Masking as an Efficient Alternative to Finetuning for Pretrained Language Models

Mengjie Zhao[†*], **Tao Lin**[‡*], **Fei Mi**[‡], **Martin Jaggi**[‡], **Hinrich Schütze**[†]
[†] LMU Munich, Germany [‡] EPFL, Switzerland
mzhao@cis.lmu.de, {tao.lin, fei.mi, martin.jaggi}@epfl.ch

Abstract

We present an efficient method of utilizing pretrained language models, where we learn selective binary masks for pretrained weights in lieu of modifying them through finetuning. Extensive evaluations of masking BERT, RoBERTa, and DistilBERT on eleven diverse NLP tasks show that our masking scheme yields performance comparable to finetuning, yet has a much smaller memory footprint when several tasks need to be inferred. Intrinsic evaluations show that representations computed by our binary masked language models encode information necessary for solving downstream tasks. Analyzing the loss landscape, we show that masking and finetuning produce models that reside in minima that can be connected by a line segment with nearly constant test accuracy. This confirms that masking can be utilized as an efficient alternative to finetuning.

1 Introduction

Finetuning a large pretrained language model like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019) often yields competitive or even state-of-the-art results on NLP benchmarks (Wang et al., 2018, 2019). Given an NLP task, standard finetuning stacks a linear layer on top of the pretrained language model and then updates all parameters using mini-batch SGD. Various aspects like brittleness (Dodge et al., 2020) and adaptiveness (Peters et al., 2019) of this two-stage transfer learning NLP paradigm (Dai and Le, 2015; Howard and Ruder, 2018) have been studied.

Despite the simplicity and impressive performance of finetuning, the prohibitively large number of parameters to be finetuned, e.g., 340 million in BERT-large, is a major obstacle to wider deployment of these models. The large memory footprint of finetuned models becomes more prominent when multiple tasks need to be solved – several copies of the millions of finetuned parameters have to be saved for inference.

Recent work (Gaier and Ha, 2019; Zhou et al., 2019) points out the potential of searching neural architectures within a fixed model, as an alternative to optimizing the model weights for downstream tasks. Inspired by these results, we present *masking*, a simple yet efficient scheme for utilizing pretrained language models. Instead of directly updating the pretrained parameters, we propose to *select* weights important to downstream NLP tasks while *discarding* irrelevant ones. The selection mechanism consists of a set of binary masks, one learned per downstream task through end-to-end training.

We show that masking, when being applied to pretrained language models like BERT, RoBERTa, and DistilBERT (Sanh et al., 2019), achieves performance comparable to finetuning in tasks like part-of-speech tagging, named-entity recognition, sequence classification, and reading comprehension. This is surprising in that a simple subselection mechanism that does not change any weights is competitive with a training regime – finetuning – that can change the value of every single weight. We conduct detailed analyses revealing important factors and possible reasons for the desirable performance of masking.

Masking is parameter-efficient: only a set of 1-bit binary masks needs to be saved per task after training, instead of all 32-bit float parameters in finetuning. This small memory footprint enables deploying pretrained language models for solving multiple tasks on edge devices. The compactness of masking also naturally allows parameter-efficient ensembles of pretrained language models.

Our **contributions**: (i) We introduce *masking*, a new scheme for utilizing pretrained language models by learning selective masks for pretrained weights, as an efficient alternative to finetuning.

* Equal contribution.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2226–2241,
November 16–20, 2020.

We show that masking is applicable to models like BERT/RoBERTa/DistilBERT, and produces performance on par with finetuning. (ii) We carry out extensive empirical analysis of masking, shedding light on factors critical for achieving good performance on eleven diverse NLP tasks. (iii) We study the binary masked language models' loss landscape and language representations, revealing potential reasons why masking has task performance comparable to finetuning.

2 Related Work

Two-stage NLP paradigm. Pretrained language models (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019b; Yang et al., 2019; Radford et al., 2019) advance NLP with contextualized representation of words. Finetuning a pretrained language model (Dai and Le, 2015; Howard and Ruder, 2018) often delivers competitive performance partly because pretraining leads to a better initialization across various downstream tasks than training from scratch (Hao et al., 2019). However, finetuning on individual NLP tasks is not parameter-efficient. Each finetuned model, typically consisting of hundreds of millions of floating point parameters, needs to be saved individually. Stickland and Murray (2019) use projected attention layers with multi-task learning to improve efficiency of finetuning BERT. Houlsby et al. (2019) insert adapter modules to BERT to improve memory efficiency. The inserted modules alter the forward pass of BERT, hence need to be carefully initialized to be close to identity.

We propose to directly pick parameters appropriate to a downstream task, by learning selective binary masks via end-to-end training. Keeping the pretrained parameters untouched, we solve several downstream NLP tasks with minimal overhead.

Binary networks and network pruning. Binary masks can be trained using the "straight-through estimator" (Bengio et al., 2013; Hinton, 2012). Hubara et al. (2016), Rastegari et al. (2016), Hubara et al. (2017), *inter alia*, apply this technique to train efficient binarized neural networks. We use this estimator to train selective masks for pretrained language model parameters.

Investigating the lottery ticket hypothesis (Frankle and Carbin, 2018) of network pruning (Han et al., 2015a; He et al., 2018; Liu et al., 2019c; Lee et al., 2019; Lin et al., 2020), Zhou et al. (2019) find that applying binary masks to a neural network is a form of training the network. Gaier and Ha (2019) propose to search neural architectures for reinforcement learning and image classification tasks, without any explicit weight training. This work inspires our masking scheme (which can be interpreted as implicit neural architecture search (Liu et al., 2019c)): applying the masks to a pretrained language model is similar to finetuning, yet is much more parameter-efficient.

Perhaps the closest work, Mallya et al. (2018) apply binary masks to CNNs and achieve good performance in computer vision. We learn selective binary masks for pretrained language models in NLP and shed light on factors important for obtaining good performance. Mallya et al. (2018) explicitly update weights in a task-specific classifier layer. In contrast, we show that end-to-end learning of selective masks, consistently for both the pretrained language model and a randomly initialized classifier layer, achieves good performance. Radiya-Dixit and Wang (2020) investigate finetuning of BERT by employing a number of techniques, including what they call sparsification, a method similar to masking. Their focus is analysis of finetuning BERT whereas our goal is to provide an efficient alternative to finetuning.

3 Method

3.1 Background on Transformer and finetuning

The encoder of the Transformer architecture (Vaswani et al., 2017) is ubiquitously used when pretraining large language models. We briefly review its architecture and then present our masking scheme. Taking BERT-base as an example, each one of the 12 transformer blocks consists of (i) four linear layers[1] $\mathbf{W}_K$, $\mathbf{W}_Q$, $\mathbf{W}_V$, and $\mathbf{W}_{AO}$ for computing and outputting the self attention among input wordpieces (Wu et al., 2016). (ii) two linear layers $\mathbf{W}_I$ and $\mathbf{W}_O$ feeding forward the word representations to the next transformer block.

