PARAMETER-EFFICIENT TRANSFER LEARNING WITH DIFF PRUNING

Demi Guo
Harvard University
dguo@college.harvard.edu

Alexander M. Rush
Cornell University
Hugging Face

Yoon Kim
MIT-IBM Watson AI
MIT CSAIL

arXiv:2012.07463v1 [cs.CL] 14 Dec 2020

ABSTRACT

While task-specific finetuning of pretrained networks has led to significant empirical advances in NLP, the large size of networks makes finetuning difficult to deploy in multi-task, memory-constrained settings. We propose *diff pruning* as a simple approach to enable parameter-efficient transfer learning within the pretrain-finetune framework. This approach views finetuning as learning a task-specific "diff" vector that is applied on top of the pretrained parameter vector, which remains fixed and is shared across different tasks. The diff vector is adaptively pruned during training with a differentiable approximation to the L_0-norm penalty to encourage sparsity. Diff pruning becomes parameter-efficient as the number of tasks increases, as it requires storing only the nonzero positions and weights of the diff vector for each task, while the cost of storing the shared pretrained model remains constant. It further does not require access to all tasks during training, which makes it attractive in settings where tasks arrive in stream or the set of tasks is unknown. We find that models finetuned with diff pruning can match the performance of fully finetuned baselines on the GLUE benchmark while only modifying 0.5% of the pretrained model's parameters per task.

1 INTRODUCTION

Task-specific finetuning of pretrained deep networks has become the dominant paradigm in contemporary NLP, achieving state-of-the-art results across a suite of natural language understanding tasks (Devlin et al., 2019; Liu et al., 2019c; Yang et al., 2019; Lan et al., 2020). While straightforward and empirically effective, this approach is difficult to scale to multi-task, memory-constrained settings (e.g. for on-device applications), as it requires shipping and storing a full set of model parameters for each task. Inasmuch as these models are learning generalizable, task-agnostic language representations through self-supervised pretraining, finetuning the entire model for each task seems especially profligate.

A popular approach to parameter-efficiency with pretrained models is to learn sparse models for each task where a subset of the final model parameters are exactly zero (Gordon et al., 2020; Sajjad et al., 2020; Zhao et al., 2020; Sanh et al., 2020). Such approaches often face a steep sparsity/performance tradeoff, and a substantial portion of nonzero parameters (e.g. 10%-30%) are still typically required to match the performance of the dense counterparts. An alternative is to use multi-task learning or feature-based transfer for more parameter-efficient transfer learning with pretrained models (Liu et al., 2019b; Clark et al., 2019; Stickland & Murray, 2019; Reimers & Gurevych, 2019; Feng et al., 2020). These methods learn only a small number of additional parameters (e.g. a linear layer) on top of a shared model. However, multi-task learning generally requires access to all tasks during training to prevent catastrophic forgetting (French, 1999), while feature-based transfer learning (e.g. based on task-agnostic sentence representations) is typically outperformed by full finetuning (Howard & Ruder, 2018).

Adapters (Rebuffi et al., 2018) have recently emerged as a promising approach to parameter-efficient transfer learning within the pretrain-finetune paradigm (Houlsby et al., 2019; Pfeiffer et al., 2020a;b;c). Adapter layers are smaller, task-specific modules that are inserted between layers of a pretrained model, which remains fixed and is shared across tasks. These approaches do not require access to all tasks during training, making them attractive in settings where one hopes to obtain and

Our code is available at https://github.com/dguo98/DiffPruning share performant models as new tasks arrive in stream. Houlsby et al. (2019) find that adapter layers trained on BERT can match the performance of fully finetuned BERT on the GLUE benchmark (Wang et al., 2019a) while only requiring 3.6% additional parameters (on average) per task.

In this work, we consider a similar setting as adapters but propose a new *diff pruning* approach with the goal of even more parameter-efficient transfer learning. Diff pruning views finetuning as learning a task-specific difference vector that is applied on top of the pretrained parameter vector, which remains fixed and is shared across different tasks. In order to learn this vector, we reparameterize the task-specific model parameters as $\boldsymbol{\theta}_{\text{task}} = \boldsymbol{\theta}_{\text{pretrained}} + \boldsymbol{\delta}_{\text{task}}$, where the pretrained parameter vector $\boldsymbol{\theta}_{\text{pretrained}}$ is fixed and the task-specific diff vector $\boldsymbol{\delta}_{\text{task}}$ is finetuned. The diff vector is regularized with a differentiable approximation to the L_0-norm penalty (Louizos et al., 2018) to encourage sparsity. This approach can become parameter-efficient as the number of tasks increases as it only requires storing the nonzero positions and weights of the diff vector for each task. The cost of storing the shared pretrained model remains constant and is amortized across multiple tasks. On the GLUE benchmark (Wang et al., 2019a), diff pruning can match the performance of the fully finetuned BERT baselines while finetuning only 0.5% of the pretrained parameters per task, making it a potential alternative to adapters for parameter-efficient transfer learning.

2 Background: Transfer Learning for NLP

The field of NLP has recently seen remarkable progress through *transfer learning* with a pretrain-and-finetune paradigm, which initializes a subset of the model parameters for all tasks from a pretrained model and then finetunes on a task specific objective. Pretraining objectives include context prediction (Mikolov et al., 2013), autoencoding (Dai & Le, 2015), machine translation (McCann et al., 2017), and more recently, variants of language modeling (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019) objectives.

