Finding the Dominant Winning Ticket in Pre-Trained Language Models

Zhuocheng Gong[1], Di He[2], Yelong Shen[3], Tie-yan Liu[2],
Weizhu Chen[3], Dongyan Zhao[1,4,5*], Ji-rong Wen[6] and Rui Yan[6*]
[1]Wangxuan Institute of Computer Technology, Peking University, China
[2]Microsoft Research, Beijing, China, [3]Microsoft Azure AI
[4]Artificial Intelligence Institute of Peking University
[5]State Key Laboratory of Media Convergence Production
[6]Gaoling School of Artificial Intelligence, Renmin University of China
{gzhch, zhaody}@pku.edu.cn, {jrwen, ruiyan}@ruc.edu.cn
{dihe, yelong.shen, tyliu, wzchen}@microsoft.com

Abstract

The Lottery Ticket Hypothesis suggests that for any over-parameterized model, a small subnetwork exists to achieve competitive performance compared to the backbone architecture. In this paper, we study whether there is a winning lottery ticket for pre-trained language models, which allow the practitioners to fine-tune the parameters in the ticket but achieve good downstream performance. To achieve this, we regularize the fine-tuning process with L1 distance and explore the subnetwork structure (what we refer to as the "dominant winning ticket"). Empirically, we show that (a) the dominant winning ticket can achieve performance that is comparable with that of the full-parameter model, (b) the dominant winning ticket is transferable across different tasks, (c) and the dominant winning ticket has a natural structure within each parameter matrix. Strikingly, we find that a dominant winning ticket that takes up 0.05% of the parameters can already achieve satisfactory performance, indicating that the PLM is significantly reducible during fine-tuning.

1 Introduction

Pre-trained Language Models (PLMs) have shown significant performance on various natural language processing (NLP) tasks (Devlin et al., 2018; Liu et al., 2019). However, as the number of model parameters gets huge, fine-tuning such models becomes inefficient. Many previous works target parameter-efficient fine-tuning approaches by freezing the PLM parameters. One can either freeze a subset of the parameters (Zaken et al., 2021) and fine-tune the remainings or freeze all of them and plug in light modules with new learnable parameters (Houlsby et al., 2019; Mahabadi et al., 2021; Hu et al., 2021).

*corresponding authors: Dongyan Zhao (zhaody@pku.edu.cn) and Rui Yan (ruiyan@ruc.edu.cn)

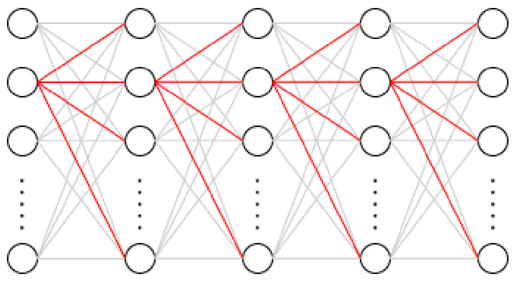

Figure 1: An illustration of the structure of the dominant winning ticket.

In parallel to this line of study, an emerging subfield has explored the possibility of training smaller subnetworks in place of the full models without hurting performance (Lee et al., 2018; Wang et al., 2020). Among them, the lottery ticket hypothesis (LTH) (Frankle and Carbin, 2018) has attracted much attention. LTH demonstrates that an over-parameterized network contains "winning tickets" (small-scale subnetworks) that can 1) match the performance of the full model; and 2) outperform randomly sampled subnetworks of the same size. Winning tickets have been verified to exist in PLMs (Prasanna et al., 2020).

In this paper, we provide an interesting result by showing that a subnetwork, which we refer to as the dominant winning ticket, exists in the PLM. It can make us to freeze all other parameters but only train the parameters of the subnetwork and obtain competitive performance for any downstream tasks. To achieve this, we fine-tune the PLM on one task (e.g., MNLI) and enforce the parameter weights close to their initial weights (pre-trained weights) by using the L1-distance penalty. This allows us to identify which parts of the model parameters change greater from the pre-trained weights during fine-tuning. We observe this ticket has some novel properties:

- The dominant winning ticket (i.e., the subnetwork) is extremely sparse, which only takes up to 0.05% of the total parameters on RoBERTa-large. But fine-tuning the subnetwork can achieve comparable performance with fine-tuning the whole model.
- Compared with randomly sampled subnetworks of the same size, the dominant winning ticket can achieve better performance with a faster convergence rate.
- The dominant winning ticket is insensitive to random seed and transferable across different downstream tasks. In other words, it is intrinsically determined by the pre-trained weights and can adapt to various downstream tasks.
- The dominant winning ticket is highly structured. It is like the "skeleton" of the network, which may give us some insights into the mechanism of PLMs.

We organize the paper as follows. After a brief overview of related works in section 2, we introduce how we are aware of the existence of the dominant winning ticket in PLMs and how we identify and extract it in section 3. Then we experiment on the dominant winning ticket in section 4. In section 5, we do further discussions about the dominant winning ticket. Section 6 is about some implications and future directions of the work.

2 Related Work

2.1 Pruning

Multiple studies of BERT concluded that it is considerably overparametrized (Kovaleva et al., 2019; Michel et al., 2019). In particular, it is possible to ablate elements of its architecture without loss in performance or even with slight gains (Voita et al., 2019; Li et al., 2021). There has been much recent work on compressing PLM. See overviews by Ganesh et al. (2021). Pruning is a promising line of work for model compression which involves obtaining smaller subnetworks with minimal performance loss (Gordon et al., 2020; Sajjad et al., 2020). A common approach is selecting the weights to be pruned by magnitude (Han et al., 2015).

