Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System

Ze Yang*, Linjun Shou*, Ming Gong, Wutao Lin, Daxin Jiang
STCA NLP Group, Microsoft
Beijing, China
{yaze,lisho,migon,wutlin,djiang}@microsoft.com

ABSTRACT

Deep pre-training and fine-tuning models (such as BERT and OpenAI GPT) have demonstrated excellent results in question answering areas. However, due to the sheer amount of model parameters, the inference speed of these models is very slow. How to apply these complex models to real business scenarios becomes a challenging but practical problem. Previous model compression methods usually suffer from information loss during the model compression procedure, leading to inferior models compared with the original one. To tackle this challenge, we propose a Two-stage Multi-teacher Knowledge Distillation (TMKD for short) method for web Question Answering system. We first develop a general Q&A distillation task for student model pre-training, and further fine-tune this pre-trained student model with multi-teacher knowledge distillation on downstream tasks (like Web Q&A task, MNLI, SNLI, RTE tasks from GLUE), which effectively reduces the overfitting bias in individual teacher models, and transfers more general knowledge to the student model. The experiment results show that our method can significantly outperform the baseline methods and even achieve comparable results with the original teacher models, along with substantial speedup of model inference.

CCS CONCEPTS

• **Information systems** **Retrieval tasks and goals**; **Question answering**.

KEYWORDS

model compression; two-stage; multi-teacher; knowledge distillation; distillation pre-training

ACM Reference Format:
Ze Yang*, Linjun Shou*, Ming Gong, Wutao Lin, Daxin Jiang. 2020. Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371792

*These authors contributed equally.

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371792

1 INTRODUCTION

Question Answering relevance, which aims to rank the text passages to natural language questions issued by users, is a critical task in Question Answering (Q&A) system [1]. In recent years, almost all commercial web search engines provide Question Answering service, in addition to the traditional web documents links. Table 1 shows an example for Question Answering from a commercial search engine. Compared with the "ten-blue-links", Q&A is a more natural interface, and thousands of millions of users enjoy the efficiency of directly accessing the information for their questions.

Table 1: An example of Q&A relevance task.

Question:	*What can I do when I have headache?*
Passage:	*Drinking warm water mixed with juice squeezed from one-half of a lemon will reduce the intensity of a headache. This particular remedy is beneficial for headaches caused by gas in the stomach. Another option is to apply lemon crusts, pounded into a paste, on your forehead to immediately relieve pain...*
Label:	*Relevant*

In recent years, deep pre-training approaches [5, 25] have brought big break-through in NLP tasks. They also show very promising results for the particular task of Q&A relevance. However, due to the huge parameter size of these models [1], both model training and inference become very time-consuming. Although several works have studied the optimization of model training [30], there is little work discussing the model inference challenge of deep pre-training models like BERT/GPT models. In fact, for a web scale Q&A system, the efficiency of model inference may be even more critical than that of model training, due to the concerns of both offline throughput and online latency. Table 2 shows the inference speed of BERT models [5] with a 1080Ti GPU. The throughout of Q&A pairs are 624 and 192 per second on average for $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$, respectively. In other words, the average latency are 1.6 and 5.2 milliseconds respectively.

In a commercial web Q&A system, there are often two complementary pipelines for the Q&A service. One pipeline is for popular queries that frequently appear in the search traffic. The answers are pre-computed offline in a batch mode and then served online by simple look-up. The magnitude of the number of Q&A pairs processed is around 10 billions. The other pipeline is for tail queries that are rarely or never seen before. For such tail queries, the answers are ranked on the fly and the latency budget for online model

[1] For example, GPT/$\text{BERT}_{\text{base}}$ has 110M parameters, and $\text{BERT}_{\text{large}}$ has 340M.

inference is typically within 10 milliseconds. Therefore, for both offline or online pipelines, it is critical to improve model inference efficiency.

Table 2: The inference speed of BERT on 1080Ti GPU.

