0.129	.720	.001
3-Way Interactions									
S × A × P	1	0.362	0.175	.676	.001	0.041	0.140	.709	.001
Residual (Error)	232	480.756	–	–	–	67.186	–	–	–