Parameter-Efficient Fine-Tuning: Is There An Optimal Subset of Parameters to Tune?

Anonymous ACL submission

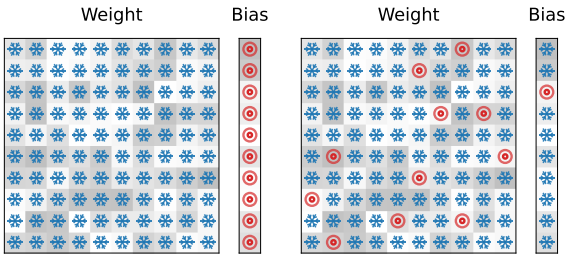
Abstract

The ever-growing size of pretrained language models (PLM) presents a significant challenge for efficiently fine-tuning and deploying these models for diverse sets of tasks within memory-constrained environments. In light of this, recent research has illuminated the possibility of selectively updating only a small subset of a model's parameters during the fine-tuning process. Since no new parameters or modules are added, these methods retain the inference speed of the original model and come at no additional computational cost. However, an open question pertains to which subset of parameters should best be tuned to maximize task performance and generalizability. To investigate, this paper presents comprehensive experiments covering a large spectrum of subset selection strategies. We comparatively evaluate their impact on model performance as well as the resulting model's capability to generalize to different tasks. Surprisingly, we find that the gains achieved in performance by elaborate selection strategies are, at best, marginal when compared to the outcomes obtained by tuning a random selection of parameter subsets. Our experiments also indicate that selection-based tuning impairs generalizability to new tasks.

1 Introduction

In recent years, the number of parameters used in language models has risen much faster than the memory available in GPUs (Lialin et al., 2023). Attaining the ability to efficiently fine-tune such large models on the available hardware necessitates methods that reduce the memory footprint. Additionally, a single pretrained model is often adapted to a wide range of tasks. The storage requirements for such a collection of model versions can be significantly reduced if the difference between these models can be represented in a compact way.

Parameter-efficient fine-tuning techniques (PEFT) aim to reduce the number of parameters that need to be stored and fine-tuned while maintaining a performance that is comparable to the training of the complete model. One of the most popular classes of these methods is referred to as *selective parameter-efficient fine-tuning* (Lialin et al., 2023). Here, a subset of the parameters is selected for PEFT, keeping the remaining parameters frozen during training. We illustrate this intuition in Figure 1 for a single weight matrix and bias vector in which most parameters are frozen and only a small subset kept unfrozen for optimization.



(a) BitFit (b) Random subset

Figure 1: Only a small subset of the parameters (marked with red circles in this illustration) is used during training; the others are frozen. The BitFit approach tunes only the bias weights, while other approaches select a tuneable subset from all model parameters.

Since only a few parameters are fine-tuned, the sparse difference between the adapted and the pretrained model can be stored in a compact way (Zaken et al., 2022; Guo et al., 2021). The same applies to gradient statistics that are stored for optimization. Reducing the required memory frees up space for the use of larger batches and therefore speeds up training. However, an open question pertains to which subset of parameters should best be tuned to maximize task performance and generalizability.

Contributions. In this paper, we investigate several theoretical questions that have been raised in the context of selective PEFT methods and the

lottery ticket hypothesis for pretrained (language) models (Gong et al., 2022; Zheng et al., 2022). Our aim is to explore if an optimal subset for tuning exists and how subset tuning affects generalizability of the model. In more detail, we examine the following questions:

- We comparatively evaluate a broad range of approaches for identifying the ideal subset of parameters to tune. Our analysis considers the size of the subset and the computational costs for its identification. For instance, it has been shown that an effective subset can be obtained through an initial fine-tuning step of the complete model (potentially incorporating some form of regularization), followed by the selection of parameters exhibiting the largest magnitude of change (Guo et al., 2021; Xu et al., 2021). This, however, still requires a costly full fine-tuning step. Hence, the possibility of identifying a promising subset without an initial fine-tuning step would be beneficial (Prasanna et al., 2020; Gong et al., 2022). Figure 1 illustrates two such strategies.

- We analyze how sparse fine-tuning affects the generalizability of the resulting network. This is motivated by Zaken et al. (2022)'s observation that their parameter-efficient method "Bitfit" generalizes better: They report that the gap between the train and test score is substantially smaller compared to a full fine-tuning of the model.

To address these questions, we systematically conduct experiments using a large number of subset sizes and various subset selection strategies. We conduct a comprehensive grid search over hyperparameters to identify optimal training parameters for each selection strategy. We compare these hyperparameter-optimized subset selection strategies to full fine-tuning (including the use of regularization), as well as an additional (non-selective) parameter-efficient fine-tuning technique, which recently gained a lot of popularity: Low-Rank Adaption (Hu et al., 2021, LoRA).

We make a number of observations in our experiments: First, the differences between different subset selection methods are marginal when hyperparameters are properly optimized, and do not significantly outperform even a random subset selection baseline. Second, subset-tuning methods tend to modify embedding networks significantly more since they are limited to a small number of parameters and hence need to make a more drastic changes. The prior function of the network which can exhibit a certain degree of general language capabilities can be more affected by these local but more drastic changes.

2 Background

Our work is informed by two lines of research: *Selective parameter-efficient fine-tuning* and the *lottery ticket hypothesis* for pretrained language models. In the remainder of this section, we discuss aspects of these two areas that are relevant to the work we present in this paper.

2.1 Selective Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) methods reduce the number of parameters that are tuned in a model. The benefits of this are twofold: (1) The cost of storage for each task-specific adaption is smaller and (2) the memory used during fine-tuning is reduced. For example, Adam(W) (Kingma and Ba, 2017; Loshchilov and Hutter, 2019), a commonly used optimizer for fine-tuning language models, not only stores the current gradient for each parameter, but additionally estimates of two lower-order moments. When using PEFT methods, the weights of the model still need to be kept in memory. Still, since fewer parameters are tuned, a much smaller number of estimates needs to be stored, significantly freeing up space for processing a larger number of samples per batch and hence speeding up training overall.

Lialin et al. (2023) arrange a large variety of PEFT methods into a comprehensive taxonomy and identify three major classes: *Additive* (which includes *adapters* and *soft prompts*), *Selective*, and *Reparametrization*-based approaches. In selection-based approaches, only a certain subset of the parameters is tuned while other parameters which are not part of the set remain frozen.

Heuristically Motivated Subsets

Zaken et al. (2022) offer a particularly simple variant: In BitFit, only the bias terms (or in a variation of this approach, only certain bias terms) are tuned. This removes the need to compute and handle parameter masks. Qi et al. (2022) propose LN-tuning (tuning only the LayerNorm modules) and suggest combining this with other methods (such as prefix tuning).

Empirical Fisher Information

Sung et al. (2021) attempt to determine the subset by a less heuristics-based approach and instead propose to use the empirical Fisher information of the network parameters to determine each parameter's importance (compare with Kirkpatrick et al., 2017). The Fisher information estimates the impact of a parameter on the model's prediction. Since the Fisher information matrix is intractable to compute, a common approximation is to only use the diagonal and approximate the sample distribution with the available N samples $x_1, ..., x_N$. The estimated Fisher information $\hat{F}_\theta$ of each parameter can then be expressed as:

$$\hat{F}_\theta = \frac{1}{N} \sum_{i=0}^{N} \mathbb{E}_{y \sim p_\theta(y|x_i)} \left(\nabla_\theta \log p_\theta(y|x_i) \right)^2 \quad (1)$$

In cases where many classes are available, calculating the expected value requires a large number of backward passes. Hence, it is common to simplify this using the "empirical Fisher" $\tilde{F}_\theta$ which can be derived by replacing the expected value with the observed label y_i of each sample. [1]

$$\tilde{F}_\theta = \frac{1}{N} \sum_{i=0}^{N} \left(\nabla_\theta \log p_\theta(y_i|x_i) \right)^2 \quad (2)$$

To retrieve a fine-tuning mask, the k parameters with the respective largest values are selected. All other parameters will remain frozen.

Using a fine-tuning mask trades off simplicity for a more theoretically substantiated method for determining the subset to be fine-tuned.

2.2 Lottery Ticket Hypothesis

A different line of research tests the lottery ticket hypothesis (Frankle and Carbin, 2019) for pretrained language models. The lottery ticket hypothesis states that the performance of a randomly initialized dense neural network can be matched by only training a certain subnetwork (i.e. only a subset of the parameters). Typically, these subsets can only be found by training the complete network and pruning connections iteratively (Frankle and Carbin, 2019; Zhou et al., 2020; Chen et al., 2021). More recent literature has tried to translate these findings to pretrained language models (Chen et al., 2020; Zheng et al., 2022; Liang et al., 2021; Gong et al., 2022). Recent research seems to suggest that it might be feasible to find suitable subnetworks without prior training (and pruning) since the weights are no longer random (Sung et al., 2021; Prasanna et al., 2020).