Previous work has found that there exist subnetworks inside the neural network, which is called the lottery ticket hypothesis (Frankle and Carbin, 2018). Some of the recent findings are that the lottery ticket hypothesis holds for PLMs: inside large-scale pre-trained model there exist subnetworks that can be retrained alone to reach the performance close to that of the full model (Chen et al., 2020; Prasanna et al., 2020; Liang et al., 2021). Prasanna et al. (2020) claimed that "When BERT plays the lottery, all tickets are winning". Liang et al. (2021) shows that there exist super tickets inside PLMs that can improve generalization.

2.2 Parameter-efficient Fine-tune

Parameter-efficient fine-tuning aims at reducing the number of trainable parameters when fine-tuning the models across different downstream domains. Various approaches are invented to achieve the goal. Some inserted and only trained adapters, which have much lesser trainable parameters, between existing layers (Houlsby et al., 2019; Mahabadi et al., 2021; Rebuffi et al., 2017). Another line of the study proposed to update only a subset of parameters when fine-tuning. For example, Gordon et al. (2020) leveraged L0 regularization to limit the non-zero elements in the update vectors. Zaken et al. (2021) proposed that only tune bias terms can reach a decent performance. Zhao et al. (2020) applied the sparse binary mask to the pre-trained weights to reduce the trainable parameter size. Besides, some proposed that the PLM has a low intrinsic dimensionality (Aghajanyan et al., 2020). Hu et al. (2021) proposed a low-rank decomposition-based method that can also significantly reduce the number of trainable parameters. Chen et al. (2021) combined low-rank decomposition and sparse mask during fine-tuning.

3 The Dominant Winning Ticket in PLMs

We start with a question: which parts of the model parameters are more important when adapting PLMs to downstream tasks? To answer this question, we design the L1-regularized fine-tuning approach and then analyze the L1-regularized weights and reveal the existence of the dominant winning ticket.

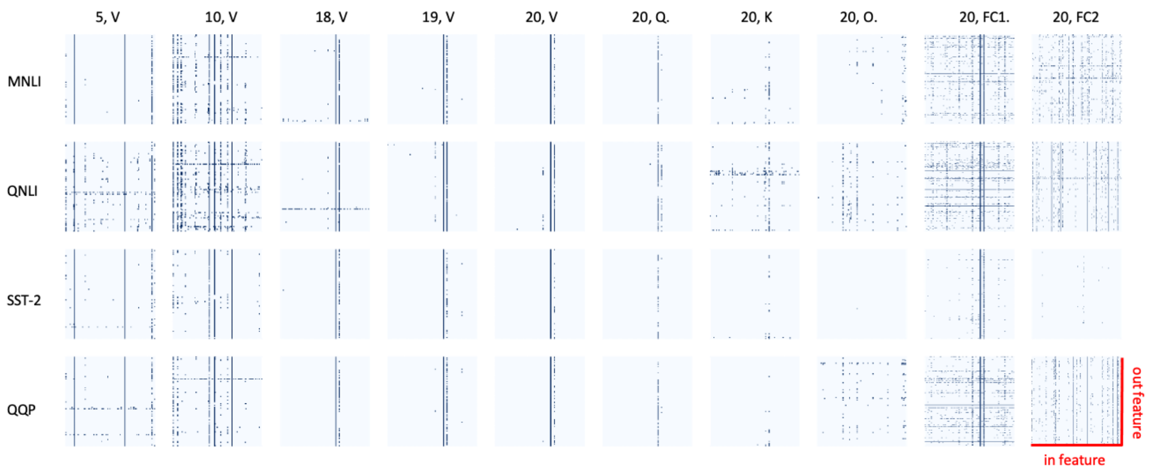

Figure 2: Visualization of out weighted parameters in RoBERTa-large. Each row is a model fine-tuned on different downstream tasks. Each column is the parameter matrix of different layers. For example, the top left block represents the value projection matrix of the 5th transformer layer from the model fine-tuned with the MNLI dataset. ("K", "Q", "V", "O" stand for the key, query, value, and output projection metrics of the self-attention module, "FC1" and "FC2" stand for the successive two fully-connected transformation metrics in each layer.)

3.1 L1-regularized Fine-tuning

To identify the subnetwork, we apply L1 regularization to all the transformer parameters. Specifically, we modify the original training objective, which results in the following minimization problem,

$$\min_{\boldsymbol{\theta}} L(\mathcal{D}, f, \boldsymbol{\theta}) + \lambda ||\boldsymbol{\theta} - \boldsymbol{\theta}_0||_1, \tag{1}$$

where $\mathcal{D}$ represents for the task-specific data, $\boldsymbol{\theta}$ is the model configuration(i.e., parameters), and $\boldsymbol{\theta}_0$ is the pre-trained model, which is fixed. λ is a hyper-parameter to control the weight of the regularization term, i.e., the strength of the sparsity. This training objective can make a large part of the parameters being close to their initial weights.

3.2 Regularized Weights Analysis

We use L1-regularized fine-tuning to answer the previous proposed question: which parts of the model parameters are more important when adapting PLMs to downstream tasks? We compute the difference between the post-fine-tune weights and the pre-trained weights to see how the weights change. Specifically, we get $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}_0$. Inspired by magnitude weight pruning (Han et al., 2015), we hypothesize that the magnitude of $\Delta\boldsymbol{\theta}$ can be an indicator. We observe that the magnitude of $\Delta\boldsymbol{\theta}$ is very small for most of the parameters (smaller than 1e-5 for 99% parameters), indicating that the L1 regularization does take effect. However, while most of the parameters remain close to their initial weights during fine-tuning, a very small fraction of parameters have much greater change. We define these parameters as out weighted parameters. Intuitively, we can choose a threshold σ to select out weighted parameters. Formally, we define σ-bounded out weighted parameters:

$$\boldsymbol{\theta}_\sigma = \boldsymbol{m}_\sigma \odot \boldsymbol{\theta}, \quad \boldsymbol{m}_\sigma \in \{0,1\}^{|\boldsymbol{\theta}|} \tag{2}$$

where $\boldsymbol{m}_\sigma$ is a binary mask vector, $\boldsymbol{m}_{\sigma,i} = \mathbb{1}\{|\Delta\theta_i| > \sigma\}$.

