Knowledge Distillation

- Idea : • learn a highly cumbersome model but then distill that knowledge into a nimbler model
  - model for learning doesn't have to be the model for serving
- Whats a neural nets "knowledge"?
  - it's not the learned parameters -> independent from architectures
  - learned mapping from input to output
  - relative probabilties of all the different classes
- KD loss

  double objective works best

  learn to predict correct class + learn to generalize
- Soft labels / targets

  instead of using hard targets from the training data we are taking the outputs from the teacher model as soft targets

Patient Knowlege Distillation

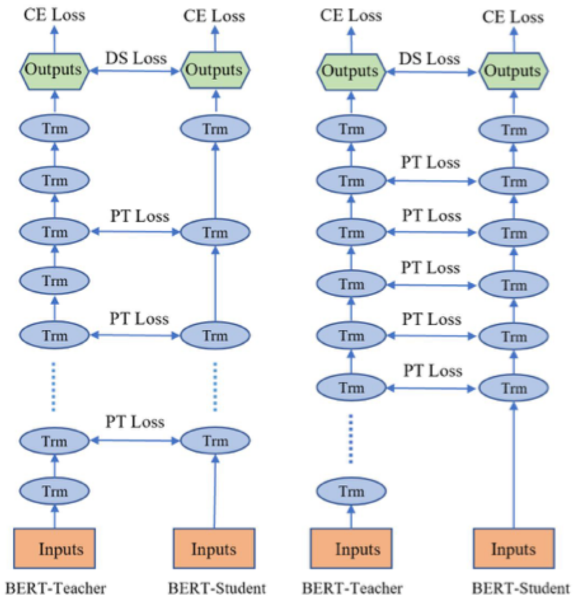
- don't use only the teachers last layer for distillation use multiple intermediary layers aswell
  - PKD-last : look at the last k-layers
  - PKD-skip : skip every k-layers ← usually works better
- BERT is problematic

  -> huge, millions of parameters

  focus on BERT for classification : [CLS]

⇒ Vanilla KD (only using the last layer) doesn't cut it

⇒ Instead Patient KD:



(a) PKD-*Skip* (b) PKD-*Last*

- Mean-sq loss between normalized hidden states
- Student only imitates the hidden states for the [CLS] token

Patient Knowledge Distillation for BERT Model Compression

Siqi Sun, Yu Cheng, Zhe Gan, Jingjing Liu
Microsoft Dynamics 365 AI Research
{Siqi.Sun,Yu.Cheng,Zhe.Gan,jingjl}@microsoft.com

Abstract

Pre-trained language models such as BERT have proven to be highly effective for natural language processing (NLP) tasks. However, the high demand for computing resources in training such models hinders their application in practice. In order to alleviate this resource hunger in large-scale model training, we propose a Patient Knowledge Distillation approach to compress an original large model (teacher) into an equally-effective lightweight shallow network (student). Different from previous knowledge distillation methods, which only use the output from the last layer of the teacher network for distillation, our student model *patiently* learns from multiple intermediate layers of the teacher model for incremental knowledge extraction, following two strategies: (i) PKD-Last: learning from the last k layers; and (ii) PKD-Skip: learning from every k layers. These two patient distillation schemes enable the exploitation of rich information in the teacher's hidden layers, and encourage the student model to *patiently* learn from and imitate the teacher through a multi-layer distillation process. Empirically, this translates into improved results on multiple NLP tasks with significant gain in training efficiency, without sacrificing model accuracy.[1]

1 Introduction

Language model pre-training has proven to be highly effective in learning universal language representations from large-scale unlabeled data. ELMo (Peters et al., 2018), GPT (Radford et al., 2018) and BERT (Devlin et al., 2018) have achieved great success in many NLP tasks, such as sentiment classification (Socher et al., 2013), natural language inference (Williams et al., 2017), and question answering (Lai et al., 2017).

Despite its empirical success, BERT's computational efficiency is a widely recognized issue because of its large number of parameters. For example, the original BERT-Base model has 12 layers and 110 million parameters. Training from scratch typically takes four days on 4 to 16 Cloud TPUs. Even fine-tuning the pre-trained model with task-specific dataset may take several hours to finish one epoch. Thus, reducing computational costs for such models is crucial for their application in practice, where computational resources are limited.

Motivated by this, we investigate the redundancy issue of learned parameters in large-scale pre-trained models, and propose a new model compression approach, *Patient Knowledge Distillation* (Patient-KD), to compress original teacher (e.g., BERT) into a lightweight student model without performance sacrifice. In our approach, the teacher model outputs probability logits and predicts labels for the training samples (extendable to additional unannotated samples), and the student model learns from the teacher network to mimic the teacher's prediction.

