BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models

Anonymous ACL submission

Abstract

We show that with small-to-medium training data, fine-tuning only the bias terms (or a subset of them) of pre-trained BERT models is competitive with (and sometimes better than) fine-tuning the entire model. For larger data, bias-only fine-tuning is competitive with other sparse fine-tuning methods. Besides their practical utility, these findings are relevant for the question of understanding the commonly-used process of finetuning: they support the hypothesis that finetuning is mainly about exposing knowledge induced by language-modeling training, rather than learning new task-specific linguistic knowledge.

1 Introduction

Large pre-trained transformer based language models, and in particular bidirectional masked language models from the BERT family (Devlin et al., 2018; Liu et al., 2019; Joshi et al., 2019), are responsible for significant gains in many NLP tasks. Under the common paradigm, the model is pre-trained on large, annotated corpora with the LM objective, and then *finetuned* on task-specific supervised data. The large size of these models make them expensive to train and, more importantly, expensive to deploy. This, along with theoretical questions on the extent to which finetuning must change the original model, has led researchers to consider fine-tuning variants where one identifies a small subset of the model parameters which need to be changed for good performance in end-tasks, while keeping all others intact (§2).

We present a simple and effective approach to fine tuning (§3), which has the following benefits:

1. Changing very few parameters per fine-tuned task.
2. Changing the same set of parameters for every tasks (task-invariance).
3. The changed parameters are both isolated and localized across the entire parameter space.
4. For small to medium training data, changing only these parameters reaches the same task accuracy as full fine-tuning, and sometimes even improves results.

Specifically, we show that freezing most of the network and **fine-tuning only the bias-terms** is surprisingly effective. Moreover, if we allow the tasks to suffer a small degradation in performance, we can fine-tune only two bias components (the "query" and "middle-of-MLP" bias terms), amounting to half of the bias parameters in the model, and only 0.04% of all model parameters.

This result has a large practical utility in deploying multi-task fine-tuned models in memory-constrained environments, as well as opens the way to trainable hardware implementations in which most of the parameters are fixed.

2 Background: fine-tuning and parameter-efficient fine-tuning

In transfer-learning via model fine-tuning, a pre-trained encoder network takes the input and produces contextualized representations. Then, a task-specific classification layer (here we consider linear classifiers) is added on top of the encoder, and the entire network (encoder+task specific classifiers) is trained end-to-end to minimize the task loss.

Desired properties. While fine-tuning per-task is very effective, it also results in a unique, large model for each pre-trained task, making it hard to reason about as well as hard to deploy, especially as the number of tasks increases. Ideally, one would want a fine-tuning method that:
(i) matches the results of a fully fine-tuned model; (ii) changes only a small portion of the model's parameters; and (iii) enables tasks to arrive in a stream, instead of requiring simultaneous access to all datasets. For efficient hardware based deployments, it is further preferred that (iv): the set of parameters that change values is consistent across

different tasks.

Learning vs. Exposing. The feasibility of fulfilling the above requirements depends on a fundamental question regarding the nature of the fine-tuning process of large pre-trained LMs: to what extent does the fine-tuning process induces the *learning of new capabilities*, vs. the *exposing of existing capabilities*, which were learned during the pre-training process.

Existing approaches. Two recent works have demonstrated that adaptation to various end-tasks can in fact be achieved by changing only a small subset of parameters. The first work, by Houlsby et al. (2019) ("Adapters"), achieves this goal by injecting small, trainable task-specific "adapter" modules between the layers of the pre-trained model, where the original parameters are shared between tasks. The second work, by Guo et al. (2020) ("Diff-Pruning"), achieves the same goal by adding a sparse, task-specific difference-vector to the original parameters, which remain fixed and are shared between tasks. The difference-vector is regularized to be sparse. Both methods allow adding only a small number of trainable parameters per-task (criteria ii), and each task can be added without revisiting previous ones (criteria iii). They also partially fulfill criteria (i), suffering only a small drop in performance compared to full fine-tuning. The Adapter method, but not the Diff-Pruning method, also supports criteria (iv). However, Diff-Pruning is more parameter efficient than the Adapter method (in particular, it adds no new parameters), and also achieves better task scores. We compare against Diff-Pruning and Adapters in the experiments section, and show that we perform favorably on many tasks while also satisfying criteria (iv).