While the lottery ticket hypothesis typically induces a different perspective, there are important ties between this line of research and parameter-efficient fine-tuning. The ability to find transferable (or general) true (in the sense of perfectly matching performance) "winning lottery tickets" would have considerable implications for parameter-efficient fine-tuning. Vice-versa, well-working methods to select subsets to be fine-tuned might reveal information about winning lottery tickets in general.

3 Subset Selection and Downstream Task Performance

In this first series of experiments, we aim to investigate the impact of the subset selection strategy and the subset size on the performance of the embedding network on a downstream task. Each configuration is evaluated with respect to the performance on each of the four downstream tasks. We first describe the used selection strategies and the experimental setup, before discussing the observed impact of these two variables.

3.1 Subset Selection Strategies

We compare a number of different selection strategies. Some of the strategies are task-independent while others rely on the task's training data to select the parameters to be tuned. As a baseline, we include a **random** selection of parameters

One of the simplest strategies is **BitFit** (Zaken et al., 2022). Here, all bias terms are selected for tuning while all other parameters remain frozen (see Figure 1). The tuned portion depends on the model's architecture and is not flexible. The authors offer a second variant that uses only some of the bias terms. However, we exclude this second variant from our analysis since we compare subset selections of similar size. Where not noted differently, we use the resulting portion of active parameters as target portion for the other methods.

In Diff pruning (Guo et al., 2021), the model is fine-tuned completely (with some regularization) before pruning away the smallest differences to the pretrained model. The pruned weights are not set to zero but to their original value. We test two variants (using L1 instead of L0-regularization):

[1] Note that the result is identical to the sum of the squared gradients of the cross-entropy loss over a given dataset.

One where we **prune without re-training** and one where we prune **with retraining** the remaining weights (initialized with the pretrained parameters). We only prune and re-train a single time to mimic the other subset methods as closely as possible (i.e. using a pre-computed mask for a single training run). The first variant cannot be considered a subset tuning method. The second does include a subset tuning step, but still requires a costly initial full-finetuning step. It might be possible to approximate the subset selection by training the model for a shorter period, but this is outside the scope of this paper.

Sung et al. (2021) propose choosing a subset based on the empirical Fisher information on the downstream data $\tilde{F}_{\theta,downstr.}$. This is equivalent to picking the largest sum of squared gradients (**largest downstr. sq-grad**) of the cross-entropy loss.

Inspired by elastic weight consolidation (Kirkpatrick et al., 2017), we decided to additionally consider the gradient statistics on a portion of the pretraining data $\tilde{F}_{\theta,pretr.}$ (using 30508 samples of wikitext, Merity et al., 2016). While choosing the k parameters with the smallest empirical Fisher information would be more in line with their proposal, we found that (this binarized version) leads to a selection of parameters that receive minimal gradient flow. For the fine-tuning to have a non-negligible effect would require a learning rate that is to high for the decoder to remain stable. We hence pick the largest values instead (**largest pretr. sq-grad**), expecting the subset to be particularly bad.

Finally, we propose a **combined** measure that selects parameters with large squared downstream gradients and lower squared pretraining gradients. This is an attempt to force the selection to consider task-specific information not merely the received gradient magnitudes. The strategy selects parameters with the largest values of:

$$G_{combined} = \frac{\tilde{F}_{\theta,downstr.}}{1 + \tilde{F}_{\theta,pretr.}} \quad (3)$$

3.2 Experimental Setup

Evaluation datasets. We evaluate all considered subset selection strategies (together with full fine-tuning baselines) to tune a `RoBERTa-base` model (125M parameters) on four tasks:

- SST-2 (Socher et al., 2013), a sentiment classification task,
- QNLI (Wang et al., 2018) a question answering natural language inference task,
- CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003), a named entity recognition tasks,
- TREC-6 (Hovy et al., 2001; Li and Roth, 2002), a question classifcation task.

In the case of SST-2 and QNLI which both are part of the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), we use the development set in place of the test set (as the test set is not readily available and requires a submission for each set of predictions).

Decoder initialization. As each task requires a randomly initialized decoder on top of the PLM, we first execute a decoder-tuning step in which we train the decoder over the frozen PLM (Cui et al., 2023). Fine-tuning the decoder first (while initially keeping the embedding network frozen) helps to mitigate the effect of the different selections of learning rates used in the experiments on the degree to which the decoder adapts to the embedding network versus vice-versa. The much higher learning rate required by some of the variants can be quite an advantage or disadvantage as a randomly initialized decoder requires significantly more tuning. The hyperparameters used to tune the decoders can be found in Table 4 in Appendix A.

Like the fine-tuned task-specific decoder, the gradient statistics can also be shared across multiple repetitions of the experiment. A different decoder initialization leads to different gradients. Hence, using the same initialization of the decoder across the experiments is required to allow sharing of the gradient statistics.

Experimental framework and hyperparameters. All experiments were conducted using the Flair-framework (Akbik et al., 2019), using their default implementations for the embeddings and task-specific decoders. [2]

Most of the hyperparameters used in fine-tuning the embedding network are set to standard values and are kept consistent over all experiments. There is no indication that these settings favor any of the variants (though this cannot be entirely ruled out). These hyperparameters can be found in Table 2.

To ensure a level playing field, an exhaustive grid search was performed over an equally spaced

[2]The configurations, code, and resulting metadata will be published upon acceptance.

Task Variant	CoNLL-2003	QNLI	SST-2	TREC-6	Avg.
Full fine-tuning	**0.9217 ± 0.0009**	**0.9290 ± 0.0018**	**0.9468 ± 0.0013**	**0.9752 ± 0.0046**	**0.9432**
LoRA (rank 4)	*0.9139 ± 0.0018*	*0.9165 ± 0.0021*	0.9406 ± 0.0031	0.9708 ± 0.0041	*0.9354*
Random subset	0.9087 ± 0.0012	0.9048 ± 0.0029	0.9342 ± 0.0028	*0.9720 ± 0.0032*	0.9299
Bitfit	0.9080 ± 0.0014	0.9039 ± 0.0018	0.9383 ± 0.0026	0.9592 ± 0.0059	0.9273
Largest pretr. sq-grad	0.9073 ± 0.0016	0.9037 ± 0.0028	0.9378 ± 0.0052	0.9552 ± 0.0061	0.9260
Largest downstr. sq-grad	0.9073 ± 0.0020	0.9075 ± 0.0010	0.9399 ± 0.0031	0.9580 ± 0.0049	0.9282
Combined gradient stats	0.9082 ± 0.0022	0.9100 ± 0.0020	*0.9431 ± 0.0033*	0.9644 ± 0.0030	0.9314
Pruning with re-training	0.9108 ± 0.0025	0.9059 ± 0.0026	0.9390 ± 0.0044	0.9696 ± 0.0017	0.9313
Pruning w/o re-training	0.9002 ± 0.0011	0.9102 ± 0.0016	0.9376 ± 0.0059	0.9556 ± 0.0022	0.9259

Table 1: Performance of the tested variants using roberta-base and a subset size similar to bitfit (except full fine-tuning). All scores are averaged over 5 runs (seeds).

Hyperparameter	Value
Number of epochs	2 or 4
Batch size	16
Weight decay	*none*
Gradient norm clipping	*5.0*
Learning rate schedule	*Linear with warm-up*
Warm-up fraction	*10%*

Table 2: The hyperparameters used in the fine-tuning experiments. Default values of Flair (Akbik et al., 2019) for fine-tuning are denoted in *italics*. For the larger task (QNLI) 2 epochs were used, in all other tasks 4.

grid (in the case of the learning rate, in logarithmic space). In cases where the limit of the range was selected, we considered the experiment to be invalid and repeated it with a larger range. This ensures that a sufficiently large range is selected (assuming the objective is convex with respect to the respective hyperparameters).

3.3 Results

We present the experimental results, first focusing on the different subset selection strategies (Section 3.3.1) and then present an ablation study where we vary the size of the subset (Section 3.3.2).

3.3.1 Selection Strategies

Table 1 reports the performance on each of the four downstream tasks. We make the following observations:

Full fine-tuning best, followed by LoRA. Unsurprisingly, we note that the full fine-tuning baseline outperforms all parameter-efficient fine-tuning methods on all of the tasks. It therefore represents the upper bound that selection-based approaches can achieve. Through not a selection-based approach, we also find that LoRA is always among the top two PEFT methods.