Different from previous knowledge distillation methods (Hinton et al., 2015; Sau and Balasubramanian, 2016; Lu et al., 2017), we adopt a *patient* learning mechanism: instead of learning parameters from only the last layer of the teacher, we encourage the student model to extract knowledge also from previous layers of the teacher network. We call this 'Patient Knowledge Distillation'. This patient learner has the advantage of distilling rich information through the deep structure of the teacher network for multi-layer knowledge distillation.

We also propose two different strategies for the distillation process: (i) PKD-Last: the student learns from the last k layers of the teacher, under the assumption that the top layers of the original network contain the most informative knowledge

[1] Code will be avialable at https://github.com/intersun/PKD-for-BERT-Model-Compression.

to teach the student; and (ii) PKD-Skip: the student learns from every k layers of the teacher, suggesting that the lower layers of the teacher network also contain important information and should be passed along for incremental distillation.

We evaluate the proposed approach on several NLP tasks, including Sentiment Classification, Paraphrase Similarity Matching, Natural Language Inference, and Machine Reading Comprehension. Experiments on seven datasets across these four tasks demonstrate that the proposed Patient-KD approach achieves superior performance and better generalization than standard knowledge distillation methods (Hinton et al., 2015), with significant gain in training efficiency and storage reduction while maintaining comparable model accuracy to original large models. To the authors' best knowledge, this is the first known effort for BERT model compression.

2 Related Work

Language Model Pre-training Pre-training has been widely applied to universal language representation learning. Previous work can be divided into two main categories: (i) feature-based approach; (ii) fine-tuning approach.

Feature-based methods mainly focus on learning: (i) context-independent word representation (e.g., word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017)); (ii) sentence-level representation (e.g., Kiros et al. (2015); Conneau et al. (2017); Logeswaran and Lee (2018)); and (iii) contextualized word representation (e.g., Cove (McCann et al., 2017), ELMo (Peters et al., 2018)). Specifically, ELMo (Peters et al., 2018) learns high-quality, deep contextualized word representation using bidirectional language model, which can be directly plugged into standard NLU models for performance boosting.

On the other hand, fine-tuning approaches mainly pre-train a language model (e.g., GPT (Radford et al., 2018), BERT (Devlin et al., 2018)) on a large corpus with an unsupervised objective, and then fine-tune the model with in-domain labeled data for downstream applications (Dai and Le, 2015; Howard and Ruder, 2018). Specifically, BERT is a large-scale language model consisting of multiple layers of Transformer blocks (Vaswani et al., 2017). BERT-Base has 12 layers of Transformer and 110 million parameters, while BERT-Large has 24 layers of Transformer and 330 million parameters. By pre-training via masked language modeling and next sentence prediction, BERT has achieved state-of-the-art performance on a wide-range of NLU tasks, such as the GLUE benchmark (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016).

However, these modern pre-trained language models contain millions of parameters, which hinders their application in practice where computational resource is limited. In this paper, we aim at addressing this critical and challenging problem, taking BERT as an example, *i.e.,* how to compress a large BERT model into a shallower one without sacrificing performance. Besides, the proposed approach can also be applied to other large-scale pre-trained language models, such as recently proposed XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019b).

Model Compression & Knowledge Distillation Our focus is model compression, *i.e.,* making deep neural networks more compact (Han et al., 2016; Cheng et al., 2015). A similar line of work has focused on accelerating deep network inference at test time (Vetrov et al., 2017) and reducing model training time (Huang et al., 2016).

A conventional understanding is that a large number of connections (weights) is necessary for training deep networks (Denil et al., 2013; Zhai et al., 2016). However, once the network has been trained, there will be a high degree of parameter redundancy. Network pruning (Han et al., 2015; He et al., 2017), in which network connections are reduced or sparsified, is one common strategy for model compression. Another direction is weight quantization (Gong et al., 2014; Polino et al., 2018), in which connection weights are constrained to a set of discrete values, allowing weights to be represented by fewer bits. However, most of these pruning and quantization approaches perform on convolutional networks. Only a few work are designed for rich structural information such as deep language models (Changpinyo et al., 2017).

Knowledge distillation (Hinton et al., 2015) aims to compress a network with a large set of parameters into a compact and fast-to-execute model. This can be achieved by training a compact model to imitate the soft output of a larger model. Romero et al. (2015) further demonstrated that intermediate representations learned by the large model can serve as hints to improve the training process and the final performance of the compact model. Chen et al. (2015) introduced techniques for efficiently transferring knowledge from an existing network to a deeper or wider network. More recently, Liu et al. (2019a) used knowledge from ensemble models to improve single model performance on NLU tasks. Tan et al. (2019) tried knowledge distillation for multilingual translation. Different from the above efforts, we investigate the problem of compressing large-scale language models, and propose a novel *patient knowledge distillation* approach to effectively transferring knowledge from a teacher to a student model.