More concretely, consider an input sentence $\mathbf{X} \in \mathbb{R}^{N \times d}$ where N is the maximum sentence length and d is the hidden dimension size. $\mathbf{W}_K$, $\mathbf{W}_Q$, and $\mathbf{W}_V$ are used to compute transformations of $\mathbf{X}$:

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

[1] We omit the bias terms for brevity.

and the self attention of $\mathbf{X}$ is computed as:

$$\text{Attention}(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}.$$

The attention is then transformed by $\mathbf{W}_{AO}$, and subsequently fed forward by $\mathbf{W}_I$ and $\mathbf{W}_O$ to the next transformer block.

When finetuning on a downstream task like sequence classification, a linear classifier layer $\mathbf{W}_T$, projecting from the hidden dimension to the output dimension, is randomly initialized. Next, $\mathbf{W}_T$ is stacked on top of a pretrained linear layer $\mathbf{W}_P$ (the *pooler layer*). All parameters are then updated to minimize the task loss such as cross-entropy.

3.2 Learning the mask

Given a pretrained language model, we do not finetune, i.e., we do not update the pretrained parameters. Instead, we *select* a subset of the pretrained parameters that is critical to a downstream task while *discarding* irrelevant ones with binary masks. We associate each linear layer $\mathbf{W}^l \in \{\mathbf{W}^l_K, \mathbf{W}^l_Q, \mathbf{W}^l_V, \mathbf{W}^l_{AO}, \mathbf{W}^l_I, \mathbf{W}^l_O\}$ of the l-th transformer block with a real-valued matrix $\mathbf{M}^l$ that is randomly initialized from a uniform distribution and has the same size as $\mathbf{W}^l$. We then pass $\mathbf{M}^l$ through an element-wise thresholding function (Hubara et al., 2016; Mallya et al., 2018), i.e., a binarizer, to obtain a binary mask $\mathbf{M}^l_{\text{bin}}$ for $\mathbf{W}^l$:

$$(m^l_{\text{bin}})_{i,j} = \begin{cases} 1 & \text{if } m^l_{i,j} \geq \tau \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

where $m^l_{i,j} \in \mathbf{M}^l$, i, j indicate the coordinates of the 2-D linear layer and τ is a global thresholding hyperparameter.

In each forward pass of training, the binary mask $\mathbf{M}^l_{\text{bin}}$ (derived from $\mathbf{M}^l$ via Eq. 1) selects weights in a pretrained linear layer $\mathbf{W}^l$ by Hadamard product:

$$\hat{\mathbf{W}}^l := \mathbf{W}^l \odot \mathbf{M}^l_{\text{bin}}.$$

In the corresponding backward pass of training, with the associated loss function $\mathcal{L}$, we cannot backpropagate through the binarizer, since Eq. 1 is a hard thresholding operation and the gradient with respect to $\mathbf{M}^l$ is zero almost everywhere. Similar to the treatment[2] in Bengio et al. (2013); Hubara et al. (2016); Lin et al. (2020), we use $\frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l_{\text{bin}}}$ as a noisy estimator of $\frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l}$ to update $\mathbf{M}^l$, i.e.:

$$\mathbf{M}^l \leftarrow \mathbf{M}^l - \eta \frac{\partial \mathcal{L}(\hat{\mathbf{W}}^l)}{\partial \mathbf{M}^l_{\text{bin}}}, \qquad (2)$$

where η refers to the step size. Hence, the whole structure can be trained end-to-end.

We learn a set of binary masks for an NLP task as follows. Recall that each linear layer $\mathbf{W}^l$ is associated with a $\mathbf{M}^l$ to obtain a masked linear layer $\hat{\mathbf{W}}^l$ through Eq. 1. We randomly initialize an additional linear layer with an associated $\mathbf{M}^l$ and stack it on top of the pretrained language model. We then update each $\mathbf{M}^l$ through Eq. 2 with the task objective during training.

After training, we pass each $\mathbf{M}^l$ through the binarizer to obtain $\mathbf{M}^l_{\text{bin}}$, which is then saved for future inference. Since $\mathbf{M}^l_{\text{bin}}$ is binary, it takes only $\approx 3\%$ of the memory compared to saving the 32-bit float parameters in a finetuned model. Also, we will show that many layers – in particular the embedding layer – do not have to be masked. This further reduces memory consumption of masking.

3.3 Configuration of masking

Our masking scheme is motivated by the observation: the pretrained weights form a good initialization (Hao et al., 2019), yet a few steps of adaptation are still needed to produce competitive performance for a specific task. However, not every pretrained parameter is necessary for achieving reasonable performance, as suggested by the field of neural network pruning (LeCun et al., 1990; Hassibi and Stork, 1993; Han et al., 2015b). We now investigate two configuration choices that affect how many parameters are "eligible" for masking.

Initial sparsity of $\mathbf{M}^l_{\text{bin}}$. As we randomly initialize our masks from uniform distributions, the sparsity of the binary mask $\mathbf{M}^l_{\text{bin}}$ in the mask initialization phase controls how many pretrained parameters in a layer $\mathbf{W}^l$ are assumed to be irrelevant to the downstream task. Different initial sparsity rates entail different optimization behaviors.

It is crucial to better understand how the initial sparsity of a mask impacts the training dynamics and final model performance, so as to generalize our masking scheme to broader domains and tasks. In §5.1, we investigate this aspect in detail. In practice, we fix τ in Eq. 1 while adjusting the uniform distribution to achieve a target initial sparsity.

[2] Bengio et al. (2013); Hubara et al. (2016) describe it as the "straight-through estimator", and Lin et al. (2020) provide convergence guarantee with error feedback interpretation.

Which layers to mask. Different layers of pretrained language models capture distinct aspects of a language during pretraining, e.g., Tenney et al. (2019) find that information on part-of-speech tagging, parsing, named-entity recognition, semantic roles, and coreference is encoded on progressively higher layers of BERT. It is hard to know a priori which types of NLP tasks have to be addressed in the future, making it non-trivial to decide layers to mask. We study this factor in §5.2.

We do not learn a mask for the lowest embedding layer, i.e., the uncontextualized wordpiece embeddings are completely "selected", for all tasks. The motivation is two-fold. (i) The embedding layer weights take up a large part, e.g., almost 21% (23m/109m) in BERT-base-uncased, of the total number of parameters. Not having to learn a selective mask for this layer reduces memory consumption. (ii) Pretraining has effectively encoded context-independent general meanings of words in the embedding layer (Zhao et al., 2020). Hence, learning a selective mask for this layer is unnecessary. Also, we do not learn masks for biases and layer normalization parameters as we did not observe a positive effect on performance.

4 Datasets and Setup

Datasets. We present results for masking BERT, RoBERTa, and DistilBERT in part-of-speech tagging, named-entity recognition, sequence classification, and reading comprehension.