Here we consider applying transfer learning to multiple tasks. We consider a setting with a potentially unknown set of tasks, where each $\tau \in \mathcal{T}$ has an associated training set $\{x_\tau^{(n)}, y_\tau^{(n)}\}_{n=1}^N$. For all tasks, the goal is to produce (possibly tied) model parameters $\boldsymbol{\theta}_\tau$ to minimize the empirical risk,

$$\min_{\boldsymbol{\theta}_\tau} \ \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(f(x_\tau^{(n)}; \boldsymbol{\theta}_\tau), y_\tau^{(n)}\right) + \lambda R(\boldsymbol{\theta}_\tau)$$

where $f(\cdot; \boldsymbol{\theta})$ is a parameterized function over the input (e.g. a neural network), $\mathcal{L}(\cdot,\cdot)$ is a loss function (e.g. cross-entropy), and $R(\cdot)$ is an optional regularizer with hyperparameter λ.

This multi-task setting can use the pretrain-then-finetune approach by simply learning independent parameters for each task; however the large size of pretrained models makes this approach exceedingly parameter inefficient. For example, widely-adopted models such as $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ have 110M and 340M parameters respectively, while their contemporaries such as T5 (Raffel et al., 2020), Megatron-LM (Shoeybi et al., 2019), and Turing-NLG (Rajbhandari et al., 2019) have parameter counts in the billions. Storing the fully finetuned models becomes difficult even for a moderate number of tasks.[1] A classic approach to tackling this parameter-inefficiency (Caruana, 1997) is to train a single shared model (along with a task-specific output layer) against multiple tasks through joint training. However, the usual formulation of multi-task learning requires the set of tasks $\mathcal{T}$ to be known in advance in order to prevent catastrophic forgetting (French, 1999),[2] making it unsuitable for applications in which the set of tasks is unknown (e.g. when tasks arrive in stream).

3 Diff Pruning

Diff pruning formulates task-specific finetuning as learning a diff vector $\boldsymbol{\delta}_\tau$ that is added to the pretrained model parameters $\boldsymbol{\theta}_{\text{pretrained}}$. We first reparameterize the task-specific model parameters,

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\text{pretrained}} + \boldsymbol{\delta}_\tau,$$

[1]An intriguing line of work suggests that large-scale language models can be used *without* finetuning for a variety of tasks if given the appropriate context (Radford et al., 2019; Brown et al., 2020). While interesting, these models generally underperform task-specific models and require billions of parameters, though recent work suggests that they can be made substantially smaller (Schick & Schutze, 2020).

[2]However, work on *continual learning* mitigates these issues to an extent (Shin et al., 2017; Lopez-Paz & Ranzato, 2017; Lee et al., 2017; Kirkpatrick et al., 2017; Parisi et al., 2018).

which results in the following empirical risk minimization problem,

$$\min_{\boldsymbol{\delta}_\tau} \ \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(f(x_\tau^{(n)}; \boldsymbol{\theta}_{\text{pretrained}} + \boldsymbol{\delta}_\tau), y_\tau^{(n)}\right) + \lambda R(\boldsymbol{\theta}_{\text{pretrained}} + \boldsymbol{\delta}_\tau).$$

This trivial reparameterization is equivalent to the original formulation. Its benefit comes in the multi-task setting where the cost of storing the pretrained parameters $\boldsymbol{\theta}_{\text{pretrained}}$ is amortized across tasks, and the only marginal cost for new tasks is the diff vector. If we can regularize $\boldsymbol{\delta}_\tau$ to be sparse such that $\|\boldsymbol{\delta}_\tau\|_0 \ll \|\boldsymbol{\theta}_{\text{pretrained}}\|_0$, then this approach can become more parameter-efficient as the number of tasks increases. We can specify this goal with an L_0-norm penalty on the diff vector,

$$R(\boldsymbol{\theta}_{\text{pretrained}} + \boldsymbol{\delta}_\tau) = \|\boldsymbol{\delta}_\tau\|_0 = \sum_{i=1}^{d} \mathbb{1}\{\boldsymbol{\delta}_{\tau,i} \neq 0\}.$$

3.1 Differentiable approximation to the L_0-norm

This regularizer is difficult to directly optimize as it is non-differentiable. In order to approximate this L_0 objective, we follow the standard approach for gradient-based learning with L_0 sparsity using a relaxed mask vector (Louizos et al., 2018). This approach involves relaxing a binary vector into continuous space, and then multiplying it with a dense weight vector to determine how much of the weight vector is applied during training. After training, the mask is deterministic and a large portion of the diff vector is true zero.

To apply this method we first decompose $\boldsymbol{\delta}_\tau$ into a binary mask vector multiplied with a dense vector,

$$\boldsymbol{\delta}_\tau = \mathbf{z}_\tau \odot \mathbf{w}_\tau, \qquad \mathbf{z}_\tau \in \{0,1\}^d, \mathbf{w}_\tau \in \mathbb{R}^d$$

We can now instead optimize an expectation with respect to $\mathbf{z}_\tau$, whose distribution $p(\mathbf{z}_\tau; \boldsymbol{\alpha}_\tau)$ is initially Bernoulli with parameters $\boldsymbol{\alpha}_\tau$,

$$\min_{\boldsymbol{\alpha}_\tau, \mathbf{w}_\tau} \ \mathbb{E}_{\mathbf{z}_\tau \sim p(\mathbf{z}_\tau;\boldsymbol{\alpha}_\tau)}\left[\frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(f(x_\tau^{(n)}; \boldsymbol{\theta}_{\text{pretrained}} + \mathbf{z}_\tau \odot \mathbf{w}_\tau,), y_\tau^{(n)}\right) + \lambda\|\boldsymbol{\delta}_\tau\|_0\right].$$

