Computationally Efficient Methods for Jointly Generating Predictions and Explanations

Mardhiyah Sanni

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2023

Abstract

In this dissertation, we explore the effectiveness of existing parameter-efficient fine-tuning methods in adapting the Text-to-Text Transfer Transformer (T5) and its UnifiedQA variant to jointly generate predictions and explanations using a few training examples. We also propose some novel combinations of these fine-tuning techniques, hypothesizing that this would result in orthogonal improvement of downstream performance. By evaluating all the fine-tuning methods against the Few Explanations Benchmark (FEB), our results reveal that on two of the four FEB tasks, our novel methods achieved performance competitive with fully fine-tuning T5-large by updating less than 1.2 % of the model's parameters. The insights drawn from this research show that no single computationally efficient method reigns supreme across all FEB tasks. Rather, the choice of the optimal fine-tuning strategy largely depends on factors such as the task's complexity, hyperparameter tuning, and computational constraints.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Mardhiyah Sanni*)

Acknowledgements

I am profoundly grateful to the numerous individuals and entities who have played pivotal roles in the completion of this research endeavor. Their unwavering support, guidance, and encouragement have been instrumental in shaping the trajectory of this study.

I am deeply grateful to all the individuals who have played pivotal roles in ensuring the successful completion of this study. Firstly, I extend my heartfelt gratitude to my esteemed supervisor, Dr. Pasquale Minervini, for his invaluable mentorship, insightful feedback, and scholarly contributions to the success of this research. I am also grateful to my colleagues in the MSc. Artificial Intelligence program for their camaraderie and discussions that allowed for continuous learning. I want to express my heartfelt thanks to my family and friends who provided unwavering encouragement and understanding throughout this academic journey. Lastly, I wish to acknowledge the countless mentors, scholars, and researchers whose contributions have paved the way for the advancement of knowledge in this field.

The contributions of these individuals have left an indelible mark on this work and my personal development. Thank you.

Mardhiyah Sanni
August 2023

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The integration of Large Language Models (LLMs) into practical applications such as virtual assistants and language translation systems has recently extended to decision-making systems that have a direct impact on human end-users. An example of this is the use of LLMs for automated content moderation in social media platforms. To ensure the trustworthiness of these models at deployment time, researchers have recently employed the use of natural language to provide human parseable explanations of the models' predictions. The choice of these Natural Language Explanations (NLEs) stems from the fact that they are easier to interpret than other forms of neural network explanations such as saliency maps [43, 47]. Models that generate these NLEs along with their predictions are termed self-rationalization models.

Despite the growing need for self-rationalization models, their adoption remains limited because training them often requires a vast amount of human-authored text explanations for every new task. As the cost of obtaining these explanations is prohibitively expensive, recent research in this space has centered around developing few-shot strategies for training self-rationalization models. A common approach used is fully fine-tuning LLMs using a few training examples. However, this often requires significant computational and storage resources to train and deploy these models, making it challenging and sometimes infeasible to develop and use self-rationalization models in resource-constrained applications. This motivates our study of computationally efficient methods for jointly generating predictions and their rationales.

1.2 Research Objectives

The primary objective of this study is to evaluate the suitability and effectiveness of existing parameter-efficient fine-tuning methods in fine-tuning both T5 [36] and its UnifiedQA [18] variant to perform the tasks in the Few Explanations Benchmark (FEB). We extend the work of Marasovic et. al. [29] that explored how prompt-based fine-tuning can be used to induce few-shot self-rationalization behaviour in T5 and UnifiedQA. To achieve this overarching goal, our research is guided by the following specific objectives:

1. Evaluate the performance of T5 and UnifiedQA on the four FEB tasks, after fine-tuning both the base and large versions of these models using the following parameter-efficient fine-tuning methods:
   - Prefix Tuning [21]
   - Low-Rank Adaptation (LoRA) [15]
   - Adaptive Low-Rank Adaptation (AdaLoRA) [55]
   - Infused Adapter by Inhibiting and Amplifying Inner Activations [25]
2. Explore and evaluate novel hybrid fine-tuning approaches that combine orthogonal parameter-efficient fine-tuning methods, leveraging their individual strengths with the aim of achieving improved downstream performance.
3. Investigate the suitability of all evaluated fine-tuning methods in the context of the FEB tasks, highlighting their strengths and limitations.

By addressing these research objectives, this study aims to contribute to the ongoing discourse surrounding computationally efficient methods of jointly generating predictions and explanations in the few-shot setting. Hence, enabling the wider adoption of self-rationalization models across various resource-constrained domains that require model transparency.

1.3 Expected Research Outcomes

Upon successful completion of this research, we expect to have provided meaningful contributions to the field of few-shot self-rationalization models, and the broader NLP space. By investigating the efficacy of various parameter-efficient fine-tuning methods of LLMs for few-shot self-rationalization tasks, we anticipate the following key outcomes:

1. Through an in-depth evaluation of existing parameter-efficient fine-tuning methods, we expect to gain insights into the suitability of each method for each of the four FEB tasks. In addition, by presenting the computational demands (i.e., the number of trainable parameters) and identifying possible avenues to save on these requirements (by studying the effects of hyperparameters that impact the number of parameters tuned) for each of these methods, we will contribute to the optimization of resource utilization. These insights would guide the selection of appropriate techniques for fine-tuning T5 to perform FEB tasks, in low-resource applications.

2. Also, by exploring the combination of orthogonal parameter-efficient fine-tuning methods, we aim to uncover new insights into how and why different fine-tuning strategies can either enhance or degrade a model's performance on FEB tasks. In-depth analysis of these combinations may offer novel avenues for future research.

Through these anticipated outcomes, this study seeks to inform future research on parameter-efficient fine-tuning methods as well as serve as a valuable resource for researchers, practitioners, and developers looking to deploy self-rationalization models in applications that require model transparency and interpretability. To ensure the research can be reproduced and extended, the code and evaluation results would be made publicly available.

1.4 Dissertation Structure

The dissertation is structured to comprehensively present our study's findings on parameter-efficient fine-tuning strategies for enhancing T5's downstream performance on few-shot self-rationalization tasks. This chapter, the Introduction, lays out the groundwork by introducing the research problem and its motivation, the objectives, and the expected outcomes of our study. Following this, the Related Work chapter delves into a thorough review of the background literature, examining works on transformer-based LLMs, parameter-efficient fine-tuning methods in Natural Language Processing (NLP), and self-rationalization models. In the Methodology chapter, the investigated methods and experimental setup are meticulously detailed, showcasing the justification behind the choice of tuned hyper-parameters for the various parameter-efficient fine-tuning techniques. The Results and Discussion chapter critically analyzes the outcomes of our experiments, providing insights into the effectiveness of the examined methods.

Lastly, the Conclusion chapter synthesizes the study’s findings, reiterating the research contributions, highlighting its limitations, and suggesting potential directions for future research in this evolving domain.

Chapter 2

Related Work

In this chapter, we explore the background concepts that underpin our study on applying parameter-efficient fine-tuning methods to the T5 model for joint generation of labels and rationales within the context of the Few Explanations Benchmark (FEB) tasks. We begin by reviewing the broader landscape of transformer-based Large Language Models (LLMs). Subsequently, we examine the existing parameter-efficient fine-tuning techniques for tuning these models and their effect on the model's performance for a variety of Natural Language Processing (NLP) tasks. We also explore the need and design evolution of self-rationalization models. Afterward, we discuss the architecture of the T5 model and its use in the development of self-rationalization models. Lastly, we identify gaps in the reviewed research and highlight the novelty and timeliness of our study.

2.1 Transformer-Based Large Language Models

Originally proven to be effective for machine translation tasks, the Transformer architecture [46] has re-shaped the Natural Language Processing (NLP) space by equipping researchers and practitioners with formidable building blocks for language modeling. Due to the ubiquitous nature of these models and their excellence at capturing the patterns and nuances of language, transformer-based models have widely been adopted to solve a wide array of NLP problems including text-to-text [36] and question-answering tasks [18, 30]. The key characteristic separating these transformer-based Large Language Models (LLMs) from previous recurrent neural networks used for language modeling, is their prowess at processing entire sequences simultaneously, allowing them model capture global dependencies between input and output sequences.

At the core of the transformer architecture lies blocks of multi-head attention mechanisms and point-wise fully connected layers, encapsulated in an encoder-decoder structure that initially encodes an input sequence to a sequence of continuous representations before autoregressively decoding this encoded representation on step at a time [46]. However, unlike previous recurrent neural networks that inherently model the temporal relationship between input embeddings, the attention mechanism is position-invariant in nature. To resolve this challenge, the transformer incorporates the use of positional encodings to inject sequential order information into the embeddings.

The training procedure of these LLMs generally involves two steps – pre-training on large, unlabeled, general-domain data in an unsupervised manner, followed by fine-tuning for specific NLP tasks. This two-step approach allows the model to initially capture general language knowledge by learning rich linguistic representations during pre-training, before learning task-specific nuances during fine-tuning. Following this technique, many LLMs such as BERT [12], GPT-3 [7] and T5 [36] have been developed, each demonstrating state-of-the-art performance on numerous NLP tasks as at the time they were introduced. These models generally have millions to billions of parameters, signifying their remarkable complexity.

However, this complexity and remarkable performance comes with several costs. Their need for vast training data, coupled with concerns that the inherent biases embedded within data can seep into their outputs, raises concerns about their ethical deployment. Additionally, the computational and storage implications for training and deploying these powerful, parameter-heavy models poses another challenge [15, 14]. Nevertheless, this field continues to evolve, with the development of new training methodologies, bias mitigation strategies and parameter-efficient fine-tuning techniques.

2.2 Parameter Efficient Fine-Tuning

In the preceding section, we noted that despite the exceptional ability of LLMs to understand and generate coherent and contextually appropriate text, their parameter-heavy nature often demands high computational and storage costs to fully fine-tune them on task-specific data. This has prompted research in the development of a family of model adaptation methods that optimize a relatively small percentage of the total parameters of the LLMs, consequently reducing the compute and storage requirements necessary for fine-tuning. These methods are commonly referred to as parameter-efficient fine-tuning methods. The existing parameter-efficient fine-tuning techniques either modify the structure of the model being adapted itself or focus on only changing the inner activations of the model without altering its overall architecture [25]. Methods that add trainable parameters to the model by modifying the model's architecture often result in additional inference latency due to the increased depth of the pre-trained model [25].