We try to understand the mechanism of fine-tuning by analyzing the distribution of σ-bounded out weighted parameters. We visualize the distributions of the out weighted parameters on several different tasks (MNLI, QNLI, SST-2, QQP) as seen in Figure 2, where the parameter matrics are

$$W^Q, W^K, W^V, W^O \in \mathbb{R}^{d_{model} \times d_{model}}$$
$$W^{FC1} \in \mathbb{R}^{d_{model} \times d_{fc}}, W^{FC2} \in \mathbb{R}^{d_{fc} \times d_{model}} \tag{3}$$

For RoBERTa-large, d_{model} is 1024 and d_{fc} is 4096.

Observations 1. The locations of the out weighted parameters have strong correlations among different tasks. Each column in Figure 2 shows the out weighted parameters of the same matrix on different tasks, which has a very similar phenomenon. This high similarity indicates that the location of the out weighted parameters may be downstream task-agnostic. That is to say, the PLM

Table 1: The structure of the dominated ticket. For each matrix, we select the top 3 dominated dimensions. We highlight several "popular" dimensions across different layers with colors. Dimension `4096` in W^{FC2} and `1024` in W^K, W^Q, W^V, W^O, W^{FC1} are the symbol for the bias term.

Layer	W^Q			W^K			W^V			W^O			W^{FC1}			W^{FC2}		
0	981	673	304	1019	781	108	673	981	472	647	766	986	673	981	1019	487	584	3291
1	981	673	106	106	981	673	673	981	106	593	929	587	673	106	981	1559	3646	1995
2	673	981	106	981	106	673	673	106	981	1024	579	237	673	106	981	3708	3058	2753
3	673	106	981	981	106	673	673	106	981	810	819	784	673	106	981	1591	480	609
4	673	106	981	981	673	106	106	673	981	101	1024	76	673	106	981	3445	1906	2828
5	673	106	981	673	106	981	673	106	981	1024	579	189	673	106	981	682	2123	841
6	673	106	981	673	106	981	673	106	981	1024	430	498	673	106	981	1267	2920	2686
7	673	106	981	673	474	106	673	106	412	692	670	699	673	106	412	2058	3485	1660
8	673	106	412	474	673	547	673	412	474	1024	534	69	673	106	412	2780	2387	1015
9	673	412	2	474	673	256	673	474	412	11	1024	608	673	2	412	3892	448	616
10	673	412	474	474	673	256	673	474	412	254	534	1024	673	412	2	1028	592	1462
11	673	2	93	474	256	623	673	474	412	330	758	1024	673	474	547	1014	3730	2157
12	673	547	93	474	256	673	673	474	547	151	909	1024	673	474	547	145	2436	2338
13	673	547	2	474	256	673	673	474	547	113	1024	151	673	547	474	902	2503	1700
14	673	547	631	474	256	673	673	474	547	141	1024	307	673	631	547	1219	1318	1535
15	673	631	547	631	673	474	673	631	474	1024	523	724	673	631	547	2413	608	15
16	673	631	547	673	631	256	631	673	547	1024	657	819	631	673	547	3454	4096	1288
17	631	673	547	673	631	256	631	673	547	254	640	209	631	673	547	3433	2	3962
18	631	673	547	673	631	256	631	673	547	986	999	966	631	673	547	628	2617	4096
19	631	673	547	673	631	256	631	673	399	657	1	845	631	673	399	1221	827	4096
20	631	673	547	673	631	256	631	673	399	975	453	1006	631	673	914	1442	4096	1669
21	631	673	None	673	631	None	631	673	547	1024	975	400	631	673	914	3290	4096	2850
22	631	None	None	None	None	None	631	673	547	559	1024	46	631	673	914	3341	3176	1078
23	631	673	None	631	673	None	631	673	914	167	871	891	631	673	422	2498	2596	1565

itself determines which parameters tend to be out weighted.

Observations 2. If looking into each block in Figure 2, we can see that the out weighted parameters are distributed along with the output dimension. The out weighted parameters in the matrix tend to be dominated by a few output dimensions, what we refer to as dominated dimensions. We identify dominated dimensions by counting the number of out weighted parameters in each dimension and observe that this phenomenon exists in most of the parameter matrices in the PLM. We list all the dominated dimensions of the model to further analyze. Details can be found in Table 2. An interesting finding is that some dimensions consistently dominate several successive layers. For example, dimension `673`[1] and `631` dominate W^Q, W^K, W^V, and W^{FC1} for more than 10 successive transformer layers. These dimensions are like the skeleton of the PLM that exists from bottom to top.

With the above observations, now we can propose our hypothesis about the dominant winning ticket of PLMs.

Hypothesis. There exists a dominant winning ticket inside a PLM that is intrinsically determined by the pre-trained weights. When fine-tuned in isolation, we can only finetune the parameters of this ticket which can match the performance of full-parameter fine-tuning while converging faster than other methods.

3.3 Extracting the Dominant Winning Ticket

When it comes to extracting the dominant winning ticket, the first question is to decide the sparsity of the subnetwork (the number of trainable parameters). Generally, we extract the dominant winning ticket with algorithm 1.

