Finding the Winning Ticket of BERT for Binary Text Classification via Adaptive Layer Truncation before Fine-tuning

Jing Fan\textsuperscript{1*}, Xin Zhang\textsuperscript{2*}, Sheng Zhang\textsuperscript{3*}
\textsuperscript{1}National University of Defense Technology
fanjing19@nudt.edu.cn, shinezhang_nudt@qq.com, zhangsheng@nudt.edu.cn

Abstract

In light of the success of transferring language models into NLP tasks, we ask whether the full BERT model is always the best and does it exist a simple but effective method to find the winning ticket in state-of-the-art deep neural networks without complex calculations. We construct a series of BERT-based models with different size and compare their predictions on 8 binary classification tasks. The results show there truly exist smaller sub-networks performing better than the full model. Then we present a further study and propose a simple method to shrink BERT appropriately before fine-tuning. Some extended experiments indicate that our method could save time and storage overhead extraordinarily with little even no accuracy loss.

1 Introduction

Recently, pretrain-then-finetune paradigm has dominated the NLP tasks. Some pre-trained language models, like BERT(Devlin et al., 2019), achieve state-of-the-art performance on numerous tasks, such as machine translation and sentiment analysis(Wang et al., 2018)(Yang et al., 2019a)(Yang et al., 2020a)(Hoang et al., 2019). For processing general knowledge as much as possible in pre-training stage, the language models are constructed with excess units (i.e., the total parameters in BERT\textsubscript{BASE} are 110M, and in BERT\textsubscript{LARGE} are 340M)(Devlin et al., 2019). Take BERT\textsubscript{BASE} as an example, training it from scratch takes 4 days on 4 to 16 Cloud TPUs, and even just fine-tuning on some complex downstream tasks requires several hours to finish one epoch(Strubell et al., 2019). That makes them impractical for resource-limited deployment scenarios and incurring cost concerns, and hinders the applications. These drawbacks force us to compress models and accelerate their calculations according to specific tasks(McCarley et al., 2019).

Existing model compression methods orienting to particular tasks fall into two categories, namely knowledge distillation(Bucilua et al., 2006) and pruning(Frankle and Carbin, 2018). For using knowledge distillation method, there needs to construct an ensemble or a teacher model and a much simple student. After the teacher fine-tuned thoroughly, the student imitates the teacher to predict and then makes decisions independently in test period. By doing so, resources like time and memory are saved in test period(Mirzadeh et al., 2020)(Yang et al., 2020b)(Wu et al., 2021)(Chia et al., 2019). Differently, pruning method only requires constructing one model, but mask parameters should be preserved to imitate dropping both in fine-tuning and testing. As a result, there almost costs more storage than ever before in test processing(Gordon et al., 2019)(Wang et al., 2020b)(Joulin et al., 2016)(Liu et al., 2018)(Voita et al., 2019).

Some common issues could be concluded that (1) they all target at accelerating calculations in test stage at the cost of more resource-consuming during fine-tuning. And (2) all the downstream tasks apply the standard language models indiscriminately without taking the complexities of different tasks into consideration.

According to the lottery ticket hypothesis(Frankle and Carbin, 2018), there is some redundancy in over-parameterization models. While some experiments about binary text classification tasks (like CoLA(Warstadt et al., 2019) and ChnSentiCorpHou(Tan and Zhang, 2008)) indicate that the standard BERT model is not always the best choice, and sometimes we could find sub-networks (winning tickets) surpassing the full BERT on some tasks via layer truncation (Details are provided in section 2). Motivated by this phenomenon, we propose to truncate BERT before fine-tuning and save resources both in fine-tuning and test stages. A simple way to estimate the truncating position is also provided by us. In addition, we conduct
some supplementary experiments on 8 binary text classification tasks (Binary text classification tasks are simple enough to highlight the issues and are the basis of most complex NLP tasks). The results show that our method is simple and effective.

In summary, our contributions include:

• We observe that full BERT is not always the best when fine-tuning on some particular downstream tasks. Sometimes shrinking BERT by simply truncating layers leads to capable even better performances with less overhead.

• We put forward a simple but effective truncation method to find out the winning ticket in BERT before fine-tuning, and provide a measure to estimate truncating position.