This objective is still difficult in practice due to $\mathbf{z}_\tau$'s being discrete (which requires the score function gradient estimator), but the expectation provides some guidance for empirically effective relaxations. We follow prior work (Louizos et al., 2018; Wang et al., 2019b) and relax $\mathbf{z}_\tau$ into continuous space $[0,1]^d$ with a stretched Hard-Concrete distribution (Jang et al., 2017; Maddison et al., 2017), which allows for the use of pathwise gradient estimators. Specifically, $\mathbf{z}_\tau$ is now defined to be a deterministic and (sub)differentiable function of a sample $\mathbf{u}$ from a uniform distribution,

$$\mathbf{u} \sim U(\mathbf{0}, \mathbf{1}), \qquad \mathbf{s}_\tau = \sigma\left(\log \mathbf{u} - \log(1 - \mathbf{u}) + \boldsymbol{\alpha}_\tau\right),$$
$$\bar{\mathbf{s}}_\tau = \mathbf{s}_\tau \times (r - l) + l, \qquad \mathbf{z}_\tau = \min(\mathbf{1}, \max(\mathbf{0}, \bar{\mathbf{s}}_\tau)).$$

Here $l < 0$ and $r > 1$ are two constants used to stretch $\mathbf{s}_\tau$ into the interval $(l, r)^d$ before it is clamped to $[0,1]^d$ with the $\min(\mathbf{1}, \max(\mathbf{0}, \cdot))$ operation. In this case we have a differentiable closed-form expression for the expected L_0-norm,

$$\mathbb{E}\left[\|\boldsymbol{\delta}_\tau\|_0\right] = \sum_{i=1}^{d} \mathbb{E}\left[\mathbb{1}\{\mathbf{z}_{\tau,i} > 0\}\right] = \sum_{i=1}^{d} \sigma\left(\boldsymbol{\alpha}_{\tau,i} - \log\frac{-l}{r}\right).$$

Thus the final optimization problem is given by,

$$\min_{\boldsymbol{\alpha}_\tau, \mathbf{w}_\tau} \ \mathbb{E}_{\mathbf{u} \sim U[\mathbf{0},\mathbf{1}]}\left[\frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(f(x_\tau^{(n)}; \boldsymbol{\theta}_{\text{pretrained}} + \mathbf{z}_\tau \odot \mathbf{w}_\tau,), y_\tau^{(n)}\right)\right] + \lambda \sum_{i=1}^{d} \sigma\left(\boldsymbol{\alpha}_{\tau,i} - \log\frac{-l}{r}\right),$$

and we can now utilize pathwise gradient estimators to optimize the first term with respect to $\boldsymbol{\alpha}_\tau$ since the expectation no longer depends on it.[3] After training we obtain the final diff vector $\boldsymbol{\delta}_\tau$ by sampling $\mathbf{u}$ once to obtain $\mathbf{z}_\tau$ (which is not necessarily a binary vector but has a significant number of dimensions equal to exactly zero due to the clamping function), then setting $\boldsymbol{\delta}_\tau = \mathbf{z}_\tau \odot \mathbf{w}_\tau$.[4]

[3] To reduce notation clutter we subsume the parameters of the task-specific output layer, which is not pretrained, into $\boldsymbol{\theta}_{\text{pretrained}}$. We do not apply the L_0-norm penalty on these parameters during training.

[4] We found sampling once to work as well as other alternatives (e.g. based on multiple samples).

3.2 L_0-BALL PROJECTION WITH MAGNITUDE PRUNING FOR SPARSITY CONTROL

Differentiable L_0 regularization provides a strong way to achieve high sparsity rate. However, it would be ideal to have more fine-grained control into the exact sparsity rate in the diff vector, especially considering applications which require specific parameter budgets. As λ is just the Lagrangian multiplier for the constraint $\mathbb{E}\left[\|\boldsymbol{\delta}_\tau\|_0\right] < \eta$ for some η, this could be achieved in principle by searching over different values of λ. However we found it more efficient and empirically effective to achieve an exact sparsity rate by simply projecting onto the L_0-ball after training.

Specifically we use magnitude pruning on the diff vector $\boldsymbol{\delta}_\tau$ and target a sparsity rate $t\%$ by only keeping the top $t\% \times d$ values in $\boldsymbol{\delta}_\tau$.[5] Note that unlike standard magnitude pruning, this is based on the magnitude of the diff vector values and not the model parameters. As is usual in magnitude pruning, we found it important to further finetune $\boldsymbol{\delta}_\tau$ with the nonzero masks fixed to maintain good performance (Han et al., 2016). Since this type of parameter-efficiency through projection onto the L_0-ball can be applied without adaptive diff pruning,[6] such an approach will serve as one of our baselines in the empirical study.

3.3 STRUCTURED DIFF PRUNING

Diff pruning, as presented above, is architecture-agnostic and does not exploit the underlying model structure—each dimension of $\mathbf{z}_\tau$ is independent from one another. While this makes the approach potentially more flexible, we might expect to achieve better sparsity/performance tradeoff through a structured formulation which encourages active parameters to group together and other areas to be fully sparse. Motivated by this intuition, we first partition the parameter indices into G groups $\{g(1), \dots, g(G)\}$ where $g(j)$ is a subset of parameter indices governed by group $g(j)$.[7] We then introduce a scalar $\mathbf{z}_\tau^j$ (with the associated parameter $\boldsymbol{\alpha}_\tau^j$) for each group $g(j)$, and decompose the task-specific parameter for index $i \in g(j)$ as $\boldsymbol{\delta}_{\tau,i}^j = \mathbf{z}_{\tau,i} \times \mathbf{z}_\tau^j \times \mathbf{w}_{\tau,i}$. The expected L_0-norm is then given by,

$$\mathbb{E}\left[\|\boldsymbol{\delta}_\tau\|_0\right] = \sum_{j=1}^{G} \sum_{i \in g(j)} \mathbb{E}\left[\mathbb{1}\{\mathbf{z}_{\tau,i} \cdot \mathbf{z}_\tau^g > 0\}\right] = \sum_{j=1}^{G} \sum_{i \in g(j)} \sigma\left(\boldsymbol{\alpha}_{\tau,i} - \log \frac{-l}{r}\right) \times \sigma\left(\boldsymbol{\alpha}_\tau^j - \log \frac{-l}{r}\right),$$

and we can train with gradient-based optimization as before.