3 Patient Knowledge Distillation

In this section, we first introduce a vanilla knowledge distillation method for BERT compression (Section 3.1), then present the proposed Patient Knowledge Distillation (Section 3.2) in details.

Problem Definition The original large teacher network is represented by a function $f(\mathbf{x};\theta)$, where $\mathbf{x}$ is the input to the network, and θ denotes the model parameters. The goal of knowledge distillation is to learn a new set of parameters θ' for a shallower student network $g(\mathbf{x};\theta')$, such that the student network achieves similar performance to the teacher, with much lower computational cost. Our strategy is to force the student model to imitate outputs from the teacher model on the training dataset with a defined objective L_{KD}.

3.1 Distillation Objective

In our setting, the teacher $f(\mathbf{x};\theta)$ is defined as a deep bidirectional encoder, e.g., BERT, and the student $g(\mathbf{x};\theta')$ is a lightweight model with fewer layers. For simplicity, we use BERT$_k$ to denote a model with k layers of Transformers. Following the original BERT paper (Devlin et al., 2018), we also use BERT-Base and BERT-Large to denote BERT$_{12}$ and BERT$_{24}$, respectively.

Assume $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ are N training samples, where $\mathbf{x}_i$ is the i-th input instance for BERT, and $\mathbf{y}_i$ is the corresponding ground-truth label. BERT first computes a contextualized embedding $\mathbf{h}_i = \text{BERT}(\mathbf{x}_i) \in \mathbb{R}^d$. Then, a softmax layer $\hat{\mathbf{y}}_i = P(\mathbf{y}_i|\mathbf{x}_i) = softmax(\mathbf{W}\mathbf{h}_i)$ for classification is applied to the embedding of BERT output, where $\mathbf{W}$ is a weight matrix to be learned.

To apply knowledge distillation, first we need to train a teacher network. For example, to train a 12-layer BERT-Base as the teacher model, the learned parameters are denoted as:

$$\hat{\theta}^t = \arg\min_\theta \sum_{i\in[N]} L^t_{CE}(\mathbf{x}_i, \mathbf{y}_i; [\theta_{\text{BERT}_{12}}, \mathbf{W}]) \tag{1}$$

where the superscript t denotes parameters in the teacher model, $[N]$ denotes set $\{1, 2, \ldots, N\}$, L^t_{CE} denotes the cross-entropy loss for the teacher training, and $\theta_{\text{BERT}_{12}}$ denotes parameters of BERT$_{12}$.

The output probability for any given input $\mathbf{x}_i$ can be formulated as:

$$\begin{aligned}\hat{\mathbf{y}}_i = P^t(\mathbf{y}_i|\mathbf{x}_i) &= softmax\Big(\frac{\mathbf{W}\mathbf{h}_i}{T}\Big)\\ &= softmax\Big(\frac{\mathbf{W}\cdot\text{BERT}_{12}(\mathbf{x}_i;\hat{\theta}^t)}{T}\Big)\end{aligned} \tag{2}$$

where $P^t(\cdot|\cdot)$ denotes the probability output from the teacher. $\hat{\mathbf{y}}_i$ is fixed as soft labels, and T is the temperature used in KD, which controls how much to rely on the teacher's soft predictions. A higher temperature produces a more diverse probability distribution over classes (Hinton et al., 2015). Similarly, let θ^s denote parameters to be learned for the student model, and $P^s(\cdot|\cdot)$ denote the corresponding probability output from the student model. Thus, the distance between the teacher's prediction and the student's prediction can be defined as:

$$\begin{aligned}L_{DS} = -\sum_{i\in[N]}\sum_{c\in C}\Big[&P^t(\mathbf{y}_i = c|\mathbf{x}_i;\hat{\theta}^t)\cdot\\ &\log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\theta^s)\Big]\end{aligned} \tag{3}$$

where c is a class label and C denotes the set of class labels.

Besides encouraging the student model to imitate the teacher's behavior, we can also fine-tune the student model on target tasks, where task-specific cross-entropy loss is included for model training:

$$\begin{aligned}L^s_{CE} = -\sum_{i\in[N]}\sum_{c\in C}\Big[&\mathbb{1}[\mathbf{y}_i = c]\cdot\\ &\log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\theta^s)\Big]\end{aligned} \tag{4}$$

Thus, the final objective function for knowledge distillation can be formulated as:

$$L_{KD} = (1-\alpha)L^s_{CE} + \alpha L_{DS} \tag{5}$$

where α is the hyper-parameter that balances the importance of the cross-entropy loss and the distillation loss.