Early parameter-efficient fine-tuning research introduced adapter tuning [14, 24] as an alternative to storing all the parameters of a new model for every task by simply adding (and storing) only a few trainable parameters per task while keeping the initial parameters of the pre-trained model frozen. These adapters are new modules with near-identity initialization, injected between layers of a pre-trained network. There have been multiple proposed variations in the structure of these modules, and where they are injected in pre-trained LLMs. Houlsby et. al. [14] proposed a simple bottleneck architecture, that maps the original d-dimensional features to a smaller m dimension, applies a non-linearity, and then projects the m-dimensional vectors back to the initial d-dimension. In this proposed use of adapters, they injected 2 adapter layers per Transformer block in BERT - one after the projection following multi-head attention and the other after the two feed-forward layers [14]. However, Houlsby et. al. [14] noted that some injected adapters have more influence on the model's performance on new tasks than others. At the time this method was introduced, it achieved near state-of-the-art performance on the GLUE benchmark [48], while only adding and optimizing 3.6% parameters per task. Another variant of adapter tuning proposed by Lin et. al. [24] introduced one adapter module per Transformer block, but added additional layer normalization. Adapter tuning generally has the downside of introducing small but non-negligible inference latency due to the increased model depth. Also, the architecture of the adapted model requires sequential processing of tasks, thus not being able to take advantage of the parallelism most deep learning architectures provide.

Another model adaptation strategy similar to adapter tuning, in that they both inject trainable layers into pre-trained LLMs is Low-Rank Adaptation (LoRA) [15]. Hu et. al. [15] proposed LoRA, a tuning technique where trainable rank decomposition matrices are injected into each layer of the transformer and optimized during training while the pre-trained weights are kept frozen. These decomposition matrices are parameterizations of the updates to the weight matrices during training. LoRA decomposition matrices, like the adapters presented by [14], have a bottleneck structure that maps d-dimensional feature vectors to r-dimensional vectors, then projects the r-dimensional vectors back to d-dimensional ones, in essence, imposing a low-rank constraint on the weight increments during training [15]. Also like adapter tuning, [15] identified that the injected decomposition matrices have varying effects on model performance, depending on the layers of the transformer they are injected into. However, unlike the adapters, during inference, the learned weights of the incremental matrices can be seamlessly merged with the original weights of the pre-trained LLM, resulting in no additional inference latency. In addition, Hu et. al. [15] were able to show that this method generalizes to full-fine-tuning when the LoRA rank is set to the rank of the pre-trained weight matrices, unlike adapter-based tuning that generalizes to an MLP [15]. When LoRA was proposed, the experiments testing its effectiveness were limited to adapting the Query and Value matrices, but further experiments by [25, 55] have shown that for some models like DeBERTa and BART, it was beneficial to adapt both self-attention and MLP layers. When used to fine-tune RoBERTa [27], GPT2 [35] and GPT3 [6], Hu et. al. [15] noted that their method achieved on-par or better performance that full-fine-tuning while also stating that it is preferable to adapt more weight matrices than adapting a single type of weight with a larger rank.

Still in the space of leveraging the use of low-rank updates for parameter-efficient fine-tuning, Zhang et. al. [55] identified the ineffectiveness of evenly distributing the rank budget to all weight matrices during model adaptation with LoRA, since this essentially overlooked the varying importance of the different weight parameters to the downstream task. To resolve this, Zhang et. al. [55] proposed Adaptive Low-Rank Adaptation (AdaLORA), a method that adaptively allocates the rank budget among the incremental weight matrices during training according to their importance score. AdaLoRA iteratively prunes less important singular values after parameterizing the changes to the weight matrices to mimic Singular Value Decomposition (SVD). It uses this parametrization approach, instead of directly computing SVD of the weight matrices and pruning the smallest singular vectors, to avoid incurring the expensive computational cost of iteratively applying SVD for high-dimensional weight matrices. Via this approach, AdaLoRA consistently out-performed full-fine-tuning when used to fine-tune DeBERTa V3-base [13] and BART-large on GLUE [48], SQuADv1 [38] and SQuADv2 [37] benchmarks. This method offers the same advantages as LoRA, but in addition, removes the dependence on heuristics to select the optimal weight matrices to apply low-rank updates to.

There has also been parameter-efficient fine-tuning research geared in the direction of altering the inner activations of the model instead of re-parameterising its layers or changing the model's architecture. This activation modification often comes in the form of either concatenating learned vectors to input sequences or activations, or re-scaling the pre-trained model's inner activations. In the case of concatenating learned vectors to input sequences or activations, the most common approaches are prompt tuning [20] and prefix tuning [21]. Both these methods overcome the drawback of manual prompt design that requires extensive human involvement in defining optimal conditioning texts to steer a pre-trained model to perform new tasks without additional training. Prefix tuning concatenates continuous task-specific vectors, known as prefixes, to the actual input sequence, allowing the actual input tokens to attend to the prefixes as if they were 'virtual tokens' [21]. This results in the modification of the inner activations after every layer in the transformer during a gradient-based training that optimizes only the appended vectors. A simpler version of this method that achieves comparable performance to full-fine-tuning of large pre-trained models was presented by Lester et. al. [20]. In this method, the embeddings of selected prompts, known as 'soft prompts', are concatenated with the embedded input sequence, and the resulting sequence then flows through the encoder-decoder as normal [20]. Only the 'soft prompts' are optimised during training. However, research [20, 26] has shown that prompt tuning only performs comparably to the full-fine-tuning of models with over 10 billion parameters. To overcome this drawback, Liu et. al. [26] proposed an approach to prompt tuning that applies the continuous prompts to every layer of the pre-trained model, instead of only the input layer as proposed by Lester et. al. [20]. This slight modification allowed prompt tuning to give comparable performance to full-fine-tuning universally across various model scales [20]. It is important to note that this approach of tuning that generally appends vectors to input sequences leads to a reduction in the amount of actual input tokens that can be processed at the same time. Recently, a fine-tuning method where the intermediate activations of the pre-trained model are multiplied by trainable vectors has been presented as a viable alternative to prompt tuning that does not limit the sequence length of the actual input but still achieves strong performance on new tasks using few examples. This method is known as Infused Adapter by Inhibiting and Amplifying Inner Activations $(IA)^3$ [25].

Another line of research in parameter-efficient fine-tuning, referred to as sparse fine-tuning, capitalizes on the sparsity of the weight matrices of existing LLMs by pruning weights with small magnitudes or choosing a subset of parameters to modify. An example is BifFit [54], a task-invariant method where only the bias terms of the pre-trained model (or a subset of them) are updated during training. Zaken et. al. [54] showed that with only learning 0.1% of the model's parameters, this approach achieved competitive and sometimes, better performance than full-fine-tuning of BERT [12] on the GLUE benchmark. Later on, SparseFit [43] extended this method to fine-tune T5 [36] on the FEB [29] tasks, experimenting with training various layers and pairs of layers, to study their relative effects on the quality of the generated Natural Language Explanations (NLEs). SparseFit[43] was able to achieve competitive results with full-fine-tuning with only updating 6.8% of the T5's parameters. Lin et. al. [23] presented a sparsification method in conjunction with their Deep Gradient Compression (DGC) strategy, that prunes based on the magnitude of the gradient during training, instead of the weight's magnitude while retaining the original structure and number of parameters of the model. They were able to show that their proposed strategy allowed for distributed training and led to faster convergence during training.

2.3 Explainability of Deep Neural Networks

Recent research have pursued the interpretability and understanding of the inner workings of Deep Neural Networks. Many of these have been directed towards generating prediction rationales in human understandable form, and have explored two main approaches of achieving this - via post-hoc explanations, or developing of self-rationalisation models. Both these classes of methods have the same goal of increasing prediction confidence and ensuring that predictions are not based on spurious correlations or artifacts present in the training data. Post-hoc explanation generation generally involves analyzing a pre-trained model to uncover its decision strategy [40]. Many post-hoc explanation methods have been applied in a wide variety of fields including medical diagnosis [16, 3], recommender systems [32, 9, 53], and engineering systems [31, 39]. [4, 5, 45] all proposed post-hoc explanation methods that perform local analyses of the decision functions, and then provide prediction rationales in the form of relevance scores of the input features. These explanations generated by these methods often rely on the assumption that the end-user has an understanding of the training data, and therefore would be able to make sense of the relevance scores.

On the other hand, self-rationalization models are trained with the sole purpose of interpretability in mind, therefore, requiring explanations as part of their training data. Recent research has focused more on using free-form text explanations as these are easily understood by the end-user. Camburu et. al. [8] extended the Infersent architecture [11] with a module designed to generate the rationale behind the deep neural network's predictions. At training time, the true labels were pre-pended to their respective free-form text explanations, and the new model was trained to generate this new sequence while optimising a combination of both classification and negative log-likelihood losses. Nevertheless, the expensive cost of obtaining human-authored explanations to train self-rationalization models, and hence their scarcity, has led to research focused on zero and few-shot self-rationalisation models. An approach that integrates the generation of extractive rationales into the learning problem, without requiring any rationales provided at training time, was proposed by Lei et. al. [19]. Leveraging the intrinsic knowledge of LLMs on general domain data, [29] designed optimal prompts that can steer T5 and its UnifiedQA variant to jointly generate predictions and explanations given only a few examples at training time. While [29, 51, 43] noted that the generated explanations may not reflect how the models work internally, Narang et. al. [33] argued that these explanations generated by the self-rationalization models are more faithful and stable than post-hoc explanations since they are intrinsic to the model.

2.4 T5 and Fine-Tuning for Rationale Generation

The Text-to-Text Transfer Transformer (T5) [36], an architectural innovation based on the Transformer, has been a pivotal breakthrough in the field of NLP and in particular, Natural Language Generation (NLG). It is a unified framework that reframes a wide range of text-based tasks as text-to-text transformations [36]. At its core, it is an encoder-decoder structure, that encodes a continuous representation of input sequences and then decodes them into desired output sequences. The encoder consists of a stack of ‘blocks’, each of which comprises two main parts – a self-attention layer, and a following feed-forward network. The decoder’s structure is similar to that of the encoder, except it includes a standard attention mechanism after each self-attention layer, that attends to the output of the encoder. The UnifiedQA [18] model, on the other hand, is a pre-trained QA model that has the same architecture as the T5 model, but a different set of pre-trained weights. For brevity, we do not provide a comprehensive description of the architectures and pre-training objectives of both T5 and UnifiedQA, and thus refer interested readers to their original papers [36, 18].