Algorithm 1 Extracting the dominant winning ticket

```
1: Fine-tune a PLM $f(x;\theta_0)$ with L1 regularization on any dowmstream task dataset $\mathcal{D}$, get $f(x;\theta)$.
2: Calculate $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}_0$, then select out out weighted parameters $\boldsymbol{\theta}_\sigma$ with threshold $\sigma$.
3: Select the $k$ most dominated dimensions each matrix, which forms the dominant winning ticket.
```

We use hyperparameter k to control the sparsity of the subnetwork. We empirically find that the scale of the subnetwork is extremely small. k equals 1 (at most one dominated dimension for each matrix) is enough to achieve comparable results to full-parameter fine-tuning for most of the tasks.

It is worth noticing that our extracted dominant winning ticket excludes embedding and layer nor-

[1] All dimensions in the paper are zero-indexed.

malization. Besides, as bias terms are vital to some extend (Zaken et al., 2021), we include bias terms when identifying the dominated dimensions, so the bias term in each matrix also has a chance to be selected.

4 Testing the Dominant Winning Ticket for Fine-tuning

The previous section is about how we identify and extract the dominant winning ticket. In this section, we discuss the properties of the dominant winning ticket by conducting systematic experiments.

To extract the dominant winning tickets, we perform L1-regularized fine-tuning on MNLI, QNLI, SST-2, and QQP respectively. We find that the dominant winning tickets corresponding to different downstream tasks look very close to each other (detailed statistics can be found in Appendix A.1), which matches the observation in Section 3.2. So, for simplicity, we regard the subnetwork extracted from MNLI as the standard dominant winning ticket. Our evaluation experiments are conducted upon it.

We compare the dominant winning ticket with the subnetwork that has the same size as the dominant winning ticket, denoted as *Dominant-k* and *Random-k* respectively. The dimensions of *Random-k* are chosen from uniform distribution. As the scale of the extracted subnetwork is adjustable by choosing different k, we consider two different compression ratios, i.e., $k = 1$ (one dimension per matrix at most) and $k = 3$ (three dimension per matrix at most).

4.1 Exprimental Setup

Datasets and models. We conduct experiments on the GLUE benchmark (Wang et al., 2018). The evaluation is performed on the GLUE dev sets. We use the publicly available RoBERTa-large[2] (Liu et al., 2019) as pre-trained language models in all our experiments.

Implementation details. Our implementation is based on the fairseq toolkit[3] (Ott et al., 2019). We fine-tune on the GLUE tasks following the standard procedures. We optimize using AdamW (Loshchilov and Hutter, 2018), with batch size of 16. For L1-regularized fine-tuning, we empirically set the weight of L1 regularizer λ to 0.001, and the threshold σ to 5e-5. For dominant winning ticket fine-tuning, we perform a hyperparameter search over initial learning rate in $\{$5e-5, 1e-4, 2e-4, 4e-4$\}$. For full-parameter fine-tuning, we search initial learning rate in $\{$1e-5, 2e-5, 3e-5, 5e-5$\}$.

4.2 Performance

Our main results on the GLUE benchmark are shown in Table 2. Fine-tuning the dominant winning ticket can match the performance of fine-tuning the whole model while only requiring less than 0.2% trainable parameters per task. Performance of *Dominant-3* and *Dominant-1* has no significant difference for most of the tasks. For small datasets like MRPC and RTE, the smaller subnetwork even performs slightly better because less trainable parameters means free from overfitting. This phenomenon also indicates that the scale of the dominant winning ticket inside the PLMs can be extremely small. When comparing the dominant winning ticket with random tickets that has the same size, we can see that *Random-3* performs pretty well. We think this benifits from the strong reducibility of PLMs. However, when the sparsity grows, the performance of *Random-1* is much worse than *Dominant-1*, indicating that randomly sampled subnetworks unavoidable deletes useful information and become less expressive at such a level of sparsity.

Figure 3 shows the training and evaluation curves of different methods. We can clearly see that the dominant winning ticket has advantages over random subnetworks in terms of convergence rate. For large datasets like QNLI, *Dominant-1* and *Dominant-3* reach the best validation accuracy in the first several epochs, while random subnetworks require much more training steps to warm up. The dominant winning ticket also gets lower training losses in all tasks, indicating it fits the data better. Another observation is that, though *Dominant-3* and *Random-3* have no significant performance gaps in most of the tasks as seen in Table 2, their learning behaviors vary quite a lot. *Random-3* gets satisfactory results eventually, but it requires much more efforts to train. Meanwhile, there is no obvious difference between *Dominant-1* and *Dominant-3* from Figure 3, which is a good sign. This phenomenon suggests that when shrinking the parameter size from 0.19% to 0.05%, the capacity of the model almost keeps unchanged.

[2]The dominant winning ticket in the paper is extracted from this specific model checkpoint. https://dl.fbaipublicfiles.com/fairseq/models/roberta.large.tar.gz

