BitFit+: Fine-tuning bias and gamma parameters

Yan (Roger) Weng
Princeton University
 yw0947@princeton.edu

Sean Kerrigan
Princeton University
sk5009@princeton.edu

Kevin Yang
Princeton University
ky6374@princeton.edu

Abstract

In our paper, we reproduce and expand on the BitFit paper (Ben-Zaken et al., 2021), which introduced a method for fine-tuning called BitFit where only bias parameters are trained. After replicating results, we added model ablations, testing new parameters for fine-tuning on three GLUE benchmark tasks (Wang et al., 2018). In order to see if b_{m_2} and b_q, specific bias parameters in the BERT model (Devlin et al., 2018), are important for fine-tuning, we first fine-tuned only a random subset of bias parameters. Then, we fine-tuned using all bias parameters except for b_{m_2} and b_q. Finally, we fine-tuned using combinations of the gamma parameters of the BERT model, g_{LN_1} and g_{LN_2}, to see if the bias parameters are special in regards to fine-tuning. We found that BitFit remains the best computationally feasible fine-tuning method, b_{m_2} and b_q may not be the most important bias parameters for fine-tuning, and the gamma parameters are inferior to the bias parameters for fine-tuning. All code used in this study is available on GitHub [1].

1 Introduction

To perform a specific task in NLP, fine-tuning a pre-trained model is often used because it allows the model to update its weights to adapt to specific tasks or datasets, improving its performance on new data that was not part of its initial training. This is often better than training a new model from scratch because it leverages knowledge the pre-trained model has already learned, reducing the need for large amounts of data and extensive computational resources.

However, the traditional method of fine-tuning all parameters can still be time-consuming for certain tasks. This process can delay deployment, especially in scenarios where rapid model updating is crucial. In addition, in settings where memory is constrained, fine-tuning all model parameters is not desirable. Instead, it would be advantageous to fine-tune a small subset of the model's parameters while keeping the rest fixed.

The BitFit paper (Ben-Zaken et al., 2021) proposes fine-tuning the bias terms of pre-trained models, which achieves results comparable to full fine-tuning while using only 0.09% of all parameters. We replicated the paper's fine-tuning approaches on three different tasks in the GLUE benchmark (Wang et al., 2018).

In addition, we implemented new methods of fine-tuning by (1) only fine-tuning a random set of bias terms, (2) only fine-tuning bias terms different than the ones covered by BitFit (Ben-Zaken et al., 2021), (3) only fine-tuning the gamma scale parameters at layer normalization stages in the multi-layer perceptron (MLP), and (4) only fine-tuning the bias parameters at layer normalization stages. We then compared our results to the original BitFit paper.

2 Related work

Pre-trained LLMs are fine-tuned with task-specific datasets. As fine-tuning all parameters can be computationally expensive, finding ways to change fewer parameters while still maintaining high accuracy is a crucial problem. This fine-tuning is known as parameter-efficient fine-tuning (PEFT) (Xu et al., 2023).

Several approaches have been taken to accomplish this task, including "Adapters" by (Houlsby et al., 2019). Instead of changing parameters of the original pre-trained model, they add task-specific "adapter" modules between each of the layers of the model, allowing the pre-trained model parameters to stay unchanged after. This falls under the umbrella of a type of PEFT known as additive fine-tuning, which involves adding new trainable parameters to the model for task-specific fine-tuning (Xu et al., 2023). Though their fine-tuned models achieved performances within 0.04% of full

[1] https://github.com/Roger-Weng/NLP_Final_Project

fine-tuning, they added 3.6% parameters per task, whereas BitFit requires just 0.09% of the model's original parameters to change.

Another approach is "Diff-Pruning" by (Guo et al., 2020). Unlike the Adapter method, Diff-Pruning does not add new parameters to the pre-trained model. Diff-Pruning is more parameter efficient, changing only 0.05% of the model's parameters, and achieves better scores on tasks. Diff-Pruning instead learns a task-specific diff vector during fine-tuning. This diff vector represents the difference between the original model parameters and the task-specific updated parameters. Diff-Pruning uses regularization to keep the diff vector sparse and adds it to the original parameters to update the model. The Diff-Pruning method and BitFit are both examples of partial fine-tuning PEFT methods, where a small portion of original model parameters are changed (Xu et al., 2023). Despite Diff-Pruning being an effective fine-tuning method that uses few parameters, one problem is that parameters are inconsistently updated between tasks. Fine-tuning for different tasks will change different parameters, which is less efficient for hardware based deployments. On the other hand, BitFit satisfies this ideal, fine-tuning the same bias terms every time.