The design of T5 enables it to seamlessly generate coherent and contextually relevant text sequences when performing various text-based language problems including classification, translation, summarisation, and question-answering. Numerous prior research [10, 44, 50] have explored the potential of fine-tuning T5 for various text generation problems. The process generally involves initializing the T5 architecture with the pre-trained weights, then via gradient-based training on the task-specific data, adjusting the model's parameters to align with the downstream task's target output distribution. Most self-rationalization models are currently based on T5 because it has a pre-training objective that optimises its performance on both classification and generation tasks, the two main tasks of self-rationalization models [29].

2.5 Summary of Existing Research Gaps

Amidst the recent surge in the development of few-shot self-rationalization models that generate free-form text NLEs and the increase in the development of parameter-efficient fine-tuning methods, gaps persist in the integration of these domains. Specifically, there is a scarcity of research investigating the application (and effectiveness) of these parameter-efficient fine-tuning techniques to pre-trained language models to perform the Few Explanations Benchmark (FEB) tasks. The urgency to address these gaps stems from the need for models that not only generate accurate predictions but also deliver plausible explanations that enhance user trust and foster interpretability in various resource-constrained domains.

This chapter sets the stage for our study by situating it within the broader landscape of natural language generation, parameter-efficient fine-tuning, and few-shot self-rationalization modeling. The synthesis of these domains guides our exploration into the effectiveness of various parameter-efficient fine-tuning methods in the fine-tuning of T5 and its UnifeidQA variant using few examples, while also exploring novel avenues of integrating the fine-tuning methods to obtain orthogonal improvement.

Chapter 3

Methodology

In this research, our primary focus is to study the effects of various parameter-efficient fine-tuning techniques in the fine-tuning process of T5 and its UnifiedQA variant for the generation of predictions and free-text rationales using only a few examples. Using the Few Explanations Benchmark (FEB) [29] as our dataset, we explore the efficacy of Prefix Tuning [21], Infused Adapter by Inhibiting and Amplifying Inner Activations [25], and leverage the intrinsic sparsity of the weights of these encoder-decoder transformer models by adapting them with Low-Rank Adaptation [15] and Adaptive Low-Rank Adaptation [55]. In addition, we also capitalize on the orthogonality of some of these methods to further improve the performance of the fine-tuned models on FEB.

3.1 The Few Explanations Benchmark

The Few Explanations Benchmark (FEB) is a meticulously curated collection comprising four English-language datasets and their associated metrics, with each dataset having human-authored free-text explanations. This benchmark thus encompasses four diverse tasks including natural language inference and commonsense question-answering tasks. In Table 3.1, we provide an overview of these datasets, their original end tasks, and the number of training samples per label (i.e. shots) included within the benchmark.

Following the work of [29] that fine-tuned the T5 model on the tasks in FEB, we reframe the self-rationalization problem as follows: Given an input sequence x, which is a concatenation of an 'instruction' (or hard prompt) and the task's features or main concepts, the objective is to autoregressively generate an output sequence y. This generated sequence is a concatenation of a prediction and the rationale behind that prediction. For example, in the case of the NLI task, x can be the sequence: ***explain nli***

Task	Dataset	Number of Shots
Natural Language Inference (NLI)	e-SNLI [8]	16
Multiple Choice Commonsense Question Answering	ECQA [1]	48
Nonsensical Sentence Selection	ComVE [49]	24
Offensiveness Classification	SBIC [42]	24

Table 3.1: The FEB Tasks

hypothesis: Pedestrians cross the water on the bridge premise: People walking across a bridge.$< /s >$, while the expected generated sequence would be: ***entailment because people walking are pedestrians***$< /s >$.

As we aim to generate these explanations with limited training samples in a computationally efficient manner, leveraging parameter-efficient fine-tuning methods becomes crucial. These techniques enable the optimization of model performance with reduced computational and memory costs while preserving the model's ability to generate plausible and coherent explanations for the tasks with the FEB.

3.2 Parameter-Efficient Fine-Tuning Methods for Rationale Generation

3.2.1 Prefix Tuning

Prefix tuning serves as a valuable technique to guide the pre-trained language model, T5, by introducing context through the pre-pending of continuous task-specific vectors to the actual input sequence x. In this setting, the tokens of x attend to these continuous vectors, known as prefixes, as if they were 'virtual tokens' [21], resulting in indirect modification of the inner activations of the model. During the fine-tuning process, all the pre-trained parameters remain frozen, and only the prefix is optimized for the specific end task. By doing so, this method effectively steers the model to generate more contextually appropriate explanations. Moreover, this approach yields a significant reduction in the amount of tuned parameters, making it a desirable parameter-efficient solution for generation tasks.

We evaluate this tuning strategy on all four tasks encompassed within FEB and investigate the influence of prefix length on the model's overall performance. Via this analysis, we assess the efficacy or suitability of prefix tuning for fine-tuning T5 to perform few-shot generation of Natural Language Explanations (NLEs).

3.2.2 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) of LLMs is a tuning strategy grounded in the premise that the low 'intrinsic dimension' [2] of over-parameterized LLMs leads to their weight updates during fine-tuning to have low intrinsic rank. This insight is leveraged via the parametrization of the weight updates, also referred to as incremental matrices, as a product of two low-rank decomposition matrices. These rank decomposition matrices are injected into layers of the transformer and are the only matrices optimized during model adaptation to a downstream task.

The key advantage of this technique lies in the substantial reduction of trainable parameters without increasing inference latency [15]. Also, this method does not reduce the usable sequence length for the actual input, unlike in prefix tuning where a portion of the model's input is allocated to the prefix. However, an essential consideration when tuning with LoRA within a limited resource budget is the determination of the layers within the transformer model where these decomposition matrices should be injected, as well as the optimal rank for these matrices. The selection criteria generally hinges on the importance of these modules (or layers) in performing the specific end task the model is being adapted to.

In this study, we limit our search to T5 layers that have shown the most significance in generation tasks in general, as the research on T5 layer significance in NLE generation is still quite limited. We also investigate the effect of the rank of the decomposition matrices on the downstream performance of the adapted models. Through these analyses, we aim to evaluate the applicability and effectiveness of LoRA in fine-tuning T5 for few-shot NLE tasks.

3.2.3 Adaptive Low-Rank Adaptation

Adaptive Low-Rank Adaptation (AdaLoRA) is a fine-tuning technique that addresses the limitations of LoRA when given a fixed budget of tunable parameters. LoRA evenly distributes the parameter budget by allocating equal ranks to all injected trainable matrices, in essence, overlooking the varying importance of the weight matrices in the transformer layers, in relation to the downstream task that the model is being adapted to. In contrast, AdaLoRA offers a more adaptive and efficient importance-aware rank allocation [55] of the budget.

In particular, AdaLoRA is a tuning strategy that performs LoRA-like fine-tuning but adaptively allocates the parameter budget among incremental weight matrices according to their respective importance score [55]. It parameterizes these incremental matrices in the form of Singular Value Decomposition (SVD) and then prunes the redundant singular values. Via this 'adaptive allocation', this method assigns a greater number of trainable parameters (i.e., by increasing rank) to critical weight matrices, allowing the model to capture more task-specific information, and therefore, improving the overall model performance. Conversely, for less important weight updates, where additional trainable parameters offer marginal performance improvement, or in some cases, even result in performance degradation, AdaLoRA prunes their singular values, resulting in lower-rank matrices.

To achieve this parameter allocation, Zhang et. al. [55] defined the parameter budget, $b^{(t)}$, as the total rank of all the incremental matrices and presented a cubic schedule to gradually decrease $b^{(t)}$ from $b^{(0)}$ to the target budget, $b^{(T)}$, via iterative pruning of redundant singular values. This global budget scheduler is given as:

$$b^{(t)} = \begin{cases} b^{(0)} & 0 \leq t < t_i \\ b^{(T)} + (b^{(0)} - b^{(T)})(1 - \frac{t-t_i-t_f}{T-t_i-t_f})^3 & t_i \leq t < T - t_f \\ b^{(T)} & \text{otherwise} \end{cases}$$

During fine-tuning with AdaLoRA, the model is initially trained (i.e., warmed-up) for t_i steps, before pruning commences. After pruning is completed due to the target budget being reached, the model is then further trained for t_f steps.

This technique offers similar advantages to LoRA, as described in section 3.2.2, but unlike LoRA, it does not require a manual search for the optimal layers to inject the trainable matrices when faced with a limited trainable parameter budget. In this research, we study the effect of the duration of pre-pruning training (i.e., warm up) on the performance of the adapted model for each of the FEB tasks.

3.2.4 Infused Adapter by Inhibiting and Amplifying Inner Activations

Infused Adapter by Inhibiting and Amplifying Inner Activations $(IA)^3$ is a fine-tuning approach, that like prefix tuning, modifies the inner activations of the model being adapted to a new task. It achieves this by scaling the intermediate activations of the model with learned vectors. These added vectors, comprising a relatively small number of parameters, are the sole trainable parameters during the fine-tuning process, thereby qualifying this as a parameter-efficient fine-tuning strategy.

In each targeted transformer layer, (IA)^3 redefines the Scaled Dot Product Attention presented by Vaswani et. al. [46] as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{Q(l_k \odot K^T)}{\sqrt{d_k}}\right)(l_v \odot V)$$

In addition to the modified attention mechanism, it also scales the intermediate activation of the position-wise feed-forward networks using $(l_{ff} \odot \gamma V(W_1 x))W_2$. In both modifications, $l_r \in \mathbb{R}^{d_k}$, $l_v \in \mathbb{R}^{d_v}$ and $l_{ff} \in \mathbb{R}^{d_{ff}}$ are the introduced vectors and they are initialized with ones to avoid modifying the model's parameters at the start of training, while γ represents the non-linearity in the feed-forward network.

Notably, (IA)^3 was specifically designed to achieve strong performance in few-shot settings and was validated on a model with a similar architecture to T5, making it a compelling parameter-efficient fine-tuning strategy in this study. Similar to LoRA, the model's architecture remains unchanged with the addition of the learned vectors, offering the additional advantage of no increased inference latency.