4 EXPERIMENTS

4.1 MODEL AND DATASETS

For evaluation we mainly use the GLUE benchmark (Wang et al., 2019b), a popular finetuning dataset. Following adapters (Houlsby et al., 2019), we test our approach on the following subset of the GLUE tasks: Multi-Genre Natural Language Inference (MNLI), where the goal is two predict whether the relationship between two sentences is entailment, contradiction, or neutral (we test on both MNLI_m and MNLI_{mm} which respectively tests on matched/mismatched domains); Quora Question Pairs (QQP), a classification task to predict whether two question are semantically equivalent; Question Natural Language Inference (QNLI), which must predict whether a sentence is a correct answer to the question; Stanford Sentiment Treebank (SST-2), a sentence classification task to predict the sentiment of movie reviews; Corpus of Linguistic Acceptability (CoLA), where the goal is predict whether a sentence is linguistically acceptable or not; Semantic Textual Similarity Benchmark (STS-B), which must predict a similarity rating between two sentences; Microsoft Research Paraphrase Corpus (MRPC), where the goal is to predict whether two sentences are semantically equivalent; Recognizing Textual Entailment (RTE), which must predict whether a second sentence is entailed by the first. The benchmark uses Matthew's correlation for CoLA, Spearman for STS-B,

[5] Wang et al. (2019b) show that it also is possible to inject such a constraint softly into the training objective by regularizing the expected model size towards a certain rate. However, since the constraint is soft this approach also makes it difficult to target an exact sparsity rate.

[6] Concretely, one can obtain $\boldsymbol{\theta}_\tau$ through usual finetuning, set $\boldsymbol{\delta}_\tau = \boldsymbol{\theta}_\tau - \boldsymbol{\theta}_{\text{pretrained}}$, and then apply magnitude pruning followed by additional finetuning on $\boldsymbol{\delta}_\tau$.

[7] While groups can be defined in various ways, we found that defining groups based on each matrix/bias vector of the pretrained model was simple and worked well enough.

F_1 score for MRPC/QQC, and accuracy for MNLI/QNLI/SST-2/RTE. Finally, to test for generalization beyond the GLUE tasks, we also test our approach on the SQuAD extractive question answering dataset (Rajpurkar et al., 2016).

For all experiments, we use the $\text{BERT}_{\text{LARGE}}$ model from Devlin et al. (2019), which has 24 layers, 1024 hidden size, 16 attention heads, and 340M parameters. We use the Huggingface Transformer library (Wolf et al., 2019) to conduct our experiments.

4.2 BASELINES

We compare both structured and non-structured variants of diff pruning against the following baselines: **Full finetuning**, which fully finetunes $\text{BERT}_{\text{LARGE}}$ as usual; **Last layer finetuning**, which only finetunes the penultimate layer (along with the final output layer)[8]; **Adapters** from Houlsby et al. (2019), which train task-specific bottleneck layers between between each layer of a pretrained model, where parameter-efficiency can be controlled by varying the size of the bottleneck layers; and **Non-adaptive diff pruning**, which performs diff pruning just based on magnitude pruning (i.e., we obtain $\boldsymbol{\theta}_\tau$ through usual finetuning, set $\boldsymbol{\delta}_\tau = \boldsymbol{\theta}_\tau - \boldsymbol{\theta}_{\text{pretrained}}$, and then apply magnitude pruning followed by additional finetuning on $\boldsymbol{\delta}_\tau$). For diff pruning we set our target sparsity rate to 0.5% and investigate the effect of different target sparsity rates in section 5.2.

4.3 IMPLEMENTATION DETAILS AND HYPERPARAMETERS

Diff pruning introduces additional hyperparameters l, r (for stretching the Hard-Concrete distribution) and λ (for weighting the approximate L_0-norm penalty). We found $l = -1.5, r = 1.5, \lambda = 1.25 \times 10^{-7}$ to work well across all tasks. We also initialize the weight vector $\mathbf{w}_\tau$ to $\mathbf{0}$, and $\boldsymbol{\alpha}_\tau$ to a positive vector (we use $\mathbf{5}$) to encourage $\mathbf{z}_\tau$ to be close to $\mathbf{1}$ at the start of training.[9] While we mainly experiment with $\text{BERT}_{\text{LARGE}}$ to compare against prior work with adapters (Houlsby et al., 2019), in preliminary experiments we found these hyperparameters to work for finetuning RoBERTa (Liu et al., 2019c) and XLNet (Yang et al., 2019) models as well.

For all tasks we initially train for 3 epochs and perform a hyperparameter search over batch size $\in \{5, 8, 12, 16\}$ and learning rate $\in \{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$. However we found the default settings used for regular finetuning as suggested in the original BERT paper to work well for most tasks. Finetuning with the fixed mask after projecting onto the L_0-ball with magnitude pruning is done for 3 epochs with a learning rate of 5×10^{-5} for all datasets except for MRPC/STS-B/RTE/SST-2 dataset, where we finetune for 5 epochs. The exact hyperparameters for each task are given in section A.2 of the appendix. Grouping for the structured version of diff pruning is based on the matrix/bias vectors (i.e. parameters that belong to the same matrix or bias vector are assumed to be in the same group), which results in 393 groups.[10]

5 RESULTS AND ANALYSIS

5.1 RESULTS ON GLUE BENCHMARK

Our main results on the GLUE benchmark are shown in Table 1. Structured diff pruning can match the performance of a fully finetuned $\text{BERT}_{\text{LARGE}}$ model while only requiring 0.5% additional parameters per task. Diff pruning without structured sparsity also performs well, though slightly worse than the structured approach. Non-adaptive diff pruning, which magnitude prunes the diff vector without learning the binary mask $\mathbf{z}_\tau$, performs significantly worse, indicating the importance of learning the masking vector. Compared to adapters, diff pruning obtains similar performance while requiring fewer parameters per task, making it a potential alternative for parameter-efficient transfer learning.[11]

[8]Wu et al. (2020) observe that finetuning later layers generally performs better than finetuning earlier layers

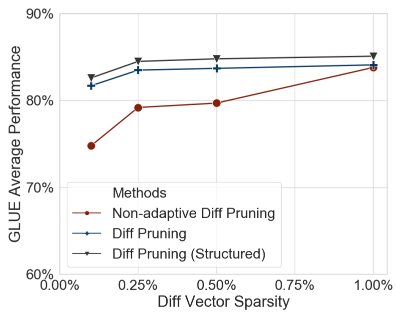
[9]These values were found via by a light hyperparameter search on the SST-2 validation set.