• Experiments in this paper show our method could save approximate 50% storage and time compared with original models both in fine-tuning and test, and remain near to 99% accuracy on average.

2 Full BERT is NOT always the best

Knowledge distillation and pruning methods accelerate testing of models by devoting additional resources during fine-tuning. That is impractical for some edge devices. In order to save cost both in fine-tuning and testing, we propose to truncate several consecutive transformer layers from top of the BERT (12-layer) and preserve the bottom part as a new encoder. In this paper, we reduce BERT ranging from 1 to 11 layers and obtain 12 different encoder. Each encoder is appended a task-specific fully-connected layer and a sigmoid layer. All these shrunk 12 models (namely trans1 to trans12) are fine-tuned on eight single-sentence classification tasks, including CoLA (Warstadt et al., 2019) and ChnSentiCorpHou (Tan and Zhang, 2008). More details of datasets, experimental settings and results are provided in Appendix A, B and C.

Here we take CoLA and ChnSentiCorpHou as examples and plot the test accuracy, training and test time of these 12 truncated models in Figure 1. There is no doubt that the training and test time are increasing dramatically as the number of transformer layer grows. However, we do not get far satisfying predictions with so many resources cost. With the increasement of the transformer layer, we find the accuracy is gradually fluctuating within a certain range. For CoLA (Figure 1 (left)), we find two winning tickets surpassing the full BERT when number of transformer layers are 9 and 10. While the models with 8 and 11 layers are almost capable of predicting as precise as full BERT-based model. As for ChnSentiCorpHou (Figure 1 (right)), although we do not find a winning ticket better than full BERT, models with 2 to 12 layers predict similarly. It seems that smaller models could perform well enough with less time and memory compared with full BERT.

3 Truncating BERT before fine-tuning

The results in section 2 show there exist truncated BERT models performing better than full BERT on some binary classification tasks. This inspires us that shrinking massive BERT for simple tasks is valuable and necessary. In order to clearly understand this phenomenon and find a way to solve problems well with less time and memory, we further take a closer look at these results. In this section, we will provide some discussion about when, where and how to truncate.

3.1 Where to truncate

As discussed in section 2, the test accuracy becomes fluctuating when the number of transformer layer larger than some certain values. We then compare the predictions of each model and capture something interesting. When the test accuracies of two models are similar, the predictions on the same instance are also close. In addition, all these 12 models provide consistent predictions on most of the instances, and divisions only arise over a few indistinguishable ones. While these divisions would become less and less with the test accuracies being closer.

Specifically, given a dataset $\mathcal{S}$ containing $n$ examples $x_1, x_2, \ldots, x_n$. We construct 12 truncated BERT-based classification models, namely $M_1$ to $M_{12}$ (here, the subscript $i$ of $M$ denotes the number of transformer layers), to make predictions on the $\mathcal{S}$. The test accuracy of a model $M_i$ is denoted as $acc_i(i = 1, \ldots, 12)$. And the calculation of $M_i$ on $x_p(p = 1, \ldots, n)$ is formulated as:

$$
M_i(x_p) = \sigma(g_i(h_i(x_p))),
$$

$$
h_i(x_p) = f_i(f_{i-1}(\cdots(f_1(x_p))\cdots)).
$$

Where $\sigma$ is the sigmoid function, $g_i(\cdot)$ is the function of a fully-connected layer, $h_i(\cdot)$ is the function of the encoder, and $f_i(\cdot)$ is the function of
the $i$-th transformer encoder layer. As mentioned above, when $acc_i$ is close to $acc_j$, we find $M_i(x_p)$ is also close to $M_j(x_p)$ ($p = 1, \cdots, n; i, j = 1, \cdots, 12, \text{and } i < j$). For classification task, we know $\sigma(g_i(\cdot))$ is a generalized linear transformation targeting at finding a hyperplane separating examples best without transforming the representations. Naturally, the prediction $M_i(\cdot)$ mainly depends on the transformation of the encoder $h_i(\cdot)$. The higher separability of the hidden states produced by the encoder is, the higher prediction probability $M_i(\cdot)$ and the higher test accuracy $acc_i$ of the model are. As a result, we could use the hidden states to help us decide where to truncate.