Different selectors score similarly. We also note that different selection-based strategies score similarly, with combined gradient statistics marginally outperforming the other two approaches using gradient statistics. On average it outperforms all proper subset tuning methods which do not require any initial full fine-tuning. The method using combined gradient statistics consistently outperforms the other two approaches using gradient statistics (though only by a small margin).

Surprisingly strong results for random subset. Even the random baseline (using a large enough learning rate), fares surprisingly well. In a single instance, it even outperforms the other PEFT methods. We conclude that the performance differences in these experiments are not drastic and that even a properly tuned random subset scores competitively with more complex approaches.

3.3.2 Subset Size

It is important to consider the joint impact of the subset size and selection method on the ideal learning rate when trying to optimize performance. Hence, in this ablation, we vary the subset size and assess its impact independently.

While the gradient flow throughout the network remains unchanged by the subset, the potential change of the network's function depends on (1) the

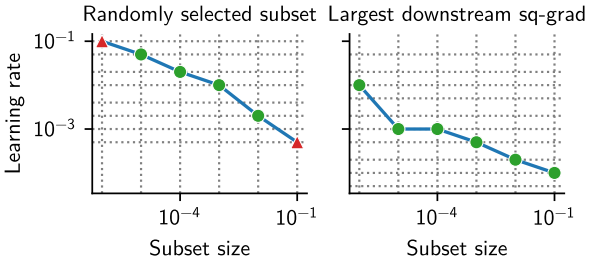


Figure 2: Selected learning rate (y-axis) based on the subset size (x-axis) and two selection strategies: Random (left) and largest average squared gradient on the downstream data (right). A red triangle indicates that the learning rate at the limit of the range was selected and might therefore be suboptimal. For more learning rate selection plots, see Figure 4.

number of parameters that can be affected and (2) the gradient these parameters receive. If the average gradient is much lower for a given set of parameters, a higher learning rate may produce better results.

This is very prominent in the comparison of a random subset and a subset selected by large Fisher Information (see Figure 2). The latter subset receives (on average) a larger gradient magnitude and may therefore require a lower learning rate.

Figure 3 illustrates the impact of the tuned subset size on the downstream performance.

The approaches of using either the combined or only the downstream gradient statistics method outperform all other selective PEFT methods when using very small subset sizes. Pruning without retraining underperforms likely due to the large amount of information that is lost during the pruning step. At small subset sizes and compared to the other approaches, the random baseline does not perform as well. It should be mentioned though, that in the case of the smallest subset size (and for TREC-6 the second smallest), the highest available learning rate of 0.1 was selected. Due to the already large range, we did not repeat this experiment with even larger learning rates.

4 Generality & Adaptability of the Embedding Network

We extend our evaluation to investigate how the generality of the embedding network is impacted by the applied fine-tuning method. To this end, we leverage the transformer networks fine-tuned with different selection strategies on a *primary task* from the previous experiment and evaluate their usefulness for a distinct *secondary task*.

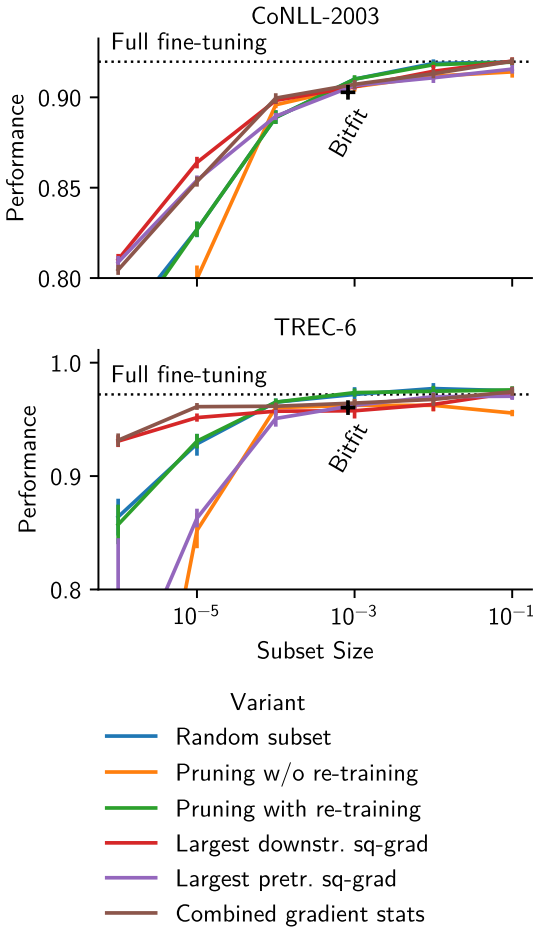


Figure 3: Performance of different variants across different subset sizes

In total, we report the following measures:

1. The test score on the primary task and the generalization gap,
2. the performance on a masked-language modeling (MLM) task using a tuned two-layer probe, and
3. the performance on a set of secondary tasks (after adapting the model).

We therefore assess how the embedding network's function changes in terms of its capability to adapt to new tasks.

4.1 Notes on Measuring Generality

We preface this experiment with the note that the "generality" of a model is no well-defined concept. Zaken et al. (2022) mention the generalization gap (the difference between the test and train performance). We are, however, not only interested in

Variant	Prim.	MLM	Gap	Sec.	Sec. (decoder)
Full fine-tuning	**0.0000 ± 0.0023**	-0.0584 ± 0.0172	-0.0402 ± 0.0189	-0.0020 ± 0.0026	0.0421 ± 0.1070
Regularized FT (L1, 0.01)	-0.0290 ± 0.0178	**-0.0274 ± 0.0087**	-0.0025 ± 0.0194	-0.0029 ± 0.0039	0.0423 ± 0.1015
Regularized FT (L1, 0.10)	-0.0527 ± 0.0264	-0.0299 ± 0.0180	**0.0068 ± 0.0174**	*-0.0018 ± 0.0029*	0.0401 ± 0.0845
Regularized FT (L2, 0.01)	*-0.0025 ± 0.0034*	-0.0431 ± 0.0114	-0.0376 ± 0.0208	-0.0035 ± 0.0036	0.0330 ± 0.1188
Regularized FT (L2, 0.10)	-0.0028 ± 0.0028	*-0.0293 ± 0.0060*	-0.0311 ± 0.0204	-0.0042 ± 0.0041	**0.0614 ± 0.1114**
LoRA (rank 4)	-0.0077 ± 0.0041	-0.0742 ± 0.0109	-0.0225 ± 0.0218	-0.0271 ± 0.1153	0.0014 ± 0.1127
Random subset	-0.0133 ± 0.0079	-0.0675 ± 0.0270	-0.0245 ± 0.0200	-0.0054 ± 0.0038	0.0387 ± 0.1290
Bitfit	-0.0159 ± 0.0068	-0.1202 ± 0.0392	-0.0066 ± 0.0174	-0.0026 ± 0.0029	-0.0096 ± 0.1583
Largest pretr. sq-grad	-0.0172 ± 0.0073	-0.1469 ± 0.0436	-0.0092 ± 0.0212	-0.0051 ± 0.0039	-0.0225 ± 0.1367
Largest downstr. sq-grad	-0.0150 ± 0.0061	-0.1162 ± 0.0328	-0.0078 ± 0.0195	-0.0046 ± 0.0040	0.0033 ± 0.1356
Combined gradient stats	-0.0118 ± 0.0061	-0.1140 ± 0.0249	-0.0072 ± 0.0187	-0.0039 ± 0.0038	0.0112 ± 0.1382
Pruning with re-training	-0.0119 ± 0.0073	-0.0613 ± 0.0195	-0.0238 ± 0.0205	-0.0054 ± 0.0039	*0.0543 ± 0.1280*
Pruning w/o re-training	-0.0173 ± 0.0057	-0.0496 ± 0.0125	*-0.0021 ± 0.0193*	**-0.0014 ± 0.0032**	0.0451 ± 0.1162

Table 3: Performance of the tested variants using roberta-base. Primary and secondary score differences compared to full fine-tuning on the pretrained embedding network. Secondary score of a tuned decoder (compared to a decoder tuned on the pretrained embedding network). Gap refers to the negative train/test gap (values smaller than zero indicate the test score is lower than the train score; the higher the better). MLM score difference to the inital MLM score. All scores are averaged over 5 runs (seeds) and all secondary tasks. Higher values are better (in all of the columns). We mark the best score (per column) in **bold** and the second best in *italics*. See Table 1 for the primary scores on each of the tasks.

whether a model generalizes well to the test data but a broader notion of generality.

Looking solely at the test score is also not sufficient as we might not be confident that the test set represents our deployment distribution. Additionally, the current fine-tuning step might not be the last in our transfer learning pipeline. In these cases, we want to preserve some general language capabilities much like we would like to preserve a good performance on some previous task in a continuous learning setting (see e.g. Kirkpatrick et al., 2017). The primary objective of this work, however, is not to attempt to resolve the question of how to quantify generality.