[10]This definition of groups is implementation-specific since it depends on how one concatenates the input vector before each affine layer. Our grouping is based on Huggingface's BERT implementation at commit `656e1386a296d696327a9db37de2ccccc79e2cc7` (available at `https://github.com/huggingface/transformers/blob/656e1386a296d696327a9db37de2ccccc79e2cc7/src/transformers/modeling_bert.py`). In preliminary experiments we found this simple definition to work well compared to alternative group definitions (e.g. based on individual neurons).

[11]However diff pruning incurs additional storage cost due to storing the nonzero positions of the diff vector.

	Total params	New params per task	QNLI*	SST-2	MNLI$_m$	MNLI$_{mm}$	CoLA	MRPC	STS-B	RTE	QQP	Avg
Full finetuning	9.00×	100%	91.1	94.9	86.7	85.9	60.5	89.3	87.6	70.1	72.1	80.9
Adapters (8-256)	1.32×	3.6%	90.7	94.0	84.9	85.1	59.5	89.5	86.9	71.5	71.8	80.4
Adapters (64)	1.19×	2.1%	91.4	94.2	85.3	84.6	56.9	89.6	87.3	68.6	71.8	79.8
Full finetuning	9.00×	100%	93.4	94.1	86.7	86.0	59.6	88.9	86.6	71.2	71.7	80.6
Last layer	1.34×	3.8%	79.8	91.6	71.4	72.9	40.2	80.1	67.3	58.6	63.3	68.2
Non-adap. diff pruning	1.05×	0.5%	89.7	93.6	84.9	84.8	51.2	81.5	78.2	61.5	68.6	75.5
Diff pruning	1.05×	0.5%	92.9	93.8	85.7	85.6	60.5	87.0	83.5	68.1	70.6	79.4
Diff pruning (struct.)	1.05×	0.5%	93.3	94.1	86.4	86.0	61.1	89.7	86.0	70.6	71.1	80.6

Table 1: GLUE benchmark test server results with BERT$_{\text{LARGE}}$ models. (Top) Results with adapter bottleneck layers (brackets indicate the size of bottlenecks), taken from from Houlsby et al. (2019). (Bottom) Results from this work. *QNLI results are not directly comparable across the two works as the GLUE benchmark has updated the test set since then. To make our results comparable the average column is calculated without QNLI.



	SQuAD New Params	F$_1$
Houlsby et al. (2019)		
Full finetuning	100%	90.7
Adapters	2.0%	90.4
This work		
Full finetuning	100%	90.8
Diff pruning	1.0%	92.1
Diff pruning (struct.)	0.5%	91.1
Diff pruning (struct.)	1.0%	93.2

Figure 1: (Left) Average performance on the GLUE validation set across different target sparsity rates for the different methods. (Right) Results with BERT$_{\text{LARGE}}$ on the SQuAD v1.1 validation set.

5.2 Varying the Target Sparsity

In Figure 1 (left), we plot results on the GLUE validation set averaged across all tasks at target sparsity rates of $0.1\%, 0.25\%, 0.5\%, 1.0\%$ for the different baselines. Structured diff pruning consistently outperforms non-structured and and non-adaptive variants across different sparsity rates. The advantage of adaptive methods becomes more pronounced at extreme sparsity rates. In Table 2, we report the breakdown of accuracy of structured diff pruning across different tasks and sparsity rates, where we observe that different tasks have different sensitivity to target sparsity rates. This suggests that we can obtain even greater parameter-efficiency through targeting task-specific sparsity rates in the diff vector.

5.3 Results on SQuAD Extractive Question Answering

To demonstrate the effectiveness of our approach beyond the GLUE tasks, we additionally experiment on SQuAD (Rajpurkar et al., 2016), an extractive question answering dataset where the model has to select the answer span to a question given a Wikipedia paragraph. To make direct comparisons with Houlsby et al. (2019), we run all experiments on SQuAD v1.1. For diff pruning, we use the same general hyperparameters as our full finetuning baseline (see section A.2). As shown in Figure 1 (right), diff pruning is able achieve comparable or better performance with only 1.0% additional parameters. Interestingly, diff pruning measurably improves the upon the full finetuning baseline while modifying fewer parameters, which indicates that diff pruning can have a useful regularization effect on top of parameter-efficiency.

5.4 Structured vs. Non-Structured Diff Pruning

Structured diff pruning introduces an additional mask per group, which encourages pruning of entire groups. This is less restrictive than traditional group sparsity techniques that have been used with L_0-norm relaxations which force all parameters in a group to share the same mask (Louizos et al., 2018; Wang et al., 2019b). However we still expect entire groups to be pruned out more often in the structured case, which might bias the learning process towards either eliminating completely or clustering together nonzero diffs. In Table 3, we indeed find that structured diff pruning leads to

Diff vector target sparsity	QNLI	SST-2	MNLI$_m$	MNLI$_{mm}$	CoLA	MRPC	STS-B	RTE	QQP	Avg
0.10%	92.7	93.3	85.6	85.9	58.0	87.4	86.3	68.6	85.2	82.5
0.25%	93.2	94.2	86.2	86.5	63.3	90.9	88.4	71.5	86.1	84.5
0.50%	93.4	94.2	86.4	86.9	63.5	91.3	89.5	71.5	86.6	84.8
1.00%	93.3	94.2	86.4	87.0	66.3	91.4	89.9	71.1	86.6	85.1
100%	93.5	94.1	86.5	87.1	62.8	91.9	89.8	71.8	87.6	85.0

Table 2: Structured diff pruning results on the validation set with different target sparsity rates. Average performance includes all 9 tasks.