Model	Parameter	Samples Per second	Latency
$\mathbf{BERT_{base}}$	110M	624	1.6ms
$\mathbf{BERT_{large}}$	340M	192	5.2ms

To improve model inference efficiency, we consider model compression approach. In other words, we aim to train a smaller model with fewer parameters to simulate the original large model. A popular method, called *knowledge distillation* [11] has been widely used for model compression. The basic idea is a teacher-student framework, in which the knowledge from a complex network (teacher model) is transferred to a simple network (student model) by learning the output distribution of the teacher model as a soft target. To be more specific, when training the student model, we not only provide the human-labeled golden ground truth, but also feed the output score from the teacher model as a secondary soft label. Compared with the discrete human labels (for classification task), the continuous scores from the teacher models give more smooth and fine-grained supervision to the student model, and thus result in better model performance. We refer to this basic knowledge distillation approach as *1-o-1 model*, in the sense that one teacher transfers knowledge to one student.

Although the 1-o-1 model can effectively reduce the number of parameters as well as the time for model inference, due to the information loss during the knowledge distillation, the performance of student model usually cannot reach the parity with its teacher model. This motivates us to develop the second approach, called *m-o-m ensemble model*. To be more specific, we first train multiple teacher models, for example, BERT (base and large) [5] and GPT [25] with different hyper-parameters. Then train a separate student model for each individual teacher model. Finally, the student models trained from different teachers are ensembled to generate the ultimate result. Our experimental results showed that the m-o-m ensemble model performs better than the 1-o-1 model. The rationale is as follows. Each teacher model is trained towards a specific learning objective. Therefore, various models have different generalization ability, and they also overfit the training data in different ways. When ensemble these models, the over-fitting bias across different models can be reduced by the voting effect. That say, the ensemble models automatically "calibrate" the results.

When we compare the m-o-m ensemble model with the 1-o-1 model, although the former has better performance, it also consumes much larger memory to host multiple student models. This motivates us to look for a new approach, which has better performance than the 1-o-1 model and consumes less memory than the m-o-m model. One observation for the m-o-m ensemble approach is that it conducts the model ensemble too late. In fact, once the training process for a student models has finished, the overfitting bias from the corresponding teacher model has already been transferred to the student model. The voting effect across student models can be considered as a "late calibration" process. On the other hand, if we feed the scores from multiple teachers to a single student model during the training stage, that model is receiving guidance from various teachers simultaneously. Therefore, the overfitting bias can be addressed by "early calibration". Based on this observation, we develop the novel *m-o-1* approach, where we train a single student model by feeding the scores from multiple teachers at the same time as the supervision signals. The experimental results showed that the m-o-1 model performs better than the m-o-m model, while the memory consumption is the same with the 1-o-1 model.

The novel m-o-1 approach results in decent compressed models. However, the performance of the compressed models still has small gap with the original large model. One obvious reason is that the original large model has a large-scale pre-training stage, where it learns the language model through an unsupervised approach. We therefore explore how to simulate a pre-training stage for the compressed models, such that it can benefit from large-scale training data and learn the feature representation sufficiently.

Our empirical study shows that the pre-training stage significantly improves the model performance. When we adopt a very large pre-training data, followed by the m-o-1 fine-tuning strategy, the compressed model can achieve comparable or even better performance than the teacher model. Another interesting finding is that although the pre-trained model is derived from Q&A pairs, it can serve as a generic baseline for multiple tasks. As we show in the experiment part, when we fine-tune the Q&A pre-trained model with various text matching tasks, such as those in GLUE [26], it outperforms the compressed model without pre-training on each task. To the best of our knowledge, this is the first work discussing the distillation pre-training and multiple teacher distillation for Web Q&A.

In this paper, we propose a **T**wo-stage **M**ulti-teacher **K**nowledge **D**istillation (**TMKD** for short) method for model compression, and make the following major contributions.

- In the first stage (i.e., the pre-training stage) of TMKD, we create a general Q&A distillation pre-training task to leverage large-scale unlabeled question-passage pairs derived from a commercial search engine. The compressed model benefits from such large-scale data and learns feature representation sufficiently. This pre-trained Q&A distillation model can be also applied to the model compression of various text matching tasks.
- In the second stage (i.e., the fine-tuning stage) of TMKD, we design a multi-teacher knowledge distillation paradigm to jointly learn from multiple teacher models on downstream tasks. The "early calibration" effect relieves the over-fitting bias in individual teacher models, and consequently, the compressed model can achieve comparable or even better performance with the teacher model.
- We conduct intensive experiments on several datasets (both open benchmark and commercial large-scale datasets) to verify the effectiveness of our proposed method. TMKD outperforms various state-of-the-art baselines and has been applied to real commercial scenarios.