We experiment with **part-of-speech tagging** (POS) on Penn Treebank (Marcus et al., 1993), using Collins (2002)'s train/dev/test split. For **named-entity recognition** (NER), we conduct experiments on the CoNLL-2003 NER shared task (Tjong Kim Sang and De Meulder, 2003).

For **sequence classification**, the following GLUE tasks (Wang et al., 2018) are evaluated: Stanford Sentiment Treebank (SST2) (Socher et al., 2013), Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019), Recognizing Textual Entailment (RTE) (Dagan et al., 2005), and Question Natural Language Inference (QNLI) (Rajpurkar et al., 2016).

In addition, we experiment on sequence classification datasets that have publicly available test sets: the 6-class question classification dataset TREC (Voorhees and Tice, 2000), the 4-class news classification dataset AG News (AG) (Zhang et al., 2015), and the binary Twitter sentiment classification task SemEval-2016 4B (SEM) (Nakov et al., 2016).

We experiment with **reading comprehension** on SWAG (Zellers et al., 2018) using the official data splits. We report Matthew's correlation coefficient (MCC) for CoLA, micro-F1 for NER, and accuracy for the other tasks.

Setup. Due to resource limitations and in the spirit of environmental responsibility (Strubell et al., 2019; Schwartz et al., 2019), we conduct our experiments on the base models: BERT-base-uncased, RoBERTa-base, and DistilBERT-base-uncased. Thus, the BERT/RoBERTa models we use have 12 transformer blocks (0–11 indexed) producing 768-dimension vectors; the DistilBERT model we use has the same dimension but contains 6 transformer blocks (0–5 indexed). We implement our models in PyTorch (Paszke et al., 2019) with the HuggingFace framework (Wolf et al., 2019).

Throughout all experiments, we limit the maximum length of a sentence (pair) to be 128 after wordpiece tokenization. Following Devlin et al. (2019), we use the Adam (Kingma and Ba, 2014) optimizer of which the learning rate is a hyperparameter while the other parameters remain default. We carefully tune the learning rate for each setup: the tuning procedure ensures that the best learning rate does not lie on the border of our search grid, otherwise we extend the grid accordingly. The initial grid is {1e-5, 3e-5, 5e-5, 7e-5, 9e-5}.

For sequence classification and reading comprehension, we use `[CLS]` as the representation of the sentence (pair). Following Devlin et al. (2019), we formulate NER as a tagging task and use a linear output layer, instead of a conditional random field layer. For POS and NER experiments, the representation of a tokenized word is its last wordpiece (Liu et al., 2019a; He and Choi, 2020). Note that a 128 maximum length of a sentence for POS and NER means that some word-tag annotations need to be excluded. Appendix §A shows our reproducibility checklist containing more implementation and preprocessing details.

5 Experiments

5.1 Initial sparsity of binary masks

We first investigate how initial sparsity percentage (i.e., fraction of zeros) of the binary mask $\mathbf{M}^l_{\text{bin}}$ influences performance of a binary masked language model on downstream tasks. We experiment on four tasks, with initial sparsities in {1%, 3%, 5%,

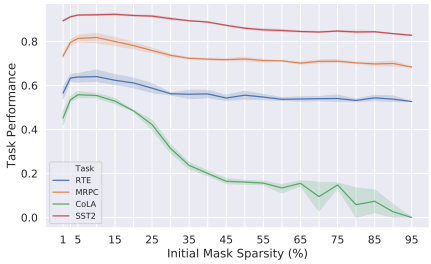


Figure 1: Dev set performance of masking BERT when selecting different amounts of pretrained parameters.

10%, 15%, 20%, . . . , 95%}. All other hyperparameters are controlled: learning rate is fixed to 5e-5; batch size is 32 for relatively small datasets (RTE, MRPC, and CoLA) and 128 for SST2. Each experiment is repeated four times with different random seeds {1, 2, 3, 4}. In this experiment, all transformer blocks, the pooler layer, and the classifier layer are masked.

Figure 1 shows that masking achieves decent performance without hyperparameter search. Specifically, (i) a large initial sparsity removing most pretrained parameters, e.g., 95%, leads to bad performance for the four tasks. This is due to the fact that the pretrained knowledge is largely discarded. (ii) Gradually decreasing the initial sparsity improves task performance. Generally, an initial sparsity in 3% $\sim$ 10% yields reasonable results across tasks. Large datasets like SST2 are less sensitive than small datasets like RTE. (iii) Selecting almost all pretrained parameters, e.g., 1% sparsity, hurts task performance. Recall that a pretrained model needs to be adapted to a downstream task; masking achieves adaptation by learning selective masks – preserving too many pretrained parameters in initialization impedes the optimization.

5.2 Layer-wise behaviors

Neural network layers present heterogeneous characteristics (Zhang et al., 2019) when being applied to tasks. For example, syntactic information is better represented at lower layers while semantic information is captured at higher layers in ELMo (Peters et al., 2018). As a result, simply masking all transformer blocks (as in §5.1) may not be ideal.

We investigate the task performance when applying the masks to different BERT layers. Figure 2 presents the optimal task performance when masking only a subset of BERT's transformer blocks on MRPC, CoLA, and RTE. Different amounts and indices of transformer blocks are masked: "bottom-up" and "top-down" indicate to mask the targeted amount of transformer blocks, either from bottom or top of BERT.

We can observe that (i) in most cases, top-down masking outperforms bottom-up masking when initial sparsity and the number of masked layers are fixed. Thus, it is reasonable to select all pretrained weights in lower layers, since they capture general information helpful and transferable to various tasks (Liu et al., 2019a; Howard and Ruder, 2018). (ii) For bottom-up masking, increasing the number of masked layers gradually improves performance. This observation illustrates dependencies between BERT layers and the learning dynamics of masking: provided with selected pretrained weights in lower layers, higher layers need to be given flexibility to select pretrained weights accordingly to achieve good task performance. (iii) In top-down masking, CoLA performance increases when masking a growing number of layers while MRPC and RTE are not sensitive. Recall that CoLA tests linguistic acceptability that typically requires both syntactic and semantic information[3]. All of BERT layers are involved in representing this information, hence allowing more layers to change should improve performance.

5.3 Comparing finetuning and masking

We have investigated two factors – initial sparsity (§5.1) and layer-wise behaviors (§5.2) – that are important in masking pretrained language models. Here, we compare the performance and memory consumption of masking and finetuning.

Based on observations in §5.1 and §5.2, we use 5% initial sparsity when applying masking to BERT, RoBERTa, and DistilBERT. We mask the transformer blocks 2–11 in BERT/RoBERTa and 2–5 in DistilBERT. $\mathbf{W}_P$ and $\mathbf{W}_T$ are always masked. Note that this global setup is surely suboptimal for some model-task combinations, but our goal is to illustrate the effectiveness and the generalization ability of masking. Hence, conducting extensive hyperparameter search is unnecessary.