	QNLI	SST-2	MNLI	CoLA	MRPC	STS-B	RTE	QQP	Avg
Non-structured	6.2%	6.1%	6.0%	6.4%	6.1%	6.4%	7.1%	6.1%	6.3%
Structured	37.7%	64.6%	28.8%	20.8%	13.2%	12.2%	12.7%	34.9%	28.1%

Table 3: Percentage of groups where all of the parameters in the group are fully zero for structured vs. non-structured diff pruning at 0.5% target sparsity. We group based on each matrix/bias vector, resulting in 393 groups in total.

finetuned models that are much more likely to leave entire groups unchanged from their pretrained values (zero diffs).

5.5 Task-Specific Sparsity

Different layers of pretrained models have been argued to encode different information (Liu et al., 2019a; Tenney et al., 2019). Given that each task will likely recruit different kinds of language phenomena embedded in the hidden layers, we hypothesize that diff pruning will modify different parts of the pretrained model through task-specific finetuning. Figure 2 shows the percentage of nonzero diff parameters attributable to the different layers for each task. We find that different tasks indeed modify different parts of the network, although there are some qualitative similarities between some tasks, for example between QNLI & QQP (both must encode questions), and MRPC & STS-B (both must predict similarity between sentences). The embedding layer is very sparsely modified for all tasks. While some of the variations in the sparsity distributions is due to simple randomness, we do observe some level of consistency over multiple runs of the same task, as shown in section A.1 of the appendix.

The ability to modify different parts of the pretrained model for each task could explain the improved parameter-efficiency of our approach compared to Houlsby et al. (2019)'s adapter layers, which can only read/write to the pretrained model at certain points of the computational graph.[12] This potentially suggests that adapter layers with more fine-grained access into model internals (e.g. adapters for key/value/query transformations) might result in even greater parameter-efficiency. While left as future work, we also note that diff pruning can be applied in conjunction with adapters, which might further improve results.

5.6 Effect of L_0-ball Projection via Magnitude Pruning

Applying magnitude pruning to project onto the L_0-ball was crucial in achieving exact sparsity targets. As shown in Table 4, we observed little loss in performance through magnitude pruning. We also found it important to finetune with a fixed mask after magnitude pruning, even for the approach that does not apply magnitude pruning.

6 Discussion

6.1 Limitations

For training, our approach requires more memory than usual finetuning due to additionally optimizing $\boldsymbol{\alpha}_\tau$ and $\mathbf{w}_\tau$. Since the majority of GPU memory is typically utilized by a minibatch's intermediate layers, this did not present a significant challenge for pretrained models that we experimented with in this study. However, this could present an issue as model sizes get larger and larger. After training, storing the task-specific diff vector requires storing a compressed version with both the

[12]To simulate this restricted setting, we tried applying diff pruning only on the dense transformations just before the output of each layer (i.e. after self-attention layers), and observed much worse performance.

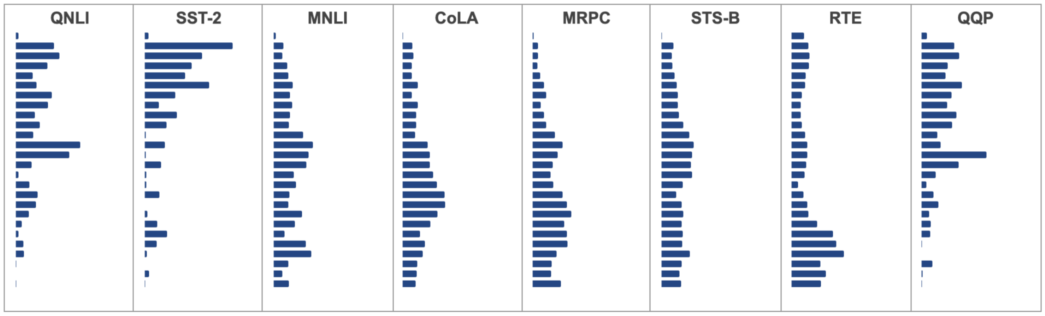


Figure 2: Percentage of modified parameters attributable to each layer for different tasks at 0.5% target sparsity. The layers are ordered from earlier to later (i.e. the embedding layer is shown at the top). The x-axis for each plot goes from 0% to 20%.

	QNLI	SST-2	MNLI_m	MNLI_{mm}	CoLA	MRPC	STS-B	RTE	QQP	Avg
Sparsity w/o Mag. Pruning	1.5%	0.6%	0.8%	0.8%	1.6%	2.4%	3.3%	0.7%	0.6%	1.4%
No Finetuning	92.1	93.8	84.3	84.8	59.8	87.7	87.1	58.9	84.4	81.4
Finetuning	93.8	94.0	86.2	86.8	63.1	91.9	89.7	71.8	86.5	84.9
Mag. Pruning + Finetuning	93.4	94.2	86.4	86.9	63.5	91.3	89.5	71.5	86.6	84.8

Table 4: (Top) Sparsity and performance without magnitude pruning on the validation set with structured diff pruning. We show results before and after finetuning with a fixed mask. (Bottom) Performance with 0.5% target sparsity and fixed mask finetuning.

nonzero positions and weights, which incurs additional storage requirements. Finally, while training efficiency was not a primary concern of this work, diff pruning was also approximately $1.5\times$ to $2\times$ slower to train per minibatch than regular finetuning.