[3]https://github.com/pytorch/fairseq

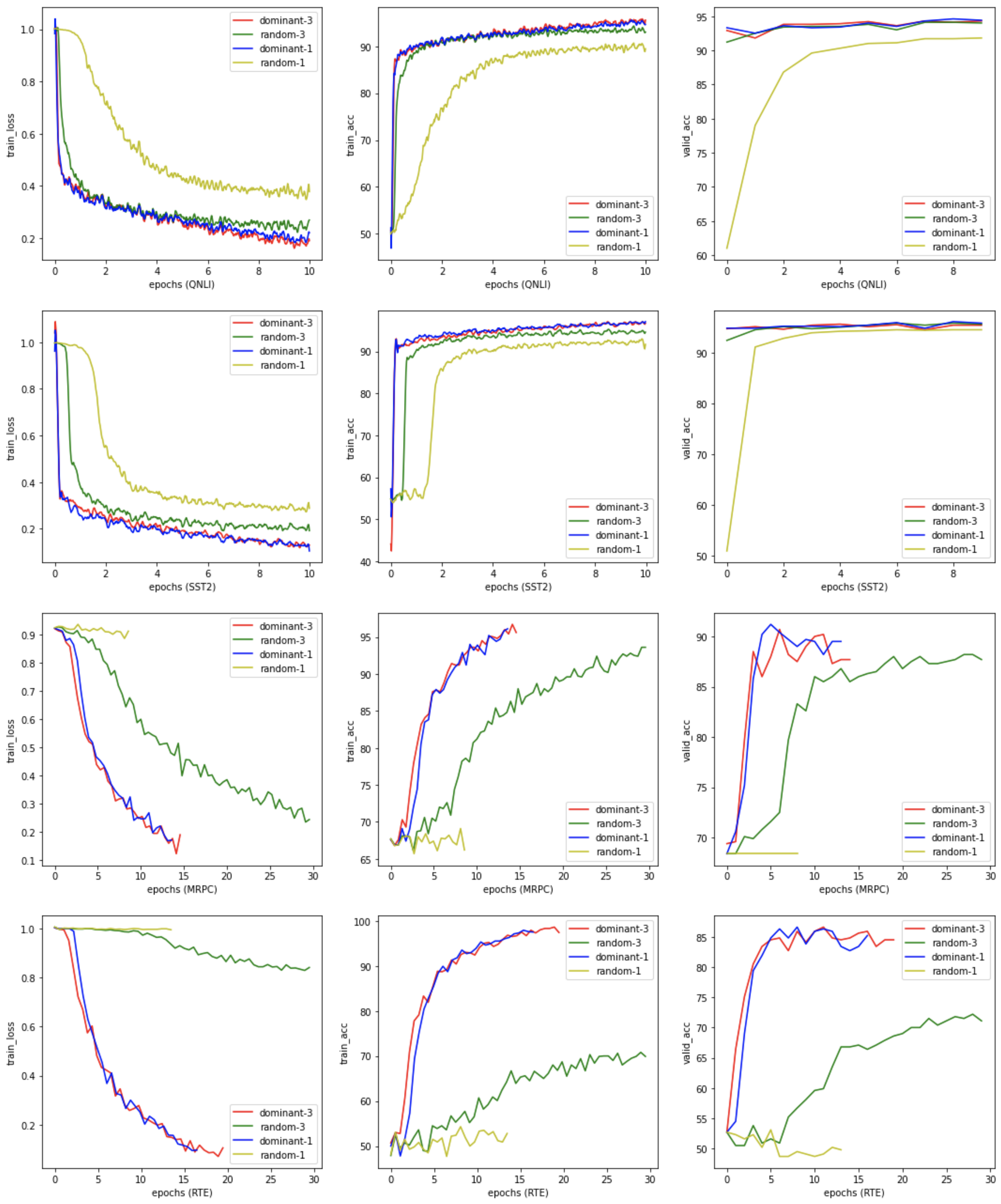

Figure 3: (Left) The training loss curves. (Middle) The training accuracy curves. (Right) The validation accracy curve. We visualize the training process for QNLI, MRPC, and RTE. For MPRC and RTE, early stopping is triggered if the validation accuracy does not increase for 8 successive epochs.

Table 2: The dominant winning ticket vs random ticket. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. All results are the average of 5 trials.

	Sparsity	**QNLI**	**SST-2**	**MNLI**	**CoLA**	**MRPC**	**STS-B**	**RTE**	**QQP**
		105k	67k	393k	8.6k	3.7k	5.8k	2.5k	364k
Full-ft	100%	94.7	96.4	90.4	68	90.9	92.4	86.4	92.2
Random-3	0.19%	94.1	96.2	89.8	68.6	89.4	91.8	72.2	91.1
Dominant-3	0.19%	94.3	96.3	90.5	69	90.2	92.2	86.6	91.7
BitFit (Zaken et al., 2021)	0.06%	94.5	96	86.7	66.3	89.7	92	86.3	88.9
Random-1	0.05%	91.7	94.4	85.4	51.4	74.8	88.7	68.5	88.3
Dominant-1	0.05%	94.6	96.1	90.4	69.7	90.9	92.2	87.7	91

Table 3: Performance comparison of full-parameter fine-tuning and 5%-parameter fine-tuning.

Params	**100%**	**5%**
QNLI	94.7	94.8
SST-2	96.4	96.6
MNLI	90.4	90.6
CoLA	68	69.7
MRPC	90.9	91.7
STS-B	92.4	92.2
RTE	86.4	88
QQP	91.9	91.6

Table 4: Dominated dimensions of W^O and W^{FC2}. Bias terms are highlighted with blue color.