3 Methods

3.1 Replications

We replicated the experiments performed by the authors of the original BitFit paper to test the effectiveness of fine-tuning the bias terms of pre-trained models.

We used the BERT-base model (Devlin et al., 2018) as the pre-trained model that we do fine-tuning on, and we describe its basic architecture below:

The BERT-base model has 12 layers, each layer starting with M self-attention heads. A self attention head has key, query and value encoders, each taking the form of a linear layer. The self-attention head is described by Figure 1.

$$\begin{aligned}\mathbf{Q}^{m,\ell}(\mathbf{x}) &= \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}\\ \mathbf{K}^{m,\ell}(\mathbf{x}) &= \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}\\ \mathbf{V}^{m,\ell}(\mathbf{x}) &= \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}\end{aligned}$$

Figure 1: Attention head

The key, query, value encoders are then combined using an attention mechanism to form h_1^l, which is fed to an MLP with layer-norm (LN). The MLP is described by Figure 2

$$\begin{aligned}\mathbf{h}_2^\ell &= \text{Dropout}\big(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell \;+\; \mathbf{b}_{m_1}^\ell\big)\\ \mathbf{h}_3^\ell &= \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell\\ \mathbf{h}_4^\ell &= \;\text{GELU}\big(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell \;+\; \mathbf{b}_{m_2}^\ell\big)\\ \mathbf{h}_5^\ell &= \text{Dropout}\big(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell \;+\; \mathbf{b}_{m_3}^\ell\big)\\ \text{out}^\ell &= \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell\end{aligned}$$

Figure 2: MLP

We adapted the fine-tuning types suggested by the original paper (Ben-Zaken et al., 2021), and fine-tuned different subsets of the BERT-base model's parameters, which include:

(1) all bias terms in the model, consisting of 0.09% of total parameters

(2) two specific bias terms b_{m_2} and b_q, consisting of 0.04% of total parameters

(3) frozen (no fine-tuning), consisting of 0.00% of total parameters

(4) a random subset of all parameters that consist of 0.09% of total parameters (this matches the number of parameters for all bias terms)

(5) random parameters that are sampled as complete rows/columns in the parameter matrices (also consist of 0.09% of total parameters)

For each fine-tuning type, we trained our model for 16 epochs. We selected this number based on observations that accuracy tends to converge within a maximum of 20 epochs. We believe that 16 epochs strike a balance between achieving optimal performance and utilizing our computational resources efficiently.

We then tested our fine-tuned models on 3 out of the 9 tasks selected from GLUE benchmark – The Corpus of Linguistic Acceptability (CoLA), determining whether an English sentence is grammatical or not; Microsoft Research Paraphrase Corpus (MRPC), determining whether two sentences are paraphrases of each other; Recognizing Textual Entailment (RTE), determining the logical relationship between two text snippets – a premise and a hypothesis. The metrics used for measuring the performance of the model are matthew's correlation for CoLA, F1 score for MRPC, and accuracy for RTE.

We ran the code provided by the original paper (Ben-Zaken et al., 2021), and used the suggested learning rates for training. The learning rate used is $7 * 10^{-4}$ on CoLA, $7 * 10^{-4}$ on MRPC, and 10^{-3} on RTE.

The results of the model performance after fine-tuning are illustrated in Section 4.

We decided not to replicate the full fine-tuning part of the BitFit paper, due to computational resource limitations.

3.2 Model Ablations

In addition to the experiments we adapted from the original paper (Ben-Zaken et al., 2021), we want to further test if bias parameters are special in the fine-tuning process and if b_{m_2} and b_q hold more importance than the other bias parameters.

To do this, we implemented new types of fine-tuning by (1) only fine-tuning a random set of bias terms, (2) only fine-tuning bias terms different than b_{m_2} and b_q, the ones covered by the original paper (Ben-Zaken et al., 2021), (3) only fine-tuning the scale parameters at layer normalization stages, and (4) only fine-tuning b_{LN_1} and b_{LN_2}.