For AG and QNLI, we use batch size 128. For the other tasks we use batch size 32. We search the optimal learning rate per task as described in §4,

[3]For example, to distinguish acceptable caused-motion constructions (e.g., "the professor talked us into a stupor") from inacceptable ones (e.g., "water talked it into red"), both syntactic and semantic information need to be considered (Goldberg, 1995).

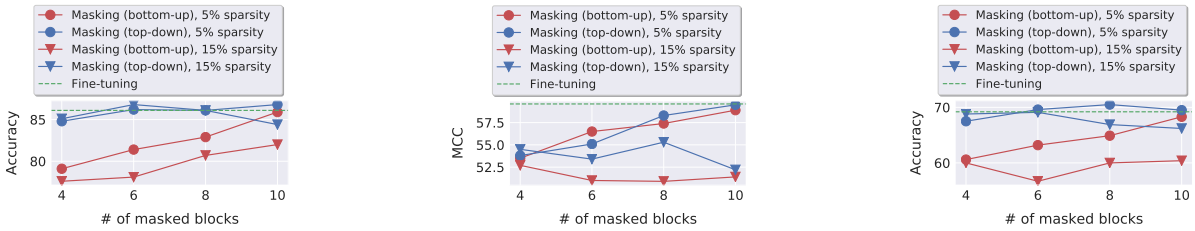


Figure 2: The impact of masking different transformer blocks of BERT for MRPC (left), CoLA (middle), and RTE (right). The number of masked blocks is shown on the x-axis; that number is either masked "bottom-up" or "top-down". More precisely, a bottom-up setup (red) masking 4 blocks means we mask the transformer blocks $\{0, 1, 2, 3\}$; a top-down setup (blue) masking 4 blocks means we mask the transformer blocks $\{8, 9, 10, 11\}$. $\mathbf{W}_P$ and $\mathbf{W}_T$ are always masked.

		MRPC 3.5k	SST2 67k	CoLA 8.5k	RTE 2.5k	QNLI 108k	SEM 4.3k	TREC 4.9k	AG 96k	POS 38k	NER 15k	SWAG 113k
BERT	Finetuning	86.1 ± 0.8	93.3 ± 0.2	59.6 ± 0.8	69.2 ± 2.7	91.0 ± 0.6	86.6 ± 0.3	96.4 ± 0.2	94.4 ± 0.1	97.7 ± 0.0	94.6 ± 0.2	80.9 ± 1.7
	Masking	86.8 ± 1.1	93.2 ± 0.5	59.5 ± 0.1	69.5 ± 3.0	91.3 ± 0.4	85.9 ± 0.5	96.0 ± 0.4	94.2 ± 0.0	97.7 ± 0.0	94.5 ± 0.1	80.3 ± 0.1
RoBERTa	Finetuning	89.8 ± 0.5	95.0 ± 0.3	62.1 ± 1.7	78.2 ± 1.1	92.9 ± 0.2	90.2 ± 0.5	96.2 ± 0.4	94.7 ± 0.0	98.1 ± 0.0	94.9 ± 0.1	83.4 ± 0.8
	Masking	88.5 ± 1.1	94.5 ± 0.3	60.3 ± 1.3	69.2 ± 2.1	92.4 ± 0.1	90.1 ± 0.1	95.9 ± 0.5	94.5 ± 0.1	98.0 ± 0.0	93.9 ± 0.1	82.1 ± 0.2
DistilBERT	Finetuning	85.4 ± 0.5	91.6 ± 0.4	55.1 ± 0.3	62.2 ± 3.0	89.0 ± 0.8	85.9 ± 0.2	95.7 ± 0.6	94.2 ± 0.1	97.6 ± 0.0	94.1 ± 0.1	72.5 ± 0.2
	Masking	86.0 ± 0.3	91.3 ± 0.3	53.1 ± 0.7	61.6 ± 1.5	89.2 ± 0.2	86.6 ± 0.6	95.9 ± 0.6	94.2 ± 0.1	97.6 ± 0.0	94.1 ± 0.2	71.0 ± 0.0

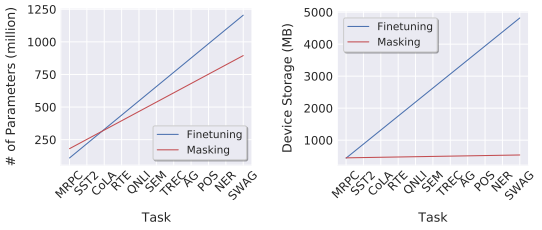
Table 1: Dev set task performances (%) of masking and finetuning. Each experiment is repeated four times with different random seeds and we report mean and standard deviation. Numbers below dataset name (second row) are the size of training set. For POS and NER, we report the number of sentences.

and they are shown in Appendix §A.4.

Performance comparison. Table 1 reports performance of masking and finetuning on the dev set for the eleven NLP tasks. We observe that applying masking to BERT/RoBERTa/DistilBERT yields performance comparable to finetuning. We observe a performance drop[4] on RoBERTa-RTE. RTE has the smallest dataset size (train: 2.5k; dev: 0.3k) among all tasks – this may contribute to the imperfect results and large variances.

Our BERT-NER results are slightly worse than Devlin et al. (2019). This may be due to the fact that "maximal document context" is used by Devlin et al. (2019) while we use sentence-level context of 128 maximum sequence length[5].

Rows "Single" in Table 2 compare performance of masking and finetuning BERT on the test set of SEM, TREC, AG, POS, and NER. The same setup and hyperparameter searching as Table 1 are used, the best hyperparameters are picked on the dev set. Results from Sun et al. (2019); Palogiannidi et al. (2016) are included as a reference. Sun et al. (2019) employ optimizations like layer-wise learning rate, producing slightly better performance than ours. Palogiannidi et al. (2016) is the best performing system on task SEM (Nakov et al., 2016). Again, masking yields results comparable to finetuning.



(a) Number of parameters. (b) Memory consumption.

Figure 3: The accumulated number of parameters and memory required by finetuning and masking to solve an increasing number of tasks.

Memory comparison. Having shown that task performance of masking and finetuning is comparable, we next demonstrate one key strength of masking: memory efficiency. We take BERT-base-uncased as our example. Figure 3 shows the accumulated number of parameters in million and memory in megabytes (MB) required when an increasing number of downstream tasks need to be solved using finetuning and masking. Masking re-

[4]Similar observations were made: DistilBERT has a 10% accuracy drop on RTE compared to BERT-base (Sanh et al., 2019); Sajjad et al. (2020) report unstableness on MRPC and RTE when applying their model reduction strategies.