Layer	W^O			W^{FC2}		
0	647	766	986	487	584	3291
1	593	929	587	1559	3646	1995
2	1024	579	237	3708	3058	2753
3	810	819	784	1591	480	609
4	101	1024	76	3445	1906	2828
5	1024	579	189	682	2123	841
6	1024	430	498	1267	2920	2686
7	692	670	699	2058	3485	1660
8	1024	534	69	2780	2387	1015
9	11	1024	608	3892	448	616
10	254	534	1024	1028	592	1462
11	330	758	1024	1014	3730	2157
12	151	909	1024	145	2436	2338
13	113	1024	151	902	2503	1700
14	141	1024	307	1219	1318	1535
15	1024	523	724	2413	608	15
16	1024	657	819	3454	4096	1288
17	254	640	209	3433	2	3962
18	986	999	966	628	2617	4096
19	657	1	845	1221	827	4096
20	975	453	1006	1442	4096	1669
21	1024	975	400	3290	4096	2850
22	559	1024	46	3341	3176	1078
23	167	871	891	2498	2596	1565

4.3 Stability of the Dominant Winning Ticket

We expect that subnetworks extracted from different L1-regularized fine-tuning runs (with different random seeds and different tasks) have similar structures. We use Jaccard similarity to measure the similarity between different tickets. Specifically, the Jaccard similarity between two sets is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{4}$$

We test four random seeds and four tasks (MNLI, QNLI, QQP, and SST-2). The average Jaccard similarity is 0.69 across seeds and 0.67 across tasks while the similarity between random tickets is nearly 0. This means that the dominant ticket has a stable structure that utilize a certain portion of pre-trained weights. This structure is task-agnostic and irrelevant to random seeds as observed in Section 3.2. Details about the structures of subnetworks can be found in Appendix A.1.

5 Discussions

5.1 When BERT Plays the Lottery, Are All Tickets Winning?

The lottery ticket hypothesis (LTH) states that dense, randomly-initialized networks contain subnetworks (winning tickets) that–when trained in isolation–reach performance comparable to the original network.

Initialization is an important factor in LTH as the winning ticket extracted from a specific initialization generally behaviors poorly on other random reinitialized networks (Frankle and Carbin, 2018). This phenomenon is often ignored when discussing LTH in PLMs as the initialization of PLMs is deterministic. PLMs initialize via pre-training weights, which contain rich information and are potentially useful. As a result, many subnetworks inside PLMs are potential winning tickets

Indeed, Prasanna et al. (2020) claimed that even the "bad" subnetworks in BERT ("good" and "bad") have acceptable performance. We further argue that any randomly sampled subnetwork with the sparsity of 95% can serve as the winning ticket. We randomly select 5% of the parameters in each matrix and freeze the rest of the model, surprisingly finding that the performance is comparable with full-parameter fine-tuning. As seen in Table 3, a simple random subnetwork requiring no sophisticated pruning techniques can already match with fine-tuning.

However, we still argue that not all tickets are winning. *The smaller the ticket is, the rarer the winning ticket is.* When the sparsity of the subnetwork increases, the performance of subnetworks begins to vary. For RoBERTa-large, at the sparsity of 99.95%, most of the "ticket" fail to win while a well-selected subnetwork (the dominant winning ticket) can suffer little performance drops. In this sense, the dominant winning ticket we found can be regarded as the smallest winning ticket.

5.2 Structured Winning Ticket

One astonishing fact about the dominant winning ticket is that it is naturally structured. When we perform L1-regularized fine-tuning, the L1 regularization is equally applied to all parameters. But weights in some dimensions tend to deviate from the pre-trained weights more than others. These dimensions, which we refer to as dominated dimensions, are shared among different layers (like the `673`, `631`, `474`, and `106` dimensions in Table 1). It seems like there is a "skeleton" inside the PLM that can serve as the dominant winning ticket.

The structure of the dominant winning ticket is different from other structured pruning studies in two ways. First, while their structures usually refer to certain parts of the model (e.g., channels in convolutional layers and attention heads in Transformers), the structure of the dominant winning ticket is micro within each parameter matrix. Second, while most of the structured subnetworks rely on structured pruning methods (Liang et al., 2021), we do not apply structure-aware regularization techniques. In other words, the structure of the dominant winning ticket is naturally formed, waiting to be found.

5.3 Connections with Bias-terms Fine-tuning

The dominant winning ticket has some connections with bias-terms fine-tuning (BitFit) (Zaken et al., 2021). The idea of BitFit is to fine-tune only the bias terms in the transformer, which only requires updating a very small subset of parameters. If we treat the bias term of an extended dimension of the weight matrix, then the subnetwork forms by the bias terms have a similar structure with the dominant winning ticket when k equals one (one trainable dimension per matrix). And the trainable parameter size is close too.

As can be seen in Table 2, BitFit is quite promising with small-to-medium training data. When the size of the training data is large, it still has acceptable performance. The overall performance of BitFit is much better than *Random-1*, indicating that the bias terms indeed catch some additional semantics. We find that the subnetwork of bias terms overlaps the dominant winning ticket. As shown in Table 4, the bias term serves as one of the dominated dimensions in W^O and W^{FC2} a lot. We think this might be an explanation of why only tuning the bias terms works well.

6 Implications and Future Work

The dominant winning ticket can be utilized for parameter-efficient fine-tuning. As the dominant winning ticket is stable across different tasks, when deploying PLMs on different scenarios, we only need to record the same small group of parameters. Besides, benefiting from the structure of the ticket, we only need to store the optimizer states for certain dimensions of each parameter matrix. With decent code implementation, we can promisingly reduce memory usage and speed up the training process. In future work, we would examine the memory reduction and speedup abilities of the dominant winning ticket.

Besides the practical value, the dominant winning ticket raises some interesting questions about PLMs. Now that the dominant winning ticket is intrinsically determined by the pre-trained weights, then *how does the subnetwork emerge during pre-training?* Another worth investigating point lies in the natural structure of the ticket. *Why certain dimensions of parameter matrices behave so differently from others?* We still know little about the mechanism behind these phenomenons. We aim to study these questions in future work.

