Evaluating Adaptive Layer Freezing through Hyperparameter Optimization for Enhanced Fine-Tuning Performance of Language Models

by

Reinaldo Figueroa Parra

B.S. Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2023

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

© 2024 Reinaldo Figueroa Parra. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Reinaldo Figueroa Parra
Department of Electrical Engineering and Computer Science
August 13, 2024

Certified by: Fiona Murray
MIT Innovation Initiative Director, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Evaluating Adaptive Layer Freezing through Hyperparameter Optimization for Enhanced Fine-Tuning Performance of Language Models

by

Reinaldo Figueroa Parra

Submitted to the Department of Electrical Engineering and Computer Science
on August 13, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Language models are initially trained on large datasets, enabling them to extract patterns and establish rich contextual connections. When dealing with data scarcity, transfer learning has become the go-to method to use these models in specialized downstream tasks via fine-tuning. However, fine-tuning on small datasets can lead to overfitting and a lack of generalization. Generalization is crucial when deploying models that perform a sensitive tasks in a real world environment, as it dictates how well it performs on unseen data. Conversely, overfitting is highly likely to occur when training on small datasets. This thesis proposes and evaluates a new method for fine-tuning language models by adaptively choosing specific learning rates for each transformer layer that provide higher performance on in-domain low-volume datasets. Additionally, we explore which layers inside the models usually hold more contextual information from pre-training that might be valuable to keep 'frozen' when fine-tuning on small datasets. This analysis provides insights into fine-tuning approaches during initial experiments when data is limited. Our results demonstrate limited performance gains on certain models while achieving more significant gains on others when fine-tuning using our proposed method. Additionally, our work also provides valuable insight into per-layer importance of language models by showing that certain layers have a stronger direct correlation with the overall model accuracy.

Thesis supervisor: Fiona Murray
Title: MIT Innovation Initiative Director

Acknowledgments

I would like to express my gratitude to Fiona Murray, Faculty Director at Mission Innovation X (MIx) and thesis supervisor, Katy Person, Senior Program Manager at MIx, and Geoff Orazem, Founder of FedScout, for welcoming me at MIx and allowing me to be part of the team as a research assistant. My time working with them, their wisdom, and guidance, really paved the way during my journey as a researcher. I am glad I was able to form part of such a great and supportive team.

Last but not least, I would also like to thank my parents. Without their full and unconditional support I would not have made it this far. They have always been my biggest pillars of strength and motivation, and for that I will be eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the advent of a new generation of Large Language Models (LLMs) has revolutionized the field of Natural Language Processing (NLP). Models such as GPT (Generative Pre-Trained Transformer), Llama, Gemini, and Gemma have emerged as the state-of-the-art and most preferred models for a wide variety of complex tasks. However, these models require extremely high computational power and memory to train and deploy, limiting their accessibility to researchers and smaller organizations with fewer resources. Moreover, limited data availability remains one of the biggest challenges for these large models across various tasks, particularly in data classification. Consequently, older and smaller models can still outperform and be more cost efficient than the latest LLMs for certain tasks, including text classification. Examples of such models include BERT [1] (Bidirectional Encoder Representations from Transformers), and its variants, such as DistilBERT[2], and RoBERTa[3]. These BERT based models require significantly less computational resources and memory, which make them more accessible to researchers and smaller entities.

Regardless of the models being used, data scarcity remains a major concern. Numerous studies have explored methods to mitigate the impact of low data availability, including transfer learning, data augmentation, and synthetic data generation using Generative Adversarial Networks (GANs) [4–6]. In this thesis, we primarily focus on transfer learning. Transfer learning entails reusing the learned representations from a model pre-trained on large datasets for a new, related task. This approach assumes that the domain of the pre-training data is related to the domain of the data in the new task. Pre-trained models have gained a substantial understanding of language semantics and common language patterns, such as contextual relationships, grammatical structures, and the meanings of words in different contexts. These capabilities enhance their ability to understand and generate human-like text and can be reused on new text. Therefore, transfer learning involves fine-tuning pre-trained models on new data, retaining the semantic relationships already learned from extensive training corpora while integrating new relationships from the new data.

In this thesis, we introduce a new fine-tuning approach with the potential to surpass traditional techniques, thereby enhancing transfer learning capabilities. Our approach is based on the observation that different layers in BERT-based models capture varying levels of semantic information [7, 8]. Firstly, we analyze the influence of assigning different learning rates to each transformer layer and the embedding layer of BERT based models. This analysis allows us to evaluate the importance of different layers for the semantic understanding

of the model, determining when it would be beneficial to 'freeze' a layer to increase model generalization capabilities on low-volume datasets. Secondly, we propose a new method to optimize per-layer learning rates, adaptively freezing layers to maximize the model's generalization capabilities. This method will enable us to select a set of per-layer hyperparameters that improves the model's performance on unseen data across datasets of similar domains, rather than using a single learning rate for all layers.