3.2.5 Hybrid Methods

This research also aims at harnessing the individual strengths and performance implications of select orthogonal fine-tuning methods. By integrating these techniques in adapting T5 to the FEB tasks, we seek to enhance the label accuracy and the quality of the generated explanations.

The integration of prefix tuning and LoRA presents a novel approach to fine-tuning T5 for the FEB tasks. Via the incorporation of prefix tuning, the model gains the advantage of explicit instructions that allow it to adapt to the peculiarities of each downstream task. In parallel, making low-rank updates to the important layers of the transformer model facilitates the learning of more task-specific information, therefore, further refining the model's explanation generation abilities for the few-shot NLE tasks. This hybrid approach of fine-tuning seeks to capitalize on the upsides of both activation-modifying and model-modifying methods.

Similarly, the parallel application of prefix tuning with (IA)^3 has the potential for achieving orthogonal improvement on downstream performance, as it combines different approaches of updating the inner activations of the model during adaptation.

It is pertinent to acknowledge that while these hybrid methods require tuning more parameters compared to their individual constituent methods, we hypothesize that the improvement in the overall model's performance would justify the additional parameters and also underscore its value in the generation of NLEs in the few-shot setting.

3.3 Experimental Setup

This section outlines the experimental setup employed to evaluate the performance of the earlier discussed parameter-efficient fine-tuning strategies applied to the T5 model to perform the FEB tasks. In this section, we discuss key experiment design choices, including data pre-processing, model architecture and initialization, hyperparameter configuration, evaluation metrics as well as the training process.

3.3.1 Data Pre-processing

We utilize the FEB datasets to assess the effectiveness of the parameter-efficient methods in jointly generating predictions and their respective natural language rationales. Consistent with Marasovic et. al. [29], we adopt the use of the 60 train-dev splits in FEB. This multiple split approach enables us to account for the variance introduced by changing the random seed, thus providing a robust estimate of explanation quality with only a few examples [29]. Prior to fine-tuning, each of the datasets undergoes standard pre-processing procedures, including tokenization and padding, in alignment with the specifications of the T5 model.

3.3.2 Model Architecture and Initialisation

In our study, the foundational model architecture is the T5 model. Our exploration extends to both T5-base and T5-large architectures, and they are initialized with their respective pre-trained weights. However, Marasovic et. al. [29] found that the UnifiedQA variant of both models produced significantly better downstream performance than the standard T5 for the SBIC, ECQA, and ComVE datasets in the FEB. Consequently, we maintain the core structure of the T5 models but initialize both with their respective UnifiedQA pre-trained weights.

3.3.3 Training Process and Hyperparameter Configuration

Our implementation uses PyTorch and leverages the Hugging Face Transformers and PEFT library [52]. During training, we employ the AdamW optimizer [28] and use bf16 for faster computations and reduced memory consumption. We will begin our experiments by reproducing the results presented by Marasovic et. al. [29] of prompt-based fine-tuning of the models described in section 3.3.2 on the FEB tasks. This preliminary experiment serves as a baseline to compare the results of our subsequent experiments.

Hyperparameter configuration is a critical component of our study, aimed at understanding the effects of different settings of the parameter-efficient fine-tuning methods. The following is an overview of the configurations used for each method under investigation.

- **Prefix Tuning:** We vary prefix length over the range of 10 to 50, in steps of 10.
- **LoRA:** To streamline the number of experiments carried out here, we build upon the language generation recommendations from Hu et. al. [15] and FEB results from [43], by re-parameterizing the incremental matrices of the Query and Value projection matrices. Specifically, we apply low-rank updates to these projection matrices individually, and to both. For each of these configurations, we also explore rank variations from $\{4, 8, 16, 32\}$. Additionally, we draw upon the empirical findings documented in [55, 25] which showed that applying lower rank updates to all attention and feed-forward layer matrices can lead to comparable or even better performance than targeting both the Query and Value projection matrices, in the context of natural language understanding and generation tasks. Hence, we extend our experimentation to target all the attention matrices (i.e., Query, Key, Value, and Output projection matrices) and jointly target both the attention matrices and the feed-forward layer matrices. For both these configurations, we use a uniform rank of 4 for all incremental matrices.
- **AdaLoRA:** Based on the best-performing target module configuration in LoRA, we systematically vary the number of warm-up steps preceding the pruning of redundant singular vectors. For each incremental matrix, we use an initial rank of 8 and set the lowest possible rank to 4. The exploration of warm-up steps encompasses the range $\{0, 100, 300, 500\}$. For all the experiments, we set a default number of training steps after all pruning has been completed to 1000.

- **(IA)^3:** Following Liiu et. al. [25], we apply activation scaling to the Key, Value, and the intermediate feed-forward layer matrices. We also investigate the effect of learning scaling vectors for all the attention and feed-forward layer matrices in the model.
- **Hybrid Methods:** We use the best-performing hyperparameters from the individual experiments with prefix tuning, (IA)^3 and LoRA.

Beyond these specific hyperparameters, we align with the settings used by Marasovic et. al. [29], fixing the learning rate and per GPU batch size to 3e-5 and 4 respectively. Also, maximum training and evaluation steps are set to 300, except for AdaLoRA experiments, where the required number of training steps we are experimenting withexceeds this threshold. For this method, we set the maximum training step is set to 2000. Detailed specifications for each hyperparameter are provided in the next chapter while presenting the results of our experiments. Decoding involves greedy decoding with a maximum sequence length of 100 for all tasks in FEB, as per Marasovic et. al. [29]. All experiments were run on NVIDIA A100 GPUs.

3.3.4 Evaluation Metrics

In this study, the evaluation of the performance of each of the developed models encompasses key metrics that assess the model's efficacy in the context of the FEB tasks. For all models, we only evaluate their performance on the validation set, to allow for comparison with the baseline performance presented by Marasovic et. al. [29] and Solano et. al. [43]. The metrics used include:

- **Label Accuracy:** This quantifies the percentage of correctly predicted labels (or answers, depending on the task).
- **Normalized BERTscore [56]:** BERTscore leverages Bidirectional Encoder Representations from Transformers (BERT) [12] to measure the similarity between the generated explanations and their respective reference explanations by computing and aggregating token-level similarity scores. It was specifically chosen over other conventional natural language generation metrics such as BLEU [34] or ROUGE [22] due to its incorporation of contextual information resulting in stronger alignment with human judgments of the quality and coherence of generated text. Nevertheless, it is worth noting that Zhang et. al. [56] and Solano

et. al. [43] have extensively shown that when NLE plausibility is considered, the correlation of BERTscore with human judgment of explanation plausibility is only moderate (although still higher than other currently known metrics). In this study, we follow [17, 29] and assign zero BERTscore to explanations of incorrectly predicted instances. This implies a low prediction accuracy inadvertently leads to a low reported BERTscore.

By considering a combination of all three evaluation methods, we aim to gain a comprehensive insight into the strengths and weaknesses of the parameter-efficient fine-tuning methods, in the context of jointly generating predictions and explanations with the FEB.

Chapter 4

Results and Discussion

This chapter methodically presents the results from all our experiments discussed in the previous chapter. It provides comprehensive insights into the effects of the various parameter-efficient fine-tuning methods on the T5 model by analyzing the downstream performance of the fine-tuned models for each of the Few Explanations Benchmark (FEB) tasks. We begin with a summary of the best results obtained from these methods in comparison with established baselines, in order to provide an initial perspective on their relative efficacy. Subsequently, each method is painstakingly examined in separate sections, with an emphasis on a detailed presentation of the quantitative results followed by an analytical discussion on the peculiarities of each method. Via this comprehensive analysis, we aim to contribute to the existing discourse around computationally efficient methods of jointly generating predictions and explanations in the few-shot setting.

We start our experiments by fully fine-tuning T5-base for all four FEB tasks. Our results were consistent with the findings of Marasovic et. al. [29]. In order to optimize computation resources, we leveraged the results of full fine-tuning of T5-Large on the FEB datasets, presented in Marasovic et. al. [29] and Solano et. al. [43] as our baselines. An overview of the best-performing methods for fine-tuning T5-Large is presented in Table 4.1. For a comprehensive examination of the outcomes yielded by each of the methods presented in Table 4.1, we refer the reader to subsequent sections of this chapter.

		e-SNLI	ECQA	ComVE	SBIC
Full Fine-Tuning (737.64M)	Accuracy	$84.8_{2.6}$	$57.6_{2.6}$	$\mathbf{80.5}_{4.5}$	$\mathbf{70.1}_{3.4}$
	BERTscore	$76.9_{2.5}$	$\mathbf{51.7}_{2.4}$	$\mathbf{74.2}_{4.2}$	$\mathbf{67.8}_{3.3}$
Sparsefit (50.45M)	Accuracy	$82.6_{2.7}$	$55.8_{3.1}$	$74.9_{5.3}$	$67.0_{4.4}$
	BERTscore	$75.6_{2.5}$	$45.9_{3.7}$	$69.0_{4.8}$	$64.3_{4.7}$
AdaLoRA (8.65M)	Accuracy	$84.5_{2.0}$	$46.6_{4.0}$	$51.6_{3.0}$	$61.8_{2.7}$
	BERTscore	$77.4_{1.8}$	$41.2_{3.7}$	$69.7_{3.5}$	$58.9_{2.9}$
Prefix Tuning+LoRA (2.85M)	Accuracy	$84.7_{1.7}$	$47.7_{5.2}$	$71.3_{3.8}$	$61.7_{3.5}$
	BERTscore	$\mathbf{77.6}_{1.6}$	$42.5_{4.7}$	$65.7_{3.5}$	$59.2_{3.5}$
LoRA (2.36M)	Accuracy	$84.1_{1.8}$	$39.0_{4.1}$	$69.0_{3.7}$	$62.7_{3.5}$
	BERTscore	$77.0_{1.7}$	$33.5_{3.8}$	$63.5_{3.4}$	$59.8_{3.7}$
Prefix Tuning+(IA)^3 (0.64M)	Accuracy	$86.5_{1.9}$	$57.5_{2.4}$	$8.4_{3.5}$	$0.1_{0.1}$
	BERTscore	$72.1_{1.6}$	$46.7_{1.9}$	$7.0_{2.9}$	$0.1_{0.1}$
Prefix Tuning (0.49M)	Accuracy	$86.6_{1.8}$	$57.6_{2.4}$	$4.4_{2.7}$	$0.1_{0.1}$
	BERTscore	$72.1_{1.5}$	$46.8_{1.9}$	$3.6_{2.2}$	$0.0_{0.1}$
(IA)^3 (0.15M)	Accuracy	$\mathbf{86.7}_{1.8}$	$\mathbf{59.5}_{2.3}$	$23.2_{1.6}$	$0.2_{0.2}$
	BERTscore	$72.3_{1.5}$	$48.3_{1.9}$	$19.4_{1.3}$	$0.2_{0.2}$

Table 4.1: Comparison between all evaluated parameter-efficient fine-tuning methods on T5-large, including both baselines - full fine-tuning and SparseFit. We provide the number of trainable parameters underneath each of the methods, where 'M' means 'million'. The results are arranged in decreasing order of the number of trained parameters. The results are rounded to 1 decimal place to match the format reported by both baselines. Best results are written in bold.