7 Conclusion

In this paper, we reveal the existence of the dominant winning ticket inside pre-trained models and introduce the L1-regularized fine-tuning to extract it. The dominant ticket is an extremely sparse subnetwork that can reach comparable performance with fine-tuning the whole model. We observe that the ticket has some novel properties. First, it is stable across different random seeds and tasks, which means once identified on one task, it can be transferred to other tasks with no performance loss. Second, the ticket has a natural structure within each parameter matrix, and this structure is shared across layers. Our study not only has practical values for parameter-efficient fine-tuning but also raises some questions about the pre-trained models.

Acknowledgments

This work is supported in part by the National Hi-Tech RD Program of China (No.2020AAA0106600).

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

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*.

Xuxi Chen, Tianlong Chen, Yu Cheng, Weizhu Chen, Zhangyang Wang, and Ahmed Hassan Awadallah. 2021. Dsee: Dually sparsity-embedded efficient tuning of pre-trained language models. *arXiv preprint arXiv:2111.00160*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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 4365–4374.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *arXiv preprint arXiv:2108.04657*.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. *arXiv preprint arXiv:2105.12002*.

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

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *arXiv preprint arXiv:2106.04647*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32:14014–14024.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 506–516.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv e-prints*, pages arXiv–2004.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

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.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

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

A Appendix

A.1 Structure Comparison of Dominant Winning Tickets

Table 5: Structure comparison of W^Q.

Layer	MNLI			QNLI			SST			QQP		
0	304	673	981	673	981	1019	8	673	981	673	981	1019
1	106	673	981	106	673	981	106	673	981	106	673	981
2	106	673	981	106	673	981	106	673	981	106	673	981
3	106	673	981	106	673	981	106	673	981	106	673	981
4	106	673	981	106	673	981	106	673	981	106	673	981
5	106	673	981	106	673	981	106	673	981	106	673	981
6	106	673	981	106	673	981	106	673	981	106	673	981
7	106	673	981	106	673	981	106	412	673	106	412	673
8	106	412	673	106	412	673	106	412	673	106	412	673
9	2	412	673	2	412	673	106	412	673	2	412	673
10	412	474	673	2	412	673	2	412	673	2	412	673
11	2	93	673	2	547	673	474	623	673	2	412	673
12	93	547	673	2	547	673	474	547	673	2	547	673
13	2	547	673	2	547	673	51	547	673	2	547	673
14	547	631	673	2	547	673	2	547	673	547	631	673
15	547	631	673	547	631	673	547	631	673	547	631	673
16	547	631	673	547	631	673	547	631	673	2	631	673
17	547	631	673	547	631	673	547	631	673	547	631	673
18	547	631	673	547	631	673	547	631	673	547	631	673
19	547	631	673	256	631	673	547	631	673	547	631	673

Table 6: Structure comparison of W^K.

Layer	MNLI			QNLI			SST			QQP		
0	108	781	1019	673	981	1019	93	781	1019	328	981	1019
1	106	673	981	106	673	981	106	673	981	106	673	981
2	106	673	981	106	673	981	106	673	981	106	673	981
3	106	673	981	106	673	981	106	673	981	106	673	981
4	106	673	981	106	673	981	106	673	981	106	673	981
5	106	673	981	106	673	981	106	673	981	106	673	981
6	106	673	981	106	673	981	106	673	981	106	673	981
7	106	474	673	106	673	981	106	547	673	106	547	673
8	474	547	673	474	547	673	474	547	673	474	547	673
9	256	474	673	2	474	673	256	474	673	2	474	673
10	256	474	673	256	474	673	256	474	673	474	547	673
11	256	474	623	256	474	673	2	474	673	2	256	474
12	256	474	673	256	474	673	2	474	673	256	474	673
13	256	474	673	256	474	673	2	474	673	256	474	673
14	256	474	673	256	474	673	256	474	673	256	474	673
15	474	631	673	256	631	673	474	631	673	474	631	673
16	256	631	673	256	631	673	547	631	673	256	631	673
17	256	631	673	256	631	673	256	631	673	256	631	673
18	256	631	673	256	631	673	534	631	673	256	631	673
19	256	631	673	256	631	673	631	673	842	631	673	914

Table 7: Structure comparison of W^V.

Layer	MNLI			QNLI			SST			QQP		
0	472	673	981	8	673	981	328	673	981	8	673	981
1	106	673	981	106	673	981	106	673	981	106	673	981
2	106	673	981	106	673	981	106	673	981	106	673	981
3	106	673	981	106	673	981	106	673	981	106	673	981
4	106	673	981	106	673	981	106	673	981	106	673	981
5	106	673	981	106	673	981	106	673	981	106	673	981
6	106	673	981	106	673	981	106	673	981	106	412	673
7	106	412	673	106	412	673	106	412	673	106	412	673
8	412	474	673	412	474	673	106	474	673	412	474	673
9	412	474	673	412	474	673	412	474	673	412	474	673
10	412	474	673	412	474	673	412	474	673	412	474	673
11	412	474	673	2	474	673	412	474	673	412	474	673
12	474	547	673	2	474	673	140	474	673	474	547	673
13	474	547	673	2	474	673	474	547	673	474	547	673
14	474	547	673	474	547	673	474	547	673	474	547	673
15	474	631	673	474	631	673	474	631	673	474	631	673
16	547	631	673	547	631	673	547	631	673	547	631	673
17	547	631	673	547	631	673	547	631	673	547	631	673
18	547	631	673	547	631	673	547	631	673	547	631	673
19	399	631	673	547	631	673	631	673	842	547	631	673
20	399	631	673	547	631	673	547	631	673	547	631	673
21	547	631	673	547	631	673	547	631	673	547	631	673
22	547	631	673	547	631	673	547	631	673	547	631	673
23	631	673	914	547	631	673	547	631	673	41	631	673

Table 8: Structure comparison of W^O.