Chapter 2

Related Work

Extensive research has been conducted on fine-tuning [4, 8–10], as it is regarded as an effective transfer learning method for utilizing powerful language models pre-trained on large datasets. Fine-tuning also has its shortcomings, which have been thoroughly researched and are going to be briefly introduced. In this section we dive deeper into our inspiration of why fine-tuning is and will stay as an important tool to reuse pre-trained models in the present, and the future. More specifically, we introduce other similar methods that have been explored to implement layer-wise learning rates, although different from the one being presented in our thesis.

The way we think about fine-tuning varies according to the type of model we are referring to. For example, due to the inherent differences in their architectures and the domain of the data they process, fine-tuning methods for Convolutional Neural Networks (CNNs) and LLMs are often approached differently. Therefore, while many of the ideas being discussed here relate to language models, they might not have the same effect on different architectures. Our focus in this section and this work entirely revolves around fine-tuning LLMs for text classification.

2.1 Fine-tuning as a Powerful Transfer Learning Method

Training LLMs from scratch demands vast amounts of data, significant computational power, and a robust infrastructure. Additionally, the energy costs associated with running this infrastructure is often underestimated. Consequently, pre-training Language Models is not a practical choice for the average researcher today. This highlights the importance of fine-tuning as the most viable solution. In this section, we provide insights on research on fine-tuning, discussing when it should be employed, and how it should be conducted based on the specific task, or domain.

2.1.1 Feature Extraction vs Direct Fine-tuning

When deciding how to best adapt a pre-trained model to a target task, two major approaches have been explored: feature extraction and direct fine-tuning. The paper "To Tune or Not to Tune? Adaptive Pretrained Representations to Diverse Tasks" by Peters et al. [9] provides

empirical results across multiple NLP tasks on these two approaches. Feature extraction involves freezing all pre-trained weights and training only the classification layers (often referred to as the 'classification head' in sequence classification tasks).

Peters et al. tested both strategies on the ELMo and BERT models for a variety of tasks and datasets. Overall, they found that both methods have comparable performance for both approaches. However, there was a clear distinction in performance depending on the task being performed. For example, for the task of text classification, freezing the model layers consistently outperformed fine-tuning in the case of ELMo, while there was little to no difference when trying these two approaches on BERT. Yet, BERT fine-tuning showed outstanding performance when used to perform Semantic Textual Similarity (STS) on three different databases(STS-B, SICK-R, and MRPC)[9]. This is attributed to the fact that models that use next-sentence prediction (NSP) objective during pre-training, such as BERT, perform particularly well on the STS task, indicating a strong alignment between the pre-training and target task. These results confirm that the similarity (or dissimilarity) between the pre-training and target tasks determines how effectively feature extraction or fine-tuning can enable a model to excel in the target task.

Therefore, we show in this thesis that an intermediate approach between full layer freezing and complete model fine-tuning ensures better performance for text classification tasks.

2.1.2 Other Approaches to Fine-tuning

There is no single correct approach to fine-tuning. The optimal method depends on the specific task, available data, and computational resources. In the paper "How to Fine-Tune BERT for Text Classification?", Sun et al. (2019) investigate different fine-tuning methods on BERT and compare their performance. Firstly, they reaffirm our statement that pre-trained models on large corpora are highly beneficial for NLP tasks, including text classification, as they eliminate the need to train a new model from scratch. The following are a subset of the approaches they proposed and evaluated:

1. This approach is threefold:
    - 1) Further Pre-training BERT on In-Domain Training Data.
    - 2) Optional Fine-Tuning BERT with multi-task learning if multiple related tasks are available.
    - 3) Fine-Tune BERT for the target task.
2. Layer selection, and layer-wise learning rate.

For (1), further pre-training is considered an important task since the data distribution that BERT was trained on might potentially be entirely different from the target domain. Therefore, pre-training BERT on the new data using BERT's original training objectives, masked language modeling (MLM) and NPS, becomes a clear first step. Then, one can optionally perform multi-task fine-tuning, by fine-tuning on all tasks simultaneously, which has proven effective in exploiting shared knowledge among multiple tasks. Finally, single-task

fine-tuning is performed for the task in question. Results showed that "BERT + withIn-Task Pre-Training + Fine-Tuning (BERT-IDPT-FiT)" method produced the lowest error rate, with the average error rate reduced by 18.57% when compared to the baseline.

In (2), Sun et al. investigate another set of strategies. The intuition behind layer selection is that the lower the layer is in the BERT model, the more general information it contains. Similarly, the higher the layer, the more specialized information it holds. Thus, it is a good idea to choose different learning rates, in decreasing order from the lower layer to the highest layer, for all transformer layers in BERT.