Inspired by the distribution of points in high dimensional space, we treat the output representations of transformer layers in encoder as coordinates of point, and apply the separability theory of point set to expressing the distribution of examples in our datasets. We truncate BERT targeting at producing representations of dataset with max separability.

Here, we introduce three widely used separability measures, namely Class Scatter Matrices (CSM)(Duda et al., 1973), Thornton’s Separability Index (SI)(Thornton, 2002), and Hypothesis Margin (HM)(Gilad-Bachrach et al., 2004).

**Class Scatter Matrices, CSM**: is a fraction of trace of between-cluster scatter and within-cluster scatter. It evaluates the overall distribution of an entire dataset.

\[
CSM = \frac{tr(S_B)}{tr(S_W)},
\]

\[
S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^T,
\]

\[
S_W = \sum_{i=1}^{c} \sum_{j=1}^{n_i} (x_{ij} - m_i)(x_{ij} - m_i)^T.
\]

Where $tr(\cdot)$ denotes the trace operation: calculating the sum of one’s diagonal elements. $c$ and $n_i$ are the number of classes in a dataset and number of examples in class $i$ respectively. While $m$ and $m_i$ are the mean vectors of whole dataset and class $i$ separately. $x_{ij}$ is the $j$-th instance in class $i$. The within-cluster scatter and between-cluster scatter are denoted as $S_W$ and $S_B$.

**Separability Index, SI**: calculates the average number of instances that share the same category label as their nearest neighbors. It is a measure evaluating the class overlap.

\[
SI = \frac{\sum_{i=1}^{n}(\phi(x_i) + \phi(x'_i) + 1) \bmod 2}{n},
\]

Where $\phi(x_i), \phi(x'_i) \in \{0, 1\}$ is the label of $x_i$, $x'_i$ is the nearest neighbor of $x_i$, and $n$ is the number of examples in a dataset.

**Hypothesis Margin, HM**: measures the sum of the distance between the hypothesis and the closest hypothesis that assigns the alternative label to the
given instance.

\[ HM = \sum_{i=1}^{n} \frac{1}{2} (\|x_i - \text{nearmiss}(x_i)\| - \|x_i - \text{nearhit}(x_i)\|), \]  

(7)

Where \( \text{nearhit}(x_i) \) and \( \text{nearmiss}(x_i) \) denote the nearest instance to \( x_i \) with the same and different label, respectively. And \( n \) is the number of instances in a dataset. \( \| \cdot \| \) denotes the Euclidean norm.

According to the definitions above mentioned, the larger these three separability values are, the more easily the examples are classified. We could apply our truncation method with the assistance of the separability measures.

3.2 When to truncate
As we all known, BERT is pre-trained on two general task, namely masked language model (MLM) and next sentence prediction (NSP). It means there is little disturbance of wrong prediction corresponding to specific task brought in. Making decisions on the basis of original pre-trained BERT is more objective.

What’s more, once the fine-tuning processing starts, the predicting error would radially propagate back to each layer of the entire network. Breaking the network during this stage leads to reconstruct new matching relations among whole net, which is far more time-consuming.

On account of these factors, we decide to truncate BERT before fine-tuning. If we can make a reliable decision about where to truncate before fine-tuning, resources like time and storage would be saved extraordinarily.

3.3 How to truncate
In this sub-section, we provide a general algorithm applying separability measures to truncating BERT (12-layer) properly for downstream tasks (Algorithm 1).

Given a target binary classification task and the corresponding dataset, we can estimate where to truncate before fine-tuning without building up the network. Our three-step method is simple enough and time-saving. In particular, all the instances in the dataset should be vectorized with BERT embeddings in the first step. Secondly, the vectorized instances are sent into the pre-trained BERT model and propagate forward once without backward propagation. In this step, we will obtain 12 hidden representations from different transformer layers of the BERT for every instance. The hidden representations produced by the same transformer layer are packed into a common set. Then in the third step, we calculate separability values for these 12 representation sets. The number of transformer layers related to the max separability is the number of layers in the truncated BERT. We truncate BERT according to that and append the task-specific layer on to form the final model.