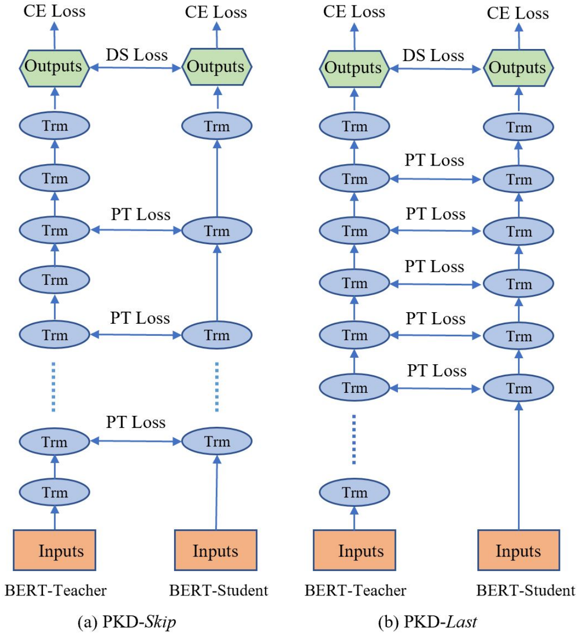


Figure 1: Model architecture of the proposed Patient Knowledge Distillation approach to BERT model compression. (Left) PKD-Skip: the student network learns the teacher's outputs in every 2 layers. (Right) PKD-Last: the student learns the teacher's outputs from the last 6 layers. Trm: Transformer.

3.2 Patient Teacher for Model Compression

Using a weighted combination of ground-truth labels and soft predictions from the last layer of the teacher network, the student network can achieve comparable performance to the teacher model on the training set. However, with the number of epochs increasing, the student model learned with this vanilla KD framework quickly reaches saturation on the test set (see Figure 2 in Section 4).

One hypothesis is that overfitting during knowledge distillation may lead to poor generalization. To mitigate this issue, instead of forcing the student to learn only from the logits of the last layer, we propose a "*patient*" teacher-student mechanism to distill knowledge from the teacher's intermediate layers as well. Specifically, we investigate two patient distillation strategies: (i) PKD-Skip: the student learns from every k layers of the teacher (Figure 1: Left); and (ii) PKD-Last: the student learns from the *last* k layers of the teacher (Figure 1: Right).

Learning from the hidden states of all the tokens is computationally expensive, and may introduce noise. In the original BERT implementation (Devlin et al., 2018), prediction is performed by only using the output from the last layer's `[CLS]` token. In some variants of BERT, like SDNet (Zhu et al., 2018), a weighted average of all layers' `[CLS]` embeddings is applied. In general, the final logit can be computed based on $\mathbf{h}_{\text{final}} = \sum_{j\in[k]} w_j \mathbf{h}_j$, where w_j could be either learned parameters or a pre-defined hyper-parameter, $\mathbf{h}_j$ is the embedding of `[CLS]` from the hidden layer j, and k is the number of hidden layers. Derived from this, if the compressed model can learn from the representation of `[CLS]` in the teacher's intermediate layers for any given input, it has the potential of gaining a generalization ability similar to the teacher model.

Motivated by this, in our Patient-KD framework, the student is cultivated to imitate the representations only for the `[CLS]` token in the intermediate layers, following the intuition aforementioned that the `[CLS]` token is important in predicting the final labels. For an input $\mathbf{x}_i$, the outputs of the `[CLS]` tokens for all the layers are denoted as:

$$\mathbf{h}_i = [\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \dots, \mathbf{h}_{i,k}] = \text{BERT}_k(\mathbf{x}_i) \in \mathbb{R}^{k\times d} \tag{6}$$

We denote the set of intermediate layers to distill knowledge from as I_{pt}. Take distilling from BERT$_{12}$ to BERT$_6$ as an example. For the PKD-Skip strategy, $I_{pt} = \{2,4,6,8,10\}$; and for the PKD-Last strategy, $I_{pt} = \{7,8,9,10,11\}$. Note that $k = 5$ for both cases, because the output from the last layer (e.g., Layer 12 for BERT-Base) is omitted since its hidden states are connected to the softmax layer, which is already included in the KD loss defined in Eqn. (5). In general, for BERT student with n layers, k always equals to $n - 1$.

The additional training loss introduced by the patient teacher is defined as the mean-square loss between the normalized hidden states:

$$L_{PT} = \sum_{i=1}^{N}\sum_{j=1}^{M} \left|\left| \frac{\mathbf{h}^s_{i,j}}{||\mathbf{h}^s_{i,j}||_2} - \frac{\mathbf{h}^t_{i,I_{pt}(j)}}{||\mathbf{h}^t_{i,I_{pt}(j)}||_2} \right|\right|_2^2 \tag{7}$$

where M denotes the number of layers in the student network, N is the number of training samples, and the superscripts s and t in $\mathbf{h}$ indicate the student and the teacher model, respectively. Combined with the KD loss introduced in Section 3.1, the final objective function can be formulated as:

$$L_{PKD} = (1-\alpha)L^s_{CE} + \alpha L_{DS} + \beta L_{PT} \tag{8}$$

where β is another hyper-parameter that weights the importance of the features for distillation in the intermediate layers.