To do so, Sun et al. tested splitting the parameters θ into $\theta^1, ..., \theta^L$ where θ^l holds the parameters for the l-th layer of BERT. Then the following equation shows how the parameters are updated

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta), \tag{2.1}$$

Here, η^l represents the learning rate of the l-th layer, and we set η^L to be the base learning rate. Then, the learning rates for each layer is defined as $\eta^{k-1} = \xi \cdot \eta^k$, and ξ is the decay factor such that $\xi < 1$. The factor ξ guarantees that the lower layers have a lower learning rate than the higher layers. Several experiments were conducted using different decay factors and learning rates, yielding the following results:

Learning rate	Decay factor ξ	Test error rates(%)
2.5e-5	1.00	5.52
2.5e-5	0.95	5.46
2.5e-5	0.90	**5.44**
2.5e-5	0.85	5.58
2.0e-5	1.00	5.42
2.0e-5	0.95	**5.40**
2.0e-5	0.90	5.52
2.0e-5	0.85	5.65

Figure 2.1: Graph representing the test error rates for different learning rates and decay factors. Taken from Sun et al. (2020) analysis of layer-wise decreasing layer rate.[4]

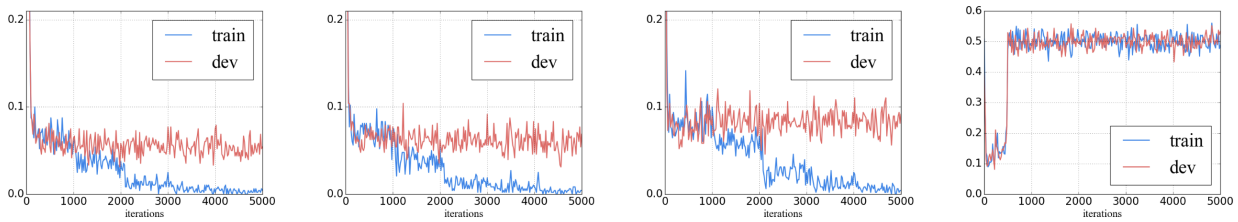
These results show that we can achieve a slight decrease on the model's test error rates by using a decay factor ξ. In our thesis, we explore this idea of layer-wise learning rate further.

2.2 Catastrophic Forgetting

The problem of 'Catastrophic Forgetting' was also introduced and explored by Sun et al. (2020)[4]. Catastrophic Forgetting is an outcome of choosing a large learning rate for the transformer layers. It aggressively overwrites the previously acquired knowledge, such as the semantic information learned during pre-training, with new but limited knowledge from the task-specific data.

The definition of a 'large' learning rate varies depending on the model in question. In their work, Sun et al. state that a value bigger than 1e-4 for BERT's transformer layers already lead to catastrophic forgetting. In our thesis, early experimental trials consistently showed the presence of catastrophic forgetting whenever any transformer layer had a learning rate above the 1e-4 threshold. Consequently, we refrain from using learning rate values above 1e-4 in our proposed method.

The following graphs vividly illustrate the implications of Catastrophic Forgetting [4]. They depict the learning curves of error rates over 5000 iterations while fine-tuning BERT on the IMDb dataset with various learning rates ranging from 2e-5 to 4e-4.



(a) lr=2e-5 (b) lr=5e-5 (c) lr=1e-4 (d) lr=4e-4

Figure 2.2: Model error rates over 5000 iterations during fine-tuning BERT on the IMDb dataset with different learning rates. Shows how (d) fails to converge due to a large learning rate. Taken from Sun et al. (2020) on the Catastrophic Forgetting problem.[4]

The graphs show that for an aggressive learning rate of 4e-4 the model fails to converge. A learning rate of 1e-4 or lower is required for the model to effectively mitigate catastrophic forgetting.

Overall, all these results show that we can achieve a slight decrease on the model's test error rates by using a decay factor ξ.

From these results, Sun et al. were able to empirically confirm the following claims:

- The top layer of BERT is particularly useful for text classification.
- BERT can overcome catastrophic forgetting with an appropriately chosen layer-wise decreasing learning rate.
- Further pre-training using within-task and in-domain data can significantly boost the model's performance.

Chapter 3

Methodology

In this thesis, we analyze the impact of varying learning rates across transformer layers during fine-tuning. Additionally, we propose a novel method to fine-tune pre-trained language models on small datasets by leveraging the learned patterns of training hyperparameters. We benchmark our approach using a variety of language models and compare the results to traditional fine-tuning methods that do not employ specific per-layer hyperparameters.

3.1 BERT-based models for sequence classification

We briefly introduce the main characteristics of the models used in our work. It is important to note that both BERT and RoBERTa have two versions: 'base' and 'large'. The main differences between the two versions are the number of learnable parameters and attention heads. However, we use only the 'base' versions of these models in our work.