4.1 Prefix Tuning

4.1.1 Prefix Length and its Effect on Performance

We examined the relationship between prefix length and model performance on the four FEB tasks. Figure 4.1 shows a general performance reduction (in both label accuracy and explanation BERTscore) as the prefix length increases from 10 to 50. This trend, albeit with a few exceptions, suggests that while longer prefixes are intended (and have been shown) to provide more task-specific instructions, they can however inadvertently introduce noise or shift the model's focus away from the actual input sequence.

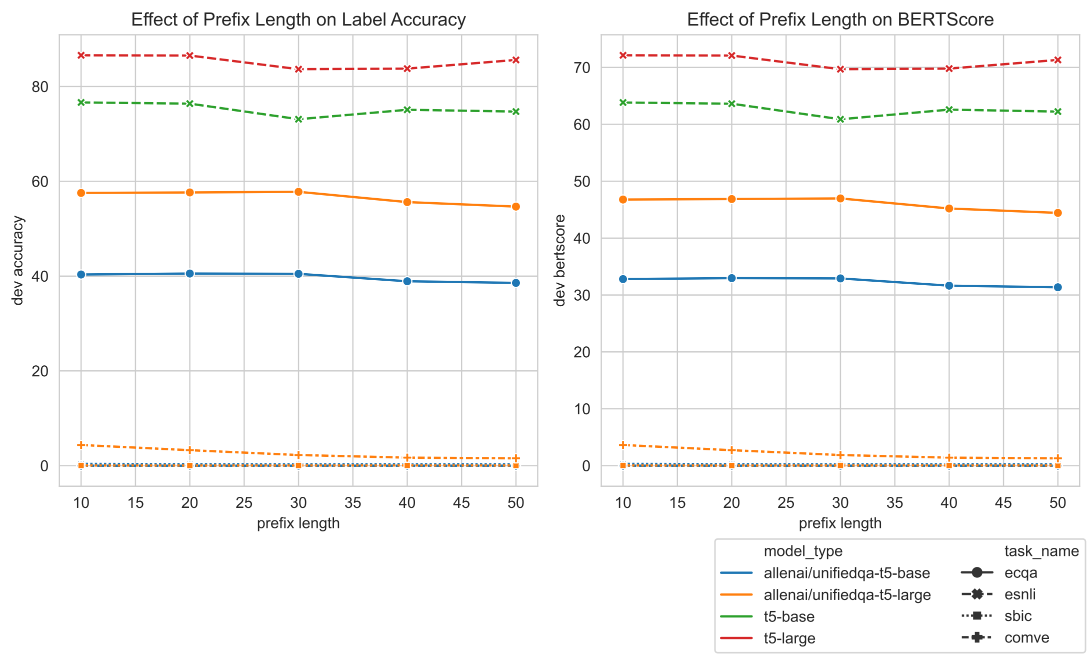

Figure 4.1: Effect of prefix length on the performance of the fine-tuned base and large models for each of the FEB tasks.

4.1.2 Task-Specific Performance

Applying prefix tuning to steer the models to perform the FEB tasks resulted in diverse trends for each of the FEB tasks, as shown in Table 4.2. Notably, we observed closely comparable performance to full fine-tuning for the ECQA (Question Answering) and e-SNLI (Natural Language Inference) tasks. However for both the SBIC (Offensive Classification) and ComVE (Nonsensical Sentence Selection) tasks, all fine-tuned models had their respective performances close to 0.

We initially surmised that this performance pattern was potentially due to the relatively longer average sequence length of the SBIC and ComVE tasks compared to that of the ECQA and e-SNLI tasks. This led us to hypothesize that contextual comprehension of the actual input sequence by the model was hindered because prefix tuning limits actual sequence length. However, further analysis revealed that all FEB tasks have comparable sequence lengths, dismissing this hypothesis. A more plausible explanation for this performance disparity could be the intrinsic complexity of the offensiveness classification and nonsensical sentence selection tasks, such that simply providing some form of task-specific instruction alone is not enough to steer the model to adequately perform these tasks. This observation underscores the importance of considering the nature of downstream tasks when deciding on the suitability of parameter-efficient

Dataset	Prefix Length	# Parameters		Base		Large	
		Base	Large	Accuracy	BERTscore	Accuracy	BERTscore
	10	0.18M	0.49M	$\mathbf{76.64}_{2.53}$	$\mathbf{63.83}_{2.12}$	$\mathbf{86.59}_{1.84}$	$\mathbf{72.13}_{1.53}$
	20	0.37M	0.98M	$76.39_{2.46}$	$63.62_{2.06}$	$86.54_{1.81}$	$72.09_{1.50}$
e-SNLI	30	0.55M	1.47M	$73.10_{2.51}$	$60.89_{2.10}$	$83.66_{2.14}$	$69.67_{1.78}$
	40	0.74M	1.97M	$75.10_{3.05}$	$62.58_{2.54}$	$83.78_{1.91}$	$69.79_{1.58}$
	50	0.92M	2.46M	$74.72_{2.28}$	$62.24_{1.90}$	$85.61_{1.59}$	$71.32_{1.33}$
	10	0.18M	0.49M	$40.34_{2.19}$	$32.80_{1.78}$	$57.55_{2.38}$	$46.78_{1.93}$
	20	0.37M	0.98M	$\mathbf{40.54}_{2.18}$	$\mathbf{32.97}_{1.77}$	$57.65_{2.33}$	46.87_{90}
ECQA	30	0.55M	1.47M	$40.48_{2.09}$	$32.92_{1.70}$	$\mathbf{57.79}_{2.25}$	$\mathbf{46.97}_{1.83}$
	40	0.74M	1.97M	$38.91_{2.12}$	$31.64_{1.73}$	$55.62_{2.32}$	$45.21_{1.89}$
	50	0.92M	2.46M	$38.57_{2.23}$	$31.36_{1.81}$	$54.67_{2.10}$	$44.43_{1.71}$
	10	0.18M	0.49M	$0.00_{0.00}$	$0.00_{0.00}$	$\mathbf{4.37}_{2.65}$	$\mathbf{3.64}_{2.22}$
	20	0.37M	0.98M	$0.00_{0.00}$	$0.00_{0.00}$	$3.27_{2.35}$	$2.72_{1.96}$
ComVE	30	0.55M	1.47M	$0.00_{0.00}$	$0.00_{0.00}$	$2.25_{1.73}$	$1.87_{1.44}$
	40	0.74M	1.97M	$0.00_{0.00}$	$0.00_{0.00}$	$1.70_{1.40}$	$1.41_{1.16}$
	50	0.92M	2.46M	$0.00_{0.00}$	$0.00_{0.00}$	$1.55_{1.14}$	$1.29_{0.95}$
	10	0.18M	0.49M	$\mathbf{0.39}_{0.31}$	$\mathbf{0.33}_{0.26}$	$0.05_{0.11}$	$0.04_{0.09}$
	20	0.37M	0.98M	$0.32_{0.26}$	$0.27_{0.22}$	$0.04_{0.10}$	$0.03_{0.08}$
SBIC	30	0.55M	1.47M	$0.30_{0.25}$	$0.25_{0.21}$	$0.05_{0.12}$	$0.04_{0.10}$
	40	0.74M	1.97M	$0.31_{0.26}$	$0.26_{0.22}$	$\mathbf{0.06}_{0.13}$	$\mathbf{0.05}_{0.10}$
	50	0.92M	2.46M	$0.3_{0.25}$	$0.25_{0.21}$	$0.05_{0.12}$	$0.04_{0.10}$

Table 4.2: Effect of prefix length on the performance of the fine-tuned base and large models for each of the FEB tasks.

fine-tuning techniques.

4.2 Low-Rank Adaptation

4.2.1 Rank Influence on Model Performance

In this study, we began our investigation of the efficacy of the LoRA method by analyzing the effect of the rank of the low-rank updates on model performance. In this exploration, we applied these updates individually to the Query matrices, the Value matrices, and both. Across the majority of the FEB tasks, our results indicated that the rank of these updates generally had little to no impact on the overall model