3 Bias-terms Fine-tuning (BitFit)

We propose a method we call BitFit (BIas-Term FIne-Tuning), in which we freeze most of the transformer-encoder parameters, and train only the bias-terms and the task-specific classification layer.

The approach is parameter-efficient: each new task requires storing only the bias terms parameter vectors (which amount to less than 0.1% of the total number of parameters), and the task-specific final linear classifier layer.

Concretely, the BERT encoder is composed of L layers, where each layer ℓ starts with M self-attention heads, where a self attention head (m, ℓ) has *key*, *query* and *value* encoders, each taking the form of a linear layer:

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

Where $\mathbf{x}$ is the output of the former encoder layer (for the first encoder layer $\mathbf{x}$ is the output of the embedding layer). These are then combined using an attention mechanism that does not involve new parameters:

$$\mathbf{h}_1^\ell = att\big(\mathbf{Q}^{1,\ell}, \mathbf{K}^{1,\ell}, \mathbf{V}^{1,\ell}, \ldots, \mathbf{Q}^{m,\ell}, \mathbf{K}^{m,\ell}, \mathbf{V}^{m,l}\big)$$

and then fed to an MLP with layer-norm (LN):

$$\mathbf{h}_2^\ell = \text{Dropout}\big(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell \;+\; \mathbf{b}_{m_1}^\ell\big) \quad (1)$$
$$\mathbf{h}_3^\ell = \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell \quad (2)$$
$$\mathbf{h}_4^\ell = \text{GELU}\big(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell \;+\; \mathbf{b}_{m_2}^\ell\big) \quad (3)$$
$$\mathbf{h}_5^\ell = \text{Dropout}\big(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell \;+\; \mathbf{b}_{m_3}^\ell\big) \quad (4)$$
$$\text{out}^\ell = \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell \quad (5)$$

The collection of all matrices $\mathbf{W}_{(\cdot)}^{\ell,(\cdot)}$ and vectors $\mathbf{g}_{(\cdot)}^\ell$, $\mathbf{b}_{(\cdot)}^{\ell,(\cdot)}$, indicated in blue and purple are the network's *parameters* Θ, where the subset of purple vectors $\mathbf{b}_{(\cdot)}^{\ell,(\cdot)}$ are the *bias terms*.[1]

The bias terms are additive, and correspond to a very small fraction of the network, in $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ bias parameters make up 0.09% and 0.08% of the total number of parameters in each model, respectively.

We show that by freezing all the parameters $\mathbf{W}^{(\cdot)}$ and $\mathbf{g}^{(\cdot)}$ and fine-tuning only the additive bias terms $\mathbf{b}^{(\cdot)}$, we achieve transfer learning performance which is comparable (and sometimes better!) than fine-tuning of the entire network,

We also show that we can fine-tune only a subset of the bias parameters, namely those associated with the *query* and the *second MLP layer* (only $\mathbf{b}_q^{(\cdot)}$ and $\mathbf{b}_{m_2}^{(\cdot)}$), and still achieve accuracies that rival full-model fine-tuning.

4 Experiments and Results

Datasets. We evaluate BitFit on the GLUE benchmark (Wang et al., 2018).[2] Consistent with previous work (Houlsby et al., 2019; Guo et al., 2020)

[1] In Appendix §A.1 we relate this notation with parameter names in HuggingFace implementation.

[2] Appendix §A.3 lists the tasks and evaluation metrics.