### Algorithm 1: Truncating algorithm

Data: \( S = \{x_1, x_2, \ldots, x_n\} \)

1. for \( x_i (i = 1, \ldots, n) \) in \( S \) do
2. Vectorize \( x_i \) as \( x_i \);
3. Send \( x_i \) in BERT;
4. for each transformer layer \( l_j (j = 1, \ldots, 12) \) in BERT do
5. Calculate hidden state \( h_{ij} \);
6. Collect \( h_{ij} \) into \( v_j (v_j = \{h_{1,j}, \ldots, h_{n,j}\}) \);
7. end
8. end
9. for each \( v_j (j = 1, \ldots, 12) \) do
10. Calculate separability value (csm\(_j\), si\(_j\) or hm\(_j\));
11. end
12. Truncate BERT according to the max separability value

In this way, we can truncate BERT adaptively before fine-tuning according to target tasks. Putting these truncated BERT into force could save as much as time and memory with little expense.

4 Experiments
Observations in section 2 inspires us that the full BERT is not always the best choice for some binary classification tasks. And in section 3 we provide a simple but clear analysis about this phenomenon, and introduce an effective method to shrink BERT for downstream tasks. In this section we will apply our method to 8 binary classification tasks and give a further exploration.

4.1 Datasets and tasks
In this sub-section, we will give a brief introduction about 8 corpora used in this paper, including ChnSentiCorpHou, CU, SST-2, CoLA, Nikon-JD, AFQMC, MRPC and RTE. The ChnSentiCorpHou is a sub-set of a series of Chinese sentiment...
Corpus | Size | Input Style | Task | language | Degree of Difficulty
--- | --- | --- | --- | --- | ---
ChnSentiCorpHou | 6k | single sentence | sentiment classification | Chinese | ★
CU | 6k | single sentence | event detection | Chinese | ★
SST-2 | 67.8k | single sentence | sentiment classification | English | ★★★
CoLA | 9k | single sentence | acceptability analysis | English | ★★★★
Nikon-JD | 2.6k | sentence pair | aspect-level sentiment classification | Chinese | ★★
AFQMC | 38.6k | sentence pair | semantic similarity | Chinese | ★★★
MRPC | 5.8k | sentence pair | semantic similarity | English | ★★★
RTE | 2.7k | sentence pair | natural language inference | English | ★★★★

Table 1: Corpora information. All tasks are binary text classification with a single sentence or sentence pair as the input. In the last column, the more the stars are, the more difficult the task is.

31 analysis corpora organized by Tan et al. (Tan and Zhang, 2008). The CU is a binary classification corpus about civil unrest event organized by Xu et al. (Xu et al., 2019). The SST-2 (Socher et al., 2013), CoLA (Warstadt et al., 2019), MRPC (Dolan and Brockett, 2005) and RTE (Dagan et al., 2005) (Haim et al., 2006) (Giampiccolo et al., 2007) (Bentivogli et al., 2009) come from the general language understanding evaluation (GLUE) benchmark (Wang et al., 2018), and AFQMC \(^1\) is selected from the Chinese language understanding evaluation (CLUE) benchmark (Xu et al., 2020). The Nikon-JD corpus is organized by our team. More details of these datasets are provided in Appendix A. In Table 1, we list some critical information about these corpora. We select two Chinese and two English single sentence classification tasks. In addition, two Chinese and two English sentence pair classification tasks are also chosen. The assignments in this section vary from sentiment classification, event detection and acceptability analysis to semantic similarity and natural language inference. Different assignment is related to different size of dataset and difficulty degree.

**4.2 Implementations**

We select the 12-layer BERT model as our basic language model and apply the truncation method on these 8 datasets. For each single sentence task, we put a [CLS] and [SEP] at the beginning and the end of the given text separately. And for sentence pair task, we also concatenate two texts into a sequence with [SEP] token being the separator, besides the special pre-processing at beginning and end. Then send processed sequences into the BERT for following procedures.

Before constructing models and fine-tuning, we utilize whole dev sets of ChnSentiCorpHou, CU, Nikon-JD, MRPC and RTE, and randomly sample about 50%, 50% and 10% examples from dev sets of SST-2, CoLA and AFQMC separately. Then we calculate CSM, SI and HM with these prepared instances for each task, and according to that we truncate BERT model adaptively. Next, we construct classification models based on the truncated BERT models, then fine-tune and test them on above mentioned corpora. More experimental details are in Appendix B.