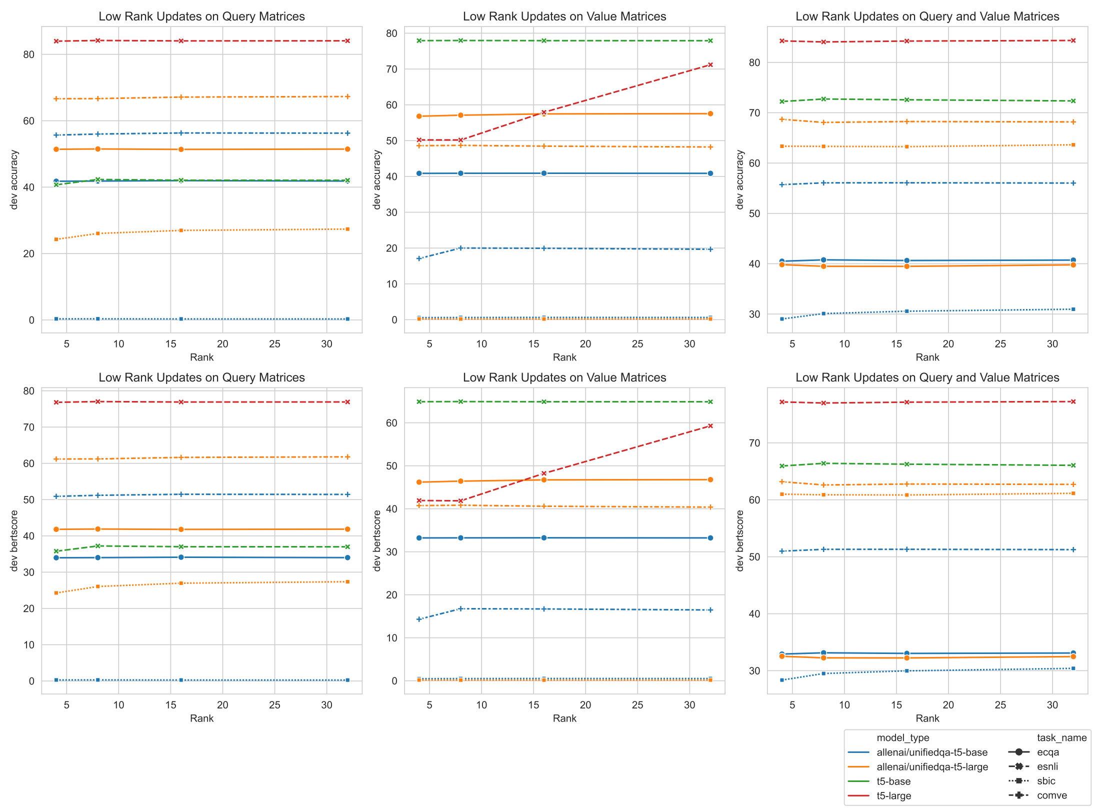

Figure 4.2: Effect of rank on model performance when LoRA is applied to different target modules

performance. However, a notable exception to this performance trend emerged when LoRA parameterization was applied to the Value matrices of the T5-base model for the e-SNLI task. Here, increase in rank, and by extension, model representation capacity, led to performance improvement suggesting that the value projection matrices in the T5-base model play a significant role in the e-SNLI task. Figure 4.2 shows a graphical representation of our findings.

4.2.2 Module Importance

Having established the apparent rank-independence of model performance across most of the FEB tasks, we shifted our focus to studying the relative importance of different model layers and their combinations, under the same rank setting, in the context of the four FEB tasks. Our results revealed an interesting insight, wherein tasks of greater complexity such as ComVE and SBIC tasks (as discussed in Section 4.1) consistently required the fine-tuning of more modules, than their less complex counterparts, e-SNLI and ECQA, for optimal downstream performance. The results of this module importance

Dataset	Target Module(s)	# Parameters		Base		Large	
		Base	Large	Accuracy	BERTscore	Accuracy	BERTscore
	Attn. Q	0.22M	0.59M	$40.71_{7.96}$	$35.78_{7.20}$	$83.97_{1.80}$	$76.82_{1.65}$
	Attn. V	0.22M	0.59M	$\mathbf{77.93}_{2.29}$	$64.90_{1.92}$	$50.20_{8.14}$	$41.92_{6.77}$
e-SNLI	Attn. Q,V	0.44M	1.18M	$72.23_{3.02}$	$65.94_{2.80}$	$\mathbf{84.27}_{1.95}$	$\mathbf{77.18}_{1.80}$
	Attn. Q,K,V,O	0.88M	2.36M	$74.03_{2.84}$	$67.75_{2.59}$	$84.05_{1.81}$	$77.04_{1.66}$
	Attn. Q,K,V,O & FFN	1.62M	4.33M	$75.72_{3.00}$	$\mathbf{69.27}_{2.77}$	$83.98_{2.18}$	$76.96_{2.01}$
	Attn. Q	0.22M	0.59M	$\mathbf{41.78}_{2.56}$	$\mathbf{33.97}_{2.08}$	$51.40_{2.81}$	$41.81_{2.29}$
	Attn. V	0.22M	0.59M	$40.86_{2.26}$	$33.22_{1.84}$	$\mathbf{56.81}_{2.23}$	$\mathbf{46.20}_{1.81}$
ECQA	Attn. Q,V	0.44M	1.18M	$40.50_{2.34}$	$32.92_{1.91}$	$39.81_{3.83}$	$32.53_{3.09}$
	Attn. Q,K,V,O	0.88M	2.36M	$37.82_{2.60}$	$30.75_{2.11}$	$39.04_{4.06}$	$33.52_{3.76}$
	Attn. Q,K,V,O & FFN	1.62M	4.33M	$30.18_{3.33}$	$24.59_{2.70}$	$48.64_{4.05}$	$42.98_{3.69}$
	Attn. Q	0.22M	0.59M	$55.68_{2.71}$	$50.90_{2.47}$	$66.63_{3.43}$	$61.18_{3.18}$
	Attn. V	0.22M	0.59M	$17.08_{3.41}$	$14.31_{2.86}$	$48.59_{0.80}$	$40.74_{0.69}$
ComVE	Attn. Q,V	0.44M	1.18M	$55.71_{2.56}$	$50.99_{2.34}$	$68.69_{3.80}$	$63.18_{3.49}$
	Attn. Q,K,V,O	0.88M	2.36M	$\mathbf{55.91}_{2.80}$	$\mathbf{51.20}_{2.57}$	$\mathbf{68.96}_{3.68}$	$63.48_{3.39}$
	Attn. Q,K,V,O & FFN	1.62M	4.33M	$55.80_{2.89}$	$51.14_{2.64}$	$69.46_{4.16}$	$\mathbf{63.98}_{3.83}$
	Attn. Q	0.22M	0.59M	$0.33_{0.30}$	$0.28_{0.26}$	$24.27_{8.73}$	$24.27_{8.73}$
	Attn. V	0.22M	0.59M	$0.57_{0.35}$	$0.48_{0.29}$	$0.21_{0.25}$	$0.17_{0.21}$
SBIC	Attn. Q,V	0.44M	1.18M	$29.02_{7.63}$	$28.35_{7.61}$	$63.34_{3.30}$	$60.98_{3.23}$
	Attn. Q,K,V,O	0.88M	2.36M	$\mathbf{59.03}_{3.12}$	$\mathbf{55.70}_{3.29}$	$62.66_{3.46}$	$59.80_{3.66}$
	Attn. Q,K,V,O & FFN	1.62M	4.33M	$57.43_{3.04}$	$54.05_{3.19}$	$\mathbf{64.05}_{3.59}$	$\mathbf{61.37}_{3.69}$

Table 4.3: Effect of applying low-rank updates (rank=4) to different modules in the T5 and UnifiedQA base and large Models. The results are presented with precision up to 2 decimal places, and the highest values are in bold.

analysis, meticulously detailed in Table 4.3, can provide guidance on allocating higher ranks or additional parameters to the incremental matrices of the identified optimal modules, particularly for applications constrained by rank or parameter budgets.

4.2.3 Task Performance and Model Size

Our findings also revealed an intriguing relationship between model performance on FEB tasks and model size. It was observed that task performance does not always scale with model size, as evidenced by the case when LoRA parameterization was applied to the Value matrices of the large models for both the e-SNLI and SBIC tasks. This anomalous behavior can be attributed to overfitting, leading us to hypothesize that for both these tasks, the Value matrices in the Large model may not carry substantial significance. This conjecture aligns with the work of Zhang et. al. [55] that indicated that fine-tuning less important modules for certain tasks might inadvertently induce overfitting.

4.3 Adaptive Low-Rank Adaptation

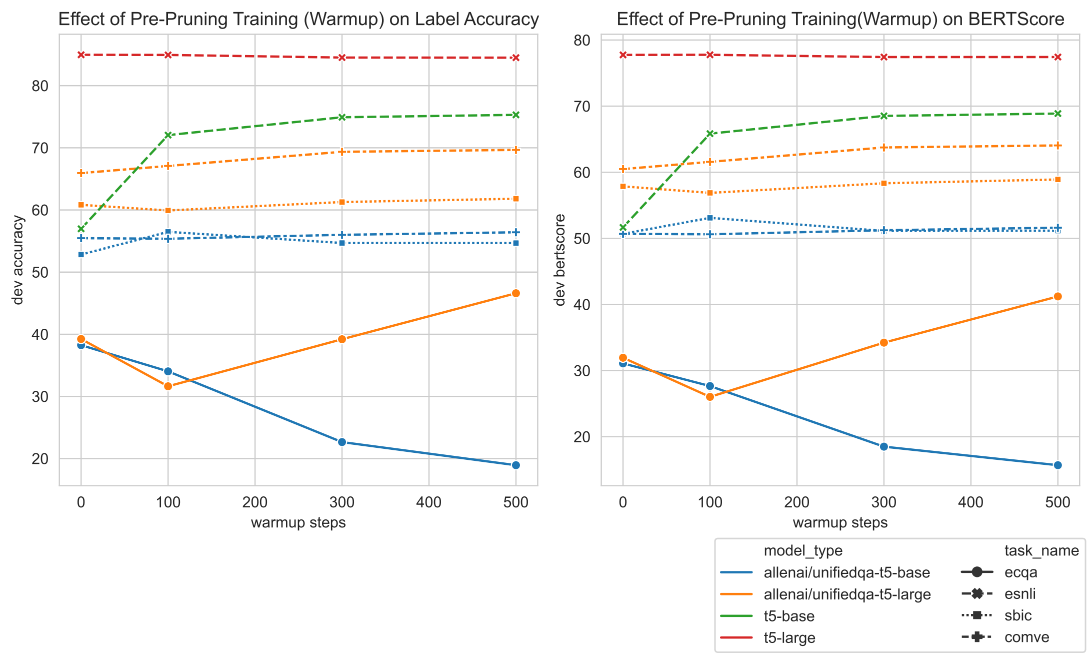

Figure 4.3: Effect of pre-pruning training (warm-up) on the performance of the fine-tuned base and large models for each of the FEB tasks.

Our experiments with this method revealed that increase in the length of training (i.e., warmup) before iteratively pruning less significant singular vectors often results in downstream performance gains, with improvements ranging from 0.95 to 18.33 in label accuracy and 0.93 to 17.21 in explanation BERTscore. It is worth mentioning that our choice of warmup duration was smaller than the range tested in the original AdaLoRA research. This leads us to hypothesize that further lengthening the warmup phase could result in better downstream performance across most of the FEB tasks. A distinct deviation from this trend, see Figure 4.3, is the consistent performance decline of the base model fine-tuned on the ECQA task, as the number of warm-up steps increases.