In light of the vague nature of the objective and due to the lack of a more suitable evaluation framework, we opt to report masked-language modeling (MLM) and performance on secondary tasks as a proxy for generality. Though we are not strictly in a continuous learning setting, these measures can be conceived of as backward and forward transfer (compare with Lopez-Paz and Ranzato, 2022). The first measure represents how much of the previous function (i.e. the masked-language modeling) was preserved, while the second describes how well each variant preserved the task-generality (see Lin et al., 2023) while fine-tuning on a specific task (or averaged across the complete set).

4.2 Experimental Setup

The experimental setup is identical to the first series of experiments (as described in Section 3.2), but extends it by a final step. After fine-tuning the model with one of the approaches, the embedding network is reused with a new task-specific classification head, fine-tuned on a secondary task, and then evaluated on the respective test sets.

During MLM probing, the embedding network remains unchanged while a two-layer MLP decoder head is tuned to solve an MLM task (a small portion of wikitext, see Table 5 in Appendix A for a detailed list of used hyperparameters). After a few epochs of training, the model is evaluated on the test set. Re-training an MLM head may not seem necessary (as one might want to conserve the original embeddings). We believe, however, that a simple transformation (e.g. a rotation, scaling, etc.) should not be counted as a reduction in the general capabilities: The underlying information content would not have changed, only the representation.

Hence, we re-train the MLM decoder to correct for such transformations.

To fine-tune the (already tuned) model on the secondary tasks, we use the same hyperparameter as presented in Table 2. Regardless of the fine-tuning strategy that is applied in the primary adaption, we first tune the task-specific decoder to adapt to the current state of the embedding network (the scores of tuning only the decoder are reported separately; this is similar to Xu et al., 2021). We then apply a full fine-tuning of the model together with the decoder. This ensures a fair evaluation and guarantees we are measuring a property of the current state of the model, not the ability of the approach to adapt the model. The learning rate is selected based on a grid search conducted on the pretrained version of the model. Thus, for all secondary fine-tuning runs, the same learning rates are used.

4.3 Results

Table 3 contains a summary of the collected results. As mentioned in the previous section, LoRA exhibits the largest average primary test scores among the parameter-efficient fine-tuning techniques. In terms of the generalization, it has a mid-range rank.

As expected, using the largest Fisher information on the pretraining data not only fares worse in regard to the primary score but also is one of the worst with respect to its generalization capabilities. Using these statistics combined with the downstream information, however, does slightly improve the subsets based on the largest downstream Fisher information (*largest downstream sq-grad*). If the embedding network is not tuned a second time (but only the task-specific decoder), this approach also outperforms BitFit.

Subset tuning impairs adaptation to new tasks. None of the strategies outperform full fine-tuning in terms of the embedding network's ability to adapt to new tasks by fine-tuning the complete model or only the decoder. Follow-up experiments would be required to determine whether the same applies when fine-tuning the model with the same strategy as in the primary adaptation.

BitFit with small train/test gap. As observed by Zaken et al. (2022), BitFit has a very low train/test gap. In our experiments, it has the lowest train/test gap among the PEFT methods. Only one of the regularized methods has a better gap (here the test score is actually higher; the primary score of this is very low). Full fine-tuning (as one might expect) has the highest overall train/test gap.

4.4 Similar vs. Dissimilar Secondary Tasks

In a follow-up experiment, we assess the impact of the similarity between the primary and secondary task. We first fine-tune a cross-lingual transformer model (`XLM-RoBERTa-base`, 279M parameters, Conneau et al., 2020) on the English version of CoNLL-2003 (a named entity recognition task) and then evaluate its performance after running a secondary fine-tuning on CoNLL-2003 in German (which we assume to be similar as the classes are identical) as well as TREC-6 which is a question classification task and thus differs more from the primary task.

Unfortunately, the data is fairly inconsistent. Since we only used two tasks (one for each category of similar vs. dissimilar), it is not possible to draw any definite conclusions from this. Nonetheless, we include these results in the appendix. Table 8 in the appendix contains a detailed report of these results.

5 Conclusion

In our evaluation of fine-tuning strategies, full fine-tuning consistently outperforms all parameter-efficient fine-tuning (PEFT) methods across various tasks. LoRA consistently ranks among the top two PEFT methods in our experiments.

Examining the utilization of gradient statistics, we observe that the method using combined gradient statistics consistently outperforms its counterparts, although the performance improvement is marginal. On average, this approach surpasses all proper subset tuning methods that do not necessitate initial full fine-tuning.

Nevertheless, it is worth noting that the differences in performance across these experiments may not be substantial enough to justify the added complexity. Surprisingly, even the random baseline, with a sufficiently high learning rate, demonstrates competitive performance, occasionally outperforming other PEFT methods.

Given the strong results of the random baseline and the generally similar performance on primary tasks, our results call into question whether there is a clear optimal subset of parameters to tune. Furthermore, our generalization experiments indicate that selective PEFT strategies impair rather than increase generalizability to secondary tasks, likely due to PEFT affecting more localized and severe changes to the transformer network.

Limitations

The experiments we report on in this paper were performed using a single model (`roberta-base`) and on a limited number of tasks. Hence, there is no guarantee that these findings transfer to large models and more complex transfer-learning scenarios. Due to the exhaustive learning rate search, we set out to conduct and given the resources that were available to us, testing the observations on a larger set of models and tasks was not possible. Testing specific hypotheses on a broader set of models and tasks may be part of future work.

Impact Statement

Large language models have the potential to reproduce multiple forms of stereotypes due to their ability to absorb societal biases ingrained in the training data. Research into parameter-efficient fine-tuning methods is unlikely to change this behavior. Additionally, training of language models is computationally demanding and carries a substantial environmental burden. This complexity further hampers the prospects of reproducing research findings and conducting subsequent studies in an academic setting. Parameter-efficient fine-tuning aims to reduce the required computational resources and might enable broader use of such models.

The experiments we conducted in the context of this paper amount to an estimated number of 150 GPU days using a mix of GPUs (mostly Nvidia Tesla V100S and some Nvidia Ampere A100).

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pages 15834–15846, Red Hook, NY, USA. Curran Associates Inc.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Jingjing Liu, and Zhangyang Wang. 2021. The Elastic Lottery Ticket Hypothesis.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale.

Ganqu Cui, Wentao Li, Ning Ding, Longtao Huang, Zhiyuan Liu, and Maosong Sun. 2023. Decoder Tuning: Efficient Language Understanding as Decoding.

Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv:1803.03635 [cs]*.

Zhuocheng Gong, Di He, Yelong Shen, Tie-Yan Liu, Weizhu Chen, Dongyan Zhao, Ji-Rong Wen, and Rui Yan. 2022. Finding the Dominant Winning Ticket in Pre-Trained Language Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1459–1472, Dublin, Ireland. Association for Computational Linguistics.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward Semantics-Based Answer Pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization.

Yong Lin, Lu Tan, Hangyu Lin, Zeming Zheng, Renjie Pi, Jipeng Zhang, Shizhe Diao, Haoxiang Wang, Han Zhao, Yuan Yao, and Tong Zhang. 2023. Speciality vs Generality: An Empirical Study on Catastrophic Forgetting in Fine-tuning Foundation Models.

David Lopez-Paz and Marc'Aurelio Ranzato. 2022. Gradient Episodic Memory for Continual Learning.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning.

Wang Qi, Yu-Ping Ruan, Yuan Zuo, and Taihao Li. 2022. Parameter-Efficient Tuning on Layer Normalization for Pre-trained Language Models.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training Neural Networks with Fixed Sparse Masks.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a Child in Large Language Model: Towards Effective and Generalizable Fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models.

Rui Zheng, Bao Rong, Yuhao Zhou, Di Liang, Sirui Wang, Wei Wu, Tao Gui, Qi Zhang, and Xuanjing Huang. 2022. Robust Lottery Tickets for Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2211–2224, Dublin, Ireland. Association for Computational Linguistics.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2020. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. *arXiv:1905.01067 [cs, stat]*.

A Additional Setup Details

This section contains the remaining hyperparameters used to produce the results presented in this paper. The code, configurations, and metadata of the results will be published upon acceptance.

Hyperparameter	Value
Number of epochs	5
Learning rate	4×10^{-4}
Batch size	64
Weight decay	*none*
Gradient norm clipping	*5.0*
Learning rate schedule	*Linear with warm-up*
Warm-up fraction	*10%*

Table 4: The hyperparameters used to fine-tune the task-specific decoders. Default values of Flair (Akbik et al., 2019) for fine-tuning are denoted in *italics*.