For comparison, we also set up some baselines:

**BERT base**: We construct binary classification models based on BERT\(_{BASE}\) (Devlin et al., 2019) as the standard models.

**Distil-Pretrain**: We build up classification models based on the distilled language models (DistilBERT (Sanh et al., 2019) and albert-chinese-tiny\(^2\)). In this way, the knowledge distillation theory is used in pre-training period.

**Distil-task-biLSTM**: We fine-tune a BERT-based classification model as the teacher, and construct a student using the biLSTM as the encoder with the assistance of the attention mechanism. Then let the student imitates prediction of the teacher.

**Distil-task-kimCNN**: We construct a student according to the kimCNN (Kim, 2014). While the teacher models are same as those of Distil-task-biLSTM.

**Random truncation**: randomly truncation means selecting a random number ranging from 0 to 11 to truncate. Here, we use the mean test accuracy of 12 various truncated BERT-based models to denote the random accuracy.

Note that all these baselines use the BERT embeddings, the batch size and max sequence length are same as those of our models (Table 6). Differently, during implementing task-specific distillation, we set the learning rate as 1e-3.

\(^1\)https://dc.cloud.alipay.com/index#/topic/intro?id=3
\(^2\)https://huggingface.co/clue/albert_chinese_tiny
Table 2: Pearson’s correlation coefficient of CSM, SI and HM.

|               | CSM | SI  | HM  |
|---------------|-----|-----|-----|
| ChnSentiCorpHou | 0.7942 | 0.8204 | 0.4208 |
| CU            | 0.5949 | 0.4435 | 0.1898 |
| SST-2         | 0.8697 | -0.1703 | 0.5242 |
| CoLA          | 0.9655 | 0.5444 | 0.8269 |
| Nikon-JD      | 0.9517 | 0.8014 | 0.7963 |
| AFQMC         | 0.7709 | -0.2119 | 0.7073 |
| MRPC          | 0.2544 | 0.7389 | 0.4105 |
| RTE           | 0.8720 | 0.1090 | -0.2889 |
| Avg           | 0.7591 | 0.3844 | 0.4483 |

4.3 Results

In order to evaluate whether our separability measures are appropriate or not and then choose a better one for truncating BERT, we firstly plot three measures with the test accuracies in Figure 2 (Here we only provide the results of CoLA and ChnSentiCorpHou, while others are shown in Appendix D). Obviously, we could easily find that the tendencies of these three measures are similar to the test accuracies to some extent. As revealed in the Equation 3-7, distance-based values (CSM and HM) are more exact than the one based on number of instances (SI), while HM is easily influenced by some specific examples and loses effectiveness. Among these three measures, the CSM is more stable. To make a quantitative analysis, we use the Pearson’s correlation coefficient (PCC) to evaluate the relations. The average of PCC in Table 2 also indicates that the CSM is more relevant to test accuracy. As a result, we choose the CSM measure to help us truncate BERT.

As shown in Figure 2 and Appendix D, when the number of transformer layers are 7, 6, 12, 10, 11, 12, 5 and 9, the CSM values reach maximum on ChnSentiCorpHou, CU, SST-2, CoLA, Nikon-JD, AFQMC, MRPC and RTE, respectively. So we accept these suggestions and truncate BERT for these tasks. In this way, the test accuracies of our truncated model are 93.05%, 96.58%, 92.26%, 85.96%, 97.31%, 74.12%, 76.52% and 63.68%.

The test accuracy, processing time and scale of parameters are listed in Table 3, Table 4 and Table 5 respectively. Obviously, our method, Truncated-CSM, is capable of the 99% language understanding capabilities of the full model in average, and almost surpasses the other compressing methods on all these tasks except the MRPC. Also, about 50% of time and memory are saved both in fine-tuning and test. What’s more, our method is simple to implement and transfer to other tasks. On the contrary, we can draw conclusions from Table 4 and Table 5 that distillation methods are too complex to apply and consume far more time and storage from building up the teacher model to obtaining the student. The pruning methods are of the same circumstances. The overhead of knowledge distillation and pruning methods in fine-tuning is awful and unbearable. However, we do not get satisfying predictions with these two kinds of ways.