4 Experiments

In this section, we describe our experiments on applying the proposed Patient-KD approach to four different NLP tasks. Details on the datasets and experimental results are provided in the following sub-sections.

4.1 Datasets

We evaluate our proposed approach on Sentiment Classification, Paraphrase Similarity Matching, Natural Language Inference, and Machine Reading Comprehension tasks. For Sentiment Classification, we test on Stanford Sentiment Treebank (SST-2) (Socher et al., 2013). For Paraphrase Similarity Matching, we use Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) and Quora Question Pairs (QQP)[2] datasets. For Natural Language Inference, we evaluate on Multi-Genre Natural Language Inference (MNLI) (Williams et al., 2017), QNLI[3] (Rajpurkar et al., 2016), and Recognizing Textual Entailment (RTE).

More specifically, SST-2 is a movie review dataset with binary annotations, where the binary label indicates positive and negative reviews. MRPC contains pairs of sentences and corresponding labels, which indicate the semantic equivalence relationship between each pair. QQP is designed to predict whether a pair of questions is duplicate or not, provided by a popular online question-answering website Quora. MNLI is a multi-domain NLI task for predicting whether a given premise-hypothesis pair is entailment, contradiction or neural. Its test and development datasets are further divided into in-domain (MNLI-m) and cross-domain (MNLI-mm) splits to evaluate the generality of tested models. QNLI is a task for predicting whether a question-answer pair is entailment or not. Finally, RTE is based on a series of textual entailment challenges, created by General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018).

For the Machine Reading Comprehension task, we evaluate on RACE (Lai et al., 2017), a large-scale dataset collected from English exams, containing 25,137 passages and 87,866 questions. For each question, four candidate answers are provided, only one of which is correct. The dataset is further divided into RACE-M and RACE-H, containing exam questions for middle school and high school students.

4.2 Baselines and Training Details

For experiments on the GLUE benchmark, since all the tasks can be considered as sentence (or sentence-pair) classification, we use the same architecture in the original BERT (Devlin et al., 2018), and fine-tune each task independently.

For experiments on RACE, we denote the input passage as P, the question as q, and the four answers as $a_1, \dots, a_4$. We first concatenate the tokens in q and each a_i, and arrange the input of BERT as `[CLS]` P `[SEP]` $q{+}a_i$ `[SEP]` for each input pair $(P, q + a_i)$, where `[CLS]` and `[SEP]` are the special tokens used in the original BERT. In this way, we can obtain a single logit value for each a_i. At last, a softmax layer is placed on top of these four logits to obtain the normalized probability of each answer a_i being correct, which is then used to compute the cross-entropy loss for modeling training.

We fine-tune BERT-Base (denoted as BERT$_{12}$) as the teacher model to compute soft labels for each task independently, where the pretrained model weights are obtained from Google's official BERT's repo[4], and use 3 and 6 layers of Transformers as the student models (BERT$_3$ and BERT$_6$), respectively. We initialize BERT$_k$ with the first k layers of parameters from pre-trained BERT-Base, where $k \in \{3, 6\}$. To validate the effectiveness of our proposed approach, we first conduct direct fine-tuning on each task without using any soft labels. In order to reduce the hyperparameter search space, we fix the number of hidden units in the final softmax layer as 768, the batch size as 32, and the number of epochs as 4 for all the experiments, with a learning rate from {5e-5, 2e-5, 1e-5}. The model with the best validation accuracy is selected for each setting.

Besides direct fine-tuning, we further implement a vanilla KD method on all the tasks by optimizing the objective function in Eqn. (5). We set the temperature T as {5, 10, 20}, $\alpha = \{0.2, 0.5, 0.7\}$, and perform grid search over T, α and learning rate, to select the model with the best validation accuracy. For our proposed Patient-KD approach, we conduct additional search over β

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] The dataset is derived from Stanford Question Answer Dataset (SQuAD).

