Masking as an Efficient Alternative to Finetuning for Pretrained Language Models

Mengjie Zhao[†*], **Tao Lin**[‡*], **Martin Jaggi**[‡], **Hinrich Schütze**[†]
[†] CIS, LMU Munich, Germany [‡] MLO, EPFL, Switzerland
mzhao@cis.lmu.de, {tao.lin, 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 and RoBERTa on a series of 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 simultaneously. Through intrinsic evaluations, we show that representations computed by 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., 2019c), 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, finetuning stacks a linear layer on top of the pretrained language model and then updates all parameters using 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 simultaneously – several copies of the millions of finetuned parameters have to be saved for inference.

Combining finetuning with multi-task learning (Collobert and Weston, 2008; Ruder, 2017) helps reduce the overall number of required parameters. But multi-task NLP models may produce inferior results compared with their single-task counterparts (Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017). Solving this problem is non-trivial and more complicated techniques, e.g., knowledge distillation (Hinton et al., 2015; Clark et al., 2019), adding extra modules (Stickland and Murray, 2019), or designing sophisticated task-specific layers (Liu et al., 2019b), may be necessary. In this work, we present a method that efficiently utilizes pretrained language models and potential interferences among tasks are eliminated.

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 directly optimizing the model weights for downstream tasks. Inspired by these results, we present 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 and RoBERTa, achieves performance comparable to finetuning in tasks like sequence classification, part-of-speech tagging, 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.

* Equal contribution.

We conduct detailed analyses revealing factors important for and possible reasons contributing to the good task performance.

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: learning selective masks of pretrained weights. Masking is an efficient alternative to finetuning. We show that masking is applicable to models like BERT and RoBERTa, 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 a series of NLP tasks. (iii) We study the loss landscape and compute representations of masked language models, 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., 2019c; Yang et al., 2019; Radford et al.) 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 for parameter-efficient transfer learning. Since the newly inserted modules alter the forward pass of BERT, they need to be carefully initialized to be close to identity.

In this work, we propose to directly pick parameters appropriate to a downstream task, by learning selective binary masks via end-to-end training. With the pretrained parameters being untouched, we solve several downstream NLP tasks simultaneously with minimal overhead.

Binary networks and network pruning. The rationale of training binary masks is 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., 2019d; 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., 2019d)): 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, for both the pretrained language model and a randomly initialized classifier layer, achieves good performance.

3 Method

3.1 Transformer

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

[1] We omit the bias terms for brevity.

representations 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 with SGD-based optimization. Instead, we carefully *select* a subset of the pretrained parameters that is critical to a downstream task while *discarding* irrelevant ones. 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 back-propagate 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.:

反向传播

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

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 the training.

After training, we pass each $\mathbf{M}^l$ through the binarizer to obtain the eventual $\mathbf{M}^l_{\text{bin}}$, 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 model computed by finetuning. Additionally, 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 while adjusting the uniform distribution to achieve a target initial sparsity.

Which layers to mask. Different layers of pretrained language models capture distinct aspects of a language during pretraining. For example, Tenney et al. (2019) find that information on POS

[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.

tagging, parsing, NER, semantic roles and coreference is encoded on progressively higher layers of BERT; Jawahar et al. (2019) show that higher transformer layers of BERT better encode long-term dependency than lower layers. 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) We assume that pretraining has effectively encoded context-independent meanings of words in wordpiece and position embeddings. Hence, learning a selective mask for the embedding layer is unnecessary. In addition, 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

4.1 Datasets

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

评估任务

For **sequence classification**, the following GLUE (Wang et al., 2018) tasks 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-6 (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 **part-of-speech tagging** (POS) on Peen Treebank (Marcus et al., 1993), using Collins (2002)'s train/dev/test split. 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 and accuracy for the other tasks.

4.2 Setup

Due to resource limitations and in the spirit of environmental responsibility (Strubell et al., 2019; Schwartz et al., 2019), we conduct our main experiments on the base models: BERT-base-uncased and RoBERTa-base. We implement[3] our models in PyTorch (Wolf et al., 2019; Paszke 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). For POS experiments, the representation of a tokenized word is its last wordpiece (Liu et al., 2019a; He and Choi, 2019).