6.2 Information-efficient Transfer Learning

Efficiently representing pretrained models adapted to new tasks is becoming an increasingly important problem in contemporary NLP. This paper focuses on a rather narrow definition of efficiency—parameter-efficiency. An interesting direction might be to target generalizations of parameter-efficiency, for example, information-efficiency, which aims to minimize the number of bits required to represent the task-specific model when given the pretrained model for free. This view can suggest other avenues for achieving information-efficient transfer learning: for example, "what is the minimum number of (potentially synthetic) datapoints that we can finetune BERT on to obtain a good task-specific model?",[13] or "what is the shortest prefix string that we can condition GPT3 on for it to become a good task-specific model?"

7 Related Work

Multi-task learning Multi-task learning (Caruana, 1997), broadly construed, aims to learn models and representations that can be utilized across a diverse range of tasks, and offers a natural approach to training parameter-efficient deep models. Several works have shown that a single BERT model can obtain good performance across multiple tasks when jointly trained (Liu et al., 2019b; Clark et al., 2019; Stickland & Murray, 2019). An alternative approach to multi-task learning that does not require access to all tasks during training involve training smaller task-specific layers that interact with a fixed pretrained model (Rebuffi et al., 2018; Zhang et al., 2020a). In particular, adapter layers (Rebuffi et al., 2018), which learn to read and write to layers of a shared model, have been applied to obtain parameter-efficient BERT models (Houlsby et al., 2019; Pfeiffer et al., 2020a;b;c). A related line of work targets extreme parameter-efficiency through task-agnostic sentence representations that can be used without finetuning for downstream tasks (Le & Mikolov, 2014; Kiros et al., 2015; Wieting et al., 2016; Hill et al., 2016; Arora et al., 2017; Conneau et al., 2017; Cer et al., 2018; Zhang et al., 2018; Subramanian et al., 2018; Reimers & Gurevych, 2019; Zhang et al., 2020b). These feature-based transfer learning methods are however generally outperformed by fully finetuned models (Howard & Ruder, 2018).

[13] Dataset distillation (Wang et al., 2018) tackles this question in the context of vision models.

Model compression There has been much recent work on compressing pretrained trained with self-supervision (see Ganesh et al. (2020) for a recent survey). A particularly promising line of work focuses on obtaining smaller pretrained models (for subsequent finetuning) through weight pruning (Gordon et al., 2020; Sajjad et al., 2020; Chen et al., 2020) and/or knowledge distillation (Sanh et al., 2019; Sun et al., 2019; Turc et al., 2019; Jiao et al., 2019; Sun et al., 2020b). It would be interesting to see whether our approach can be applied on top of these smaller pretrained models to for even greater parameter-efficiency.

Learning to mask Our work is closely related to the line of work on learning to mask parts of deep networks with differentiable relaxations of binary masks for model pruning and parameter sharing (Wang et al., 2019b; Zhao et al., 2020; Sanh et al., 2020; Radiya-Dixit & Wang, 2020; Mallya et al., 2018; Guo et al., 2019; Sun et al., 2020a). While these works also enable parameter-efficient transfer learning, they generally apply the masks directly on the pretrained parameters instead of on the difference vector as in the present work.

Regularization towards pretrained models Finally, diff pruning is also related to works which regularize the learning process towards pretrained/shared models for continual learning (Rusu et al., 2016; Kirkpatrick et al., 2017; Schwarz et al., 2018), domain adaptation (Wiese et al., 2017; Miceli Barone et al., 2017), and stable finetuning (Lee et al., 2020). These works typically do not utilize sparse regularizers and target a different goal than parameter-efficiency.

8 Conclusion

We propose diff pruning as a simple approach for parameter-efficient transfer learning with pretrained models. Experiments on standard NLP benchmarks and models show that diff pruning can match the performance of fully finetuned baselines while requiring only a few additional parameters per task. We also propose a structured variant of diff pruning which provides further improvements. Avenues for future work include (i) applying this approach to other architectures (e.g. ConvNets for vision applications), (ii) injecting parameter-efficiency objectives directly into the pretraining process (to pretrain models that are better suited towards sparse transfer learning), and (iii) combining diff pruning with other techniques (e.g. adapters) to achieve even greater parameter-efficiency.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable feedback on the initial draft. AMR was supported by NSF 1704834 and NSF Career 2037519.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings . In *Proceedings of ICLR*, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. 2020.

Rich Caruana. Multitask Learning. *Machine Learning*, 1997.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of EMNLP: System Demonstrations*, 2018.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. *arXiv:2007.12223*, 2020.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Proceedings of ACL*, 2019.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of EMNLP*, 2017.

Andrew Dai and Quoc V. Le. Semi-Supervised Sequence Learning. In *Proceedings of NIPS*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*, 2019.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic BERT Sentence Embedding. *arXiv:2007.01852*, 2020.

Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3, 1999.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *arXiv:2002.11985*, 2020.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. In *Proceedings of Rep4NLP 2020 Workshop at ACL 2020*, 2020.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. In *Proceedings of CVPR*, 2019.

Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *Proceedings of ICLR*, 2016.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of ACL*, 2016.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, and Mona Attariyanand Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*, 2019.

Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL*, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of ICLR*, 2017.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv:1909.10351*, 2019.

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. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 14:3521–3526, 2017.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-Thought Vectors. In *Proceedings of NeurIPS*, 2015.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *Proceedings of ICLR*, 2020.

Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of ICML*, 2014.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *Proceedings of ICLR*, 2020.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*. 2017.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of ACL*, 2019a.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of ACL*, 2019b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*, 2019c.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Proceedings of NeurIPS*, 2017.