Characteristics	**BERT**	**RoBERTa**	**DistilBERT**
# of Layers	12	12	6
Hidden Size	768	768	768
Attention Heads	12	12	12
# of Parameters	110M	123M	66M
Max Sequence Length	512	512	512
Training Objective	MLM, NSP	MLM	MLM
Training Dataset(s)	BC + Wiki	BC + Wiki+	Same as BERT

Table 3.1: Architectural Characteristics of BERT-base, RoBERTa-base, and DistilBERT Models. All models were trained on BookCorpus(BC) and Wikipedia(Wiki), with RoBERTa being trained on additional datasets(+). [1–3]

3.2 Benchamrking Traditional Fine-Tuning Approach

In order to evaluate how well our newly proposed fine-tuning strategy performs, we establish a baseline by fine-tuning three widely popular transformer-based models already introduced.

We do so by using traditional fine-tuning methods which also implies using one single learning rate for the entire model. We will directly compare the performance of the traditional methods to that of our fine-tuning approach using layer-wise learning rates.

3.3 Hyperparameters

In this section, we present the hyperparameters we have picked for our baseline fine-tuned models and our reasoning behind this selection. We also clarify what parameters stay the same, and what changes during our adaptive layer freezing optimization task.

3.3.1 Batch Size, Epochs, and Learning Rate

Choosing a proper batch size plays an important role on guaranteeing training efficiency, model generalization, and convergence. A larger batch size can lead to significant time saving benefits during training due to higher parallelization at the cost of higher GPU memory utilization. In order to keep a consistent value across all models and experiments given the resources at our disposal, we decided to set our batch size to 32.

Epochs is a hyperparameter that is more directly involved in our experiments. When we use layer-wise learning rates, some learning rates will get extremely small and will technically 'freeze' said layer. Consequently, fine-tuning a model will take longer to converge. Therefore, for all our experiments we chose the number of epochs to be 35. We also implemented an stopping condition which stops fine-tuning if average training loss has stopped decreasing to avoid overfitting.

The learning rate determines the step size that an algorithm takes in the direction opposite to the gradient during optimization. A learning rate that is too high can cause the training to diverge and the model to lack generalization, and a learning rate too small can lead to slow convergence, or raises the chance of the algorithm getting stuck at a local minima.

Additionally, the learning rate is one of the most critical hyperparameters in our experiments. For all our experiments, the learning rates for all layers, except the transformer and embedding layers, is set to 2e-5. We found that this value is very common in the industry and that we should stick to it as a baseline learning rate value.

3.4 Hyperparameter Optimization Tools

In this section, we present an optimization framework that we utilize to perform our layer-wise learning rate freezing optimization experiments. This hyperparameter optimizer tool is named Optuna, and is widely known in the machine learning space. Optuna is a useful tool to find the optimal combination of hyperparameters as well as which hyperparameters have the biggest impact on the model's performance.

3.4.1 Optuna

Optuna is considered automatic hyperparameter optimization framework for machine learning. It is a powerful tool that allows us to define a list of parameters, their desired search spaces, and efficient state-of-the-art algorithms for sampling and search. Additionally, it is capable of pruning trials that are most likely to perform worse than the average. Optuna allows us to create a "study" which we declare our training cycle, and let it run for a certain number of "trials". After all trials have been completed, Optuna returns a lot of valuable information about the model and the training process. This information includes from the best set of hyperparameters to a graph that establishes their overall "Importance" on the model's performance, which we define next.

3.4.2 Optuna's Hyperparameter "Importance"

Optuna's 'Importance' module allows us to retrieve the impact of each modifiable hyperparameter on the performance of the evaluation metric we attempt to maximize. The 'Importance' module takes all the completed trials in a given study and returns each hyperparameter with their computed impotance scores. Optuna uses fANOVA [11] as its preferred (and default) method for analyzing hyperparameter importance. Other importance evaluators are Mean Decrease Impurity (MDI) and PED-ANOVA [12]; however, we use the default importance evaluator, fANOVA, for all of our experiments.

3.4.3 fANOVA

fANOVA [11] constitutes a tool that takes the hyperparameters and their performance metrics from all the complete trials of an Optuna study, and fits a random forest in order to capture the intrinsic relationships between the hyperparameters and the trial performance. Subsequently, it applies functional ANOVA to decompose the variance attributed to each hyperparameter and their interactions, to then aggregate these contributions and compute the overall importance of each hyperparameter interaction. One detail to keep in mind is that due to the inherent variability of the random initialization in random forests, each run of the fANOVA algorithms might yield different values. However, with enough trials, fANOVA yields similar results every time that is executed.

Sampler and Pruners

Samplers dictate how to sample from the distribution of each parameter present in the optimization problem. We use the default and preferred sampler already implemented in Optuna: Tree-structured Parzen Estimator (TPESampler). TPESampler is a bayesian optimization based method that iteratively models the objective function, focuses on balancing exploration-exploitation, but explores promising areas.