		%Param	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
	Train size		105k	67k	393k	393k	8.5k	3.7k	7k	2.5k	364k	
(V)	Full-FT†	100%	**93.5**	**94.1**	**86.5**	**87.1**	**62.8**	**91.9**	89.8	71.8	**87.6**	**84.8**
(V)	Full-FT	100%	91.7±0.1	93.4±0.2	85.5±0.4	85.7±0.4	62.2±1.2	90.7±0.3	**90.0±0.4**	**71.9±1.3**	87.5±0.4	84.1
(V)	Diff-Prune†	0.5%	**93.4**	**94.2**	**86.4**	**86.9**	63.5	91.3	89.5	71.5	**86.6**	**84.6**
(V)	BitFit	0.08%	91.4±2.4	93.2±0.4	84.4±0.2	84.8±0.1	**63.6±0.7**	**91.7±0.5**	**90.3±0.1**	**73.2±3.7**	85.4±0.1	84.2
(T)	Full-FT‡	100%	91.1	94.1	86.7	**86.0**	59.6	88.9	86.6	**71.2**	71.7	81.2
(T)	Full-FT†	100%	**93.4**	**94.9**	86.7	85.9	**60.5**	**89.3**	**87.6**	70.1	**72.1**	**81.8**
(T)	Adapters‡	3.6%	90.7	94.0	84.9	85.1	59.5	89.5	**86.9**	71.5	**71.8**	81.1
(T)	Diff-Prune†	0.5%	**93.3**	94.1	**86.4**	**86.0**	**61.1**	**89.7**	86.0	70.6	71.1	**81.5**
(T)	BitFit	0.08%	92.0	**94.2**	84.5	84.8	59.7	88.9	85.5	**72.0**	70.5	80.9

Table 1: BERT_LARGE model performance on the GLUE benchmark validation set (V) and test set (T). Lines with † and ‡ indicate results taken from Guo et al. (2020) and Houlsby et al. (2019) (respectively).

we exclude the WNLI task, on which BERT models do not outperform the majority baseline.

Models and Optimization. We use the publicly available pre-trained BERT_BASE, BERT_LARGE (Devlin et al., 2018) and $\text{RoBERTa}_\text{BASE}$ (Liu et al., 2019) models, using the HuggingFace (Wolf et al., 2020) interface and implementation. Appendix §A.2 lists optimization details.

Comparison to Diff-Pruning and Adapters (Table 1) In the first experiment, we compare BitFit to Diff-Pruning method and Adapters method, when using a fewer number of parameters. Table 1 reports the dev-set and test-set performance compared to the Diff-Pruning and Adapters numbers reported by Guo et al. (2020) and Houlsby et al. (2019) (respectively). This experiment used the BERT_LARGE model.

On validation set, BitFit outperforms Diff-Pruning on 4 out of 9 tasks, while using 6x fewer trainable parameters [3]. As for test-set results, two clear wins compared to Diff-Pruning and 4 clear wins compared to Adapters while using 45x fewer trainable parameters.

Different Base-models (Table 2) We repeat the BERT_LARGE results on different base-models (the smaller BERT_BASE and the better performing $\text{RoBERTa}_\text{BASE}$). The results in Table 2 show that the trends remain consistent.

Are bias parameters special? Are the bias parameters special, or will any random subset do? We randomly sampled the same amount of parameters as in BitFit from the entire model, and fine-tuned only them ("rand uniform" line in Table 3). The results are substantially worse across all tasks; similar patterns are observed when the random parameters are sampled as complete rows/columns in the parameter matrices ("rand row/col" line in Table 3).

[3] QNLI results are not directly comparable, as the GLUE benchmark updated the test set since then.

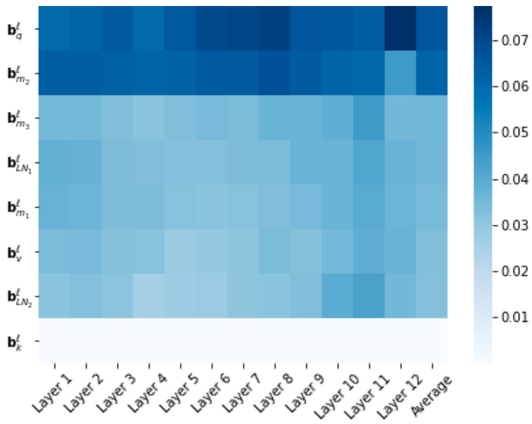

Figure 1: Change in bias components (RTE task).

Fewer bias parameters (Table 3) Can we fine-tune on only a subset of the bias-parameter?