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 masked language model on downstream tasks. We experiment on four tasks, with initial sparsities in {1%, 3%, 5%, 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. 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 when most of the connections are removed. (ii) Gradually decreasing the initial sparsity improves task performance. Generally, an initial sparsity in 3% $\sim$ 10% yields reasonable results across tasks.

[3] We will release our code.

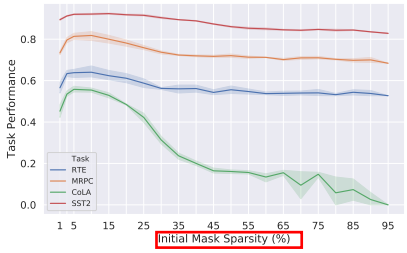


Figure 1: Task performance of masked BERT on dev set with different amounts of pretrained parameters being selected. Without hyperparameter search, the masked models already achieve decent performance on the four tasks.

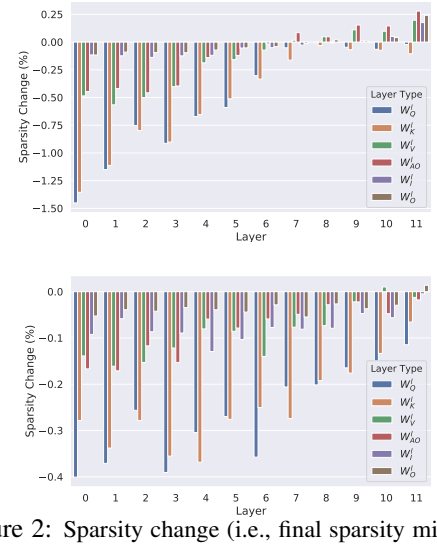


Figure 2: Sparsity change (i.e., final sparsity minus initial sparsity) of the six types of linear layers in the l-th transformer block in masked BERT, trained on POS (top) and SST2 (bottom).

Large datasets like SST2 (67k) are less sensitive than small datasets like RTE (2.5k). (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. Also, the classifier layer is randomly initialized then masked – selecting too many randomly initialized parameters from this layer similarly makes optimization difficult.

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. It is possible that removing too many pretrained parameters in lower layers hurts upper layers when creating high quality representations.

We investigate layer-wise behaviors of BERT layers when using the masking scheme. Specifically, in Figure 2, we observe the *sparsity change*, i.e., final sparsity percentage minus initial sparsity percentage, of a masked layer $\hat{\mathbf{W}}^l \in \{\hat{\mathbf{W}}^l_K, \hat{\mathbf{W}}^l_Q, \hat{\mathbf{W}}^l_V, \hat{\mathbf{W}}^l_{AO}, \hat{\mathbf{W}}^l_I, \hat{\mathbf{W}}^l_O\}$ of the l-th transformer block, the masked pooler layer $\hat{\mathbf{W}}_P$, and the masked classifier layer $\hat{\mathbf{W}}_T$.

We run our masking scheme on BERT for 10 epochs for POS and SST2. Initial sparsity of $\mathbf{M}^l_{\text{bin}}$ of all masked layers is purposely set to a high value of 50% to encourage strong effects. Such a high initial sparsity leads to inferior but better than baseline task performance: 0.866 accuracy for SST2 and 0.970 for POS on dev.

Figure 2 presents the layer-wise sparsity change of the l-th transformer blocks on POS and SST2. Sparsity change of $\hat{\mathbf{W}}_P$ and $\hat{\mathbf{W}}_T$ is 0.00% and -9.86% for POS; and -0.04% and -1.69% for SST2. Observations are: (i) most sparsity changes are negative, meaning that the learning objective encourages $\mathbf{M}^l_{\text{bin}}$ to be less sparse than the large initial sparsity 50%. For POS, the top layer sparsities increase, reflecting that the encoded abstract semantic information is not helpful for solving this syntactic task. (ii) Sparsity decreases in the lower layers of masked BERT are ≈ 3 times larger for POS than for SST2 (sentiment classification). This is consistent with previous studies showing that syntactic information is better encoded in lower layers (Peters et al., 2018). $\hat{\mathbf{W}}_T$ sees the largest sparsity change: -9.86% for POS and -1.69% for SST2. We conjecture that the randomly initialized weights in this layer require more adaptations than pretrained layers to fit a downstream task.