The rest of the paper is organized as follows. After a summary of related work in Section 2, we describe our proposed model in details

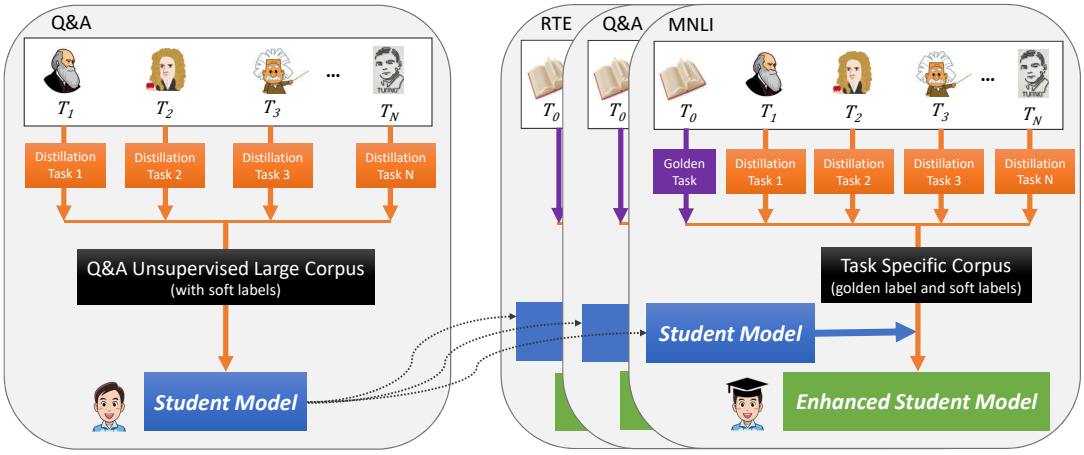

Figure 1: The Overall Architecture of Our Two-stage Multi-teacher Distillation Model.

in Section 3, followed by comprehensive evaluations in Section 4 and Section 5. Finally, Section 6 concludes this paper and discuss future directions.

2 RELATED WORK

In this section we briefly review two research areas related to our work: model compression and multi-task learning.

2.1 Model Compression

As the parameter size of neural network model is getting larger and larger [5, 12, 23], how to make it feasible to deploy and apply the models in industrial environment becomes an important problem. A natural process is to compress the model [8, 11, 16]. Low-rank approximation was a factorization method [4, 13, 31], which used multiple low rank matrices to approximate the original matrix to reduce model redundancy [9, 10, 16]. Hinton et al. proposed a knowledge distillation method (KD for short) [11]. In their work, the output of the complex network was used as a soft target for the training of simple network. By this way, the knowledge of complex models can be transferred to simple models. Distilling complex models into simple models has been shown to improve many NLP tasks to achieve impressive performance [14, 15, 18, 20]. Polino et al. [24] proposed a quantized distillation method. In their work, they incorporated distillation loss, and expressed with respect to the teacher network, into the training process of a smaller student network whose weights were quantized to a limited set of levels. Papernot et al. proposed a training data protected method based on knowledge distillation [21]. In their work, an ensemble of teachers was trained on disjoint subsets of the sensitive data, and then a student model was trained on public data labeled using the ensemble of teachers.

2.2 Multi-task Learning

Multi-task learning has been widely studied in deep learning, which leverages the information among different tasks to improve the generalization performance [3, 6, 28]. Fares et al. [7] empirically evaluated the utility of transfer and multi-task learning on semantic interpretation of noun-noun compounds. It showed that transfer learning via parameter sharing can help a neural classification model generalize over a highly skewed distribution of relations. Pentina and Lampert [22] studied a variant of multi-task learning in which annotated data was available on some of the tasks. Lee et al. [17] studied the performance of different ensemble methods under the framework of multi-task learning.

You et al. [29] presented a method to train a thin deep network by incorporating in the intermediate layers and imposing a constraint about the dissimilarity among examples. Wu et al. [27] propose a multi-teacher knowledge distillation framework for compressed video action recognition to compress this model. These efforts have tried mutiple teacher distillation methods in the field of computer vision, but little research has been done on the NLP deep pretraining based model. Concurrently with our work, several works also combine the multi-task learning with knowledge distillation [2, 18, 19]. However, they applied the knowledge distillation and multi-task learning to enhance the original model performance, instead of targeting model compression.