[4] https://github.com/google-research/bert

PKD is PKD-skip

Model	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m (393k)	MNLI-mm (393k)	QNLI (105k)	RTE (2.5k)
BERT$_{12}$ (Google)	93.5	88.9/84.8	71.2/89.2	84.6	83.4	90.5	66.4
BERT$_{12}$ (Teacher)	94.3	89.2/85.2	70.9/89.0	83.7	82.8	90.4	69.1
BERT$_6$-FT	90.7	85.9/80.2	69.2/88.2	80.4	79.7	86.7	63.6
BERT$_6$-KD	91.5	**86.2/80.6**	70.1/88.8	80.2	79.8	88.3	64.7
BERT$_6$-PKD	**92.0**	85.0/79.9	**70.7/88.9**	**81.5**	**81.0**	**89.0**	**65.5**
BERT$_3$-FT	86.4	80.5/**72.6**	65.8/86.9	74.8	74.3	84.3	55.2
BERT$_3$-KD	86.9	79.5/71.1	67.3/87.6	75.4	74.8	84.0	56.2
BERT$_3$-PKD	**87.5**	**80.7**/72.5	**68.1/87.8**	**76.7**	**76.3**	**84.7**	**58.2**

Table 1: Results from the GLUE test server. The best results for 3-layer and 6-layer models are in-bold. Google's submission results are obtained from official GLUE leaderboard. BERT$_{12}$ (Teacher) is our own implementation of the BERT teacher model. FT represents direct fine-tuning on each dataset without using knowledge distillation. KD represents using a vanilla knowledge distillation method. And PKD represents our proposed Patient-KD-Skip approach. Results show that PKD-Skip outperforms the baselines on almost all the datasets except for MRPC. The numbers under each dataset indicate the corresponding number of training samples.

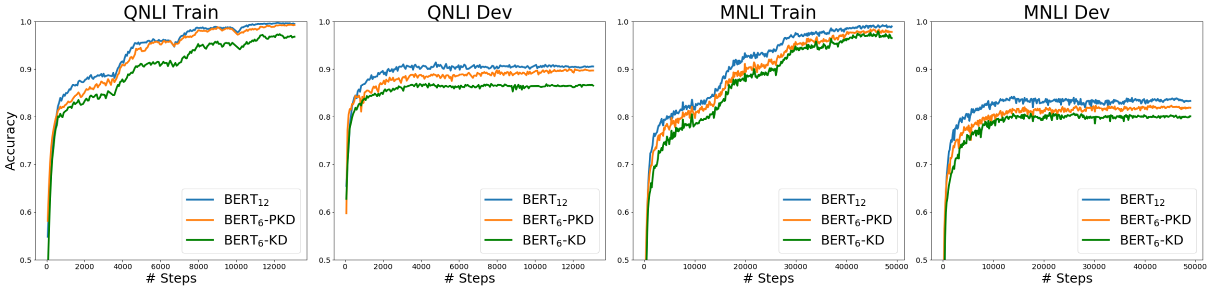


Figure 2: Accuracy on the training and dev sets of QNLI and MNLI datasets, by directly applying vanilla knowledge distillation (KD) and the proposed Patient-KD-Skip. The teacher and the student networks are BERT$_{12}$ and BERT$_6$, respectively. The student network learned with vanilla KD quickly saturates on the dev set, while the proposed Patient-KD starts to plateau only in a later stage.

from $\{10, 100, 500, 1000\}$ on all the tasks. Since there are so many hyper-parameters to learn for Patient KD, we fix α and T to the values used in the model with the best performance from the vanilla KD experiments, and only search over β and learning rate.

4.3 Experimental Results

We submitted our model predictions to the official GLUE evaluation server to obtain results on the test data. Results are summarized in Table 1. Compared to direct fine-tuning and vanilla KD, our Patient-KD models with BERT$_3$ and BERT$_6$ students perform the best on almost all the tasks except MRPC. For MNLI-m and MNLI-mm, our 6-layer model improves 1.1% and 1.3% over fine-tune (FT) baselines; for QNLI and QQP, even though the gap between BERT$_6$-KD and BERT$_{12}$ teacher is relatively small, our approach still succeeded in improving over both FT and KD baselines and further closing the gap between the student and the teacher models.

Furthermore, in 5 tasks out of 7 (SST-2 (-2.3% compared to BERT-Base teacher), QQP (-0.1%), MNLI-m (-2.2%), MNLI-mm (-1.8%), and QNLI (-1.4%)), the proposed 6-layer student coached by the patient teacher achieved similar performance to the original BERT-Base, demonstrating the effectiveness of our approach. Interestingly, all those 5 tasks have more than 60k training samples, which indicates that our method tends to perform better when there is a large amount of training data.

For the QQP task, we can further reduce the model size to 3 layers, where BERT$_3$-PKD can still have a similar performance to the teacher model. The learning curves on the QNLI and MNLI datasets are provided in Figure 2. The student model learned with vanilla KD quickly saturated on the dev set, while the proposed Patient-KD keeps learning from the teacher and improving

Model	SST-2	MRPC	QQP	MNLI-m	MNLI-mm	QNLI	RTE
BERT$_6$ (PKD-Last)	91.9	**85.1**/79.5	70.5/**88.9**	80.9	**81.0**	88.2	65.0
BERT$_6$ (PKD-Skip)	**92.0**	85.0/**79.9**	**70.7**/**88.9**	**81.5**	**81.0**	**89.0**	**65.5**

Table 2: Performance comparison between PKD-Last and PKD-Skip on GLUE benchmark.