For the first ablation, we fine-tuned on a random subset of bias terms by using a masking technique to randomly select bias parameters for training. The number of bias terms selected randomly is 0.04% of all parameters, the same percentage as b_{m_2} and b_q combined, which the authors of the BitFit paper (Ben-Zaken et al., 2021) claim is a subset of bias parameters that rivals full-model fine-tuning. To select these parameters, we implemented a method to iterate through every bias vector in each of the 12 layers of the BERT model and randomly choose elements in these vectors to be unmasked, giving us a diverse set of bias terms.

For the second ablation, we fine-tuned all bias terms other than b_{m_2} and b_q, and then compared the results with fine-tuning only b_{m_2} and b_q. To do this, we set the trainable bias terms to all bias terms apart from b_{m_2} and b_q based on the code provided by the original paper (Ben-Zaken et al., 2021).

For the third ablation, we fine-tuned only the gamma scale parameters at layer normalization stages (g_{LN_1} and g_{LN_2}) and compared the model performance to that of fine-tuning all bias parameters. We accomplished this by extracting the gamma parameters from each layer of the model using their respective names, attention.output.LayerNorm.weight and output.LayerNorm.weight. We added an option for g_{LN_1} and g_{LN_2} as arguments, allowing us to filter out all other parameters.

For the last type of ablation, we fine-tuned b_{LN_1} and b_{LN_2}. To do this, we set the trainable bias terms to b_{LN_1} and b_{LN_2} based on the code provided by the original paper (Ben-Zaken et al., 2021).

As in model replications, for every fine-tuning type, we trained our model for 16 epochs and then tested our fine-tuned models on the same 3 tasks selected from GLUE benchmark.

Due to constraints on GPU usage, we were limited to fine-tuning the BERT-base model once per task for each type of fine-tuning.

4 Results and Analysis

In the BitFit paper (Ben-Zaken et al., 2021), the authors made two main claims:

1. Fine-tuning only the additive bias terms (BitFit) is effective, and bias parameters are special, since the average score of BitFit is comparable to the average score of Full-FT and the average score of rand uniform and rand row/col is substantially worse than that of BitFit (Ben-Zaken et al., 2021).

2. Fine-tuning only the bias parameters associated with the query and the second MLP layer (b_{m_2} and b_q), is an effective subset of bias parameters, since those two bias terms have the greatest magnitude of change during fine-tuning, and the average score of fine-tuning on b_{m_2} and b_q is comparable to that of BitFit.

Our goal in this paper was to replicate those results as well as further investigate each claim.

4.1 Replications

We ran five replication tests on three different tasks for 16 epochs. We ran BitFit, b_{m_2}/b_q, Frozen, and rand uniform and rand row/column, and took the results from the original paper (Ben-Zaken et al., 2021) for the Full-FT results. Again, to save computational power, we chose the following three out of the nine GLUE Benchmark tasks: CoLA, MRPC, and RTE.

Next, as the original paper (Ben-Zaken et al., 2021) did, we averaged the scores for each fine-tune type for easy comparison. Our replication results, along with original paper's results (shown in parenthesis), are shown below in Figure 3. As a note, the average score in the table is only averaged over the three tasks of interest. Additionally, the change in bias vector b, defined as $(1/\dim(b)) \cdot \|b_0 - b_F\|_1$ (the average absolute change, across its dimensions, between the initial bias term and the fine-tuned bias terms), for all the bias terms is shown in Figures 4, 5, and 6.

	% Param	CoLA (Matthews correlation)	MRPC (F1)	RTE (Accuracy)	Avg Score
REPLICATIONS					
Full-FT *	100%	56.4	89.0	70.5	72.0
BitFit	0.09%	58.9 (58.8)	90.6 (90.4)	70.4 (72.3)	73.3 (73.8)
b_{m2}, b_q	0.04%	57.0 (57.4)	88.7 (89.0)	66.8 (68.6)	70.8 (71.7)
Frozen	0.0%	29.5 (31.9)	81.4 (81.1)	57.0 (56.9)	56.0 (56.6)
Rand uniform	0.09%	55.7 (54.1)	86.0 (84.3)	63.5 (62.9)	68.4 (67.1)
Rand row/col	0.09%	52.3 (53.4)	88.6 (88.0)	67.2 (65.1)	69.4 (68.1)