Pruners can be a powerful tool during optimization as they have the task to end trials early if they show indicators of bad performance. However, we refrain from using them in this work as our goal is to present the full picture.

3.5 Adaptive Layer Freezing through Hyperparameter Optimization

Our adaptive layer freezing approach consists of two main goals. Firstly, we want to take a deep look at the hyperparameter importance of the transformer and embedding layers. More specifically, we inspect the patterns that emerge and guarantee better performance of the model on unseen data. Secondly, we want to use this knowledge to pick layer-wise learning rates that will guarantee increased performance on other datasets.

Here is a detailed set of steps explaining our algorithm:

- Select one database for a given classification task, and split it into three sets: training, validation, and test sets. Additionally, make sure to always take a small (<10% of the total number of samples of the dataset) but balanced sample of the full dataset as your training set.
- Set up your training and validation loops to work with Optuna (via an 'study').
- Fine-tune the model on one training dataset (we will use just the IMDb dataset for all our Optuna studies) for the set amount of epochs (35 is the value we have chosen as indicated previously) for a set number of trials. Each trial assigns a different set of learning rates for all the transformer layers, and the embedding layer. Set your optimization algorithm to maximize validation accuracy.
- Note that a high number of trials will ensure a broader exploration of learning rate combinations for each layer. This process will ultimately lead to finding the most adequate set of hyperparameter for this task. Therefore, choose to do >200 trials (this number depends heavily on the sample distribution of the learning rates that are part of the optimization problem).
- After the optimization study is over, utilize the information it yields to investigate the importance and correlation of layer-wise learning rates during fine-tuning, with validation accuracy as the maximized metric. We hypothesize this method yields a set of layer-wise learning rates that achieve better generalization on unseen data.
- Take a new dataset and fine-tune the model using the newly discovered set of learning rates for each layer. It allows you to check the hypothesis that layer-wise learning rates help the model to perform better on unseen data for the same tasks. More importantly, compare this result with that of our baseline metrics which use the same learning rate for the entire model.

These set of steps will guarantee we fully test our hypothesis that carefully chosen layer-wise learning rates will perform better on unseen data than models fine-tuned with the same learning rate for every layer.

Chapter 4

Results

The results of our experiments, conducted according to the procedures established in the previous section, are showcased in sections 4.1 – 4.3 and their corresponding subsections.

Firstly, our layer-wise importance analysis reveals that for the task of binary classification using the DistilBERT model, Transformer Layer 3 (shown as *l3* in 4.1) has a significantly greater influence on the overall model performance. Specifically, the learning rate assigned to *l3* showed a direct correlation with the model's accuracy metric, which further suggested that lower learning rates values for *l3* (values between $2e-7$ and $2e-10$) consistently guaranteed higher accuracy.

For the BERT model (which has 6 additional transformer layers for a total of 12), we see a similar pattern to that of DistilBERT. However, in this case, Transformer Layer 8 (*l8*) has more relevance, while other layers such as *l11* and *l5* also showed relatively high importance scores when compared to the others. RoBERTa showed a pattern similar to BERT, but in this case, Transformer Layer 8 (*l8*) showed an even more pronounced dominant importance compared to all other layers. After running over 200 Optuna trials for each model, we can confidently guarantee that the importance scores converge to these values. Interestingly, our layer importance results do not directly align with the conclusions highlighted in 2 as the top layers of DistilBERT, BERT, and Roberta did not prove to be more relevant for the text classification task.

Our layer-wise learning rate experiments via adaptive layer freezing showed similar patterns to those shown in 2.1.2. Despite adaptive layer freezing offering a more flexible and dynamic approach than layer-wise decay for choosing learning rates, the accuracy gains were not significant with less than 1% gains on our experiments involving DistilBERT and BERT models. However, RoBERTa demonstrated a significant superiority, achieving success rates almost 3% higher than the baseline in some cases. We attribute these remarkable results to several factors: RoBERTa is trained on a much larger and more diverse dataset, its focus on the MLM training objective without NSP task used in BERT, its use of Byte-Pair Encoding (BPE) for tokenization, and other architectural optimizations. These results provide us with meaningful insight about our methods. Improving model performance across multiple datasets with different domains through the assignment of layer-wise learning rates is non-trivial, and seems to be directly related to the specifics of a given model and dataset. As a result, performing adaptive layer freezing did not consistently seem to maintain linguistic patterns inherent in the transformer layers that would boost model generalization

capabilities. Nonetheless, our experiments demonstrated that using language models such as RoBERTa can still provide more generalization on unseen data while fine-tuning using our adaptive layer freezing method presented in this thesis.

4.1 DistilBERT Binary Classification

4.1.1 Importance

Here are the importance scores (they add up to 1) that each transformer layer in DistilBERT obatined based on the influence of their assigned learning rates.