Additionally, while AdaLoRA resulted in downstream performance that is comparable to what was obtained using LoRA, we note that the former required the tuning of a significantly larger number of parameters. This observation leads us to conjecture that AdaLoRA tuning could benefit from additional pruning steps to further reduce the number of trained parameters. Moreover, this could also serve the dual purpose of reducing overfitting by allocating fewer parameters to unimportant layers. However, due to computational resource limitations during the course of this research, we could not validate this hypothesis.

These findings reveal that while AdaLoRA eliminates the need for manually searching for the best layers to be adapted for every new task, it necessitates the careful selection of pre-pruning training duration and possibly, the number of pruning iterations, in order to achieve optimal downstream performance.

Dataset	# Warmup Steps	Base		Large	
		Accuracy	BERTscore	Accuracy	BERTscore
e-SNLI	0	$56.97_{3.66}$	$51.67_{3.40}$	$\mathbf{84.97}_{1.99}$	$77.75_{1.81}$
	100	$72.04_{3.58}$	$65.84_{3.31}$	$84.94_{1.81}$	$\mathbf{77.77}_{1.66}$
	300	$74.91_{2.92}$	$68.53_{2.68}$	$84.52_{1.94}$	$77.43_{1.79}$
	500	$\mathbf{75.30}_{2.97}$	$\mathbf{68.88}_{2.73}$	$84.50_{1.95}$	$77.43_{1.79}$
ECQA	0	$\mathbf{38.25}_{2.53}$	$\mathbf{31.09}_{2.06}$	$39.24_{3.73}$	$31.94_{3.03}$
	100	$34.03_{2.96}$	$27.66_{2.41}$	$31.63_{3.46}$	$26.03_{2.80}$
	300	$22.65_{3.79}$	$18.50_{3.07}$	$39.20_{4.04}$	$34.23_{3.70}$
	500	$18.93_{3.67}$	$15.69_{3.04}$	$\mathbf{46.60}_{4.02}$	$\mathbf{41.22}_{3.65}$
ComVE	0	$55.46_{2.51}$	$50.69_{2.29}$	$65.93_{3.22}$	$60.48_{2.98}$
	100	$55.37_{2.51}$	$50.62_{2.30}$	$67.08_{3.41}$	$61.57_{3.14}$
	300	$56.00_{2.71}$	$51.23_{2.47}$	$69.36_{3.60}$	$63.75_{3.30}$
	500	$\mathbf{56.41}_{3.23}$	$\mathbf{51.62}_{2.95}$	$\mathbf{69.66}_{3.47}$	$\mathbf{64.06}_{3.19}$
SBIC	0	$52.82_{3.84}$	$50.69_{3.63}$	$60.82_{2.62}$	$57.87_{2.71}$
	100	$\mathbf{56.48}_{2.69}$	$\mathbf{53.10}_{2.72}$	$59.92_{2.88}$	$56.88_{2.97}$
	300	$54.69_{2.20}$	$51.13_{2.25}$	$61.28_{2.80}$	$58.34_{2.88}$
	500	$54.68_{2.21}$	$51.17_{2.26}$	$\mathbf{61.80}_{2.74}$	$\mathbf{58.91}_{2.86}$

Table 4.4: Effect of pre-pruning training (warm-up) on the performance of the fine-tuned base and large models for each of the FEB tasks. We updated 1.62 million and 8.56 million parameters in the base and large models respectively. The results are presented with precision up to 2 decimal places, and the highest values are in bold.

4.4 Infused Adapter by Inhibiting and Amplifying Inner Activations

Dataset	Target Module(s)	# Parameters		Base		Large	
		Base	Large	Accuracy	BERTscore	Accuracy	BERTscore
e-SNLI	Attn:K,V & FFN: W_0	0.06M	0.15M	**74.57**$_{2.27}$	**62.11**$_{1.90}$	**86.73**$_{1.78}$	**72.25**$_{1.48}$
	Attn:Q,K,V,O & FFN: W_0, W_1	0.20M	0.54M	74.42$_{2.26}$	61.98$_{1.89}$	86.64$_{1.85}$	72.18$_{1.54}$
ECQA	Attn:K,V & FFN: W_0	0.06M	0.15M	**40.73**$_{2.29}$	**33.12**$_{1.86}$	**59.46**$_{2.29}$	**48.34**$_{1.86}$
	Attn:Q,K,V,O & FFN: W_0, W_1	0.20M	0.54M	40.66$_{2.26}$	33.06$_{1.83}$	59.14$_{2.36}$	48.08$_{1.92}$
ComVE	Attn:K,V & FFN: W_0	0.06M	0.15M	0.00$_{0.00}$	0.00$_{0.00}$	23.20$_{1.59}$	19.41$_{1.33}$
	Attn:Q,K,V,O & FFN: W_0, W_1	0.20M	0.54M	0.00$_{0.00}$	0.00$_{0.00}$	**32.20**$_{2.24}$	**26.96**$_{1.88}$
SBIC	Attn:K,V & FFN: W_0	0.06M	0.15M	0.45$_{0.30}$	0.37$_{0.25}$	0.24$_{0.23}$	0.20$_{0.19}$
	Attn:Q,K,V,O & FFN: W_0, W_1	0.20M	0.54M	0.45$_{0.30}$	0.37$_{0.25}$	**0.28**$_{0.25}$	**0.23**$_{0.21}$

Table 4.5: Effect of scaling different matrices in the base and large models with learned vectors. The results are presented with precision up to 2 decimal places, and the highest values are in bold.

The results for the e-SNLI and ECQA tasks outperform all the other evaluated parameter-efficient fine-tuning methods, thus aligning with the excepted behaviour of the (IA)^3 hyperparameter configuration proposed by Liu et. al. [25]. Specifically, the scaling of only the Key, Value, and Intermediate feedforward layer matrices, as per the recipe from Liu et. al. [25], leads to improved downstream performance. The seemingly more complex ComVE and SBIC tasks, on the other hand, did not follow the performance trend of this recipe since their downstream performances were significantly lower than what was reported for other evaluated parameter-efficient fine-tuning methods. We however note that contrary to the recipe presented by Liu et. al. [25] where only specific matrices are scaled, the downstream performance of these two tasks - ComVE and SBIC, greatly benefited from scaling all attention and feedforward network matrices with trainable vectors.

Earlier in Chapter 3, we hypothesized based on the findings of Liu et. al. [25], that fine-tuning T5 with (IA)^3 could potentially yield the best downstream performance across all FEB tasks. However, based on our results, we surmise that this hypothesis may have been proven untrue due to the slight differences in the training objectives and models used. (IA)^3 was designed and evaluated on T-Few [25], a model based on T0-3b [41]. Although this model has a similar architecture to T5, it was trained to optimize a different objective function. This objective function added unlikelihood loss and length normalization loss to the standard cross-entropy loss, in order to boost its performance when fine-tuned in few-shot scenarios. These enhancements appear to have a significant impact on T-Few's downstream performance on new tasks, challenging our initial assumption that (IA)^3 was the main source of performance improvement.

We also note that the performances of the base model on the ComVE and SBIC tasks are centered around 0, similar to the results obtained in prefix tuning. This shared trend raises the question of the suitability of parameter-efficient fine-tuning methods that only update a model's inner activation for certain complex tasks.

4.5 Hybrid Fine-Tuning Methods

Dataset	Fine-Tuning Method	# Parameters		Base		Large	
		Base	Large	Accuracy	BERTscore	Accuracy	BERTscore
e-SNLI	Prefix Tuning with LoRA	1.07M	2.85M	$76.80_{2.71}$	$70.23_{2.47}$	$84.67_{1.69}$	$77.56_{1.57}$
	Prefix Tuning with (IA)^3	0.24M	0.64M	$76.07_{2.56}$	$63.36_{2.15}$	$86.52_{1.90}$	$72.08_{1.57}$
ECQA	Prefix Tuning with LoRA	1.07M	2.85M	$19.16_{4.75}$	$15.83_{3.87}$	$47.71_{5.21}$	$42.48_{4.69}$
	Prefix Tuning with (IA)^3	0.24M	0.64M	$40.39_{2.08}$	$32.84_{1.69}$	$57.49_{2.39}$	$46.74_{1.94}$
ComVE	Prefix Tuning with LoRA	1.07M	2.85M	$54.72_{2.66}$	$50.05_{2.43}$	$71.31_{3.75}$	$65.66_{3.49}$
	Prefix Tuning with (IA)^3	0.24M	0.64M	$0.01_{0.05}$	$0.01_{0.04}$	$8.40_{3.51}$	$7.01_{2.94}$
SBIC	Prefix Tuning with LoRA	1.07M	2.85M	$56.41_{2.87}$	$52.67_{3.03}$	$61.68_{3.45}$	$59.16_{3.51}$
	Prefix Tuning with (IA)^3	0.24M	0.64M	$0.38_{0.31}$	$0.32_{0.26}$	$0.06_{0.11}$	$0.05_{0.09}$

Table 4.6: Results from the combination of orthogonal parameter-efficient fine-tuning methods.

In a bid to enhance label accuracy and the quality of the generated explanations, we increased the percentage of parameters tuned by combining orthogonal parameter-efficient fine-tuning methods- Prefix Tuning and IA3, and Prefix Tuning and LoRA. For each of those individual methods, we select their respective best performing hyper-parameter setting, from the previous experiments. Given that the optimal settings for each of these methods varied across the FEB tasks, we adopted a selection approach of choosing the setting that yielded the highest performance across the most tasks (and models), while tuning a relatively few parameters. Therefore, for these experiments, we employed a prefix length of 10 for prefix tuning, scaled the Key, Value and Intermediate Feedforward layer's matrices for IA3 tuning, and re-parameterized the incremental matrices of the Query, Key, Value and Output projection matrices for LoRA. Our findings from combining these orthogonal methods are meticulously documented in Table 4.6, complemented by the comparative results presented in Table 4.1. From these results, we made the following observations:

- **Prefix Tuning with LoRA:** The integration of prefix tuning and LoRA to fine-tune both the base and large models to perform the FEB tasks yielded promising results, with most experiments demonstrating downstream performance improvement (over LoRA-only tuning) between 0.6 and 9.0 for both label accuracy and explanation BERTscore. This method also outperforms full-fine-tuning of both T5-base and T5-large on the e-SNLI task. Notably, this combined fine-tuning approach significantly mitigated the woeful performance obtained when prefix tuning only was used to fine-tune the models to perform ComVE and SBIC tasks. It yielded task performance comparable with both full fine-tuning and SparseFit methods. These observations suggest that instructing the model with an activation modification tuning method has a synergistic relationship with applying low-rank updates via a model parametrization tuning method.
- **Prefix Tuning with (IA)^3:** Conversely, while there were marginal performance improvements over standalone prefix tuning, the combination of prefix tuning with (IA)^3 resulted in a general performance decline in comparison with IA3-only tuning. This trend prompts a reiteration of the question posed in Section 4.4 about the suitability and effectiveness of fine-tuning methods that only target the inner activations of a model for certain downstream tasks. In addition, this method boasts competitive performance with full-finetuning for both the e-SNLI and ECQA tasks.