We define the amount of change in a bias vector $\mathbf{b}$ to be $\frac{1}{\mathbf{dim(b)}}\|\mathbf{b}_0 - \mathbf{b}_F\|_1$, that is, the average absolute change, across its dimensions, between the initial LM values $\mathbf{b}_0$ and its fine-tuned values $\mathbf{b}_F$. Figure 1 shows the change per bias term and layer, for the RTE task (other tasks look very similar, see Appendix §A.4). The 'key' bias $\mathbf{b}_k$ has zero change, consistent with the theoretical observation in Cordonnier et al. (2020). In contrast, $\mathbf{b}_q$, the bias of the queries, and $\mathbf{b}_{m2}$, the bias of the intermediate MLP layers (which take the input from 768-dims to 3072), change the most. Table 3 reports dev-set results when fine-tuning only the $\mathbf{b}_q^{(\cdot)}$ and $\mathbf{b}_{m2}^{(\cdot)}$ bias terms, for the BERT_BASE model. Results are only marginally lower than when tuning all bias parameters. Tuning either $\mathbf{b}_q^{(\cdot)}$ or $\mathbf{b}_{m2}^{(\cdot)}$ alone yields substantially worse results, indicating both bias types are essential. As expected, using a frozen BERT_BASE model yields much worse results.

Generalization gap. We find that the generalization gap (Shalev-Shwartz and Ben-David, 2014)—

	Method	%Param	QNLI	SST-2	MNLI$_m$	MNLI$_{mm}$	CoLA	MRPC	STS-B	RTE	QQP	Avg.
BB	Full-FT	100%	**90.7±0.2**	92.0±0.4	**83.5±0.1**	**83.7±0.3**	56.4±0.9	89.0±1.0	88.9±0.7	70.5±0.6	**87.1±0.1**	82.3
BB	BitFit	0.09%	90.2±0.2	**92.1±0.3**	81.4±0.2	82.2±0.2	**58.8±0.5**	**90.4±0.5**	**89.2±0.2**	**72.3±0.9**	84.0±0.2	**82.4**
BL	Full-FT	100%	**91.7±0.1**	**93.4±0.2**	**85.5±0.4**	**85.7±0.4**	62.2±1.2	90.7±0.3	90.0±0.4	71.9±1.3	**87.5±0.4**	84.1
BL	BitFit	0.08%	91.4±2.4	93.2±0.4	84.4±0.2	84.8±0.1	**63.6±0.7**	**91.7±0.5**	**90.3±0.1**	**73.2±3.7**	85.4±0.1	**84.2**
Ro	Full-FT	100%	**92.3±0.2**	**94.2±0.4**	**86.4±0.3**	**86.9±0.3**	61.1±0.8	**92.5±0.4**	90.6±0.2	77.4±1.0	**88.0±0.2**	**85.3**
Ro	BitFit	0.09%	91.3±0.2	93.7±0.1	84.8±0.1	85.2±0.2	**61.8±1.3**	92.0±0.4	**90.8±0.3**	**77.8±1.7**	84.5±0.2	84.6

Table 2: Dev-set results for different base models. **BB**: BERT$_{\text{BASE}}$. **BL**: BERT$_{\text{LARGE}}$. **Ro**: RoBERTa$_{\text{BASE}}$.

	% Param	QNLI	SST-2	MNLI$_m$	MNLI$_{mm}$	CoLA	MRPC	STS-B	RTE	QQP	Avg.
Full-FT	100%	**90.7±0.2**	92.0±0.4	**83.5±0.1**	**83.7±0.3**	56.4±0.9	89.0±1.0	88.9±0.7	70.5±0.6	**87.1±0.1**	82.3
BitFit	0.09%	90.2±0.2	**92.1±0.3**	81.4±0.2	82.2±0.2	**58.8±0.5**	**90.4±0.5**	**89.2±0.2**	**72.3±0.9**	84.0±0.2	**82.4**
$\mathbf{b}_{m2}, \mathbf{b}_q$	0.04%	89.4±0.1	91.2±0.2	80.4±0.2	81.5±0.2	57.4±0.8	89.0±0.2	88.4±0.1	68.6±0.6	83.7±0.2	81.1
$\mathbf{b}_{m2}$	0.03%	88.9±0.1	91.1±0.3	79.9±0.3	80.7±0.2	54.9±0.9	87.9±0.6	88.2±0.1	66.8±0.6	82.1±0.4	80.0
$\mathbf{b}_q$	0.01%	86.8±0.1	89.6±0.2	74.4±0.3	75.7±0.2	49.1±1.5	84.4±0.2	85.6±0.1	61.4±1.1	80.6±0.4	76.6
Frozen	0.0%	68.7±0.3	81.7±0.1	42.4±0.1	43.8±0.1	31.9±1.1	81.1±0.1	71.4±0.1	56.9±0.4	62.4±0.2	62.1
rand uniform	0.09%	87.8±0.3	90.5±0.3	78.3±0.3	78.8±0.2	54.1±1.0	84.3±0.3	87.2±0.4	62.9±0.9	82.4±0.3	78.5
rand row/col	0.09%	88.4±0.2	91.0±0.3	79.4±0.3	80.1±0.3	53.4±0.6	88.0±0.7	87.9±0.2	65.1±0.7	82.3±0.2	79.5