Figure 3 presents the optimal task performance when masking only subset of BERT's transformer blocks on MRPC, CoLA and RTE. We see 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 il-

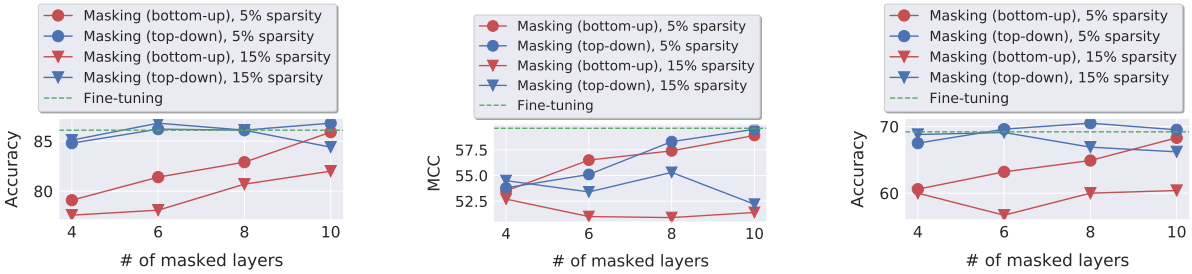


Figure 3: The impact of masking different transformer layers of BERT for MRPC (left), CoLA (middle) and RTE (right). Different numbers and indices of blocks are masked: "bottom-up" and "top-down" indicate to mask the targeted amount of transformer blocks, either from top or bottom of BERT. More precisely, a bottom-up setup (red) masking 4 layers means we mask the transformer blocks $\{0, 1, 2, 3\}$; a top-down setup (blue) masking 4 layers means we mask the transformer blocks $\{8, 9, 10, 11\}$. $\mathbf{W}_P$ and $\mathbf{W}_T$ are always masked. Numerical results can be found in Table 5 (in Appendix §A.2).

		MRPC 3.5k	SST2 67k	CoLA 8.5k	RTE 2.5k	QNLI 108k	POS 38k	SWAG 113k
BERT	Finetuning	86.1 ± 0.8	93.3 ± 0.2	59.6 ± 0.8	69.2 ± 2.7	91.0 ± 0.6	97.7 ± 0.0	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	97.7 ± 0.0	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	98.1 ± 0.0	83.4 ± 0.8
	Masking	89.0 ± 1.0	95.1 ± 0.1	61.7 ± 3.0	72.2 ± 3.3	92.6 ± 0.1	97.8 ± 0.0	82.5 ± 0.1

Table 1: Dev set task performances of masking BERT and RoBERTa, compared with results obtained from finetuning. A 5% initial sparsity of $\mathbf{M}^l_{\text{bin}}$ is used when masking BERT and 3% for RoBERTa. Transformer blocks 2–11, $\mathbf{W}_P$, and $\mathbf{W}_T$ are masked. Each experiment is repeated four times with different random seeds.

lustrates 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.[4] 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 thoroughly investigated two factors – initial sparsity (§5.1) and layer-wise behavior (§5.2) – that are important in masking pretrained language models. Here, we compare the performance of masking and finetuning on a series of NLP tasks.

We search the optimal learning rate per task as described in §4.2. We use a batch size of 32 for tasks that have <96k training examples. For AG, QNLI, and SWAG, we use batch size 128. The optimal hyperparameters per task are shown in §A.1.

Table 1 compares performance of masking and finetuning on the dev set for GLUE tasks, POS, and SWAG. We observe that applying masking to BERT and RoBERTa yields performance comparable to finetuning. We observe a performance drop[5] on RoBERTa-RTE. RTE has the smallest dataset size (2.5k in train and 0.3k in dev) among all tasks – this may contribute to the imperfect results and large performance variances.