Figure 3: Replications - fine-tuning using a subset of the bias parameters. Reported results are for the $\text{BERT}_{\text{BASE}}$ model. * indicates that this result was taken from https://arxiv.org/pdf/2106.10199

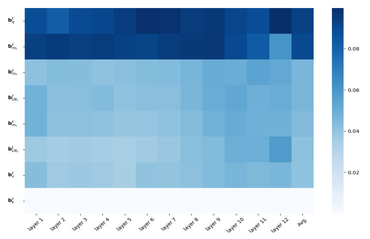


Figure 4: BitFit change in bias for RTE task

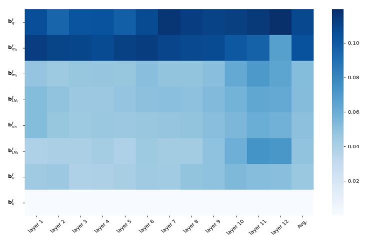


Figure 5: BitFit change in bias for MRPC task

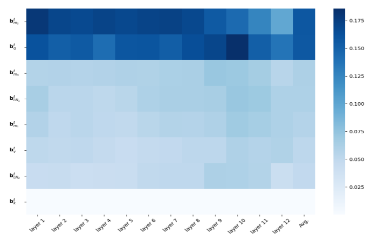


Figure 6: BitFit change in bias for CoLA task

All of these results were very close to the results obtained in the original BitFit paper. Though we were unable to run multiple trials due to computational limitations, we attempted to estimate the variance of our results, running the b_{m2}, b_q fine-tune type for the RTE task three times, and obtaining a variance of 1.04% for accuracy. We will now reflect on the author's original two claims:

1. Fine-tuning only the additive bias terms (BitFit) is effective, and bias parameters are special, since the average score of BitFit is comparable to the average score of Full-FT and the average score of rand uniform and rand row/col is substantially worse than that of BitFit.

   This seems accurate, since BitFit's average score of 73.3 is comparable to Full-FT's average score of 72.0. Additionally, the average scores of rand uniform (68.4) and rand row/col (69.4) are significantly worse than the average score of BitFit (73.3).

2. Fine-tuning only the bias parameters associated with the query and the second MLP layer (b_{m_2} and b_q) is effective.

   This seems accurate, since in the RTE, CoLA, and MRPC tasks (Figures 4, 5, and 6), b_{m_2}, b_q bias terms are the darkest across nearly all layers, indicating that they have the greatest magnitude of change throughout fine-tuning. In addition, b_{m_2}, b_q's average score of 70.8 is comparable to BitFit's score of 73.3.

4.2 Ablations

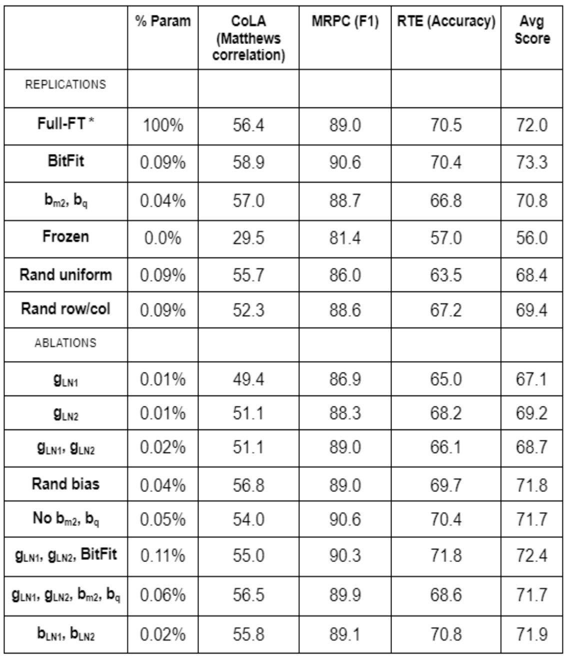
To explore the author's two claims further, we will perform the following ablations:

1. The author claimed that the bias parameters are special, since BitFit achieved a higher average score than both rand uniform and rand row/col, which are subsets of all the parameters. However, this result does not eliminate the possibility that there are other "special" parameters, since a large majority of unimportant parameters would bring the rand uniform and rand row/col average scores down.