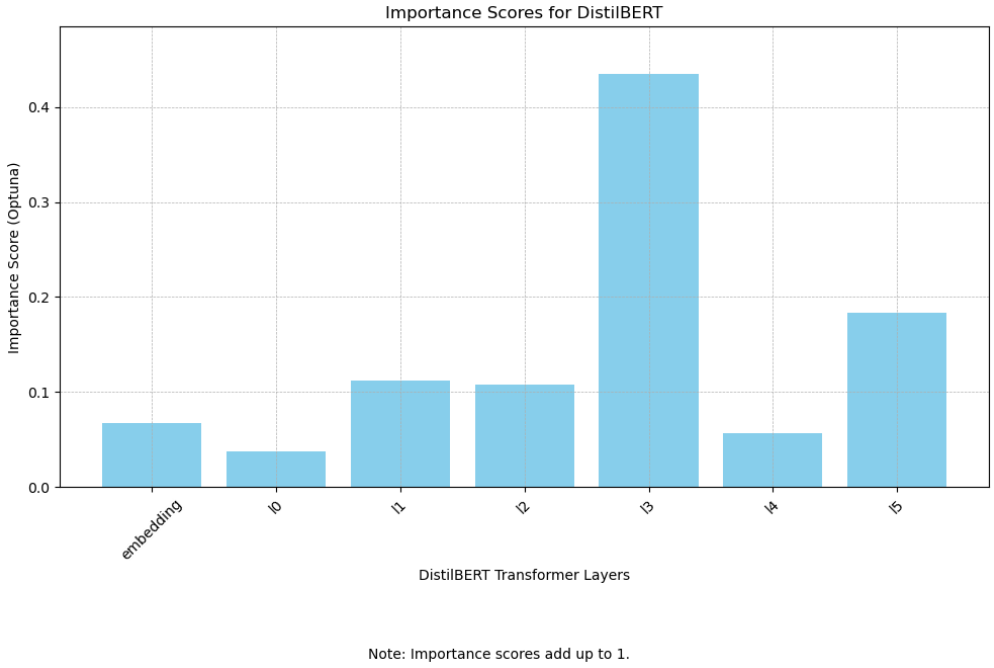


Figure 4.1: Importance scores (computed via Optuna) representing the influence of each DistilBERT transformer layer (6 layers) and the embedding layer on the model's overall performance in binary classification tasks.

4.1.2 Evaluation of Adaptive Layer Freezing on DistilBERT

	IMDb	YELP-P	SST-2	Amazon Polarity
Feature Extraction	0.7752	0.8377	0.74496	0.78528
Baseline Fine-Tuning	0.8604	0.9125	0.79234	0.87702
DistilBERT-ALF-1	**0.8729**	0.89167	**0.80746**	0.87399
DistilBERT-ALF-2	**0.875**	0.8979	**0.80141**	0.875

Table 4.1: Comparison of Different Fine-Tuning Methods Across Datasets Using DistilBERT. Values show the success rate (for a maximum of 1). DistilBERT-ALF-1 represents fine-tuning with the best set of layer-wise hyperparameters suggested by Optuna at the end of the study. DistilBERT-ALF-2 represents fine-tuning with another set of promising layer-wise hyperparameters that also produced compelling results.

Model	Emb.	l0	l1	l2	l3	l4	l5
DistilBERT-ALF-1	2e-8	2e-10	0	2e-7	2e-5	2e-8	2e-7
DistilBERT-ALF-2	2e-6	2e-10	2e-10	2e-10	2e-8	2e-10	2e-5

Table 4.2: Each set of learning rates for the embedding and transformer layers of the DistilBERT model, as a result of our Optuna studies.

4.2 BERT Binary Classification

4.2.1 Importance

As before, here are the importance scores (they add up to 1) that each transformer layer in BERT according to their assigned learning rates.