According to these results, we confirm our method is effective and provide the following suggestions:

- Only 1 layer of transformer is enough to perform well on most simple tasks.
- For some complex tasks, a few more layers (3-5 layers) could produce satisfying results.
- More exact estimation could be made by randomly sampling some instances to calculate CSM value.

5 Related Work

In this section, we briefly review some work about pre-trained language models, knowledge distillation and model pruning.

Pre-trained language models: Since about 2017, there gradually appear some complex language models. The widely used BERT proposed by Devlin et al.(Devlin et al., 2019) achieves pretty good results on GLUE(Wang et al., 2018), SQuAD(Rajpurkar et al., 2016) and SWAG(Zellers et al., 2018). Before long, XLNet(Yang et al., 2019b), RoBERTa(Liu et al., 2019) and some others(Wang et al., 2021)(Shen et al., 2020)(Zafir et al., 2019) emerge and push the evaluation scores of NLP tasks much higher. With the development of pre-trained language models, a wide range of NLP tasks obtain new state-of-the-art results again and again, including question answering(McCarley et al., 2019), language inference(Williams et al., 2018)(Levesque et al., 2012), text classification(Joulin et al., 2016) and text summarization(Barzilay and Elhadad, 1999). Consequently, the language models has been an indispensable part in solving NLP assignments.

Knowledge distillation: Knowledge distillation is a method that can transfer knowledge from an
Table 3: Test accuracy of 8 tasks in this paper (including 2 tasks in section 2). Bond font in the last row indicates the test accuracy of our truncated model is larger than other methods (except BERT base).

| Model       | ChnSentiCorpHou | CU | SST-2 | CoLA | Nikon-JD | AFQMC | MRPC | RTE |
|-------------|-----------------|----|-------|------|----------|-------|------|-----|
| BERT base   | 93.30           | 96.82 | 92.26 | 83.87 | 97.01    | 74.12 | 83.59 | 65.81 |
| Distil-Pretrain | 91.17        | 96.18 | 91.21 | 81.40 | 90.72    | 51.88 | 82.09 | 59.56 |
| Distil-task-biLSTM | 87.50    | 95.48 | 87.10 | 68.31 | 89.32    | 64.18 | 64.17 | 50.88 |
| Distil-task-kimCNN | 86.48     | 94.02 | 85.23 | 70.59 | 91.29    | 63.51 | 67.59 | 52.08 |
| Random truncation | 92.05    | 96.55 | 90.43 | 77.29 | 95.66    | 70.90 | 76.68 | 64.21 |
| Truncated-CSM | **93.05**   | **96.58** | **92.26** | **85.96** | **97.31** | **74.12** | **76.52** | **63.68** |

Table 4: Processing time of 8 tasks in this paper (including 2 tasks in section 2). For distillation-based method, it includes fine-tuning time of teacher model and distillation time of student model. While for truncation-based method, it includes calculation time of separability measure and fine-tuning time. The unit of measurement is second (s). Because the total time of Distil-Pretrain and Random truncation are not available, here we do not provide related time.

| Model       | ChnSentiCorpHou | CU | SST-2 | CoLA | Nikon-JD | AFQMC | MRPC | RTE |
|-------------|-----------------|----|-------|------|----------|-------|------|-----|
| BERT base   | 102             | 102 | 109   | 109  | 102       | 102   | 109  | 109 |
| Distil-Pretrain | -             | -  | -     | -    | -        | -     | -    | -   |
| Distil-task-biLSTM | 102+9        | 102+9 | 109+9 | 109+9 | 102+9    | 102+9 | 109+9 | 109+9 |
| Distil-task-kimCNN | 102+7       | 102+7 | 109+7 | 109+7 | 102+7    | 102+7 | 109+7 | 109+7 |
| Random truncation | -            | -  | -     | -    | -        | -     | -    | -   |
| Truncated-CSM | 67             | 60  | 109   | 95   | 102       | 53    | 88   | -   |