Table 3: Fine-tuning using a subset of the bias parameters. Reported results are for the BERT$_{\text{BASE}}$ model.

the difference between training error and test error—is substantially smaller for the BitFit models: while for full fine-tuning the train set accuracy reaches nearly 100%, in the bias-only fine-tuned models the difference between the train and test set performance is often less than 2%.

Token-level tasks. The GLUE tasks are all sentence level. We also experimented with token-level PTB POS-tagging. Full-FT results for BERT$_{\text{BASE}}$, BERT$_{\text{LARGE}}$ and RoBERTa$_{\text{BASE}}$ are 97.2, 97.4, 97.2, while BitFit results are 97.2, 97.4, 97.1.

Size of training data. The GLUE results suggest a reverse correlation between BitFit ability to reach Full-FT performance, and training set size. To test this (and to validate another token-level task), we train on increasing-sized subsets of SQuAD v1.0 Rajpurkar et al. (2016a). The results on Figure 2 show a clear trend: BitFit dominates over Full-FT in the smaller-data regime, while the trend is reversed when more training data is available. We conclude that BitFit is a worthwhile targetted fine-tuning method in small-to-medium data regimes.

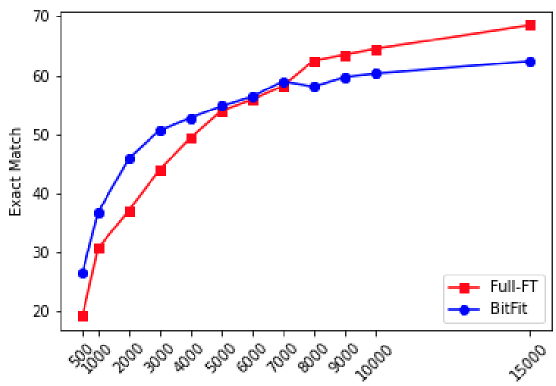

Figure 2: Comparison of BitFit and Full-FT with BERT$_{\text{BASE}}$ exact match score on SQuAD validation set.

5 Related Work

Bias terms and their importance are rarely discussed in the literature.[4] Zhao et al. (2020) describe a masking-based fine-tuning method, and explicitly mention *ignoring* the bias terms, as handling them "did not observe a positive effect on performance".

An exception is the work of Wang et al. (2019) who analyzed bias terms from the perspective of attribution method. They demonstrate that the last layer bias values are responsible for the predicted class, and propose a way to back-propagate their importance. Michel and Neubig (2018) finetuned the biases of the output softmax in an NMT systems, to personalize the output vocabulary. Finally, Cai et al. (2020) demonstrate that bias-only fine-tuning similar to ours is effective also for adaptation of pre-trained computer vision models.

6 Discussion

Besides its empirical utility, the remarkable effectiveness of bias-only fine-tuning raises intriguing questions on the fine-tuning dynamics of pre-trained transformers, and the relation between the bias terms and transfer between LM and new tasks. We aim to study those questions in a future work.

[4] Indeed, the equations in the paper introducing the Transformer model (Vaswani et al., 2017) do not include bias terms at all, and their existence in the BERT models might as well be a fortunate mistake.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. 2020. Tiny transfer learning: Towards memory-efficient on-device learning. *CoRR*, abs/2007.11622.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. Multi-head attention: Collaborate instead of concatenate. *CoRR*, abs/2006.16362.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

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.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning.