Hyperparameter	Value
Number of epochs	4
Learning rate	2×10^{-3}
Batch size	64
Weight decay	0.05
Learning rate schedule	Constant

Table 5: The hyperparameters that used to fine-tune the MLM head.

B Additional Results

In the following, we present some alternative perspectives on the experiments discussed in this paper. The results are derived from the same set of experiments and are purely a different way of presenting them.

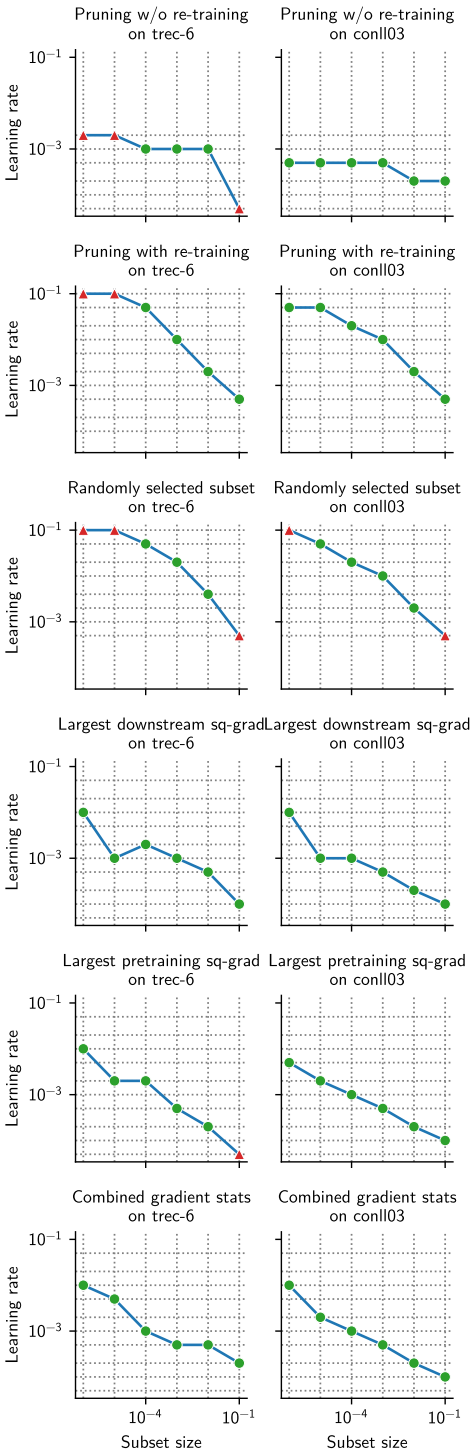


Figure 4: Selected learning rates for each subset size. Each grid intersection represents (at least) one experiment conducted in the parameter search. The best learning is represented by a marker. Learning rates that are at the limits of the tested intervals are marked red and may not be optimal given the used resolution (we used learning rates which, on a logarithmic scale, are approximately equally spaced: 1×10^{-4}, 2×10^{-4}, 5×10^{-4}, 1×10^{-3}, and so on).

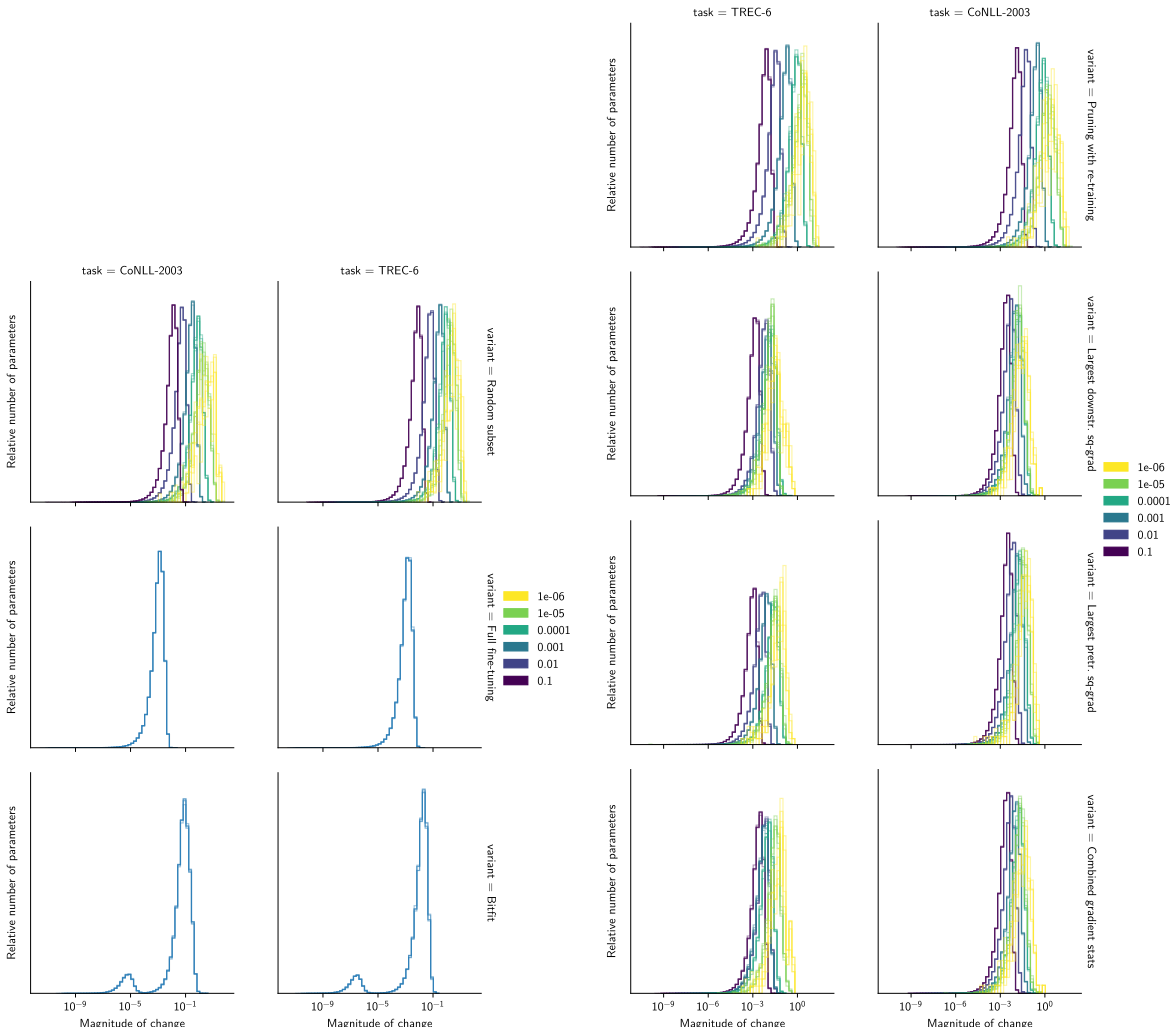


Figure 5: The relative number of parameters with a certain magnitude of change over the different subset sizes.