[5]Similar observations were made: https://github.com/huggingface/transformers/issues/64

		SEM	TREC	AG	POS	NER	Memory (MB)
Masking	Single	12.03	3.30	5.62	2.34	9.85	447
	Ensem.	11.52	3.20	5.28	2.12	9.19	474
Finetun.	Single	11.87	3.80	5.66	2.34	9.85	438
	Ensem.	11.73	2.80	5.17	2.29	9.23	1752
Sun et al. (2019)		n/a	2.80	5.25	n/a	n/a	n/a
Palogiannidi et al. (2016)		13.80	n/a	n/a	n/a	n/a	n/a

Table 2: Error rate (%) on test set and model size comparison. Single: the averaged performance of four models with different random seeds. Ensem.: ensemble of the four models.

quires a small overhead when solving a single task but is much more efficient than finetuning when several tasks need to be inferred. Masking saves a single copy of a pretrained language model containing 32-bit float parameters for all the eleven tasks and a set of 1-bit binary masks for each task. In contrast, finetuning saves every finetuned model so the memory consumption grows linearly.

Masking naturally allows light ensembles of models. Rows "Ensem." in Table 2 compare ensembled results and model size. We consider the ensemble of predicted (i) labels; (ii) logits; (iii) probabilities. The best ensemble method is picked on dev and then evaluated on test. Masking only consumes 474MB of memory – much smaller than 1752MB required by finetuning – and achieves comparable performance. Thus, masking is also much more memory-efficient than finetuning in an ensemble setting.

6 Discussion

6.1 Intrinsic evaluations

§5 demonstrates that masking is an efficient alternative to finetuning. Now we analyze properties of the representations computed by binary masked language models with intrinsic evaluation.

One intriguing property of finetuning, i.e., stacking a classifier layer on top of a pretrained language model then update all parameters, is that a linear classifier layer suffices to conduct reasonably accurate classification. This observation implies that the configuration of data points, e.g., sentences with positive or negative sentiment in SST2, should be close to linearly separable in the hidden space. Like finetuning, masking also uses a linear classifier layer. Hence, we hypothesize that upper layers in binary masked language models, even without explicit weight updating, also create a hidden space in which data points are close to linearly separable.

Figure 4 uses t-SNE (Maaten and Hinton, 2008)

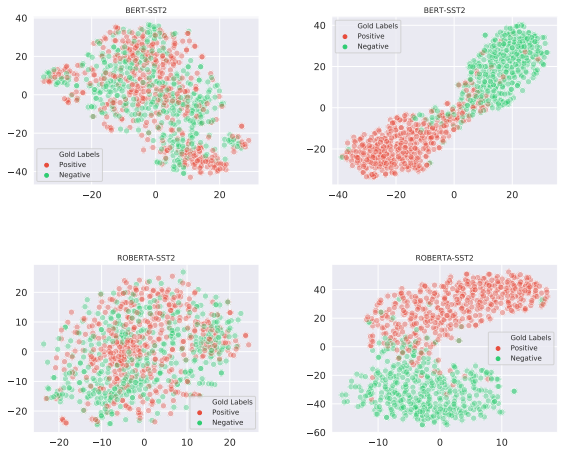


Figure 4: t-SNE visualization of the representation of `[CLS]` computed by the topmost transformer block in pretrained (left), finetuned (top right), and masked (bottom right) BERT/RoBERTa. We use `scikit-learn` (Pedregosa et al., 2011) and default t-SNE parameters.

	SST2	SEM
SST2	41.8	-13.4
SEM	20.0	11.5

(a) Masking

	SST2	SEM
SST2	41.8	-10.1
SEM	18.9	12.2

(b) Finetuning

Table 3: Generalization on dev (%) of binary masked and finetuned BERT. Row: training dataset; Column: evaluating dataset. Numbers are improvements against the majority-vote baseline: 50.9 for SST2 and 74.4 for SEM. Results are averaged across four random seeds.

to visualize the representation of `[CLS]` computed by the topmost transformer block in pretrained, finetuned, and masked BERT/RoBERTa, using the dev set examples of SST2. The pretrained models' representations (left) are clearly not separable since the model needs to be adapted to downstream tasks. The sentence representations computed by the finetuned (top right) and the binary masked (bottom right) encoder are almost linearly separable and consistent with the gold labels. Thus, a linear classifier is expected to yield reasonably good classification accuracy. This intrinsic evaluation illustrates that binary masked models extract good representations from the data for the downstream NLP task.

6.2 Properties of the binary masked models

Do binary masked models generalize? Figure 4 shows that a binary masked language model produces proper representations for the classifier layer and hence performs as well as a finetuned model. Here, we are interested in verifying that

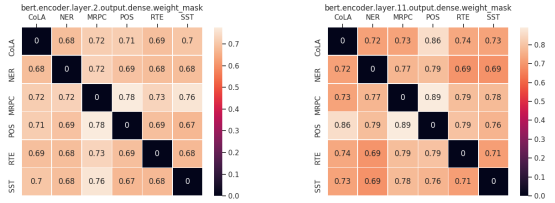


Figure 5: Scores s of two sets of masks, trained with two different tasks, of layer $\mathbf{W}_O$ in transformer blocks 2 (left) and 11 (right) in BERT. A large s means that the two masks are dissimilar.

the binary masked model does indeed solve downstream tasks by learning meaningful representations – instead of exploiting spurious correlations that generalize poorly (Niven and Kao, 2019; McCoy et al., 2019). To this end, we test if the binary masked mode is generalizable to other datasets of the same type of downstream task. We use the two sentiment classification datasets: SST2 and SEM. We simply evaluate the model masked or finetuned on SST2 against the dev set of SEM and vice versa. Table 3 reports the results against the majority-vote baseline. The finetuned and binary masked models of SEM generalize well on SST2, showing $\approx 20\%$ improvement against the majority-vote baseline.

On the other hand, we observe that the knowledge learned on SST2 does not generalize to SEM, for both finetuning and masking. We hypothesize that this is because the Twitter domain (SEM) is much more specific than movie reviews (SST2). For example, some Emojis or symbols like ":)" reflecting strong sentiment do not occur in SST2, resulting in unsuccessful generalization. To test our hypothesis, we take another movie review dataset IMDB (Maas et al., 2011), and directly apply the SST2-finetuned- and SST2-binary-masked- models on it. Masking and finetuning achieve accuracy 84.79% and 85.25%, which are comparable and both outperform the baseline 50%, demonstrating successful knowledge transfer.

Thus, finetuning and masking yield models with similar generalization ability. The binary masked models indeed create representations that contain valid information for downstream tasks.