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.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 312–318. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

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.

Shengjie Wang, Tianyi Zhou, and Jeff A. Bilmes. 2019. Bias also matters: Bias attribution for deep neural network explanation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6659–6667. PMLR.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

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. 2020. Transformers: State-of-the-art natural language processing. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

A Appendices

A.1 Layer naming

For convenience, we relate the notation used in the paper with the names of the corresponding parameters in the popular HuggingFace (Wolf et al., 2020) implementation.

HuggingFace Parameter Name	BitFit notation
attention.self.query.bias	$\mathbf{b}_q$
attention.self.key.bias	$\mathbf{b}_k$
attention.self.value.bias	$\mathbf{b}_v$
attention.output.dense.bias	$\mathbf{b}_{m_1}$
attention.output.LayerNorm.bias	$\mathbf{b}_{LN_1}$
intermediate.dense.bias	$\mathbf{b}_{m_2}$
output.dense.bias	$\mathbf{b}_{m_3}$
output.LayerNorm.bias	$\mathbf{b}_{LN_2}$

Table 4: Mapping the HuggingFace's BertLayer bias parameters names to BitFit paper bias notation.

A.2 Training Details

To perform classification with BERT, we follow the approach of Devlin et al. (2018), and attach a linear layer to the contextual embedding of the `[CLS]` token to predict the label. The GLUE tasks are fed into BERT using the standard procedures.
We optimize using AdamW (Loshchilov and Hutter, 2017), with batch sizes of 16. For full fine-tuning, we used initial learning rates in {1e-5, 2e-5, 3e-5, 5e-5}, and for the bias-only experiments we used initial learning rates in {1e-4, 4e-4, 7e-4, 1e-3} as the smaller rates took a very long time to converge on some of the tasks. With the larger learning rates, the bias-only fine-tuning converged in 8 or fewer epochs for most tasks, and up to 20 epochs on the others. We did not perform hyperparameter optimization beyond the minimal search over 4 learning rates. In each evaluation we report X±Y where X is the average result for training 5 models with 5 different random seeds, Y is the standard deviation.
To perform classification with RoBERTa$_{\text{BASE}}$, we follow the above details but without hyperparameter search over the learning rates, for bias-only fine-tuning we used 1e-4 as learning rate and for full fine-tuning we used 1e-5 as learning rate.
As Mosbach et al. (2020) show, fine-tuning BERT$_{\text{LARGE}}$ and RoBERTa$_{\text{BASE}}$ is a unstable due to vanishing gradients. BitFit allows for the usage of bigger learning rates, and overall the optimization process is much more stable, when compared with a full fine-tuning.

Task Name	Metric
QNLI	acc.
SST-2	acc.
MNLI	matched acc./mismatched acc.
CoLA	Matthews corr.
MRPC	F1
STS-B	Spearman corr.
RTE	acc.
QQP	F1

Table 5: Metrics that we use to evaluate GLUE Benchmark.

Task Name	BERT$_{\text{BASE}}$	BERT$_{\text{LARGE}}$
QNLI	1e-4	7e-4
SST-2	4e-4	4e-4
MNLI	1e-4	1e-4
CoLA	7e-4	4e-4
MRPC	7e-4	1e-3
STS-B	1e-4	1e-4
RTE	1e-3	4e-4
QQP	4e-4	4e-4

Table 6: Learning rate configurations for best performing models.

A.3 GLUE Benchmark

We provide information on the GLUE tasks we evaluated on, as well as on the evaluation metrics. We test our approach on the following subset of the GLUE (Wang et al., 2018) tasks: The Corpus of Linguistic Acceptability (CoLA; Warstadt et al. (2018)), The Stanford Sentiment Treebank (SST-2; Socher et al. (2013)), The Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett (2005)), The Quora Question Pairs (QQP; Iyer et al. (2017)), The Semantic Textual Similarity Benchmark (STS-B; Cer et al. (2017)), The Multi-Genre Natural Language Inference Corpus (MNLI; Bowman et al. (2015)), The Stanford Question Answering Dataset (QNLI; Rajpurkar et al. (2016b)) and The Recognizing Textual Entailment (RTE; Dagan et al. (2005)).