Model	RACE	RACE-M	RACE-H
BERT$_{12}$ (Leaderboard)	65.00	71.70	62.30
BERT$_{12}$ (Teacher)	65.30	71.17	62.89
BERT$_6$-FT	54.32	61.07	51.54
BERT$_6$-KD	58.74	64.69	56.29
BERT$_6$-PKD-Skip	**60.34**	**66.57**	**57.78**

Table 3: Results on RACE test set. BERT$_{12}$ (Leaderboard) denotes results extracted from the official leaderboard (http://www.qizhexie.com//data/RACE_leaderboard). BERT$_{12}$ (Teacher) is our own implementation. Results of BERT$_3$ are not included due to the large gap between the teacher and the BERT$_3$ student.

accuracy, only starting to plateau in a later stage.

For the MRPC dataset, one hypothesis for the reason on vanilla KD outperforming our model is that the lack of enough training samples may lead to overfitting on the dev set. To further investigate, we repeat the experiments three times and compute the average accuracy on the dev set. We observe that fine-tuning and vanilla KD have a mean dev accuracy of 82.23% and 82.84%, respectively. Our proposed method has a higher mean dev accuracy of 83.46%, hence indicating that our Patient-KD method slightly overfitted to the dev set of MRPC due to the small amount of training data. This can also be observed on the performance gap between teacher and student on RTE in Table 5, which also has a small training set.

We further investigate the performance gain from two different patient teacher designs: PKD-Last vs. PKD-Skip. Results of both PKD variants on the GLUE benchmark (with BERT$_6$ as the student) are summarized in Table 2. Although both strategies achieved improvement over the vanilla KD baseline (see Table 1), PKD-Skip performs slightly better than PKD-Last. Presumably, this might be due to the fact that distilling information across every k layers captures more diverse representations of richer semantics from low-level to high-level, while focusing on the last k layers tends to capture relatively homogeneous semantic information.

Results on RACE are reported in Table 3, which shows that the Vanilla KD method outperforms direct fine-tuning by 4.42%, and our proposed patient teacher achieves further 1.6% performance lift, which again demonstrates the effectiveness of Patient-KD.

4.4 Analysis of Model Efficiency

We have demonstrated that the proposed Patient-KD method can effectively compress BERT$_{12}$ into BERT$_6$ models without performance sacrifice. In this section, we further investigate the efficiency of Patient-KD on storage saving and inference-time speedup. Parameter statistics and inference time are summarized in Table 4. All the models share the same embedding layer with 24 millon parameters that map a 30k-word vocabulary to a 768-dimensional vector, which leads to 1.64 and 2.4 times of machine memory saving from BERT$_6$ and BERT$_3$, respectively.

To test the inference speed, we ran experiments on 105k samples from QNLI training set (Rajpurkar et al., 2016). Inference is performed on a single Titan RTX GPU with batch size set to 128, maximum sequence length set to 128, and FP16 activated. The inference time for the embedding layer is negligible compared to the Transformer layers. Results in Table 4 show that the proposed Patient-KD approach achieves an almost linear speedup, 1.94 and 3.73 times for BERT$_6$ and BERT$_3$, respectively.

4.5 Does a Better Teacher Help?

To evaluate the effectiveness of the teacher model in our Patient-KD framework, we conduct additional experiments to measure the difference between BERT-Base teacher and BERT-Large teacher for model compression.

Each Transformer layer in BERT-Large has 12.6 million parameters, which is much larger than the Transformer layer used in BERT-Base. For a compressed BERT model with 6 layers, BERT$_6$ with BERT-Base Transformer (denoted as BERT$_6$[Base]) has only 67.0 million parameters, while BERT$_6$ with BERT-Large Transformer (de-

no reduction
on embedding
layer

reduction only
on transformer
layers

# Layers	# Param (Emb)	# Params (Trm)	Total Params	Inference Time (s)
3	23.8M	21.3M	45.7M (2.40×)	27.35 (3.73×)
6	23.8M	42.5M	67.0M (1.64×)	52.51 (1.94×)
12	23.8M	85.1M	109M (1×)	101.89 (1×)

Table 4: The number of parameters and inference time for BERT_3, BERT_6 and BERT_{12}. Parameters in Transformers (Trm) grow linearly with the increase of layers. Note that the summation of # Param (Emb) and # Param (Trm) does not exactly equal to Total Params, because there is another softmax layer with 0.6M parameters.