Analyzing masks. We study the dissimilarity between masks learned by different BERT layers and downstream tasks. For the initial and trained binary masks $\mathbf{M}_{\text{bin}}^{t,init}$ and $\mathbf{M}_{\text{bin}}^{t,trained}$ of a layer trained on task $t \in \{t1, t2\}$. We compute:

$$s = \frac{\left\|\mathbf{M}_{\text{bin}}^{t1,trained} - \mathbf{M}_{\text{bin}}^{t2,trained}\right\|_1}{\left\|\mathbf{M}_{\text{bin}}^{t1,trained} - \mathbf{M}_{\text{bin}}^{t1,init}\right\|_1 + \left\|\mathbf{M}_{\text{bin}}^{t2,trained} - \mathbf{M}_{\text{bin}}^{t2,init}\right\|_1},$$

where $\|\mathbf{W}\|_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}|w_{i,j}|$. Note that for the same random seed, $\mathbf{M}_{\text{bin}}^{t1,init}$ and $\mathbf{M}_{\text{bin}}^{t2,init}$ are the same. The dissimilarity s measures the difference between two masks as a fraction of all changes brought about by training. Figure 5 shows that, after training, the dissimilarities of masks of higher BERT layers are larger than those of lower BERT layers. Similar observations are made for finetuning: top layer weights in finetuned BERT are more task-specific (Kovaleva et al., 2019). The figure also shows that the learned masks for downstream tasks tend to be dissimilar to each other, even for similar tasks. For a given task, there exist different sets of masks (initialized with different random seeds) yielding similar performance. This observation is similar to the results of evaluating the lottery ticket hypothesis on BERT (Prasanna et al., 2020; Chen et al., 2020): a number of subnetworks exist in BERT achieving similar task performance.

6.3 Loss landscape

Training complex neural networks can be viewed as searching for good minima in the highly non-convex landscape defined by the loss function (Li et al., 2018). Good minima are typically depicted as points at the bottom of different locally convex valleys (Keskar et al., 2016; Draxler et al., 2018), achieving similar performance. In this section, we study the relationship between the two minima obtained by masking and finetuning.

Recent work analyzing the loss landscape suggests that the local minima in the loss landscape reached by standard training algorithms can be connected by a simple path (Garipov et al., 2018; Gotmare et al., 2018), e.g., a Bézier curve, with low task loss (or high task accuracy) along the path. We are interested in testing if the two minima found by finetuning and masking can be easily connected in the loss landscape. To start with, we verify the task performance of an interpolated model $\mathbf{W}(\gamma)$ on the line segment between a finetuned model $\mathbf{W}_0$ and a binary masked model $\mathbf{W}_1$:

$$\mathbf{W}(\gamma) = \mathbf{W}_0 + \gamma(\mathbf{W}_1 - \mathbf{W}_0), 0 \leq \gamma \leq 1\,.$$

We conduct experiments on MRPC and SST2 with the best-performing BERT and RoBERTa

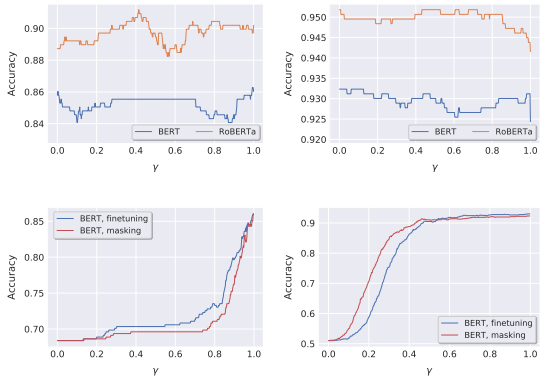


Figure 6: Mode connectivity results on MRPC (left) and SST2 (right). Top images: dev set accuracy of an interpolated model between the two minima found by finetuning ($\gamma = 0$) and masking ($\gamma = 1$). Bottom images: accuracy of an interpolated model between pretrained ($\gamma = 0$) and finetuned/masked ($\gamma = 1$) BERT.

models obtained in Table 1 (same seed and training epochs); Figure 6 (top) shows the results of mode connectivity, i.e., the evolution of the task accuracy along a line connecting the two candidate minima.

Surprisingly, the interpolated models on the line segment connecting a finetuned and a binary masked model form a high accuracy path, indicating the extremely well-connected loss landscape. Thus, masking finds minima on the same connected low-loss manifold as finetuning, confirming the effectiveness of our method. Also, we show in Figure 6 (bottom) for the line segment between the pretrained BERT and a finetuned/masked BERT, that mode connectivity is not solely due to an overparameterized pretrained language model. Bézier curves experiments show similar results, cf. Appendix §B.

7 Conclusion

We have presented masking, an efficient alternative to finetuning for utilizing pretrained language models like BERT/RoBERTa/DistilBERT. Instead of updating the pretrained parameters, we only train one set of binary masks per task to select critical parameters. Extensive experiments show that masking yields performance comparable to finetuning on a series of NLP tasks. Leaving the pretrained parameters unchanged, masking is much more memory efficient when several tasks need to be solved. Intrinsic evaluations show that binary masked models extract valid and generalizable representations for downstream tasks. Moreover, we demonstrate that the minima obtained by finetuning and masking can be easily connected by a line segment, confirming the effectiveness of applying masking to pretrained language models. Our code is available at: `https://github.com/ptlmasking/maskbert`.

Future work may explore the possibility of applying masking to the pretrained multilingual encoders like mBERT (Devlin et al., 2019) and XLM (Conneau and Lample, 2019). Also, the binary masks learned by our method have low sparsity such that inference speed is not improved. Developing methods improving both memory and inference efficiency without sacrificing task performance can open the possibility of widely deploying the powerful pretrained language models to more NLP applications.

Acknowledgments

We thank the anonymous reviewers for the insightful comments and suggestions. This work was funded by the European Research Council (ERC #740516), SNSF grant 200021_175796, as well as a Google Focused Research Award.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

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

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors,

Advances in Neural Information Processing Systems 28, pages 3079–3087. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping.

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

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. 2018. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Adam Gaier and David Ha. 2019. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, pages 5365–5379.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798.

Adele E Goldberg. 1995. *Construction grammar*. Wiley.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015a. Learning both weights and connections for efficient neural network. In *NeurIPS - Advances in Neural Information Processing Systems*, pages 1135–1143.

Song Han, Jeff Pool, John Tran, and William Dally. 2015b. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Babak Hassibi and David G. Stork. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. Morgan-Kaufmann.

Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20. Best Paper Candidate.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2234–2240.

Geoffrey Hinton. 2012. Neural networks for machine learning.

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. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4356–4365, Hong Kong, China. Association for Computational Linguistics.

Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *ICLR - International Conference on Learning Representations*.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. 2020. Dynamic model pruning with feedback. In *International Conference on Learning Representations*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019c. Rethinking the value of network pruning. In *ICLR - International Conference on Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *The European Conference on Computer Vision (ECCV)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Haris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 task 4: Sentiment analysis in twitter using semantic-affective model adaptation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 155–163, San Diego, California. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT plays the lottery, all tickets are winning.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? learning efficient language models. volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443, Online. PMLR.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green ai.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, Long Beach, California, USA. PMLR.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ellen Voorhees and Dawn Tice. 2000. The trec-8 question answering track evaluation. *Proceedings of the 8th Text Retrieval Conference*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2019. Are all layers created equal? *arXiv preprint arXiv:1902.01996*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Mengjie Zhao, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2020. Quantifying the contextualization of word representations with semantic class probing. In *Findings of EMNLP*.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pages 3592–3602.