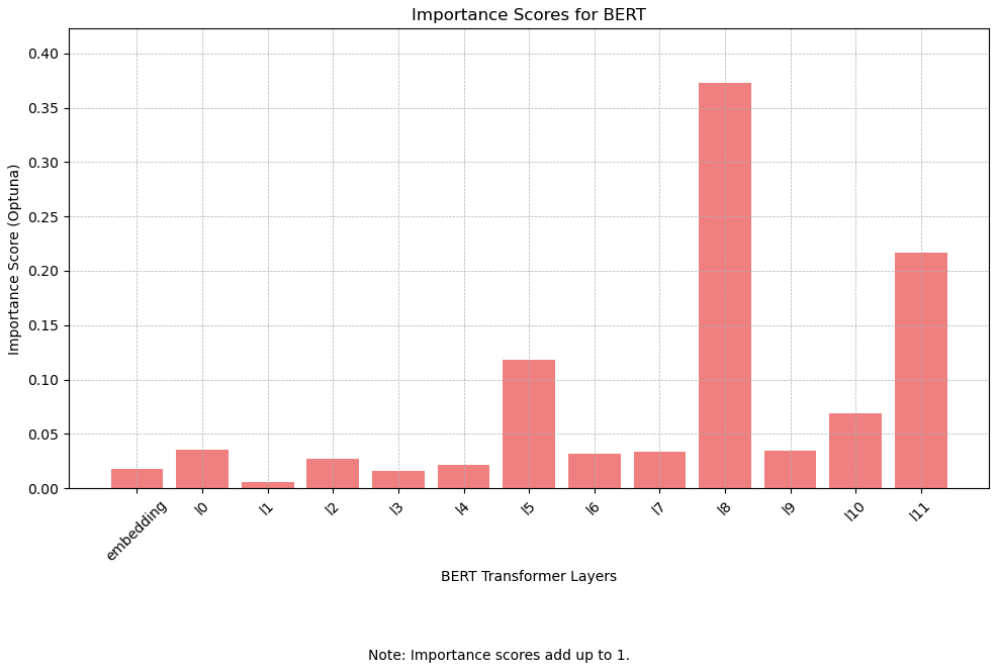


Figure 4.2: Importance scores (computed via Optuna) representing the influence of each BERT transformer (12 layers) layer and the embedding layer on the model's overall performance in binary classification tasks.

4.2.2 Evaluation of Adaptive Layer Freezing on BERT

	IMDb	**YELP-P**	**SST-2**	**Amazon Polarity**
Feature Extraction	0.55208	0.56875	0.56801	0.56985
Baseline Fine-Tuning	0.8708	0.93958	0.84375	0.92083
BERT-ALF-1	**0.87316**	0.93566	0.84167	0.89375
BERT-ALF-2	**0.88125**	0.93333	0.83542	0.89792

Table 4.3: Comparison of Different Fine-Tuning Methods Across Datasets Using BERT. Values show the success rate (for a maximum of 1). BERT-ALF-1 represents fine-tuning with the best set of layer-wise hyperparameters suggested by Optuna at the end of the study. BERT-ALF-2 represents fine-tuning with another set of promising layer-wise hyperparameters that also produced compelling results.

Model	Emb.	l0	l1	l2	l3	l4	l5	l6	l7	l8	l9	l10	l11
BERT-ALF-1	0	2e-9	2e-8	2e-5	2e-6	2e-7	2e-7	2e-6	2e-9	2e-5	2e-10	2e-6	2e-6
BERT-ALF-2	2e-6	2e-7	2e-8	2e-10	2e-6	2e-6	2e-7	2e-8	0	2e-5	2e-9	2e-8	2e-6

Table 4.4: Each set of learning rates for the embedding and transformer layers of the BERT model, as a result of our Optuna studies.

4.3 RoBERTa Binary Classification

4.3.1 Importance

Lastly, here are the importance scores (they add up to 1) that each transformer layer in RoBERTa according to their assigned learning rates.

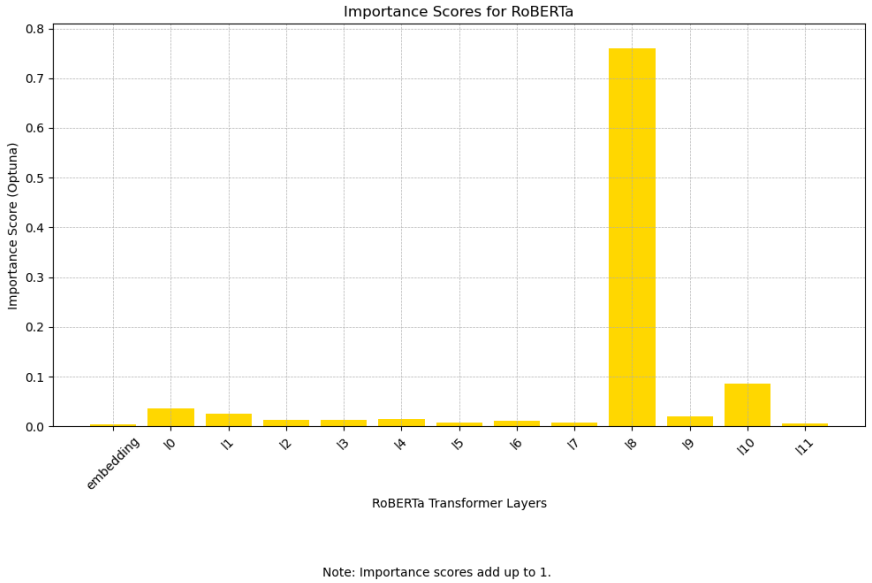


Figure 4.3: Importance scores (computed via Optuna) representing the influence of each RoBERTa transformer (12 layers) layer and the embedding layer on the model's overall performance in binary classification tasks.

