Memory-Efficient Selective Fine-Tuning

Antoine Simoulin [1] **Namyong Park** [1] **Xiaoyi Liu** [1] **Grey Yang** [1]

Abstract

We propose an approach for reducing the memory required to fine-tune transformer-based models. During the backward pass, our approach only propagates the gradient through a small number of input positions, while freezing the others. Thus, we only save a subset of the intermediate activations during the forward pass, for which the computed gradient will not be zero. We show that our approach leads to performance on-par with full fine-tuning, while requiring only up to a third of the GPU memory. Our approach is specifically efficient in fine-tuning language models with a number of parameters lying around hundreds of millions. It allows to fine-tune such models on consumer hardware, while maintaining a large batch size.

Table 1. Using two models requiring roughly the same GPU memory, we observe that the memory breakdown and the impact of PEFT methods application are very different. For each model, we show the evolution of the GPU memory (in MiB) required for performing one training step for OPT-1B3 (Zhang et al., 2022) with a batch size of 1 and a sequence length of 128 and BERT-base (Devlin et al., 2019) with a batch size of 256, a sequence length of 128. Fwd (w/o grad) corresponds to the execution of the forward pass, while disabling gradient computation.

			w/ LoRA	
	BERT	OPT	BERT	OPT
Cuda Context	780	780	780	780
+ Model weights	1,250	**5,806**	1,250	**5,828**
+ Fwd (w/o grad)	2,860	6,050	2,860	6,072
+ Fwd (w/ grad)	**24,754**	6,290	**20,554**	6,294
+ Bwd	25,242	**11,258**	20,948	6,322
+ Optimizer step	25,242	**21,390**	20,948	6,322

1. Introduction

Large Language Models (LLMs) come with the challenge of adapting them for specific tasks. Their high number of parameters increases the compute requirement to fine-tune them. Alternative methods such as zero-shot tuning or prompting usually underperform fine-tuning (Brown et al., 2020). Thus, in many cases, fine-tuning medium size LLMs may offer a better balance in term of cost and performance, compared with fine-tuning large LLMs or conditioning their outputs with prompt based approaches (Li et al., 2022; Schick & Schütze, 2021). Medium size LLMs may also be used as individual components, co-trained to encode information for a larger system (Pfeiffer et al., 2023).

Parameter-Efficient Fine-Tuning (PEFT) approaches aim at reducing the compute and storage requirements to fine-tune LLMs by only updating a small subset of the model parameters. As a result, we do not need to store any corresponding gradients and optimizer states for the frozen parameters. With parameters, gradients, and optimizer states usually representing the majority of the GPU memory usage, this is specially efficient for very large language models with billions of parameters. However, for language models with "only" hundred of millions of parameters, most of the GPU memory is actually used to store intermediate activations required for gradient computation during the backward pass.

Table 1 presents the GPU memory required to perform one training step with BERT-base (Devlin et al., 2019) and OPT (Zhang et al., 2022) on a consumer hardware GPU. We calibrate the example such that the memory requirement is roughly the same for both models. In this configuration we can only fit a single example for OPT, while we can use a batch size of 256 for BERT. We observe that the memory breakdown is very different between the two configurations. The required memory drastically increases during the forward pass for BERT and during the backward pass for OPT. When comparing the execution of forward pass with and without enabling gradient computation in pytorch, we estimate that the memory cost to store intermediate activations represents around 22 Gb for BERT and less than 1 Gb for OPT. On the contrary, we estimate that computing and storing the parameter gradients increase the memory requirement by less than 1 Gb for BERT and around 5 Gb for OPT. When applying LoRA (Hu et al., 2022), a PEFT method, we observe that the memory drastically decreases for OPT, while having a less significant impact on BERT.

[1]Meta. Correspondence to: Antoine Simoulin <antoinesimoulin@meta.com>.

Work presented at the ES-FoMo Workshop at ICML 2023, Honolulu, Hawaii, USA.