Layer	MNLI			QNLI			SST			QQP		
0	647	766	986	647	725	766	304	647	725	647	725	766
1	587	593	929	593	622	929	586	616	622	587	593	929
2	237	579	1024	237	579	612	448	579	1024	488	556	579
3	784	810	819	784	824	1024	589	784	1024	784	814	1014
4	76	101	1024	10	421	1024	367	494	730	648	696	1024
5	189	579	1024	225	579	1024	133	632	1024	203	204	232
6	430	498	1024	430	498	1024	430	498	1024	430	500	1024
7	670	692	699	665	692	1024	386	399	422	133	692	1024
8	69	534	1024	67	534	1024	442	512	574	67	464	1024
9	11	608	1024	11	608	1024	608	953	1024	11	830	1024
10	254	534	1024	254	534	1024	254	534	1024	254	534	1024
11	330	758	1024	330	758	1024	330	758	1024	330	758	843
12	151	909	1024	73	909	1024	73	669	1024	73	151	1024
13	113	151	1024	113	151	1024	113	941	1024	113	941	1024
14	141	307	1024	141	307	1024	141	307	1024	141	489	506
15	523	724	1024	523	722	1024	523	643	1024	488	523	1024
16	657	819	1024	132	183	519	111	519	1024	145	183	1024
17	209	254	640	567	640	657	74	107	125	871	984	1024
18	966	986	999	633	763	1024	763	889	891	207	986	1024
19	1	657	845	1	4	57	1	657	1024	1	405	657
20	453	975	1006	293	363	453	453	1007	1020	453	1002	1024
21	400	975	1024	437	519	903	901	975	1024	320	597	986
22	46	559	1024	331	559	790	483	559	1024	180	195	626
23	167	871	891	16	660	886	87	130	147	101	238	606

Table 9: Structure comparison of W^{FC1}.

Layer	MNLI			QNLI			SST			QQP		
0	673	981	1019	106	673	981	106	673	981	106	673	981
1	106	673	981	106	673	981	106	673	981	106	673	981
2	106	673	981	106	673	981	106	673	981	106	673	981
3	106	673	981	106	673	981	106	673	981	106	673	981
4	106	673	981	106	673	981	106	673	981	106	673	981
5	106	673	981	106	673	981	106	673	981	106	673	981
6	106	673	981	106	673	981	106	412	673	106	673	981
7	106	412	673	2	106	673	106	412	673	106	412	673
8	106	412	673	2	106	673	106	412	673	2	412	673
9	2	412	673	2	412	673	412	474	673	2	412	673
10	2	412	673	2	474	673	412	474	673	2	412	673
11	474	547	673	2	474	673	2	474	673	2	474	673
12	474	547	673	474	547	673	474	547	673	474	547	673
13	474	547	673	474	547	673	474	547	673	474	547	673
14	547	631	673	547	631	673	547	631	673	2	631	673
15	547	631	673	547	631	673	547	631	673	2	631	673
16	547	631	673	547	631	673	547	631	673	547	631	673
17	547	631	673	547	631	673	547	631	673	547	631	673
18	547	631	673	547	631	673	547	631	673	547	631	673
19	399	631	673	547	631	673	547	631	673	547	631	673
20	631	673	914	256	631	673	547	631	673	547	631	673
21	631	673	914	256	631	673	547	631	673	547	631	673
22	631	673	914	256	631	673	547	631	673	41	631	673
23	422	631	673	631	673	841	265	563	631	631	651	715

Table 10: Structure comparison of W^{FC2}.

Layer	MNLI			QNLI			SST			QQP		
0	487	584	3291	487	584	2390	487	584	3660	458	487	584
1	1559	1995	3646	1114	1559	4060	1559	3646	3912	1114	1559	4060
2	2753	3058	3708	445	3056	3708	870	3636	3708	445	3056	3708
3	480	609	1591	609	1177	1591	44	1591	1846	609	1177	1591
4	1906	2828	3445	1393	2828	3445	1773	3445	3940	1906	2828	3445
5	682	841	2123	682	841	3582	1298	1427	3624	682	841	2475
6	1267	2686	2920	425	1072	2200	909	1946	3027	425	2686	2920
7	1660	2058	3485	1660	2058	3485	317	1660	2058	1660	2058	3485
8	1015	2387	2780	1832	2780	2998	217	1247	2982	933	2387	2780
9	448	616	3892	34	1692	3892	34	1910	3892	34	1966	3892
10	592	1028	1462	592	1028	4096	592	1028	2063	592	1028	4096
11	1014	2157	3730	1014	2877	3730	1014	2877	3730	1014	2877	3730
12	145	2338	2436	145	2436	2754	145	1534	2338	145	2338	2436
13	902	1700	2503	902	1034	2503	902	1700	2503	902	1034	2503
14	1219	1318	1535	1219	1535	2787	1535	2963	3720	866	1535	3443
15	15	608	2413	15	64	2413	15	1997	2413	608	2413	3212
16	1288	3454	4096	1009	2827	3454	2639	2827	3454	1531	2827	4096
17	2	3433	3962	703	895	3235	699	2834	3475	703	1070	2588
18	628	2617	4096	1429	3091	4096	834	3130	3913	484	2982	4096
19	827	1221	4096	879	2409	3438	34	1207	4096	2556	3333	4096
20	1442	1669	4096	1299	2177	3491	1442	3329	4096	1880	2527	3894
21	2850	3290	4096	49	159	1116	2698	3290	4096	1426	3123	3290
22	1078	3176	3341	87	1669	3798	2365	2404	4096	1444	2576	3745
23	1565	2498	2596	1695	2544	3289	607	2010	2142	267	704	2084