The metrics that we used to evaluate GLUE Benchmark are in Table 5. Learning rate configurations for best performing models are in Table 6. For all the experiments we used the common train:dev:test partition of GLUE.

A.4 Amount of change in bias terms

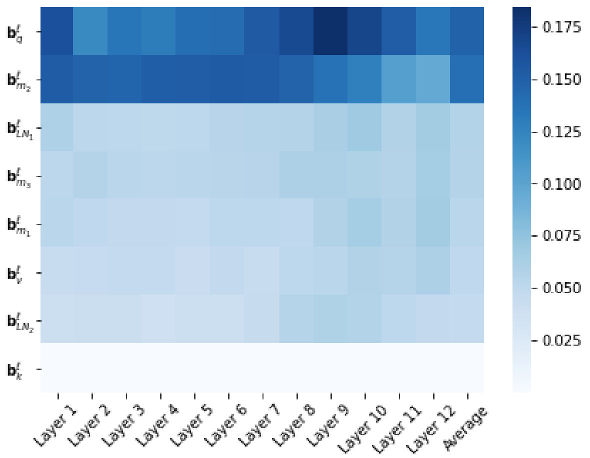

Figure 3: Change in bias components (CoLA task).

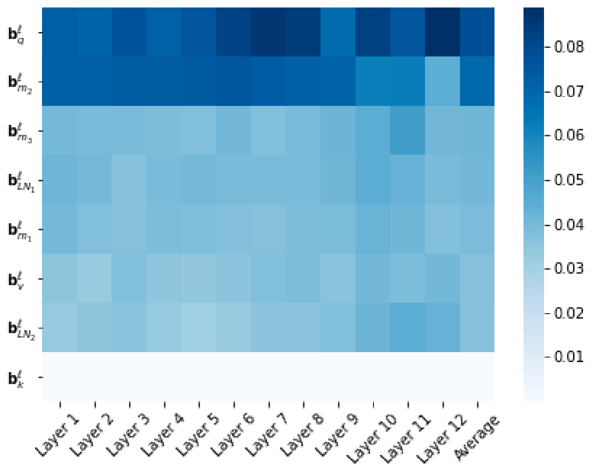

Figure 4: Change in bias components (MRPC task).

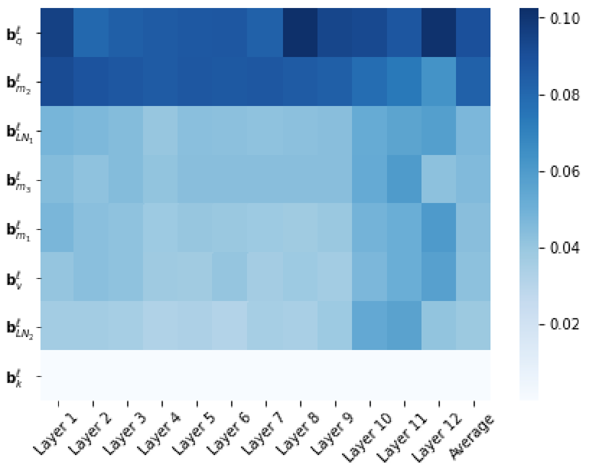

Figure 5: Change in bias components (STS-B task).

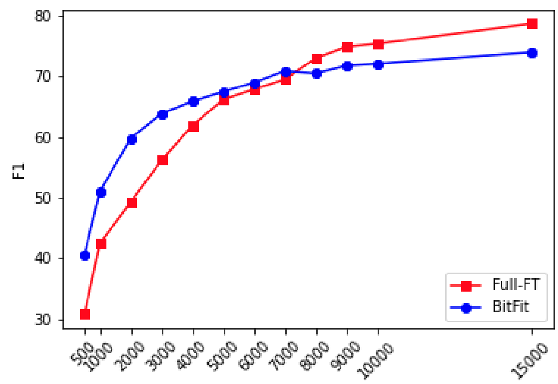

Figure 6: Comparison of BitFit and Full-FT with $\text{BERT}_{\text{BASE}}$ F1 score on SQuAD validation set.

A.5 SQuAD F1 Results