Thus, we tested the model only using g_{LN_1} and g_{LN_2} weights (separate and combined), which were only 0.01% of all parameters each. For parallel comparison (same number of parameters and same type of layer), we also ran b_{LN_1} and b_{LN_2}. In addition, we combined g_{LN_1} and g_{LN_2} with both BitFit and b_{m_2}, b_q.

2. The author claimed that fine-tuning on b_{m_2} and b_q is effective. We want to determine if b_{m_2} and b_q are special parameters. We first fine-tuned the model on all biases besides b_{m_2} and b_q. Next, similar to how the authors sampled rand uniform and rand row/col to determine if biases were special parameters, we sampled a random subset of bias terms (the same amount of bias terms as in b_{m_2}/b_q) to see if b_{m_2} and b_q are indeed special parameters. We also tested on b_{LN_1} and b_{LN_2}, two other weights that did not change nearly as much during fine-tuning compared to b_{m_2} and b_q (Figures 4, 5, 6).

Figure 7 with both the replications and ablations results is below. We have also included the change in bias terms for the RTE task for a variety of fine-tuning methods used in ablations.



	% Param	CoLA (Matthews correlation)	MRPC (F1)	RTE (Accuracy)	Avg Score
REPLICATIONS					
Full-FT*	100%	56.4	89.0	70.5	72.0
BitFit	0.09%	58.9	90.6	70.4	73.3
b_{m2}, b_q	0.04%	57.0	88.7	66.8	70.8
Frozen	0.0%	29.5	81.4	57.0	56.0
Rand uniform	0.09%	55.7	86.0	63.5	68.4
Rand row/col	0.09%	52.3	88.6	67.2	69.4
ABLATIONS					
g_{LN1}	0.01%	49.4	86.9	65.0	67.1
g_{LN2}	0.01%	51.1	88.3	68.2	69.2
g_{LN1}, g_{LN2}	0.02%	51.1	89.0	66.1	68.7
Rand bias	0.04%	56.8	89.0	69.7	71.8
No b_{m2}, b_q	0.05%	54.0	90.6	70.4	71.7
g_{LN1}, g_{LN2}, BitFit	0.11%	55.0	90.3	71.8	72.4
g_{LN1}, g_{LN2}, b_{m2}, b_q	0.06%	56.5	89.9	68.6	71.7
b_{LN1}, b_{LN2}	0.02%	55.8	89.1	70.8	71.9

Figure 7: Fine-tuning using a subset of the bias and gamma parameters. Reported results are for the $\text{BERT}_{\text{BASE}}$ model. * indicates that this result was taken from https://arxiv.org/pdf/2106.10199.pdf.