Setting	Teacher	Student	SST-2	MRPC	QQP	MNLI-m	MNLI-mm	QNLI	RTE
N/A	N/A	BERT_{12} (Teacher)	94.3	89.2/85.2	70.9/89.0	83.7	82.8	90.4	69.1
N/A	N/A	BERT_{24} (Teacher)	94.3	88.2/84.3	71.9/89.4	85.7	84.8	92.2	72.8
#1	BERT_{12}	BERT_6[Base]-KD	91.5	86.2/80.6	70.1/88.8	79.7	79.1	88.3	64.7
#2	BERT_{24}	BERT_6[Base]-KD	91.2	86.1/80.7	69.4/88.6	80.2	79.7	87.5	65.7
#3	BERT_{24}	BERT_6[Large]-KD	89.6	79.0/70.0	65.0/86.7	75.3	74.6	83.4	53.7
#4	BERT_{24}	BERT_6[Large]-PKD	89.8	77.8/68.3	67.1/87.9	77.2	76.7	83.8	53.2

Table 5: Performance comparison with different teacher and student models. BERT_6[Base]/[Large] denotes a BERT_6 model with a BERT-Base/Large Transformer in each layer. For PKD, we use the PKD-Skip architecture.

noted as BERT_6[Large]) has 108.4 million parameters. Since the size of the `[CLS]` token embedding is different between BERT-Large and BERT-Base, we cannot directly compute the patient teacher loss (7) for BERT_6[Base] when BERT-Large is used as teacher. Hence, in the case where the teacher is BERT-Large and the student is BERT_6[Base], we only conduct experiments in the vanilla KD setting.

Results are summarized in Table 5. When the teacher changes from BERT_{12} to BERT_{24} (*i.e.*, Setting #1 vs. #2), there is not much difference between the students' performance. Specifically, BERT_{12} teacher performs better on SST-2, QQP and QNLI, while BERT_{24} performs better on MNLI-m, MNLI-mm and RTE. Presumably, distilling knowledge from a larger teacher requires a larger training dataset, thus better results are observed on MNLI-m and MNLI-mm.

We also report results on using BERT-Large as the teacher and BERT_6[Large] as the student. Interestingly, when comparing Setting #1 with #3, BERT_6[Large] performs much worse than BERT_6[Base] even though a better teacher is used in the former case. The BERT_6[Large] student also has 1.6 times more parameters than BERT_6[Base]. One intuition behind this is that the compression ratio for the BERT_6[Large] model is 4:1 (24:6), which is larger than the ratio used for the BERT_6[Base] model (2:1 (12:6)). The higher compression ratio renders it more challenging for the student model to absorb important weights.

When comparing Setting # 2 and #3, we observe that even when the same large teacher is used, BERT_6[Large] still performs worse than BERT_6[Base]. Presumably, this may be due to initialization mismatch. Ideally, we should pre-train BERT_6[Large] and BERT_6[Base] from scratch, and use the weights learned from the pre-training step for weight initialization in KD training. However, due to computational limits of training BERT_6 from scratch, we only initialize the student model with the first six layers of BERT_{12} or BERT_{24}. Therefore, the first six layers of BERT_{24} may not be able to capture high-level features, leading to worse KD performance.

Finally, when comparing Setting #3 vs. #4, where for setting #4 we use Patient-KD-Skip instead of vanilla KD, we observe a performance gain on almost all the tasks, which indicates Patient-KD is a generic approach independent of the selection of the teacher model (BERT_{12} or BERT_{24}).

5 Conclusion

In this paper, we propose a novel approach to compressing a large BERT model into a shallow one via Patient Knowledge Distillation. To fully utilize the rich information in deep structure of the teacher network, our Patient-KD approach encourages the student model to patiently learn from the teacher through a multi-layer distillation process. Extensive experiments over four NLP tasks demonstrate the effectiveness of our proposed model.

For future work, we plan to pre-train BERT from scratch to address the initialization mismatch issue, and potentially modify the proposed method such that it could also help during pre-training. Designing more sophisticated distance metrics for loss functions is another exploration direction. We will also investigate Patient-KD in more complex settings such as multi-task learning and meta learning.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. 2017. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

Yu Cheng, Felix X. Yu, Rogerio S. Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *ICCV*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. 2013. Predicting parameters in deep learning. In *NIPS*.

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*.

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

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *NIPS*.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *ECCV*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

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

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Liang Lu, Michelle Guo, and Steve Renals. 2017. Knowledge distillation for small-footprint highway networks. In *ICASSP*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

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

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

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *ICLR*.

Bharat Bhusan Sau and Vineeth N Balasubramanian. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

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 *EMNLP*.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *ICLR*.

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 *NIPS*.

Dmitry P. Vetrov, Jonathan Huang, Li Zhang, Maxwell Collins, Michael Figurnov, Ruslan Salakhutdinov, and Yukun Zhu. 2017. Spatially adaptive computation time for residual networks. In *CVPR*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Mark Zhang. 2016. Doubly convolutional neural networks. In *NIPS*.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*.