A Reproducibility Checklist

A.1 Computing infrastructure

All experiments are conducted on following GPU models: Tesla V100, GeForce GTX 1080 Ti, and GeForce GTX 1080. We use per-GPU batch size 32. Thus, experiments comparing masking and finetuning on QNLI and AG take 4 GPUs and all the other tasks use a single GPU.

A.2 Number of parameters

In §5.3 we thoroughly compare the number of parameters and memory consumption of finetuning and masking. Numerical values are in Table 8.

A.3 Validation performance

The dev set performance of Table 2 is covered in Table 1. We report Matthew's correlation coefficient (MCC) for CoLA, micro-F1 for NER, and accuracy for the other tasks. We use the evaluation functions in `scikit-learn` (Pedregosa et al., 2011) and `seqeval` (https://github.com/chakki-works/seqeval).

A.4 Hyperparameter search

The only hyperparameter we searched is learning rate, for both masking and finetuning, according to the setup discussion in §4. The optimal values are in Table 4.

A.5 Datasets

For GLUE tasks, we use the official datasets from the benchmark https://gluebenchmark.com/. For TREC and AG, we download the datasets developed by Zhang et al. (2015), which are available at here. Note that this link is provided by Zhang et al. (2015) and also used by Sun et al. (2019). For SEM, we obtain the dataset from the official SemEval website: http://alt.qcri.org/semeval2016/task4/. For NER, we use the official dataset: https://www.clips.uantwerpen.be/conll2003/ner/. We obtain our POS dataset from the linguistic data consortium (LDC). We use the official dataset of SWAG (Zellers et al., 2018): https://github.com/rowanz/swagaf/tree/master/data.

For POS, sections 0-18 of WSJ are train, sections 19-21 are dev, and sections 22-24 are test (Collins, 2002). We use the official train/dev/test splits of all the other datasets.

To preprocess the datasets, we use the tokenizers provided by the `Transformers` package (Wolf et al., 2019) to convert the raw dataset to the formats required by BERT/RoBERTa/DistilBERT. Since wordpiece tokenization is used, there is no out-of-vocabulary words.

Since we use a maximum sequence length of 128, our preprocessing steps exclude some word-tag annotations in POS and NER. For POS, after wordpiece tokenization, we see 1 sentence in dev and 2 sentences in test have more than 126 (the `[CLS]` and `[SEP]` need to be considered) wordpieces. As a result, we exclude 5 annotated words in dev and 87 annotated words in test. Similarly, for NER (which is also formulated as a tagging task following Devlin et al. (2019)), we see 3 sentences in dev and 1 sentence in test have more than 126 wordpieces. As a result, we exclude 27 annotated words in dev and 8 annotated words in test.

The number of examples in dev and test per task is shown in following Table 5.

B More on Mode Connectivity

Following the mode connectivity framework proposed in Garipov et al. (2018), we parameterize the path joining two minima using a Bézier curve. Let $\mathbf{w}_0$ and $\mathbf{w}_{n+1}$ be the parameters of the models trained from finetuning and masking. Then, an n-bend Bézier curve connecting $\mathbf{w}_0$ and $\mathbf{w}_{n+1}$, with n trainable intermediate models $\theta = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$, can be represented by $\phi_\theta(t)$, such that $\phi_\theta(0) = \mathbf{w}_0$ and $\phi_\theta(1) = \mathbf{w}_{n+1}$, and

$$\phi_\theta(t) = \sum_{i=0}^{n+1} \binom{n+1}{i} (1-t)^{n+1-i} t^i \mathbf{w}_i \,.$$

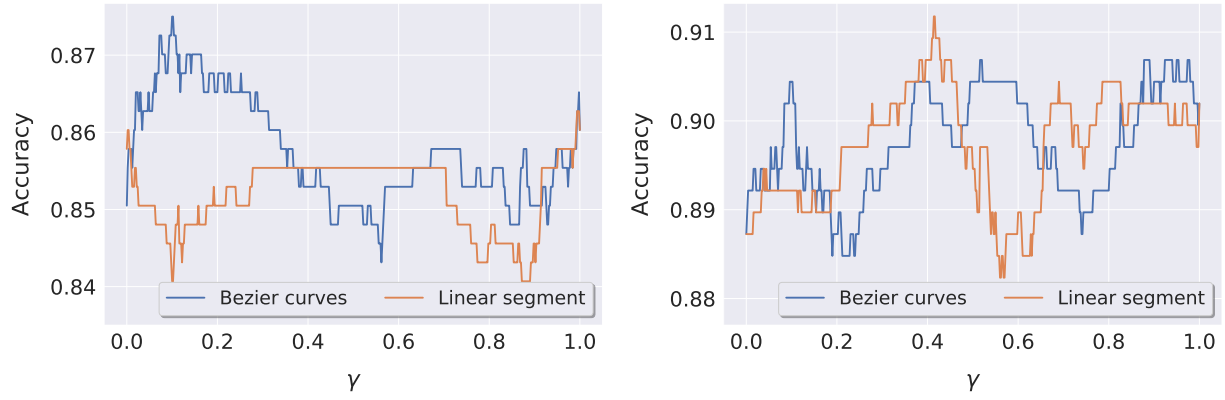
We train a 3-bend Bézier curve by minimizing the loss $\mathbb{E}_{t\sim U[0,1]} \mathcal{L}\left(\phi_\theta(t)\right)$, where $U[0,1]$ is the uniform distribution in the interval $[0,1]$. Monte Carlo method is used to estimate the gradient of this expectation-based function and gradient-based optimization is used for the minimization. The results are illustrated in Figure 7. Masking implicitly performs gradient descent, analogy to the weights update achieved by finetuning; the observations complement our arguments in the main text.

C More Empirical Results

Ensemble results of RoBERTa and DistilBERT. Following Table 6 shows the single and ensemble results of RoBERTa and DistilBERT on the test set of SEM, TREC, AG, POS, and NER.

		MRPC	SST2	CoLA	RTE	QNLI	POS	NER	SWAG	SEM	TREC	AG
BERT	Finetuning	5e-5	1e-5	3e-5	5e-5	3e-5	3e-5	3e-5	7e-5	1e-5	3e-5	3e-5
	Masking	1e-3	5e-4	9e-4	1e-3	7e-4	5e-4	7e-4	1e-4	7e-5	1e-4	5e-4
RoBERTa	Finetuning	3e-5	1e-5	1e-5	7e-6	1e-5	9e-6	3e-5	1e-5	7e-6	9e-6	3e-5
	Masking	3e-4	9e-5	3e-4	3e-4	1e-4	3e-4	3e-4	1e-4	3e-4	5e-4	5e-4
DistilBERT	Finetuning	3e-5	7e-5	3e-5	3e-5	3e-5	3e-5	1e-5	7e-6	1e-5	3e-5	3e-5
	Masking	9e-4	7e-4	9e-4	9e-4	1e-3	7e-4	7e-4	3e-4	3e-4	9e-4	1e-3

Table 4: The optimal learning rate on different tasks for BERT/RoBERTa/DistilBERT. We perform finetuning/masking on all tasks for 10 epochs with early stopping of 2 epochs.