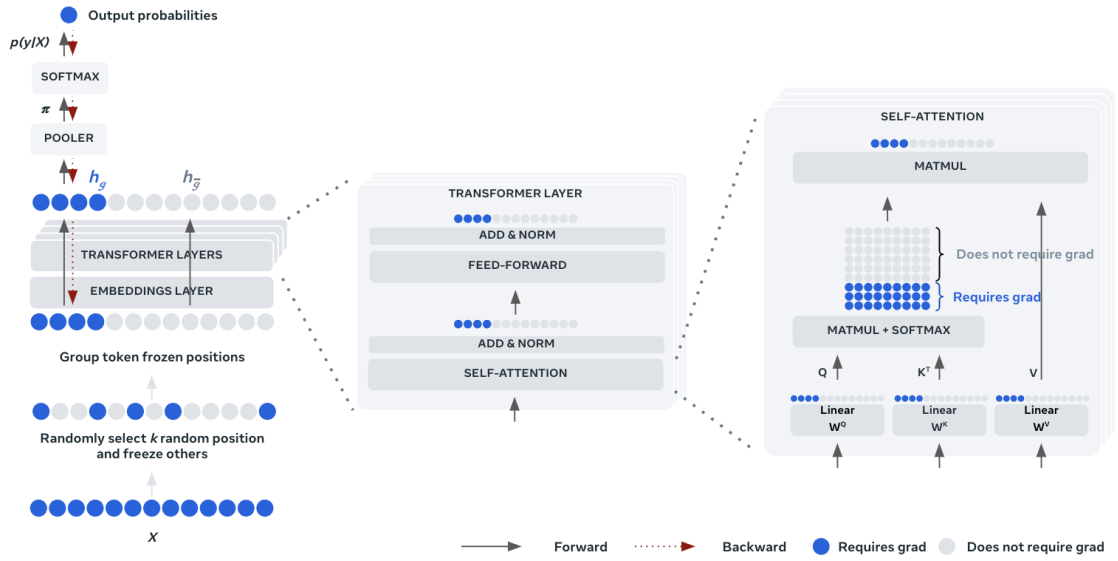

Figure 1. We illustrate our approach for memory efficient fine-tuning of LLMs. During the backward pass, we compute the gradient for only a subset of k input positions, while the other remains frozen (in grey in the figure). During the forward pass, all input positions are used, but only a subset of the layer activations is cached in memory (in blue in the figure).

In this work, we propose a method that focuses on optimizing the training of LLMs with millions of parameters, with the aim at significantly reducing the GPU memory dedicated to storing intermediate activations during the forward pass without sacrificing model quality. Our paper is segmented into the method (Section 2), the empirical evaluation results (Section 3) and related work (Section 4).

2. Selective Fine-Tuning

Figure 1 illustrates our selective fine-tuning (SFT) method, aiming at reducing the memory needed to store the intermediate activations used for gradient computation. Given an input sequence X and a class label y, a transformer network associates each token from the input sequence to an embedding and computes a corresponding sequence of hidden states h through multiple layer applications.

For each input sequence, we select k random positions.[1] We organize the input of each layer in two groups, one with the k selected input positions: $h_{\mathcal{G}}$, and one with the remaining unselected positions: $h_{\bar{\mathcal{G}}}$, such that that $h = [h_{\mathcal{G}}, h_{\bar{\mathcal{G}}}]$, with $[\ \ ]$ denoting the concatenation operator. This re-ordering does not impact the computation since the position is directly encoded in the hidden representations.

We use the average of the hidden states from the selected positions of the last layers as input for a MLP. The MLP outputs a probability distribution over the classes of the task given Eq 1. During the evaluation, we use the average from all hidden states of the last layer as input for the MLP.

$$\begin{aligned} \pi &= \text{MLP}\left(\frac{1}{k}\sum_{i=1}^{k} h_{\mathcal{G}}\right) \\ p(y|X) &= \text{softmax}(\pi) \end{aligned} \tag{1}$$

The key element of our method is that we disable the gradient computation for the un-selected tokens in $\bar{\mathcal{G}}$. Thus, only the k selected position contributions will be used for the gradient computation during the backward pass. For example, let us consider a dense layer $f(x) = \sigma(Wx + b)$ with a weight W, bias b, and nonlinear function σ as parameters. We note the output of the layer a, the input h and the pre-activation z such that $a = \sigma(z) = \sigma(Wh + b)$. When back-propagating through this layer, given a loss $\mathcal{L}$, we compute the gradient with respect to W and b given Eq 2.

$$\begin{aligned} \frac{\partial \mathcal{L}}{dW} &= \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial W} = \frac{\partial \mathcal{L}}{\partial a}\sigma' h \\ \frac{\partial \mathcal{L}}{db} &= \frac{\partial \mathcal{L}}{\partial a}\sigma' \end{aligned} \tag{2}$$

In pytorch, we disable the gradient computation for the

[1] We select the positions using a uniform distribution. However, we always include the `[CLS]` token—a special symbol prepended as the beginning of every input sentence.