Rows "Single" in Table 2 compare performance of masking and finetuning BERT on the test set of SEM, TREC-6 and AG. The same setup and hyperparameter searching of masking BERT as Table 1 are used, the best hyperparameters are picked on the dev set. Results from Sun et al. (2019) are included as a reference. Note that Sun et al. (2019) use configurations like layer-wise learning rate, producing slightly better performance than ours. Palogiannidi et al. (2016) is the best performing systems on task SEM (Nakov et al., 2016). Again, masking yields results comparable to finetuning.

Next, we compare ensembled results to better demonstrate memory efficiency of the masking

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

[5] 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.

		SEM	TREC-6	AG	Model Size
		4.3k	4.9k	96k	(MB)
Masking	Single	12.03	3.30	5.62	447
	Ensem.	11.52	3.20	5.28	474
Finetun.	Single	11.87	3.80	5.66	438
	Ensem.	11.73	2.80	5.17	1752
Sun et al. (2019)		n/a	2.80	5.25	438
Palogiannidi et al. (2016)		13.80	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.

scheme. Three ensemble methods are considered: (i) majority voting where the most voted label is the final prediction; (ii) ensemble of logits where the label with the highest overall logit is the final prediction; (iii) ensemble of probability where the label with the highest overall predicted probability is the final prediction. The best ensemble method is picked on dev then evaluated on test. Rows "Ensem." in Table 2 compare ensembled results and model size. Masking consumes only 474MB memory – much smaller than 1752MB required by finetuning – and achieves comparable performance. Thus, masking is much more memory-efficient than finetuning in an ensemble setting.

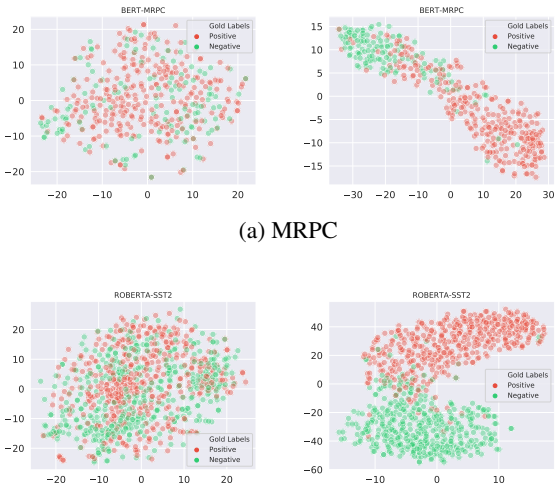
6 Discussion

6.1 Intrinsic evaluations

Through extensive evaluations on a series of NLP tasks, §5 demonstrates that the masking scheme is an efficient alternative to finetuning. Now we analyze properties of the representations computed by masked BERT/RoBERTa 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 masked BERT/RoBERTa, 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) to visualize the representation of `[CLS]` computed by the topmost transformer block in pretrained and masked BERT/RoBERTa, using the dev set examples of MRPC and SST2. As shown, representations computed by the pretrained BERT/RoBERTa



(a) MRPC

(b) SST2

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

Train \ Eval	SST2	SEM
SST2	41.8	-13.4
SEM	20.0	11.5

(a) Masking

Train \ Eval	SST2	SEM
SST2	41.8	-10.1
SEM	18.9	12.2

(b) Finetuning

Table 3: Cross-dataset performance (%) of masked and finetuned BERT models obtained in §5.3. Numbers are performance improvement against the majority baseline: 50.9 for SST2 and 74.4 for SEM. Results are averaged across four random seeds.

are clearly not distinguishable since the pretrained models need adaptations to downstream tasks. After applying the masking scheme, the computed representations are almost linearly separable and consistent with the gold labels. Thus, a linear classifier is expected to yield reasonably good classification accuracy. All t-SNE visualizations of pretrained, finetuned, and masked BERT/RoBERTa are presented in §A.3. This intrinsic evaluation illustrates that masked BERT/RoBERTa extracts good representations from the data for the downstream NLP task.

6.2 Do the masked models generalize?

Figure 4 illustrates that a masked language model extracts proper text representations for the classifier layer and hence performs as well as finetuning. Here, we are interested in verifying that our masked language model does indeed solve downstream tasks by learning meaningful representations – in-

stead of simply exploiting superficial noise within a dataset. To this end, we test if the masked language model is generalizable to other datasets of one type of downstream task.