(a) BERT

(b) RoBERTa

Figure 7: The accuracy on MRPC dev set, as a function of the point on the curves $\phi_\theta(\gamma)$, connecting the two minima found by finetuning (left, $\gamma=0$) and masking (right, $\gamma=1$).

	Dev	Test
MRPC	408	n/a
SST2	872	n/a
CoLA	1,042	n/a
RTE	277	n/a
QNLI	5,732	n/a
SEM	1,325	10,551
TREC	548	500
AG	24,000	7,600
POS	135,105	133,082
NER	51,341	46,425
SWAG	20,006	n/a

Table 5: Number of examples in dev and test per task. For POS and NER, we report the number of words.

			SEM	TREC	AG	POS	NER
RoBERTa	Masking	Single	11.12	3.15	5.06	2.11	11.03
		Ensem.	10.54	2.40	4.55	2.11	10.57
	Finetun.	Single	10.74	3.00	5.10	2.00	10.43
		Ensem.	10.74	2.60	4.50	1.96	9.54
DistilBERT	Masking	Single	11.89	3.70	5.71	2.39	10.40
		Ensem.	11.60	3.00	5.29	2.54	9.86
	Finetun.	Single	11.94	3.30	5.42	2.39	10.18
		Ensem.	11.48	3.00	4.84	2.29	9.74

Table 6: Error rate (%) on test set of tasks by RoBERTa and DistilBERT. Single: the averaged performance of four models with different random seeds. Ensem.: ensemble of the four models.

D Numerical Values of Plots

D.1 Layer-wise behaviors

Table 7 details the numerical values of Figure 2.

D.2 Memory consumption

Table 8 details the numerical values of Figure 3.

	MRPC	RTE	CoLA
Finetuning (BERT + classifier)	0.861 ± 0.008	0.692 ± 0.027	0.596 ± 0.015
Masking (BERT 00-11 + classifier, initial sparsity 5%)	0.862 ± 0.015	0.673 ± 0.036	0.592 ± 0.004
Masking (BERT 00-11 + classifier, initial sparsity 15%)	0.825 ± 0.039	0.626 ± 0.040	0.522 ± 0.027
Masking (BERT 02-11 + classifier, initial sparsity 5%)	0.868 ± 0.011	0.695 ± 0.030	0.595 ± 0.010
Masking (BERT 02-11 + classifier, initial sparsity 15%)	0.844 ± 0.024	0.662 ± 0.021	0.556 ± 0.012
Masking (BERT 04-11 + classifier, initial sparsity 5%)	0.861 ± 0.004	0.705 ± 0.037	0.583 ± 0.005
Masking (BERT 04-11 + classifier, initial sparsity 15%)	0.861 ± 0.009	0.669 ± 0.014	0.553 ± 0.014
Masking (BERT 06-11 + classifier, initial sparsity 5%)	0.862 ± 0.004	0.696 ± 0.027	0.551 ± 0.006
Masking (BERT 06-11 + classifier, initial sparsity 15%)	0.868 ± 0.008	0.691 ± 0.033	0.534 ± 0.016
Masking (BERT 08-11 + classifier, initial sparsity 5%)	0.848 ± 0.016	0.675 ± 0.034	0.538 ± 0.014
Masking (BERT 08-11 + classifier, initial sparsity 15%)	0.851 ± 0.009	0.688 ± 0.022	0.545 ± 0.005
Masking (BERT 00-09 + classifier, initial sparsity 5%)	0.859 ± 0.012	0.683 ± 0.031	0.589 ± 0.011
Masking (BERT 00-09 + classifier, initial sparsity 15%)	0.820 ± 0.052	0.604 ± 0.021	0.514 ± 0.016
Masking (BERT 00-07 + classifier, initial sparsity 5%)	0.829 ± 0.032	0.649 ± 0.053	0.574 ± 0.012
Masking (BERT 00-07 + classifier, initial sparsity 15%)	0.807 ± 0.042	0.600 ± 0.027	0.509 ± 0.004
Masking (BERT 00-05 + classifier, initial sparsity 5%)	0.814 ± 0.033	0.632 ± 0.058	0.565 ± 0.027
Masking (BERT 00-05 + classifier, initial sparsity 15%)	0.781 ± 0.032	0.567 ± 0.030	0.510 ± 0.025
Masking (BERT 00-03 + classifier, initial sparsity 5%)	0.791 ± 0.026	0.606 ± 0.027	0.535 ± 0.034
Masking (BERT 00-03 + classifier, initial sparsity 15%)	0.776 ± 0.035	0.600 ± 0.019	0.527 ± 0.014

Table 7: Numerical value of the layer-wise behavior experiment. We train for 10 epochs with mini-batch size 32. The learning rate is finetuned using the mean results on four different random seeds.

	Number of Parameters		Memory Usage (Kilobytes)	
	Finetuning	Masking	Finetuning	Masking
Pretrained	109,482,240		437,928.96	
MRPC	+ 1,536	+ 1,536 + 71,368,704 + 1,536	+ 6.144	+ 6.144 + 8,921.088 + 0.192
SST2	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
CoLA	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
RTE	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
QNLI	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
SEM	+ 1,536 + 109,482,240	+ 71,368,704 + 1,536	+ 6.144 + 437,928.96	+ 8,921.088 + 0.192
TREC	+ 4,608 + 109,482,240	+ 4,608 + 71,368,704 + 4,608	+ 18.432 + 437,928.96	+ 18.432 + 8,921.088 + 0.576
AG	+ 3,072 + 109,482,240	+ 3,072 + 71,368,704 + 3,072	+ 12.288 + 437,928.96	+ 12.288 + 8,921.088 + 0.384
POS	+ 37,632 + 109,482,240	+ 37,632 + 71,368,704 + 37,632	+ 150.528 + 437,928.96	+ 150.528 + 8,921.088 + 4.704
NER	+ 6,912 + 109,482,240	+ 6,912 + 71,368,704 + 6,912	+ 27.648 + 437,928.96	+ 27.648 + 8,921.088 + 0.864
SWAG	+ 768 + 109,482,240	+ 768 + 71,368,704 + 768	+ 3.072 + 437,928.96	+ 3.072 + 8,921.088 + 0.096

Table 8: Model size comparison when applying masking and finetuning. Numbers are based on BERT-base-uncased. Note that our masking scheme enables sharing parameters across tasks: tasks with the same number of output dimension can use the same classifier layer.