Table 2. Results from BERT-large (Devlin et al., 2019) on GLUE test tasks scored using the benchmark server. We report the Matthew's Correlation for CoLA, the Spearman correlation for STS-B, F1 score for MRPC and QQP. We report the accuracy on the MNLI matched test split and the accuracy for every other tasks. The "Param." column indicates the ratio of the number of updated parameters for each task by the number of parameters in the backbone model. We indicate in **bold** the best result for each task. † indicates models we trained. We report adapter results from Houlsby et al. (2019), BitFit from Zaken et al. (2022) and Diff Pruning from Guo et al. (2021). For LoRA (Hu et al., 2022) and Ladder Side Tuning (LST) (Sung et al., 2022), we select the best learning rate in the dev set between the values proposed in the original papers, respectively $[5e^{-4}, 4e^{-4}, 3e^{-4}, 2e^{-4}]$ and $[3e^{-4}, 1e^{-3}, 3e^{-3}]$. We do not use the initialization setup proposed in LoRA or LST nor do we drop any layers for the LST method.

Method	Param. (%)	CoLA	SST-2	MRPC	QQP	QNLI	MNLI	STS-B	Avg.
Avg. number of tokens	—	11.3	13.3	53.2	30.6	49.4	39.8	27.8	32.2
Full Tuning†	100.0	60.7	**94.6**	88.3	**72.0**	92.4	85.8	85.8	82.8
Adapters (Houlsby et al., 2019)	3.6	59.5	94.0	89.5	71.8	90.7	84.9	**86.9**	82.5
BitFit (Zaken et al., 2022)	0.1	59.7	94.2	88.9	70.5	92.0	84.5	85.0	82.1
Diff Pruning (Guo et al., 2021)	0.5	**61.1**	94.1	**89.7**	71.1	**93.3**	**86.4**	86.0	**83.1**
Ladder Side Tuning† (Sung et al., 2022)	2.4	56.4	93.4	88.0	66.9	89.1	82.9	86.6	80.5
LoRA† (Hu et al., 2022)	0.3	58.5	94.0	89.2	71.1	91.1	84.7	84.6	81.9
Selective Fine-Tuning†	100.0	59.6	93.9	88.0	70.8	91.0	85.4	86.0	82.1

positions in $\bar{\mathcal{G}}$ such that:

$$\frac{\partial \mathcal{L}}{\partial a} = \left[\frac{\partial \mathcal{L}}{\partial a_{\mathcal{G}}}, \frac{\partial \mathcal{L}}{\partial a_{\bar{\mathcal{G}}}}\right] = \left[\frac{\partial \mathcal{L}}{\partial a_{\mathcal{G}}}, 0\right] \tag{3}$$

Pluging that in Eq 2, we have:

$$\frac{\partial \mathcal{L}}{dW} = \left[\frac{\partial \mathcal{L}}{\partial a_{\mathcal{G}}}\sigma' h_{\mathcal{G}}, 0\right]; \quad \frac{\partial \mathcal{L}}{db} = \frac{\partial \mathcal{L}}{\partial a}\sigma' \tag{4}$$

Given Eq 4, we only need to cache $h_{\mathcal{G}}$ for the chain rule application instead of the full activation h.

3. Experiments

3.1. Performance on downstream tasks

We first validate the relevance of our method on the GLUE benchmark (Wang et al., 2018). We use a similar hyper-parameter search space as in Zaken et al. (2022), by performing a cross validation on the dev set using a learning rate in $[5e^{-5}, 3e^{-5}, 2e^{-5}, 1e^{-5}]$. We set the batch size to 16 and perform 3 epochs on large datasets and 20 epochs on small ones (MRPC, STS-B, CoLA). We use BERT-large (Devlin et al., 2019) and either fine-tune the model fully or use our selective fine-tuning approach and only propagate the gradient through 16 input positions. We then evaluate our model on the test set and report the results in Table 2.

As seen in the second part of Table 2 the average score is on-par with full fine-tuning, thus empirically validating the relevance of the approach. We compare our results with state-of-the art comparable PEFT method and show that the difference with full fine-tuning and SoTA approaches are comparable with our results.

3.2. Influence of the proportion of frozen token positions

Given our selective fine-tuning approach, we then evaluate the impact of the number of frozen input positions on the performance. We use our selective procedure to fine-tune BERT-base on two tasks from the GLUE benchmark: MRPC and STS-B. We set the hyper-parameters as of $5e^{-5}$ for the learning rate, 32 for the batch size and 4 epochs. But we use different values for k, the number of updated input positions, ranging between 4 and 64. We report in Figure 2 (right), the average performance on the dev set of the tasks.

As seen in Figure 2, the performance increases from 84.8 to 88.8 as the number of trained positions increases from 4 to 64. However, by only tuning 32 positions, we already reach an average performance of 88.4, close to the 88.8 obtained by training 64 input positions. In that regard, our approach performs better than freezing some bottom layers (Lee et al., 2019). Indeed Lee et al. (2019) show that only tuning the four bottom layers impacted the performance by 10% on the GLUE benchmark.