Our approach is also a knowledge distillation based method for model compression. Different from previous approaches, we develop a novel Q&A distillation pre-training task leveraging large-scale unsupervised Q&A data. Moreover, we design a multi-task paradigm in the fine-tuning stage to jointly distill the knowledge from different teacher models into a single student model.

3 OUR APPROACH

In this section, we firstly describe the overall design of our model, and then describe the proposed approach TMKD in details. Finally, we discuss the procedure of model training and prediction.

3.1 Overview

Figure 1 shows the architecture of TMKD. It consists of two stages: distillation pre-training and task specific distillation fine-tuning. In terms of teacher model for distillation, we take labeled data

by crowd sourcing judges as one specific teacher (T_0) which has the ground-truth knowledge (e.g. 0 or 1). We also have several other teachers (T_1-T_N) trained on different pre-trained models (e.g., BERT [5] and GPT [25]) or with different hyper-parameters, which provide the soft knowledge as pseudo supervision (score in [0, 1]).

3.1.1 Stage 1 - Distillation Pre-training. Deep pre-trained models like BERT/GPT benefit from the pre-training stage on large-scale unsupervised data for better representation learning. Inspired by this, we explore how to simulate a pre-training stage for the compressed models. One method is to leverage large-scale unsupervised data of specific task for knowledge distillation. However it is usually hard to obtain large-scale task-specific unsupervised data for NLP tasks, such as NLI tasks from GLUE datasets. To address this challenge, a Q&A knowledge distillation task is proposed to pre-train the compressed student model on a large-scale Q&A unlabeled data which are derived from a commercial search engine. To be more specific:

- *Step 1:* For each question, top 10 relevant documents are returned by the commercial search engine to form <Question, Url> pairs, and passages are further extracted from these documents to form <Question, Passage> pairs.
- *Step 2:* Then we leverage several Q&A teacher models (such as $\text{BERT}_{\text{large}}$ fine-tuned models) to score the above <Question, Passage> pairs.
- *Step 3:* We use the <Question, Passage> corpus as well as their corresponding teacher models' output scores as the pseudo ground truth to pre-train the student model[2].

With *Step 1* and *Step 2*, we could collect a large-scale auto labelled corpus (i.e. soft labels) for pre-training, which is several magnitudes larger than that of the human labeled training set. For *Step 3*, we propose the novel multi-teacher knowledge distillation (i.e. m-o-1 approach) for pre-training. The distillation pre-trained student model [3] with Q&A task not only greatly boosts final Q&A fine-tuned model but also other NLU tasks (like NLI tasks from GLUE), which are shown in experiment section later.

3.1.2 Stage 2 - Task Specific Distillation Fine-tuning. Through the large-scale distillation pre-training stage, our student model is able to learn decent feature representation capabilities for general NLU tasks (like Web Q&A task, MNLI, SNLI, RTE tasks from GLUE). At the fine-tuning stage, the student model is firstly initialized with the pre-trained parameters in the above *Stage 1*, and then all of the parameters are fine-tuned using labeled data from the downstream specific tasks. At this stage, we propose a novel multi-teacher knowledge distillation method (i.e. m-o-1 approach).

To be more specific, for each downstream task, we use both the golden label (i.e. ground-truth knowledge of T_0) on the task specific corpus and the soft labels of T_1-T_N (i.e. pseudo ground-truth knowledge) on the same corpus to jointly fine-tune to get an enhanced student model. This is just like the learning process of human beings that we simultaneously gain knowledge from our teachers as well as the textbooks that our teachers have studied.

[2]The BERT student model is initialized by the bottom three layers of the BERT model. Therefore, it has captured a rough language model from large corpus.
[3]github.com/microsoft/NeuronBlocks/tree/master/model_zoo/TMKD.

3.2 TMKD Architecture

TMKD is implemented from BERT [5]. Our model consists of three layers: Encoder layer utilizes the lexicon to embed both the question and passage into a low embedding space; Transformer layer maps the lexicon embedding to contextual embedding; Multi-header layer jointly learns from multiple teachers simultaneously during training, as well as generates final prediction output during inference.