We use the two sentiment classification datasets in our task pool: 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. For example, in (a), cell value -13.4 means that the SST2-masked BERT performs 13.4% worse than the majority baseline[6] on dev set of SEM.

Comparing SST2 and SEM, we can observe that knowledge learned on SST2 does not generalize to SEM, for both finetuning and masking. Note that 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. On the other hand, the finetuned and masked models of SEM generalize well on SST2, showing $\approx 20\%$ improvement against the majority baseline. Thus, the masked models indeed create representations that contain valid information for downstream tasks.

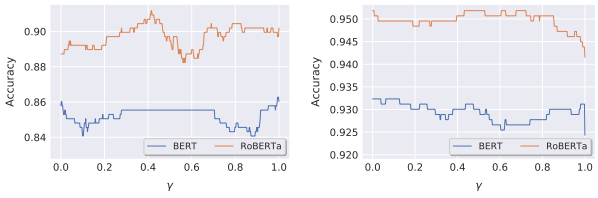
6.3 Loss landscape

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, for BERT and RoBERTa, the relationship between the two minima obtained by masking and finetuning.

Recent work analyzing the loss landscape suggests the local minima reached in the loss landscape of vanilla training 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 on 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 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 models obtained in Table 1 (same seed and training epochs); Figure 5 shows the results of mode connectivity, i.e., the evolution of the loss value along a line connecting two candidate minima.



(a) MRPC (b) SST2

Figure 5: The accuracy on dev set, as a linear interpolation of two minima found by finetuning ($\gamma = 0$) and masking ($\gamma = 1$). The training loss (omitted) has similar curves.

Surprisingly, the interpolated models on the line segment connecting a finetuned and a masked model form a high accuracy path[7], indicating the extremely well-connected loss landscape. Thus, the masking scheme finds minima on the same connected low-loss manifold as finetuning, confirming the effectiveness of our method. We also experimented with Bézier curves as proposed by Garipov et al. (2018); similar observations can be made, as shown in §A.4.

7 Conclusion

We have presented masking, a new way of utilizing pretrained language models that is more efficient than finetuning. Instead of directly modifying the pretrained parameters through additional task-specific training (as in finetuning), we only train one binary mask per task in order to select critical parameters. Extensive experiments show that masking yields performance comparable to traditional finetuning on a series of NLP tasks. Leaving the pretrained language model parameters unchanged, masking is much more parameter and memory efficient when several tasks need to be solved simultaneously. Intrinsic evaluations show that masked language models extract valid representations for

[6] A naive classifier always predicting the major class in the dataset.

[7] We additionally show in Figure 10 (§A.4), for the line segment between a pretrained language model and a finetuned/masked model, that mode connectivity is not solely due to an overparameterized pretrained language model.

downstream tasks. We further show that the representations are generalizable. Moreover, we demonstrate that the minima obtained by finetuning and masking can be easily connected by a line segment, further confirming the effectiveness of applying masking to pretrained language models.

Masking is independent of the order of downstream tasks, which avoids potential interferences between tasks, e.g., catastrophic forgetting (French, 1999). Thus, it is a natural fit to apply this scheme to continual lifelong learning (Parisi et al., 2019). We plan to exploit this possibility on NLP tasks in future work.

References

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

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

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.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

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.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

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. 2019. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

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.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

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.

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.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. 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. 2019c. Roberta: A robustly optimized bert pretraining approach.

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

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.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain. 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.

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.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

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.

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

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.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks.

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.

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 R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems.

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.