Primary task	Variant	Primary score	Secondary Score Diff.	MLM Precision @1
CoNLL-2003	Full fine-tuning	**0.9217 ± 0.0009**	-0.0025 ± 0.0025	*0.3756 ± 0.0024*
	Random subset	0.9087 ± 0.0011	-0.0039 ± 0.0038	0.3754 ± 0.0061
	Bitfit	0.9080 ± 0.0013	*-0.0023 ± 0.0023*	0.3481 ± 0.0033
	Largest pretr. sq-grad	0.9073 ± 0.0014	-0.0063 ± 0.0034	0.2965 ± 0.0048
	Largest downstr. sq-grad	0.9073 ± 0.0018	-0.0049 ± 0.0042	0.3283 ± 0.0033
	Combined gradient stats	0.9082 ± 0.0020	-0.0046 ± 0.0037	0.3316 ± 0.0019
	Pruning with re-training	*0.9108 ± 0.0023*	-0.0070 ± 0.0047	0.3552 ± 0.0063
	Pruning w/o re-training	0.9002 ± 0.0010	**-0.0015 ± 0.0030**	**0.3891 ± 0.0028**
QNLI	Full fine-tuning	**0.9290 ± 0.0016**	*-0.0020 ± 0.0016*	**0.4067 ± 0.0016**
	Random subset	0.9048 ± 0.0026	-0.0056 ± 0.0029	0.3817 ± 0.0031
	Bitfit	0.9039 ± 0.0016	-0.0038 ± 0.0029	0.2595 ± 0.0151
	Largest pretr. sq-grad	0.9037 ± 0.0026	-0.0027 ± 0.0019	0.3040 ± 0.0125
	Largest downstr. sq-grad	0.9075 ± 0.0009	-0.0037 ± 0.0035	0.3461 ± 0.0113
	Combined gradient stats	0.9100 ± 0.0018	-0.0028 ± 0.0033	0.3407 ± 0.0103
	Pruning with re-training	0.9059 ± 0.0024	-0.0051 ± 0.0036	*0.3828 ± 0.0056*
	Pruning w/o re-training	*0.9102 ± 0.0014*	**-0.0013 ± 0.0032**	0.3825 ± 0.0028
SST-2	Full fine-tuning	**0.9468 ± 0.0012**	*-0.0012 ± 0.0020*	**0.3957 ± 0.0062**
	Random subset	0.9342 ± 0.0025	-0.0077 ± 0.0038	0.3394 ± 0.0059
	Bitfit	0.9383 ± 0.0024	-0.0019 ± 0.0023	0.3393 ± 0.0036
	Largest pretr. sq-grad	0.9378 ± 0.0048	-0.0021 ± 0.0023	0.3531 ± 0.0036
	Largest downstr. sq-grad	0.9399 ± 0.0028	-0.0017 ± 0.0027	0.3611 ± 0.0029
	Combined gradient stats	*0.9431 ± 0.0030*	**-0.0011 ± 0.0025**	0.3582 ± 0.0030
	Pruning with re-training	0.9390 ± 0.0040	-0.0051 ± 0.0035	0.3854 ± 0.0036
	Pruning w/o re-training	0.9376 ± 0.0054	-0.0021 ± 0.0033	*0.3950 ± 0.0021*
TREC-6	Full fine-tuning	**0.9752 ± 0.0042**	*-0.0022 ± 0.0031*	0.3669 ± 0.0073
	Random subset	*0.9720 ± 0.0029*	*-0.0022 ± 0.0030*	*0.4135 ± 0.0033*
	Bitfit	0.9592 ± 0.0054	-0.0024 ± 0.0036	0.3505 ± 0.0056
	Largest pretr. sq-grad	0.9552 ± 0.0056	-0.0091 ± 0.0045	0.2354 ± 0.0112
	Largest downstr. sq-grad	0.9580 ± 0.0045	-0.0067 ± 0.0037	0.2781 ± 0.0133
	Combined gradient stats	0.9644 ± 0.0027	-0.0048 ± 0.0042	0.2936 ± 0.0099
	Pruning with re-training	0.9696 ± 0.0015	-0.0026 ± 0.0025	0.4097 ± 0.0037
	Pruning w/o re-training	0.9556 ± 0.0020	**-0.0003 ± 0.0024**	**0.4149 ± 0.0031**

Table 6: Performance of the tested variants using roberta-base. Primary and secondary score compared to full fine-tuning on the pretrained embedding. MLM is the MLM precision @1 score. All scores are averaged over 5 runs (seeds) and all secondary tasks.

Primary task	Variant	Primary score	MLM score	CoNLL-2003	QNLI	SST-2	TREC-6
CoNLL-2003	Bitfit	0.9080 ± 0.0013	0.3481 ± 0.0033	*0.9213 ± 0.0014*	*0.9269 ± 0.0016*	**0.9433 ± 0.0026**	0.9720 ± 0.0024
	Combined gradient stats	0.9082 ± 0.0020	0.3316 ± 0.0019	0.9197 ± 0.0020	0.9239 ± 0.0014	0.9383 ± 0.0021	0.9724 ± 0.0048
	Full fine-tuning	**0.9217 ± 0.0009**	*0.3756 ± 0.0024*	0.9206 ± 0.0019	0.9255 ± 0.0009	**0.9433 ± 0.0039**	0.9732 ± 0.0023
	Largest downstr. sq-grad	0.9073 ± 0.0018	0.3283 ± 0.0033	0.9204 ± 0.0017	0.9248 ± 0.0014	0.9358 ± 0.0021	0.9720 ± 0.0024
	Largest pretr. sq-grad	0.9073 ± 0.0014	0.2965 ± 0.0048	0.9180 ± 0.0017	0.9235 ± 0.0020	0.9381 ± 0.0032	0.9680 ± 0.0047
	Pruning w/o re-training	0.9002 ± 0.0010	**0.3891 ± 0.0028**	**0.9218 ± 0.0021**	**0.9277 ± 0.0013**	0.9424 ± 0.0031	*0.9748 ± 0.0036*
	Pruning with re-training	*0.9108 ± 0.0023*	0.3552 ± 0.0063	0.9186 ± 0.0038	0.9179 ± 0.0013	0.9369 ± 0.0041	0.9712 ± 0.0033
	Random subset	0.9087 ± 0.0011	0.3754 ± 0.0061	0.9188 ± 0.0013	0.9233 ± 0.0021	0.9394 ± 0.0040	**0.9756 ± 0.0017**
QNLI	Bitfit	0.9039 ± 0.0016	0.2595 ± 0.0151	0.9188 ± 0.0022	0.9248 ± 0.0015	0.9406 ± 0.0026	0.9736 ± 0.0033
	Combined gradient stats	0.9100 ± 0.0018	0.3407 ± 0.0103	0.9173 ± 0.0016	0.9256 ± 0.0009	0.9420 ± 0.0019	*0.9768 ± 0.0036*
	Full fine-tuning	**0.9290 ± 0.0016**	**0.4067 ± 0.0016**	**0.9206 ± 0.0013**	**0.9270 ± 0.0015**	**0.9450 ± 0.0014**	0.9720 ± 0.0020
	Largest downstr. sq-grad	0.9075 ± 0.0009	0.3461 ± 0.0113	0.9182 ± 0.0004	0.9257 ± 0.0013	0.9388 ± 0.0021	0.9752 ± 0.0033
	Largest pretr. sq-grad	0.9037 ± 0.0026	0.3040 ± 0.0125	0.9184 ± 0.0015	0.9257 ± 0.0010	0.9445 ± 0.0033	0.9732 ± 0.0011
	Pruning w/o re-training	*0.9102 ± 0.0014*	0.3825 ± 0.0028	*0.9193 ± 0.0020*	*0.9263 ± 0.0017*	*0.9447 ± 0.0030*	**0.9772 ± 0.0036**
	Pruning with re-training	0.9059 ± 0.0024	*0.3828 ± 0.0056*	0.9161 ± 0.0024	0.9240 ± 0.0020	0.9399 ± 0.0055	0.9724 ± 0.0033
	Random subset	0.9048 ± 0.0026	0.3817 ± 0.0031	0.9175 ± 0.0015	0.9246 ± 0.0027	0.9385 ± 0.0031	0.9696 ± 0.0030
SST-2	Bitfit	0.9383 ± 0.0024	0.3393 ± 0.0036	*0.9199 ± 0.0011*	*0.9281 ± 0.0018*	0.9445 ± 0.0037	0.9728 ± 0.0023
	Combined gradient stats	*0.9431 ± 0.0030*	0.3582 ± 0.0030	0.9196 ± 0.0012	0.9268 ± 0.0009	**0.9472 ± 0.0037**	**0.9748 ± 0.0027**
	Full fine-tuning	**0.9468 ± 0.0012**	**0.3957 ± 0.0062**	**0.9205 ± 0.0012**	**0.9284 ± 0.0011**	0.9443 ± 0.0019	**0.9748 ± 0.0030**
	Largest downstr. sq-grad	0.9399 ± 0.0028	0.3611 ± 0.0029	0.9192 ± 0.0017	0.9277 ± 0.0018	0.9450 ± 0.0014	0.9740 ± 0.0051
	Largest pretr. sq-grad	0.9378 ± 0.0048	0.3531 ± 0.0036	0.9197 ± 0.0010	0.9278 ± 0.0016	0.9450 ± 0.0024	0.9720 ± 0.0037
	Pruning w/o re-training	0.9376 ± 0.0054	*0.3950 ± 0.0021*	*0.9199 ± 0.0020*	0.9262 ± 0.0006	*0.9461 ± 0.0030*	0.9720 ± 0.0057
	Pruning with re-training	0.9390 ± 0.0040	0.3854 ± 0.0036	0.9166 ± 0.0007	0.9222 ± 0.0022	0.9420 ± 0.0057	0.9716 ± 0.0036
	Random subset	0.9342 ± 0.0025	0.3394 ± 0.0059	0.9142 ± 0.0029	0.9183 ± 0.0034	0.9420 ± 0.0047	0.9676 ± 0.0017
TREC-6	Bitfit	0.9592 ± 0.0054	0.3505 ± 0.0056	0.9192 ± 0.0011	**0.9306 ± 0.0004**	0.9415 ± 0.0044	0.9720 ± 0.0032
	Combined gradient stats	0.9644 ± 0.0027	0.2936 ± 0.0099	0.9161 ± 0.0008	0.9252 ± 0.0031	0.9378 ± 0.0045	*0.9744 ± 0.0030*
	Full fine-tuning	**0.9752 ± 0.0042**	0.3669 ± 0.0073	0.9197 ± 0.0009	*0.9290 ± 0.0024*	0.9420 ± 0.0047	0.9732 ± 0.0018
	Largest downstr. sq-grad	0.9580 ± 0.0045	0.2781 ± 0.0133	0.9169 ± 0.0014	0.9237 ± 0.0017	0.9365 ± 0.0021	0.9688 ± 0.0058
	Largest pretr. sq-grad	0.9552 ± 0.0056	0.2354 ± 0.0112	0.9154 ± 0.0016	0.9205 ± 0.0031	0.9365 ± 0.0033	0.9640 ± 0.0076
	Pruning w/o re-training	0.9556 ± 0.0020	**0.4149 ± 0.0031**	**0.9207 ± 0.0023**	0.9285 ± 0.0013	**0.9486 ± 0.0025**	0.9740 ± 0.0024
	Pruning with re-training	0.9696 ± 0.0015	0.4097 ± 0.0037	*0.9206 ± 0.0015*	0.9252 ± 0.0013	*0.9427 ± 0.0024*	0.9740 ± 0.0032
	Random subset	*0.9720 ± 0.0029*	*0.4135 ± 0.0033*	0.9200 ± 0.0012	0.9275 ± 0.0023	0.9415 ± 0.0014	**0.9748 ± 0.0041**