Christos Louizos, Max Welling, Diederik P, and Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *Proceedings of ICLR*, 2018.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of ICLR*, 2017.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *Proceedings of ECCV*, 2018.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Proceedings of NeurIPS*. 2017.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of EMNLP*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*, 2013.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *arXiv:1802.07569*, 2018.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of NAACL*, 2018.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Ruckle, and Kyunghyun Cho amd Iryna Gurevych. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. *arXiv:2005.00247*, 2020a.

Jonas Pfeiffer, Andreas Ruckle, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych Kyunghyun Cho. AdapterHub: A Framework for Adapting Transformers. *arXiv:2007.07779*, 2020b.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. *arXiv:2005.00052*, 2020c.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

Evani Radiya-Dixit and Xin Wang. How fine can fine-tuning be? Learning efficient language models. In *Proceedings of AISTATS*, 2020.

Colin Raffel, Noam Shazeer, Katherine Lee Adam Roberts, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21, 2020.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. *arXiv:1910.02054*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2016.

S. Rebuffi, A. Vedaldi, and H. Bilen. Efficient Parametrization of Multi-domain Deep Neural Networks. In *Proceedings of CVPR*, 2018.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP*, 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671*, 2016.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor Man's BERT: Smaller and Faster Transformer Models. *arXiv:2004.03844*, 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement Pruning: Adaptive Sparsity by Fine-Tuning. *arXiv:2005.07683*, 2020.

Timo Schick and Hinrich Schutze. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. *arXiv:2009.07118*, 2020.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. In *Proceedings of ICML*, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Proceedings of NeurIPS*. 2017.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv:1909.08053*, 2019.

Asa Cooper Stickland and Iain Murray. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of ICML*, 2019.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In *Proceedings of ICLR*, 2018.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient Knowledge Distillation for BERT Model Compression. In *Proceedings of EMNLP*, 2019.

Ximeng Sun, Rameswar Panda, and Rogerio Feris. AdaShare: Learning What To Share For Efficient Deep Multi-Task Learning. In *Proceedings of NeurIPS*, 2020a.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of ACL*, July 2020b.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of ACL*, 2019.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv:1908.08962*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*, 2019a.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset Distillation. *arXiv:1811.10959*, 2018.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured Pruning of Large Language Models. *arXiv:1910.04732*, 2019b.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. Neural domain adaptation for biomedical question answering. In *Proceedings of CoNLL*, August 2017.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards Universal Paraphrastic Sentence Embeddings. In *Proceedings of ICLR*, 2016.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

John M. Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Similarity Analysis of Contextual Word Representation Models. In *Proceedings of ACL*, 2020.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of NeurIPS*, 2019.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks. In *Proceedings of ECCV*, 2020a.

Minghua Zhang, Yunfang Wu, Weikang Li, and Wei Li. Learning universal sentence representations with mean-max attention autoencoder. In *Proceedings of EMNLP*, 2018.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. An Unsupervised Sentence Embedding Method byMutual Information Maximization. In *Proceedings of EMNLP*, 2020b.

Mengjie Zhao, Tao Lin, Martin Jaggi, and Hinrich Schutze. Masking as an Efficient Alternative to Finetuning for Pretrained Language Models. *arXiv:2004.12406*, 2020.

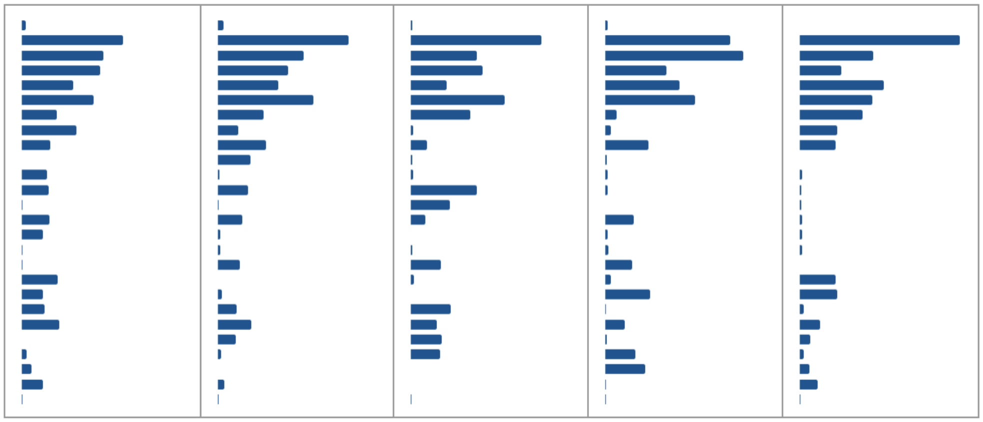


Figure 3: Percentage of modified parameters attributable to each layer for 5 different runs of SST-2 at 0.5% target sparsity. The layers are ordered from earlier to later (i.e. the embedding layer is shown at the top). The x-axis for each plot goes from 0% to 20%.

	QNLI	SST-2	MNLI_m	MNLI_{mm}	CoLA	MRPC	STS-B	RTE	QQP
Learning rate	2×10^{-5}	5×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	2×10^{-5}
Batch size	8	8	8	8	8	8	12	8	8
Training epochs	3	3	3	3	3	3	3	3	3
Finetuning epochs	3	5	3	3	3	5	5	5	3

Table 5: Best hyperparameters for the GLUE tasks based on the respective validation sets.

A Appendix

A.1 Consistency of Nonzero Parameters

Figure 3 shows the percentage of modified parameters attributable to each layer across 5 runs of SST-2. We find that there is nonotrivial variation in sparsity across runs, but also a degree of consistency. For example, the first layer is modified considerably more than other layers across all runs.

A.2 Hyperparameters

Table 5 shows hyperparameters we used for training GLUE tasks. For SQuAD v1.1 experiments, we ran distributed training across 8 GPUs, and used per gpu batch size 3, maximum sequence length 384, document stride 128, learning rate 3×10^{-5}, number of initial training epochs 2 and number of finetuning epochs 2.