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 Experimental details

A.1 Hyper-parameter tuning

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

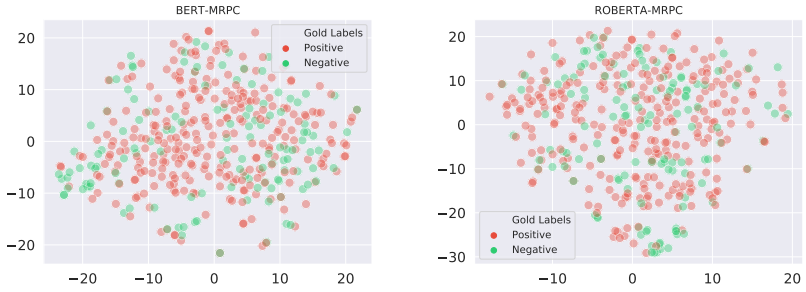
Table 4: The optimal learning rate on different tasks (evaluated through four different random seeds) for BERT and RoBERTa. We perform finetuning/masking on all tasks for 10 epochs with early stoping of 2 epochs.

A.2 Other empirical results

	MRPC	RTE	CoLA
finetuning (BERT + classifier)	0.861 ± 0.008	0.692 ± 0.027	0.596 ± 0.015
masking (BERT 00-11 + classifier, M1, initial sparsity 5%)	0.862 ± 0.015	0.673 ± 0.036	0.592 ± 0.004
masking (BERT 00-11 + classifier, M1, initial sparsity 15%)	0.825 ± 0.039	0.626 ± 0.040	0.522 ± 0.027
masking (BERT 02-11 + classifier, M1, initial sparsity 5%)	0.868 ± 0.011	0.695 ± 0.030	0.595 ± 0.010
masking (BERT 02-11 + classifier, M1, initial sparsity 15%)	0.844 ± 0.024	0.662 ± 0.021	0.556 ± 0.012
masking (BERT 04-11 + classifier, M1, initial sparsity 5%)	0.861 ± 0.004	0.705 ± 0.037	0.583 ± 0.005
masking (BERT 04-11 + classifier, M1, initial sparsity 15%)	0.861 ± 0.009	0.669 ± 0.014	0.553 ± 0.014
masking (BERT 06-11 + classifier, M1, initial sparsity 5%)	0.862 ± 0.004	0.696 ± 0.027	0.551 ± 0.006
masking (BERT 06-11 + classifier, M1, initial sparsity 15%)	0.868 ± 0.008	0.691 ± 0.033	0.534 ± 0.016
masking (BERT 08-11 + classifier, M1, initial sparsity 5%)	0.848 ± 0.016	0.675 ± 0.034	0.538 ± 0.014
masking (BERT 08-11 + classifier, M1, initial sparsity 15%)	0.851 ± 0.009	0.688 ± 0.022	0.545 ± 0.005
masking (BERT 00-09 + classifier, M1, initial sparsity 5%)	0.859 ± 0.012	0.683 ± 0.031	0.589 ± 0.011
masking (BERT 00-09 + classifier, M1, initial sparsity 15%)	0.820 ± 0.052	0.604 ± 0.021	0.514 ± 0.016
masking (BERT 00-07 + classifier, M1, initial sparsity 5%)	0.829 ± 0.032	0.649 ± 0.053	0.574 ± 0.012
masking (BERT 00-07 + classifier, M1, initial sparsity 15%)	0.807 ± 0.042	0.600 ± 0.027	0.509 ± 0.004
masking (BERT 00-05 + classifier, M1, initial sparsity 5%)	0.814 ± 0.033	0.632 ± 0.058	0.565 ± 0.027
masking (BERT 00-05 + classifier, M1, initial sparsity 15%)	0.781 ± 0.032	0.567 ± 0.030	0.510 ± 0.025
masking (BERT 00-03 + classifier, M1, initial sparsity 5%)	0.791 ± 0.026	0.606 ± 0.027	0.535 ± 0.034
masking (BERT 00-03 + classifier, M1, initial sparsity 15%)	0.776 ± 0.035	0.600 ± 0.019	0.527 ± 0.014

Table 5: A thorough investigation. We train for 10 epochs with mini-batch size 32. The learning rate is fine-tuned using the mean results on four different random seeds.