These findings highlight the significance of method compatibility when integrating orthogonal parameter-efficient fine-tuning methods to achieve optimal performance gains.

4.6 Summary of Findings

Throughout this chapter, we presented a comprehensive evaluation of various parameter-efficient fine-tuning methods applied to both T5 and its UnifiedQA variant to perform all four FEB tasks. Our results and subsequent analysis revealed some noteworthy insights that re-inforce our initial understanding of these methods while also highlighting their suitability and limitations within the context of few-shot self-rationalization models. These insights can be summarized with the following points:

1. Prefix tuning can lead to performance comparable with full fine-tuning for e-SNLI and ECQA tasks, however, the length of the prefix should remain very short to avoid introducing noise that could divert the model's focus from the actual input sequence.

2. Different layers (and their combinations) within a model have widely varying significance with respect to the downstream FEB task. For example, our findings show that applying low-rank updates to both the Query and Value matrices of T5-Large results in better downstream performance on the e-SNLI task. In contrast, updating these same matrices causes a significant decline in the downstream performance of the ECQA task. Failure to select the right layers for LoRA parametrization often leads to overfitting. However, upon selection of the most important layers, modest rank values are sufficient for model adaptation with LoRA, as performance was largely rank-independent.

3. AdaLoRA eliminates the need for manual search of optimal layers but rather introduces the necessity to meticulously set optimal warm-up duration and number of pruning iterations.

4. (IA)^3 leads to downstream performance comparable, and sometimes better than full fine-tuning for both e-SNLI and ECQA tasks, while only adding and training about 0.02% of the parameters trained in full fine-tuning.

5. Parameter-efficient fine-tuning methods that only modify the inner activations of the model, without re-parameterizing any of the model's layers, may not be suitable for certain FEB tasks like ComVE and SBIC.

6. Innovative combination of orthogonal parameter-efficient fine-tuning methods demonstrated the potential for obtaining improved downstream performance over individual approaches. Notably, both prefix tuning with LoRA and prefix tuning with (IA)^3 yield better performance than full-finetuning of T5-large on the e-SNLI tasks while only updating less than 1.2% of the model's parameters.

Chapter 5

Conclusions

5.1 Summary of Research

In this study, we comprehensively explored the application of parameter-efficient fine-tuning methods in adapting T5 and its UnifiedQA variant to perform the tasks encompassed in the Few Explanations Benchmark (FEB). Our research objectives were framed to address the suitability, effectiveness, and limitations of these fine-tuning techniques, within the context of few-shot self-rationalization. To achieve these objectives, we systematically evaluated the downstream performance of T5 and UnifiedQA on all the FEB tasks. Leveraging model adaptation methods such as Prefix Tuning, Low-Rank Adaptation (LoRA), Adaptive Low-Rank Adaptation (AdaLoRA), and Infused Adapter by Inhibiting and Amplifying Inner Activation $(IA)^3$, we uncovered diverse insights that collectively contribute to the discourse on computationally efficient methods of jointly generating predictions and their rationales in the few-shot setting.

Our findings shed light on several key aspects. Firstly, Prefix tuning demonstrated its potential for achieving comparable performance with full fine-tuning for the e-SNLI and ECQA tasks, while underscoring the importance of frugality in prefix length selection. The evaluation of LoRA highlighted the significance of only applying low-rank updates to layers critical to specific tasks, as well as the impact of rank (or lack thereof) on the downstream performance across most of the FEB tasks. AdaLoRA's ability to automatically allocate the optimal amount of parameters to different layers of the model being adapted, while requiring careful tuning of warmup duration and number of pruning iterations, brought to light the trade-offs between automation and hyperparameter sensitivity. Additionally, $(IA)^3$ stood out for its ability to achieve better performance than full fine-tuning on e-SNLI and ECQA tasks with the addition of relatively few parameters, underscoring its efficiency.

We also explored hybrid fine-tuning approaches that adapted models by applying orthogonal parameter-efficient fine-tuning methods in parallel. This exploration yielded insights into the symbiotic effects of combining certain methods and highlighted the possible limitations of fine-tuning methods that only modify the inner activations of a model, without re-parameterizing any of its layers. The overarching discovery from our study is that no single fine-tuning method reigns supreme across all FEB tasks. Rather, the choice of the optimal fine-tuning strategy largely depends on factors such as the task's complexity, hyperparameter tuning, and computational constraints. These findings contribute knowledge to the fields of few-shot self-rationalization models and parameter-efficient fine-tuning techniques, while also paving the way for future research that integrates both fields.

5.2 Limitations and Future Work

While this study has provided valuable insights into the efficacy of various parameter-efficient fine-tuning methods for few-shot self-rationalization tasks, several limitations still need to be considered. These limitations, presented below, offer a more comprehensive understanding of the scope of this study, while also highlighting possible avenues for future research.

1. The findings and conclusions drawn from this study are primarily derived from the tasks within the FEB. Generalizing these findings to other tasks, or benchmarks, might not be straightforward due to the variations in data distribution, linguistic nuances, and task requirements. Future research can examine a broader range of tasks, across several domains, to validate the generality of the identified trends.

2. Given that the success of many machine learning methods often depends on the careful selection of hyperparameters, we note that even though we have explored a wide range of hyperparameter settings for each of the evaluated methods, there might still exist more optimal configurations that could yield better downstream performance. Performing more extensive hyperparameter search could reveal deeper insights into how these methods work.

3. In this study, we only evaluated a handful of parameter-efficient fine-tuning strategies, and even studied a smaller set of hybrid fine-tuning methods, thus leaving

many possible combinations unexplored. Further investigation into novel hybrid approaches, including different parallel combinations of parameter-efficient fine-tuning methods and sequential application of these methods could result in orthogonal performance improvements.

In addition to the possible research directions garnered from these identified limitations, future research can also investigate the underlying characteristics of tasks that yield unfavourable downstream performance when models are fine-tuned with parameter-efficient methods that only modify the inner activations of the model.

Bibliography

[1] Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3050–3065, 2021.

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[3] Raisul Arefin, Manar D Samad, Furkan A Akyelken, and Arash Davanian. Non-transfer deep learning of optical coherence tomography for post-hoc explanation of macular disease classification. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pages 48–52. IEEE, 2021.

[4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[5] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

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

[7] 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, 2020.

[8] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] Alexandre Chanson, Nicolas Labroche, and Willème Verdeaux. Towards local post-hoc recommender systems explanations. In *Proceedings of the 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*, 2021.

[10] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR, 2021.

[11] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[13] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

[14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751, 2019.

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

[16] Weina Jin, Xiaoxiao Li, Mostafa Fatehi, and Ghassan Hamarneh. Generating post-hoc explanation from deep neural networks for multi-modal medical image analysis tasks. *MethodsX*, 10:102009, 2023.

[17] Maxime Kayser, Oana-Maria Camburu, Leonard Salewski, Cornelius Emde, Virginie Do, Zeynep Akata, and Thomas Lukasiewicz. e-vil: A dataset and benchmark for natural language explanations in vision-language tasks, 2021.

[18] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

[19] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association for Computational Linguistics.

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[22] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[23] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[24] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*, 2020.

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

[26] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association*

for Computational Linguistics (Volume 2: Short Papers), pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics.

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

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[29] Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew Peters. Few-shot self-rationalization with natural language prompts. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 410–424, Seattle, United States, July 2022. Association for Computational Linguistics.

[30] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering, 2018.

[31] Nijat Mehdiyev and Peter Fettke. Local post-hoc explanations for predictive process monitoring in manufacturing. *arXiv preprint arXiv:2009.10513*, 2020.

[32] Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Generating post hoc review-based natural language justifications for recommender systems. *User Modeling and User-Adapted Interaction*, 31:629–673, 2021.

[33] Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*, 2020.

[34] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL ’02, page 311–318, USA, 2002. Association for Computational Linguistics.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of

transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[37] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[39] Saumendu Roy, Gabriel Laberge, Banani Roy, Foutse Khomh, Amin Nikanjam, and Saikat Mondal. Why don't xai techniques agree? characterizing the disagreements between post-hoc explanations of defect predictions. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 444–448. IEEE, 2022.

[40] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

[41] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization, 2022.

[42] Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online, July 2020. Association for Computational Linguistics.

[43] Jesus Solano, Oana-Maria Camburu, and Pasquale Minervini. Sparsefit: Few-shot prompting with sparse fine-tuning for jointly generating predictions and natural language explanations, 2023.

[44] Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1451–1465, 2021.

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. Allennlp interpret: A framework for explaining predictions of nlp models. *arXiv preprint arXiv:1909.09251*, 2019.

[48] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[49] Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. SemEval-2020 task 4: Commonsense validation and explanation. In *Proceedings of The 14th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2020.

[50] Qingyun Wang, Semih Yavuz, Victoria Lin, Heng Ji, and Nazneen Rajani. Stage-wise fine-tuning for graph-to-text generation. *arXiv preprint arXiv:2105.08021*, 2021.

[51] Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable natural language processing. *arXiv preprint arXiv:2102.12060*, 2021.

[52] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[53] Shuyuan Xu, Yunqi Li, Shuchang Liu, Zuohui Fu, Xu Chen, and Yongfeng Zhang. Learning post-hoc causal explanations for recommendation. *arXiv preprint arXiv:2006.16977*, 2020.

[54] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199, 2021.

[55] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[56] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.