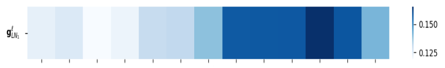


Figure 8: g_{LN_1} change in bias for RTE task

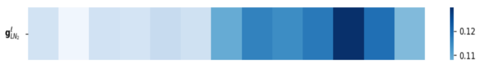


Figure 9: g_{LN_2} change in bias for RTE task

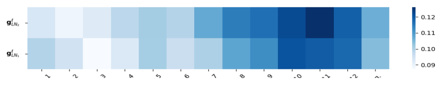


Figure 10: g_{LN_1}, g_{LN_2} change in bias for RTE task

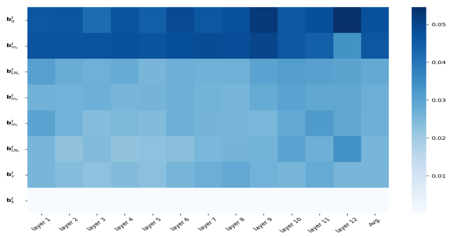


Figure 11: randbias change in bias for RTE task

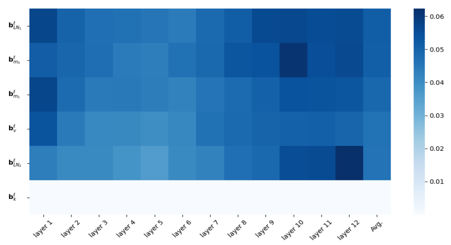


Figure 12: no b_{m_2}, b_q change in bias for RTE task

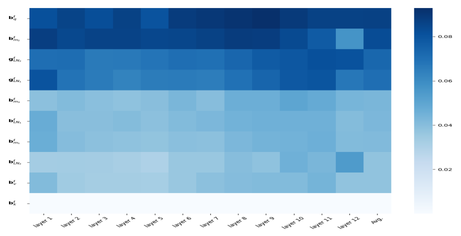


Figure 13: g_{LN_1}, g_{LN_2}, BitFit change in bias for RTE task

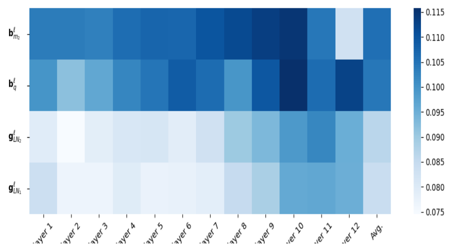


Figure 14: $g_{LN_1}, g_{LN_2}, b_{m_2}, b_q$ change in bias for RTE task

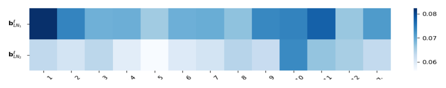


Figure 15: b_{LN_1}, b_{LN_2} change in bias for RTE task

Now let us answer whether bias parameters are special weights and whether b_{m_2}, b_q are special bias terms.

1. Are bias parameters special weights?

   (a) After running the tests mentioned in the first ablation point, we found that the average score of g_{LN_1} (67.1) was lower than the average scores of g_{LN_2} (69.2) and g_{LN_1}, g_{LN_2} (68.7), which had similar scores. Thus, it seems as though only the second gamma layer normalization weight is needed, and the first one is not as effective for fine-tuning. However, even taking the best average score (69.2), the gamma terms do not compare to the average BitFit score (73.3). One could argue that this may be due to there being more bias than gamma terms (0.09% vs. 0.02%). So, to control for this, we also ran the same layer type in the bias terms as the gamma terms (layer normalization), which had the same number of parameters. Our average score for b_{m1}, b_{m_2} was 71.9%, significantly greater than 69.2%. From this parallel comparison, it does seem as if bias parameters are special.

   (b) Next, we combined the g and b terms. Taking the perspective of adding the bias terms (BitFit or b_{m_2}, b_q) to g_{LN_1}, g_{LN_2}, we see that adding BitFit increased the average score by 3.7 (68.7 to 72.4) and adding b_{m_2}, b_q increased the average score by 3.0 (from 68.7 to 71.7). In both cases, adding bias terms increases the average score significantly. Now taking the perspective of adding the gamma terms (g_{LN_1}, g_{LN_2}) to BitFit decreased the average score by 0.9 (73.3 to 72.4) and adding the gamma terms to b_{m_2}, b_q increased the average score by 0.9 (from 70.8 to 71.7). In both cases, adding gamma terms has only a marginal effect on the bias terms' scores. Thus, since adding biases increases the gamma parameters' average score but the opposite is not true, it can be reasonably stated that bias terms are special.

2. Are b_{m_2} and b_q special bias terms? After running the tests mentioned in the second ablation point, we found that fine-tuning on all biases besides b_{m_2} and b_q yielded a higher average score (71.7) than fine-tuning on b_{m_2} and b_q (70.8). Assuming b_{m_2}, b_q biases are special, a couple of explanations for this are as follows:

   (a) There are slightly more parameters in the no b_{m_2}, b_q fine-tuning (0.05% vs. 0.04%), causing the average b_{m_2}, b_q score to be less favorable.

   (b) The variety of bias terms is essential; thus, although b_{m_2}, b_q performs worse than no b_{m_2}, b_q due to its lack of variety, b_{m_2}, b_q would perform better than any other two bias parameters.

We can test both of these hypotheses by fine-tuning on b_{LN_1}, b_{LN_2}. Its average score was 71.9, higher than the 70.8 score for b_{m_2}, b_q. This is directly contradictory to hypothesis (a) because b_{LN_1}, b_{LN_2} only make up 0.02% of parameters, directly contradictory to hypothesis (b) because the average score for b_{LN_1}, b_{LN_2} is better. Thus, we can reasonably claim that b_{m_2} and b_q parameters are likely not special bias parameters.