A.3 T-SNE visualizations

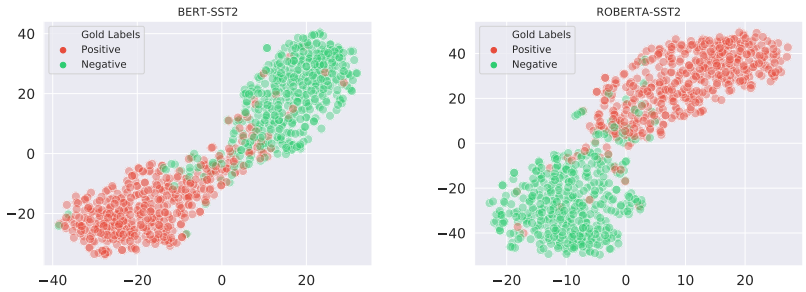


(a) MRPC

(b) SST2

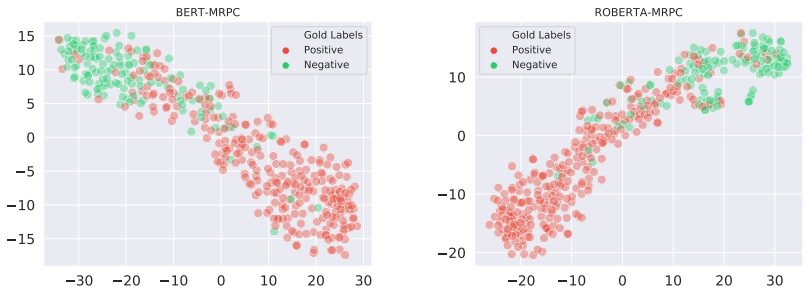
Figure 6: t-SNE of the representation of `[CLS]` computed by the topmost transformer block in pretrained BERT and RoBERTa.

(a) MRPC



(b) SST2

Figure 7: t-SNE of the representation of `[CLS]` computed by the topmost transformer block in finetuned BERT and RoBERTa.

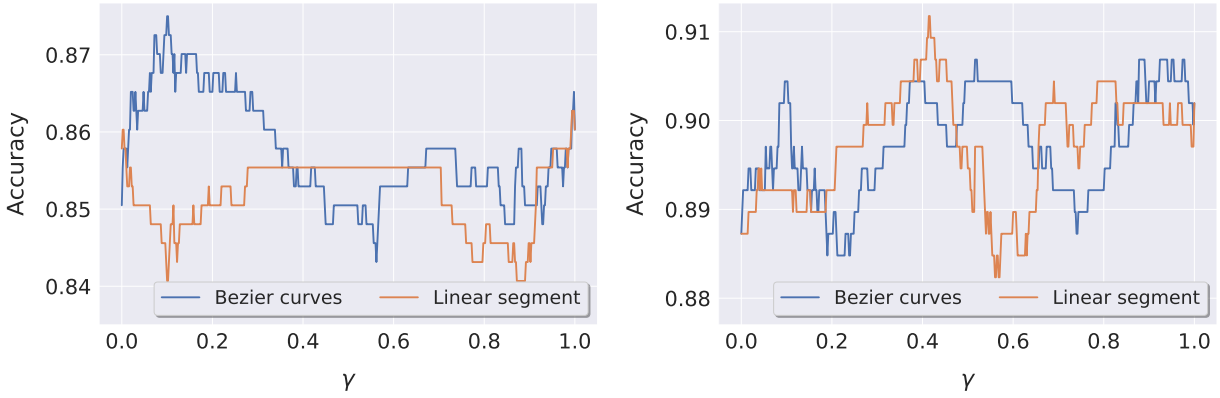


(a) MRPC

(b) SST2

Figure 8: t-SNE of the representation of `[CLS]` computed by the topmost transformer block in masked BERT and RoBERTa.

A.4 More on mode connectivity



(a) BERT (b) RoBERTa

Figure 9: 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, $\gamma = 1$).