3.2.1 Encoder Layer. In a Q&A system, each question and passage are described by a set of words. We take the word pieces as the input just like BERT. $X = \{x^{(1)}, x^{(2)}, ..., x^{(|X|)}\}$ is to denote all the instances, and each instance has a $\langle Q, P\rangle$ pair. Let $Q = \{w_1, w_2, w_3, ..., w_m\}$ be a question with m word pieces, $P = \{w_1, w_2, w_3, ..., w_n\}$ be a passage with n word pieces, and w_i is the bag-of-word representation. $C = \{c_1, c_2, \ldots, c_{|C|}\}$ represents the label set to indicate $\langle Q, P\rangle$'s relation. Each token representation is constructed by the sum of the corresponding token, segment and position embeddings. Let $\mathbf{V} = \{\vec{v}_t \in \mathbb{R}^{D_v} | t = 1, \ldots, M\}$ denote all the summed vectors in a D_v dimension continuous space.

We concatenate the $\langle Q, P\rangle$ pair, and add $\langle CLS\rangle$ as the first token, then add $\langle SEP\rangle$ between Q and P. After that, we obtain the concatenation input $x_c = \{w_1, w_2, w_3, \ldots, w_{m+n+2}\}$ of a given instance $x^{(i)}$. With the encoder layer, we map x_c into continuous representations $H_e = \{v_1, v_2, \ldots, v_{m+n+2}\}$.

3.2.2 Transformer Layer. We also use the bidirectional transformer encoder to map the lexicon embedding H_e into a sequence of continuous contextual embedding $H_s = \{h_1, h_2, h_3, \ldots, h_{m+n+2}\}$.

3.2.3 Multi-header Layer. In our proposed approach, firstly several teacher models are built with different hyper-parameters. Then, in order to let the student model to jointly learn from these teacher models, a multi-header layer is designed consisting of two parts, i.e. golden label header and soft label headers:

Golden Label Header. Given instance $x^{(i)}$, this header aims to learn the ground truth label. Following the BERT, we select $x^{(i)}$'s first token's transformer hidden state h_1 as the global representation of input. The probability that $x^{(i)}$ is labeled as class c is defined as follows:

$$P(c|\langle Q, P\rangle) = softmax(W_g^T \cdot h_1) \tag{1}$$

where W_g^T is a learnable parameter matrix, $c \in C$ indicates the relation between $\langle Q, P\rangle$. The objective function of golden label header task is then defined as the cross-entropy:

$$l_g = -\sum_{c \in C} c \cdot log(P(c|\langle Q, P\rangle)) \tag{2}$$

Soft Label Headers. Take the i-th soft label as an example, $iin[1, |N|]$, N is the number of soft labels. For a given instance $x^{(i)}$, we also select the first token's hidden state h_1 as the global representation of input. The probability that $x^{(i)}$ is labeled as class c is defined as follows:

$$P_{s_i}(c|\langle Q, P\rangle) = softmax(W_{s_i}^T \cdot h_1) \tag{3}$$

where $W_{s_i}^T$ is a learnable parameter matrix. We support $R_{s_i}(c|\langle Q, P\rangle) = W_{s_i}^T \cdot h_1$ as the logits of i-th soft header before normalization.

For a instance $\langle Q, P\rangle$, teacher model can predict probability distributions to indicate that Q and P are relevant or not. Soft label

Table 3: Statistics of experiment datasets (For DeepQA, we have a test set, which is non-overlapping with the training set. For GLUE, please note that the results on development sets are reported, since GLUE does not distribute labels for the test sets).

Dataset	Size of Samples (Train/Test)	Average Question Length (Words)	Average Answer Length (Words)
DeepQA	1M/10K	5.86	43.74
CommQA-Unlabeled	4M(base) 40M(large) 0.1B(extreme)	6.31	42.70
CommQA-Labeled	12M/2.49K	5.81	45.70
MNLI	392.70K/19.64K	20.52	10.90
SNLI	549.36K/9.84K	13.80	10.90
QNLI	108.43K/5.73K	9.93	28.07
RTE	2.49K/0.27K	45.30	9.77

headers aim to learn the teachers' knowledge through soft labels. The objective function of soft label headers is defined as mean squared error as follows:

$$
\begin{aligned}
l_{s_i} &= \frac{1}{|C|} \sum_{c \in C} (R_{s_i}(c|\langle Q, P\rangle) - R_{t_i}(c|\langle Q, P\rangle))^2 \\
l_s &= \frac{1}{N} \sum_{i=1}^{N} l_{s_i}
\end{aligned}
\tag{4}
$$

where $R_{t_i}(c|\langle Q, P\rangle)$ represents the i-th soft label teacher's logits before normalization and N is the number of soft label headers.