Table 5: Scale of parameters of 8 tasks in this paper (including 2 tasks in section 2). All values are scaled by 1M.
ensemble or a large highly regularized model into a much smaller, distilled model. This concept is proposed by Caruana et al. (Bucilă et al., 2006) and further extended to Neural Networks by Hinton’s team (Hinton et al., 2015). Some like SANH et al. (Sanh et al., 2019) and Jiao et al. (Jiao et al., 2020) only use this theory to compress models and accelerate processing. More are inspired to not only reduce the models but also transfer extra knowledge by means of changing input style or other factors during distillation (Ma et al., 2020) (Chen et al., 2020) (Wang et al., 2020) (Chen et al., 2019) (Sun et al., 2020).

**Model pruning:** Existing model pruning work could classify into two categories, namely structured pruning and unstructured pruning. The structured pruning means removing models by dropping several entire layers or some coherent groups of blocks (Wang et al., 2020) (Michel et al., 2019) (Fan et al., 2019). In contrast, unstructured pruning is a form of dropout that compresses model by reducing weights independently (Frankle and Carbin, 2018) (Gordon et al., 2019) (Prasanna et al., 2020) (Chen et al., 2020).

### 6 Conclusion

In this paper, we conduct a series of experiments to exploring whether it exist winning tickets when transferring the language models in downstream binary classification tasks and how to find them conveniently and effectively. Different from conventional model compressing methods, we insist to truncate BERT by a brute force mean. In addition, a simple way to estimate the size of BERT before fine-tuning is provided by us. In this way, the resources are saved tremendously without losing too much accuracy. More effectively, some simple task could be solved only use one transformer layer sometimes.

Currently, there is a little work studying the over-parameterization problem of state-of-the-art networks based on language models. And researchers fail to comprehend and explain these complex and massive deep neural networks. Also, there is little work solving the resource-consuming problem well. However, these problems should be attached importance to and solved well before the development of neural models reaching a higher level. Now we throw away a brick in order to attract more researchers to pay attention to then solve these problems.

In future, we will extend these experiments to other more complex NLP tasks and other language models, and explore more effective ways to find winning tickets in language models. Also, we are eager to find out more effective ways to solve these problems better.

### References

Regina Barzilay and Michael Elhadad. 1999. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. 2020a. Online knowledge distillation with diverse peers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3430–3437.

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

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2019. Distilling knowledge learned in bert for text generation. *arXiv preprint arXiv:1911.03829*.

Yew Ken Chia, Sam Witteveen, and Martin Andrews. 2019. Transformer to cnn: Label-scarce distillation for efficient text classification. *arXiv preprint arXiv:1909.03508*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT* (1).

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

Richard O Duda, Peter E Hart, and David G Stork. 1973. *Pattern classification and scene analysis*, volume 3. Wiley New York.
Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. In International Conference on Learning Representations.

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

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing, pages 1–9.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. 2004. Margin based feature selection-theory and algorithms. In Proceedings of the twenty-first international conference on Machine learning, page 43.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2019. Compressing bert: Studying the effects of weight pruning on transfer learning.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. stat, 1050:9.

Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. 2019. Aspect-based sentiment analysis using bert. In Proceedings of the 22nd Nordic Conference on Computational Linguistics, pages 187–196.

Xiaoli Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 4163–4174.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

Y. Kim. 2014. Convolutional neural networks for sentence classification. Eprint Arxiv.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning.

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.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. In International Conference on Learning Representations.

Xinyin Ma, Yongliang Shen, Gongfan Fang, Chen Chen, Chenghao Jia, and Weiming Lu. 2020. Adversarial self-supervised data free distillation for text classification. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6182–6192.

JS McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a bert-based question answering model. arXiv preprint arXiv:1910.06360.

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

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 5191–5198.

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.

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.

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.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

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.
A Datasets Information

Here, we provide more exact descriptions about corpora used in this paper.

**ChnSentiCorpHou**: The ChnSentiCorpHou is a Chinese sentiment corpus about house comments collected by Tan et al. (Tan and Zhang, 2008). It consists of 3000 positive and 3000 negative reviews about hotel. We conduct our single sentence classification task on it. When given a comment, we should predict the corresponding polarity.
CU: The civil unrest (Xu et al., 2019) dataset consists of thousands of declarative sentences. According to whether a sentence is a description about a civil unrest event, it is labeled as positive or negative example. There are 3000 positive and 3000 negative instances in this dataset.