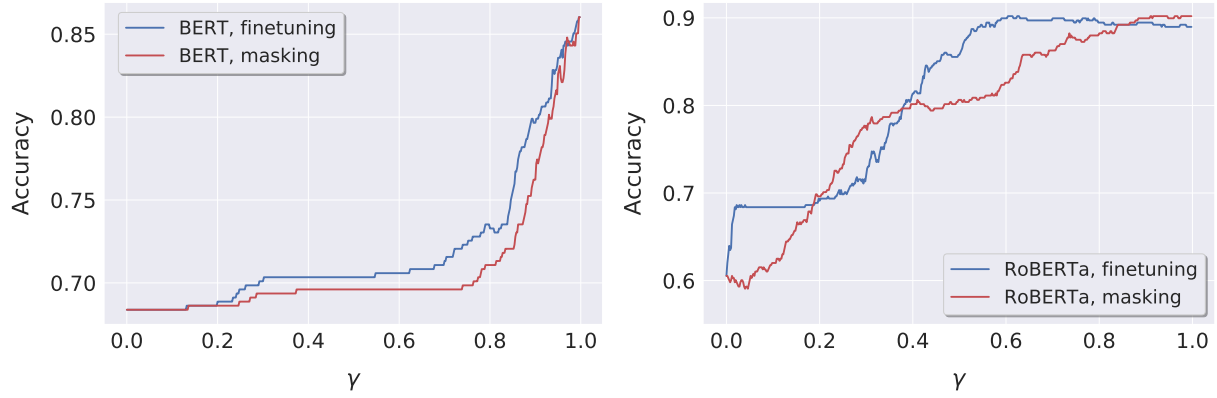
Following the mode connectivity framework proposed in Garipov et al. (2018), we parameterize the path joining two minima using a Bézier curve. Let $\mathbf{w}_0$ and $\mathbf{w}_{n+1}$ be the parameters of the models trained from fine-tuning and masking. Then, an n-bends Bézier curve connecting $\mathbf{w}_0$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

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

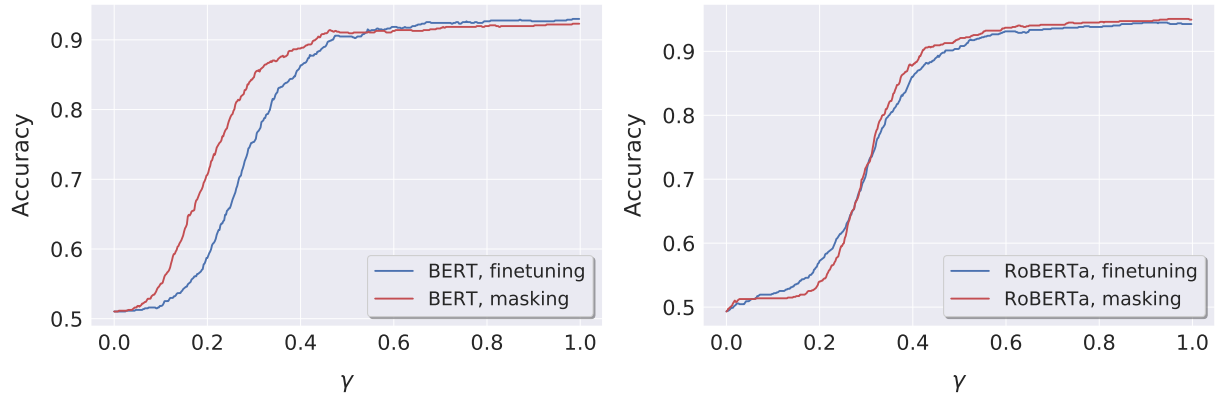
We train an 3-bends Bézier curve by minimizing the loss $\mathbb{E}_{t\sim U[0,1]} \mathcal{L}\left(\phi_\theta(t)\right)$, where $U[0,1]$ is the uniform distribution in the interval $[0,1]$. Monte Carlo method is used to estimate the gradient of this

expectation-based function and gradient-based optimization is used for the minimization. The results are illustrated in Figure 9. Masking implicitly performs gradient descent, analogy to the weights update achieved by finetuning; the observations complement our arguments in the main text.

In addition, Figure 10 visualize the line segment between a pretrained language model and a finetuned or masked model (on downstream task), highlighting the present observations of the mode connectivity are not solely due to the overparameterized pretrained language model.



(a) MRPC



(b) SST2

Figure 10: The accuracy on the dev set, as a function of the point on the curves $\phi_\theta(\gamma)$, connecting the two models between pretrained language model (left, $\gamma = 0$), and finetuned or masked model (right, $\gamma = 1$) on a downstream task.