Table 7: Performance of full fine-tuning on a secondary task after a applying each variant on the primary task using a RoBERTa (base). All scores are averaged over 5 runs (std in parentheses).

		CoNLL-2003 (English)	CoNLL-2003 (German)		TREC-6	
			Sec. (decoder)	Sec. (full)	Sec. (decoder)	Sec. (full)
Subset size	Variant					
0.000100	Combined gradient stats	**0.8874 ± 0.0017**	*0.7808 ± 0.0058*	*0.8659 ± 0.0041*	*0.5048 ± 0.0934*	0.9624 ± 0.0054
	Largest downstr. sq-grad	0.8849 ± 0.0039	0.7749 ± 0.0025	0.8648 ± 0.0033	**0.5252 ± 0.0354**	*0.9660 ± 0.0065*
	Largest pretr. sq-grad	0.8665 ± 0.0023	0.7419 ± 0.0056	0.8608 ± 0.0035	0.4928 ± 0.0966	*0.9660 ± 0.0051*
	Pruning w/o re-training	*0.8867 ± 0.0016*	**0.7926 ± 0.0025**	**0.8717 ± 0.0017**	0.4860 ± 0.0462	**0.9716 ± 0.0046**
	Pruning with re-training	0.8618 ± 0.0028	0.4617 ± 0.0134	0.8598 ± 0.0035	0.3048 ± 0.0539	0.9636 ± 0.0048
	Random subset	0.8590 ± 0.0026	0.3151 ± 0.0210	0.8504 ± 0.0049	0.2576 ± 0.0095	0.9572 ± 0.0041
0.001000	Combined gradient stats	0.8962 ± 0.0009	0.7868 ± 0.0055	0.8690 ± 0.0025	0.4708 ± 0.0768	**0.9700 ± 0.0032**
	Largest downstr. sq-grad	0.8980 ± 0.0025	*0.7883 ± 0.0041*	*0.8704 ± 0.0033*	0.4468 ± 0.0389	**0.9700 ± 0.0032**
	Largest pretr. sq-grad	0.8910 ± 0.0027	0.7655 ± 0.0087	0.8654 ± 0.0029	**0.5488 ± 0.0318**	0.9640 ± 0.0081
	Pruning w/o re-training	0.8891 ± 0.0019	**0.7899 ± 0.0019**	**0.8719 ± 0.0009**	*0.4972 ± 0.0256*	**0.9700 ± 0.0051**
	Pruning with re-training	*0.8986 ± 0.0013*	0.7410 ± 0.0030	0.8578 ± 0.0047	0.4404 ± 0.0632	0.9680 ± 0.0051
	Random subset	**0.9000 ± 0.0020**	0.7430 ± 0.0106	0.8581 ± 0.0035	0.3924 ± 0.0724	0.9684 ± 0.0057
0.010000	Combined gradient stats	0.9081 ± 0.0019	0.7896 ± 0.0066	0.8682 ± 0.0027	0.4524 ± 0.0713	0.9656 ± 0.0078
	Largest downstr. sq-grad	0.9078 ± 0.0018	**0.7991 ± 0.0027**	*0.8683 ± 0.0020*	0.4240 ± 0.0770	*0.9696 ± 0.0017*
	Largest pretr. sq-grad	0.9028 ± 0.0012	0.7671 ± 0.0040	0.8636 ± 0.0056	0.5052 ± 0.0278	0.9636 ± 0.0033
	Pruning w/o re-training	0.9078 ± 0.0015	*0.7987 ± 0.0047*	**0.8727 ± 0.0034**	**0.5352 ± 0.0276**	0.9680 ± 0.0047
	Pruning with re-training	**0.9121 ± 0.0019**	0.7912 ± 0.0040	0.8668 ± 0.0027	*0.5316 ± 0.0415*	**0.9704 ± 0.0017**
	Random subset	*0.9112 ± 0.0022*	0.7927 ± 0.0022	0.8670 ± 0.0038	0.4956 ± 0.0478	0.9636 ± 0.0038
0.100000	Combined gradient stats	*0.9135 ± 0.0026*	*0.7918 ± 0.0052*	0.8661 ± 0.0051	0.5112 ± 0.0363	0.9676 ± 0.0043
	Largest downstr. sq-grad	**0.9139 ± 0.0026**	0.7881 ± 0.0037	0.8629 ± 0.0043	*0.5420 ± 0.0248*	**0.9696 ± 0.0048**
	Largest pretr. sq-grad	0.9117 ± 0.0020	0.7811 ± 0.0049	*0.8664 ± 0.0019*	0.5196 ± 0.0528	0.9660 ± 0.0062
	Pruning w/o re-training	0.9080 ± 0.0017	**0.7989 ± 0.0047**	**0.8720 ± 0.0028**	0.5332 ± 0.0301	*0.9688 ± 0.0052*
	Pruning with re-training	0.9123 ± 0.0019	0.7885 ± 0.0057	0.8651 ± 0.0051	**0.5536 ± 0.0417**	*0.9688 ± 0.0066*
	Random subset	0.9108 ± 0.0017	0.7701 ± 0.0064	0.8638 ± 0.0044	0.4304 ± 0.1021	0.9656 ± 0.0055

Table 8: Score a full fine-tuning on CoNLL-2003 (german), compared to a baseline (full ft on pretrained) of 0.8724 (0.0023), and TREC-6 after fine-tuning using each of the variants on CoNLL03 (english). Each score is shown in the respective column.

Task	CoNLL-2003	QNLI	SST-2	TREC-6
Variant				
LoRA (rank 1, 0.03%)	0.9059 ± 0.0025	0.9072 ± 0.0026	0.9353 ± 0.0043	0.9636 ± 0.0043
LoRA (rank 2, 0.06%)	0.9114 ± 0.0020	0.9133 ± 0.0022	0.9353 ± 0.0039	0.9652 ± 0.0039
LoRA (rank 3, 0.09%)	0.9129 ± 0.0011	0.9157 ± 0.0026	0.9360 ± 0.0039	0.9692 ± 0.0048
LoRA (rank 4, 0.12%)	0.9139 ± 0.0018	0.9165 ± 0.0021	**0.9406 ± 0.0031**	0.9708 ± 0.0041
LoRA (rank 16, 0.47%)	*0.9173 ± 0.0009*	*0.9202 ± 0.0014*	*0.9404 ± 0.0054*	*0.9712 ± 0.0046*
LoRA (rank 64, 1.89%)	**0.9185 ± 0.0020**	**0.9217 ± 0.0009**	0.9399 ± 0.0029	**0.9744 ± 0.0033**

Table 9: Performance of Low-Rank adoption (Hu et al., 2021) across four different tasks (five runs each).

Primary task	Primary score	MLM score	CoNLL-2003	QNLI	SST-2	TREC-6
CoNLL-2003	0.9139 ± 0.0016	0.3659 ± 0.0125	0.9173 ± 0.0014	0.9166 ± 0.0017	0.9385 ± 0.0046	0.9708 ± 0.0033
QNLI	0.9165 ± 0.0020	0.3776 ± 0.0112	0.9167 ± 0.0005	0.9260 ± 0.0021	0.9392 ± 0.0035	0.8792 ± 0.1997
SST-2	0.9406 ± 0.0028	0.3737 ± 0.0046	0.9173 ± 0.0009	0.9218 ± 0.0013	0.9420 ± 0.0015	0.8112 ± 0.3484
TREC-6	0.9708 ± 0.0038	0.3659 ± 0.0089	0.9178 ± 0.0009	0.9227 ± 0.0025	0.9411 ± 0.0024	0.9700 ± 0.0014

Table 10: Performance of Low-Rank adoption with a rank of 4 (Hu et al., 2021) after fine-tuning on secondary task (five runs each).