3.3. GPU memory impact

Finally, we seek to analyze the GPU memory required to fine-tune LLMs using various approaches. We train our BERT-base model for 100 steps on the CoLA task using various batch sizes and report the maximum GPU memory used. We compare with the two other PEFT fine-tuning approaches closest to ours: Ladder Side Tuning (Sung et al., 2022) and LoRA (Hu et al., 2022). LoRA approach freezes

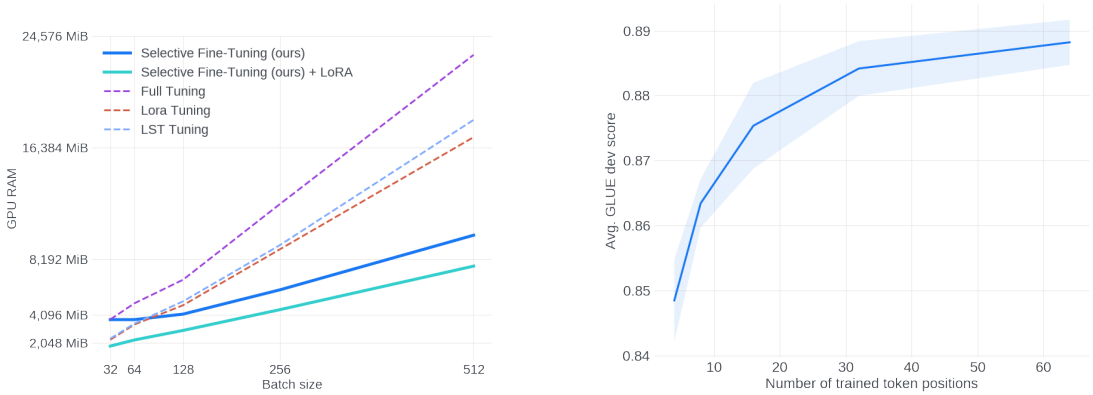

Figure 2. (left) We plot the GPU memory required to train BERT-base on the CoLA task given the batch size. We compare our approach with two PEFT approaches: Ladder Side Tuning and LoRA. (right) We plot the mean and standard deviation performance on the dev set of five runs when training BERT-base on two tasks from the GLUE benchmark: MRPC, and STS-B. We use our memory efficient fine-tuning approach with a different number of selected inputs for the gradient computation.

most of the model parameters, while only training additional matrices, which weights are added to the backbone network. Ladder Side Tuning (LST) freezes the model parameters but trains a side-network with smaller dimensions, taking as input intermediate activations from the backbone model.

Figure 2 shows the evolution of the required GPU memory with respect to the batch size. The GPU memory increases with the batch size for every approach. Our selected position approach is more memory efficient by a large margin. When using a batch size of 512, it requires two times less memory than full fine tuning, from $9,952$ MiB to $23,196$ MiB.

All methods minimize GPU memory usage. LoRA and LST aim to reduce the memory required to store optimizer states and parameter gradients, while our method reduces the memory required to store intermediate activations. Interestingly enough it is possible to use these approaches in conjunction to reduce the memory for all three contributions. Figure 2 shows that we can further reduce the memory by combining our selective fine tuning approach with LoRA, thus requiring only $7,682$ MiB with a batch size of 512, a third of the memory used for full fine-tuning.

4. Related Work

Parameter-Efficient Fine-Tuning approaches (PEFT) aim to limit the computing resources required to fine-tune LLMs, by only updating a subset of the backbone model parameters. BitFit (Zaken et al., 2022) only fine-tunes the bias-terms of the pre-trained model. Selective masking learns task-specific binary masks and applies them to the original weights, thus acting as filters selecting task-specific sub-networks (Mallya et al., 2018; Zhao et al., 2020; Radiya-Dixit & Wang, 2020).

Some approaches add a few parameters to a frozen model and train them for a specific task. Adapters are task-specific modules injected between layers of a frozen pre-trained transformer model (Houlsby et al., 2019; Stickland & Murray, 2019; Pfeiffer et al., 2021; Rücklé et al., 2021; Mahabadi et al., 2021). Prefix-tuning (Li & Liang, 2021; Qin & Eisner, 2021; Liu et al., 2021) prepends a sequence of continuous task-specific vectors to the input of a frozen model and tunes them for each task. Diff pruning (Guo et al., 2021) reparametrizes the parameters of a task-specific model as the sum of the fixed pre-trained model parameters with a task-specific sparse vector specifically tuned for each task.