Our results for randbias also support this hypothesis. We randomly sampled the same number of parameters from the biases as there were in the b_{m_2}, b_q (0.04%), and we achieved an average score of 71.8, a 1.0% increase from b_{m_2}, b_q. Since our random selection of biases achieved a higher accuracy, it is again reasonable to claim that b_{m_2} and b_q are likely not special bias parameters.

Another interesting question is whether or not the magnitude of the change in bias during fine-tuning is correlated with the performance of that bias during fine-tuning. This was implied by the authors of BitFit, when they only mentioned testing the subset of bias terms b_{m_2} and b_q because they were the bias parameters that changed the most during fine-tuning. Let us consider the following comparisons:

1. b_{m_2}, b_q vs. no b_{m_2}, b_q: b_{m_2}, b_q weights change much more than the other biases (Figure 4), yet the other biases perform better (71.7 vs. 70.8).

2. g_{LN_1} vs. g_{LN_2}: very similar changes in bias (Figure 10), yet g_{LN_2} has a better average score than g_{LN_1} (69.2 vs. 67.1)

3. g_{LN_1}, g_{LN_2} vs. b_{m_2}, b_q: b_{m_2}, b_q change more (Figure 13), and b_{m_2}, b_q's average score is higher (70.8 vs. 68.7).

4. b_{m_2}, b_q vs. b_{LN_1}, b_{LN_2}: b_{m_2}, b_q change more (Figure 4), yet b_{LN_1}, b_{LN_2} has a higher average score (71.9 vs. 70.8).

Of course, the bias parameters need to change at least somewhat in order to be better than the frozen model, but from these results, it seems as though there is no clear correlation between the change in bias and the performance. In fact, if we assume that bias parameters are inherently more effective for fine-tuning than gamma parameters, we could even investigate if there is a reverse correlation, for bias changes in certain ranges.

In summary, through our analysis, we have concluded that bias terms are special, b_{m_2} and b_q terms are not special, and there is no positive correlation between the change in magnitude of the bias term and its performance.

5 Conclusion

The significance of fine-tuning LLMs cannot be overstated. Limitations on computational resources makes efficiently fine-tuning pre-trained models an important challenge. (Ben-Zaken et al., 2021) tackle this challenge with their partial fine-tuning approach BitFit. Our replication of their results agrees that BitFit is seemingly a strong competitor for computational feasible methods of fine-tuning, scoring the best across all tasks. However, though their b_{m_2} and b_q fine-tuning scores very close to that of BitFit, we find an interesting outcome when fine-tuning on random bias terms or on all bias terms other than b_{m_2} and b_q. These two fine-tune types generally scored higher than just fine-tuning on b_{m_2} and b_q, as did combinations of BitFit with gamma parameters and fine-tuning on just b_{LN_1} and b_{LN_2} parameters. Why then do the b_{m_2} and b_q bias parameters change the most during fine-tuning? One potential explanation is that these parameters are overfitting the model, capturing specific nuances in the data that are not helpful for generalization.

Further, we used gamma parameters to test whether bias parameters are special for fine-tuning compared to other types of parameters. The BitFit paper (Ben-Zaken et al., 2021) did not fine-tune with the gamma parameters, focusing only on bias terms. While fine-tuning with gamma parameters, we found a decrease in scores. Our combination of gamma parameters with BitFit did not result in increased scores, instead performing worse than BitFit alone. We controlled for bias terms being a larger percentage of model parameters and being present throughout the model by fine-tuning with just b_{LN_1} and b_{LN_2}. Fine-tuning with these parameters resulted in higher scores than fine-tuning with gamma parameters, leading us to believe that bias terms are special. One potential explanation is that bias parameters help the model adapt to different tasks better with their simple shift in activation functions, allowing the model to adjust its predictions without overfitting to new data.

Our paper tests several ways of fine-tuning in the hopes of creating more efficient, computationally feasible methods of fine-tuning LLMs. As LLMs become larger with billions of parameters trained on bigger datasets than ever, cheap methods of fine-tuning will become more important than ever.

Acknowledgements

We would like to thank professor Karthik Narasimhan for his amazing lectures and our project mentor Yihan Wang for his guidance.

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *CoRR*, abs/2012.07463.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.