4.3.2 Evaluation of Adaptive Layer Freezing on RoBERTa

	IMDb	**YELP-P**	**SST-2**	**Amazon Polarity**
Feature Extraction	0.80882	0.86213	0.54412	0.59191
Baseline Fine-Tuning	0.93199	0.95037	0.84926	0.94301
RoBERTa-ALF-1	0.92831	**0.95772**	**0.86397**	**0.94853**
RoBERTa-ALF-2	0.93015	**0.96691**	**0.87868**	**0.94669**

Table 4.5: Comparison of Different Fine-Tuning Methods Across Datasets Using RoBERTa. Values show the success rate (for a maximum of 1). RoBERTa-ALF-1 represents fine-tuning with the best set of layer-wise hyperparameters suggested by Optuna at the end of the study. RoBERTa-ALF-2 represents fine-tuning with another set of promising layer-wise hyperparameters that also produced compelling results.

Model	**Emb.**	**l0**	**l1**	**l2**	**l3**	**l4**	**l5**	**l6**	**l7**	**l8**	**l9**	**l10**	**l11**
RoBERTa-ALF-1	2e-5	0	2e-9	2e-9	0	0	2e-7	0	2e-6	2e-5	2e-9	2e-10	2e-8
RoBERTa-ALF-2	2e-9	2e-8	2e-6	2e-9	2e-8	0	2e-8	0	2e-10	2e-5	2e-8	2e-10	2e-7

Table 4.6: Each set of learning rates for the embedding and transformer layers of the RoBERTa model, as a result of our Optuna studies.

Chapter 5

Conclusions

Our goal in this work was to increase generalization and accuracy across DistilBERT, BERT, and RoBERTa models by gaining insight from layer importance and testing our adaptive layer freezing method as a continuation over previous techniques, such as layer decay. We did so by fine-tuning these models for the task of binary text classification using our adaptive layer freezing methodology, and testing our results on popular datasets, such as IMDB, YELP-P, Stanford SST-2, and Amazon Polarity.

After careful analysis, our results conclude that layer-wise learning rates have limited effects on certain models such as DistilBERT and BERT. However, applying this method on other models such as RoBERTa did provide impressive outcomes by achieving success rates in some cases almost 3% higher than the baseline. In other words, by varying the learning rate at each layer, some models (RoBERTa in our case) are capable of keeping more valuable knowledge from pre-training and learning new linguistic patterns during fine-tuning. Our results from our adaptive layer freezing approach matches the patterns of those showed in 2.1.2, leading to moderate accuracy gains. Consequently, our findings partially support our initial hypothesis and intuition that selectively freezing layers, which could potentially retain relevant linguistic information, would benefit the overall learning process of the model since there are still other models such as DistilBERT and BERT that did not show this behavior. This discrepancy may be due to several factors, such as their different architectural properties, or the limited capacity of these models to improve further without a larger and more diverse training set. Therefore, our work suggests that generalization in language models like RoBERTa can still be achieved by leveraging our adaptive layer freezing approach, thereby enhancing the fine-tuning performance of transformer-based models.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[2] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[4] C. Sun, X. Qiu, Y. Xu, and X. Huang. *How to Fine-Tune BERT for Text Classification?* 2020. arXiv: 1905.05583 [cs.CL]. URL: https://arxiv.org/abs/1905.05583.

[5] J. Wei and K. Zou. *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. 2019. arXiv: 1901.11196 [cs.CL]. URL: https://arxiv.org/abs/1901.11196.

[6] C. Little, M. Elliot, R. Allmendinger, and S. S. Samani. *Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study*. 2021. arXiv: 2112.01925 [cs.LG]. URL: https://arxiv.org/abs/2112.01925.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG]. URL: https://arxiv.org/abs/1411.1792.

[8] J. Howard and S. Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL]. URL: https://arxiv.org/abs/1801.06146.

[9] M. E. Peters, S. Ruder, and N. A. Smith. *To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks*. 2019. arXiv: 1903.05987 [cs.CL]. URL: https://arxiv.org/abs/1903.05987.

[10] M. Mosbach, M. Andriushchenko, and D. Klakow. *On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines*. 2021. arXiv: 2006.04884 [cs.LG]. URL: https://arxiv.org/abs/2006.04884.

[11] F. Hutter, H. Hoos, and K. Leyton-Brown. "An Efficient Approach for Assessing Hyperparameter Importance". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by E. P. Xing and T. Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 754–762. URL: https://proceedings.mlr.press/v32/hutter14.html.

[12] S. Watanabe, A. Bansal, and F. Hutter. *PED-ANOVA: Efficiently Quantifying Hyperparameter Importance in Arbitrary Subspaces*. 2023. arXiv: 2304.10255 [cs.LG]. URL: https://arxiv.org/abs/2304.10255.