More close to our work, Hu et al. (2022) propose a low-rank matrix decomposition that compresses the pre-trained model weights. Side Tuning (Zhang et al., 2020; Sung et al., 2022) proposes to reduce the memory requirement for storing intermediate activations by training a small separated side network, which takes intermediate activations from the backbone network as input via shortcut connections.

5. Conclusion and Future Work

Our selective approach reduces the GPU memory required to fine-tune transformer-based language models, while maintaining the same standard benchmark performance. Our approach selects a subset of the input positions through which the gradient is propagated, while the other remains frozen. When increasing the batch size, our approach reduces the memory requirements by up to a third. We hope that our approach will facilitate the fine-tuning of large language models, in specializing them for specific domains, or co-training them with other neural components from a larger system.

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Guo, D., Rush, A. M., and Kim, Y. Parameter-efficient transfer learning with diff pruning. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4884–4896. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.378. URL https://doi.org/10.18653/v1/2021.acl-long.378.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *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*, pp. 2790–2799. PMLR, 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Lee, J., Tang, R., and Lin, J. What would elsa do? freezing layers during transformer fine-tuning. *CoRR*, abs/1911.03090, 2019. URL http://arxiv.org/abs/1911.03090.

Li, X., Tramèr, F., Liang, P., and Hashimoto, T. Large language models can be strong differentially private learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=bVuP3ltATMz.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.353. URL https://doi.org/10.18653/v1/2021.acl-long.353.

Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., and Tang, J. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602, 2021. URL https://arxiv.org/abs/2110.07602.

Mahabadi, R. K., Henderson, J., and Ruder, S. Compacter: Efficient low-rank hypercomplex adapter layers. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1022–1035, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/081be9fdff07f3bc808f935906ef70c0-Abstract.html.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pp. 72–88. Springer, 2018. doi: 10.1007/978-3-030-01225-0_5. URL https://doi.org/10.1007/978-3-030-01225-0_5.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. Adapterfusion: Non-destructive task composition for

transfer learning. In Merlo, P., Tiedemann, J., and Tsarfaty, R. (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pp. 487–503. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.39. URL https://doi.org/10.18653/v1/2021.eacl-main.39.

Pfeiffer, J., Ruder, S., Vulic, I., and Ponti, E. M. Modular deep learning. *CoRR*, abs/2302.11529, 2023. doi: 10.48550/arXiv.2302.11529. URL https://doi.org/10.48550/arXiv.2302.11529.

Qin, G. and Eisner, J. Learning how to ask: Querying lms with mixtures of soft prompts. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5203–5212. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.410. URL https://doi.org/10.18653/v1/2021.naacl-main.410.

Radiya-Dixit, E. and Wang, X. How fine can fine-tuning be? learning efficient language models. In Chiappa, S. and Calandra, R. (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2435–2443. PMLR, 2020. URL http://proceedings.mlr.press/v108/radiya-dixit20a.html.

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., and Gurevych, I. Adapterdrop: On the efficiency of adapters in transformers. In Moens, M., Huang, X., Specia, L., and Yih, S. W. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 7930–7946. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.626. URL https://doi.org/10.18653/v1/2021.emnlp-main.626.

Schick, T. and Schütze, H. It's not just size that matters: Small language models are also few-shot learners. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 2339–2352. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.185. URL https://doi.org/10.18653/v1/2021.naacl-main.185.

Stickland, A. C. and Murray, I. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *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*, pp. 5986–5995. PMLR, 2019. URL http://proceedings.mlr.press/v97/stickland19a.html.

Sung, Y.-L., Cho, J., and Bansal, M. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=isPnnaTZaP5.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. 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*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Zaken, E. B., Goldberg, Y., and Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1–9. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-short.1. URL https://doi.org/10.18653/v1/2022.acl-short.1.

Zhang, J. O., Sax, A., Zamir, A., Guibas, L. J., and Malik, J. Side-tuning: A baseline for network adaptation via additive side networks. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12348 of *Lecture Notes in Computer Science*, pp. 698–714. Springer, 2020. doi: 10.1007/978-3-030-58580-8_41. URL https://doi.org/10.1007/978-3-030-58580-8_41.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer,

L. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022. doi: 10.48550/arXiv.2205.01068. URL https://doi.org/10.48550/arXiv.2205.01068.

Zhao, M., Lin, T., Mi, F., Jaggi, M., and Schütze, H. Masking as an efficient alternative to finetuning for pretrained language models. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 2226–2241. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.174. URL https://doi.org/10.18653/v1/2020.emnlp-main.174.