Primary task	Reg.	Coeff.	Primary score	Gap	MLM score	CoNLL-2003	QNLI	SST-2	TREC-6
CoNLL-2003	l1	0.01	0.9013 ± 0.0007	-0.0337 ± 0.0004	0.4044 ± 0.0016	*0.9217 ± 0.0008*	**0.9278 ± 0.0009**	0.9399 ± 0.0039	0.9712 ± 0.0036
		0.10	0.8824 ± 0.0013	*-0.0165 ± 0.0014*	0.3869 ± 0.0026	0.9211 ± 0.0009	*0.9275 ± 0.0010*	*0.9424 ± 0.0025*	*0.9744 ± 0.0033*
		1.00	0.8387 ± 0.0015	**-0.0101 ± 0.0009**	**0.4106 ± 0.0007**	0.9215 ± 0.0013	0.9267 ± 0.0018	**0.9433 ± 0.0059**	**0.9752 ± 0.0018**
	l2	0.01	*0.9203 ± 0.0010*	-0.0704 ± 0.0009	0.3870 ± 0.0053	0.9211 ± 0.0025	0.9229 ± 0.0013	0.9404 ± 0.0018	0.9684 ± 0.0036
		0.10	**0.9210 ± 0.0025**	-0.0654 ± 0.0027	0.4067 ± 0.0035	0.9213 ± 0.0022	0.9248 ± 0.0007	0.9394 ± 0.0029	0.9720 ± 0.0028
		1.00	0.9192 ± 0.0016	-0.0595 ± 0.0009	*0.4086 ± 0.0058*	**0.9223 ± 0.0013**	0.9237 ± 0.0025	0.9413 ± 0.0053	0.9720 ± 0.0020
QNLI	l1	0.01	0.8701 ± 0.0008	*0.0149 ± 0.0009*	0.4177 ± 0.0022	0.9197 ± 0.0013	0.9258 ± 0.0010	0.9415 ± 0.0037	0.9756 ± 0.0030
		0.10	0.8323 ± 0.0008	**0.0198 ± 0.0012**	*0.4259 ± 0.0016*	*0.9211 ± 0.0021*	0.9267 ± 0.0012	0.9415 ± 0.0045	0.9748 ± 0.0023
		1.00	0.6640 ± 0.0005	0.0086 ± 0.0005	**0.4434 ± 0.0013**	**0.9218 ± 0.0018**	0.9276 ± 0.0017	*0.9436 ± 0.0038*	*0.9760 ± 0.0042*
	l2	0.01	**0.9270 ± 0.0018**	-0.0203 ± 0.0020	0.4146 ± 0.0036	0.9196 ± 0.0014	**0.9310 ± 0.0014**	0.9385 ± 0.0033	*0.9760 ± 0.0037*
		0.10	*0.9242 ± 0.0016*	-0.0174 ± 0.0016	0.4210 ± 0.0013	0.9194 ± 0.0012	0.9301 ± 0.0018	0.9344 ± 0.0033	0.9700 ± 0.0058
		1.00	0.9132 ± 0.0018	-0.0041 ± 0.0014	0.4247 ± 0.0026	0.9189 ± 0.0018	*0.9306 ± 0.0018*	**0.9443 ± 0.0039**	**0.9776 ± 0.0026**
SST-2	l1	0.01	0.9326 ± 0.0022	-0.0026 ± 0.0023	0.4279 ± 0.0017	**0.9206 ± 0.0020**	0.9275 ± 0.0018	*0.9413 ± 0.0026*	0.9752 ± 0.0018
		0.10	0.9177 ± 0.0014	*-0.0023 ± 0.0013*	*0.4328 ± 0.0012*	0.9194 ± 0.0013	**0.9284 ± 0.0013**	0.9401 ± 0.0022	*0.9772 ± 0.0023*
		1.00	0.8711 ± 0.0012	**0.0170 ± 0.0013**	**0.4400 ± 0.0015**	0.9182 ± 0.0028	*0.9277 ± 0.0024*	0.9392 ± 0.0034	**0.9776 ± 0.0022**
	l2	0.01	*0.9436 ± 0.0020*	-0.0386 ± 0.0020	0.4034 ± 0.0019	*0.9198 ± 0.0011*	0.9256 ± 0.0012	0.9411 ± 0.0032	0.9736 ± 0.0036
		0.10	**0.9438 ± 0.0021**	-0.0255 ± 0.0019	0.4169 ± 0.0027	0.9196 ± 0.0017	0.9269 ± 0.0021	0.9394 ± 0.0030	0.9736 ± 0.0048
		1.00	0.9429 ± 0.0012	-0.0118 ± 0.0010	0.4266 ± 0.0016	*0.9198 ± 0.0017*	0.9257 ± 0.0005	**0.9417 ± 0.0017**	0.9752 ± 0.0030
TREC-6	l1	0.01	0.9528 ± 0.0021	0.0114 ± 0.0022	0.4203 ± 0.0005	*0.9216 ± 0.0024*	0.9250 ± 0.0017	0.9378 ± 0.0056	0.9724 ± 0.0046
		0.10	0.9296 ± 0.0024	*0.0262 ± 0.0035*	0.4133 ± 0.0015	0.9201 ± 0.0009	*0.9269 ± 0.0021*	*0.9424 ± 0.0025*	**0.9764 ± 0.0026**
		1.00	0.4836 ± 0.0033	**0.0568 ± 0.0056**	**0.4434 ± 0.0017**	0.9190 ± 0.0012	**0.9288 ± 0.0008**	0.9411 ± 0.0015	*0.9756 ± 0.0017*
	l2	0.01	*0.9720 ± 0.0061*	-0.0209 ± 0.0056	0.4011 ± 0.0058	**0.9217 ± 0.0019**	0.9265 ± 0.0015	**0.9438 ± 0.0038**	0.9716 ± 0.0033
		0.10	**0.9724 ± 0.0030**	-0.0159 ± 0.0025	0.4166 ± 0.0023	0.9207 ± 0.0018	0.9268 ± 0.0014	0.9397 ± 0.0017	0.9680 ± 0.0047
		1.00	0.9680 ± 0.0050	-0.0086 ± 0.0057	*0.4233 ± 0.0035*	0.9203 ± 0.0021	0.9263 ± 0.0021	0.9417 ± 0.0041	0.9716 ± 0.0017

Table 11: Effect of regularization on primary and secondary scores.

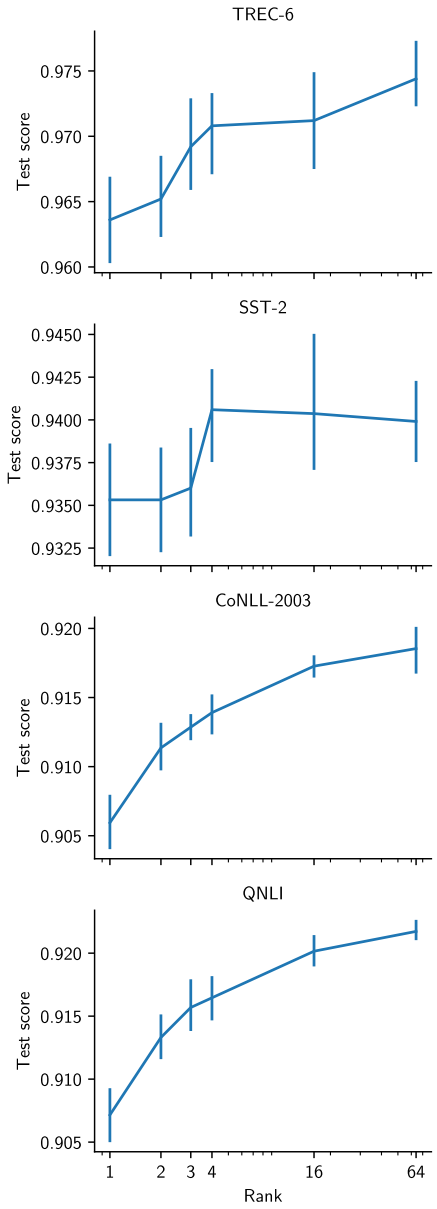


Figure 6: Primary test performance using Low-Rank adoption (Hu et al., 2021) with varying ranks.