SST-2: The Stanford Sentiment Treebank (Socher et al., 2013) consists of about 70k English sentences from movie reviews. These sentences are annotated with positive or negative polarity. The target of this task is to predict the sentiment category when given a sentence.

CoLA: The Corpus of Linguistic Acceptability (Warstadt et al., 2019) consists of a plenty of English acceptability judgments drawn from books and journal articles on linguistic theory. There are 10k examples in this corpus, and each of which is a sentence annotated with a binary label indicating whether it is a grammatical English sequence or not. Note that for privacy limitation, we only utilize the training and dev set.

Nikon-JD: The Nikon-JD is a Chinese aspect-level sentiment classification dataset consisting of 2.6k product reviews crawled from E-commerce platform, namely JingDong³. We organize 3 professors annotating these reviews. Given a review and a target aspect category, all those 3 annotators should make a decision whether the review expresses a positive polarity or not toward the aspect. Then we label it in terms of the majority.

AFQMC: The Ant Financial Question Matching Corpus (Xu et al., 2020) comes from the Ant Technology Exploration Conference Developer competition. There are 42.5k instances in this corpus, each of which is a sentence pair. It is a binary classification task that targets at predicting whether a pair of sentences are semantically equivalent. There are approximately 5.8k pairs in this corpus.

RTE: The Recognizing Textual Entailment (RTE)(Wang et al., 2018) dataset consists of 2.7k sentence pairs. Each pair of sentences is labeled with a binary category signifying whether one sentence could entail the other or not.

### B Experimental Settings

Full ChnSentiCorpHou, CU, SST-2 and MRPC corpora are obtained from the original sources. Due to the privacy limitation, we only get the training and dev set in CoLA, AFQMC and RTE. For CoLA and AFQMC, we treat the original dev sets as our new test sets and randomly sample 10% instances from the original training sets forming the new dev sets while the remaining examples forming the new training sets. Then we fine-tune classification models on the training and dev sets and implement test on the test sets for SST-2, CoLA, AFQMC and MRPC. We perform 5-fold cross-validations on the remaining four datasets, namely ChnSentiCorpHou, CU, Nikon-JD and RTE, which means partitioning the examples into 5 parts randomly, choosing a different fold as the test set with accuracy calculated on, and leaving the remaining 4 folds as training set each time. In the evaluation step, we use the average accuracy of 5 folds as the final result.

We use 12-layer BERT-wwm-ext⁴ and BERT-base-uncased⁵ as our basic language models for Chinese and English tasks separately. All the classification models are fine-tuned for 5 epochs. The batch size and other information are listed in Table 6. Following each epoch of training, we give a test on dev set and record the better model. Finally, the one performs best on dev set will be saved and applied.

### C Accuracy curves

In Figure 3, we plot the test accuracy, training time and test time of the other 6 datasets. According to these curves, we confirm that there truly exist some smaller sub-networks surpassing the full BERT model on some downstream tasks. The more difficult the task is, the more complex the corresponding model should be.

3https://www.jd.com/

4https://huggingface.co/hfl/chinese-bert-wwm-ext

5https://huggingface.co/bert-base-uncased
Figure 3: Accuracy, training time and test time of trans1-12. The reservation of accuracy arrives decimally hind two, and training and test time are rounded to the nearest whole number.
D Separability measures and test accuracy

In this section, we provide the comparisons of separability measures and test accuracy of other 6 corpora. Similar as the Figure 2, we also capture consistent trends between the CSM measure and test accuracy on these binary classification tasks. That means our truncation method assisted by CSM is resonable and effective.
Figure 4: CSM, SI and HM Curves of CU and SST-2. The test accuracy is shown with a royal blue line, and the separability value is drawn with a red line.
Figure 5: CSM, SI and HM Curves of Nikon-JD and AFQMC. The test accuracy is shown with a royal blue line, and the separability value is drawn with a red line.
Figure 6: CSM, SI and HM Curves of MRPC and RTE. The test accuracy is shown with a royal blue line, and the separability value is drawn with a red line.