3.3 Training and Prediction

In order to learn parameters of **TMKD** model, our proposed TMKD model has a two-stage training strategy. At the first stage, we use the Equation (4) to learn the generalized natural language inference capability from the unlabeled data with soft labels. At the second stage, we combine Equation (2) and Equation (4) to learn the task-specific knowledge from the labeled data with golden labels and soft labels, then obtain our final learning objective function as follows:

$$
l = (1 - \alpha) l_g + \alpha l_s \tag{5}
$$

where α is a loss weighted ratio, l_{s_i} is the loss of i-th soft header. In the inference stage, we use an aggregation operation to calculate the final result as follows:

$$
\begin{aligned}
O(c|\langle Q, P\rangle) = &\frac{1}{N+1}(P(c|\langle Q, P\rangle) + \\
&\sum_{i=1}^{N} P_{s_i}(c|\langle Q, P\rangle))
\end{aligned}
\tag{6}
$$

where P_{s_i} is the i-th student header's output and N denotes the number of soft label headers.

4 EXPERIMENT

In this section, we conduct empirical experiments to verify the effectiveness of our proposed TMKD on model compression. We first introduce the experimental settings, then compare our model to the baseline methods to demonstrate its effectiveness.

4.1 Dataset

We conduct experiments on several datasets as follows.

- **DeepQA**: An English Q&A task dataset from one commercial Q&A system, with 1 million labeled cases. Each case consists of three parts, i.e. question, passage, and binary label (i.e. 0 or 1) by crowd sourcing judges indicating whether the question can be answered by the passage. The following briefly describes how the data is collected. Firstly, for each question, top 10 relevant documents returned by the search engine are selected to form <Question, Url> pairs; Then passages are further extracted from these documents to form <Question, Url, Passage> triples; These <Query, Passage> pairs are sampled and sent to crowd sourcing judges. Specifically, each <Query, Passage> pair is required to get judged by three judges. Those cases with more than 2/3 positive labels will get positive labels, otherwise negative.
- **CommQA-Unlabeled** A large-scale unlabeled Q&A data coming from a commercial search engine. The collection method of <Query, Passage> pairs is same as DeepQA, and the difference is that the question type and domain of this dataset is more diverse than DeepQA. We sampled 4 million (named base dataset) and 40 million (named large dataset) as the pre-training data. Besides, in our commercial scenario, we have one extremely large Q&A unlabeled dataset (0.1 billion) cooked by the same data collection approach.
- **CommQA-Labeled** A large-scale commercial Q&A training data, which is sampled from CommQA-Unlabeled, and labeled by crowd sourcing judges.
- **GLUE** [26]: A collection of datasets for evaluating NLU systems, including nine language understanding tasks. Among them, we choose textual entailment tasks (MNLI, SNLI, QNLI, RTE), which are similar to Q&A task. For MNLI and QNLI, given two sentences (premise and hypothesis), the task is to predict whether the premise entails the hypothesis (entailment), contradicts (contradiction), or neither (neutral). While for SNLI and RTE, the relationship does not contain neutral type.

4.2 Evaluation Metrics

We use the following metrics to evaluate model performance:

- **Accuracy (ACC)**: Number of correct predictions divided by the total number of samples.
- **Queries Per Second (QPS)**: Average number of cases processed per second. We use this metric to evaluate the model inference speed.

Table 4: Model comparison between our methods and baseline methods. ACC denotes accuracy (all ACC metrics in the table are percentage numbers with % omitted). Specially for MNLI, we average the results of matched and mismatched validation set.

Model		Performance (ACC)					Inference	Parameters
		DeepQA	MNLI	SNLI	QNLI	RTE	Speed(QPS)	(M)
Original Model	BERT-3	75.78	70.77	77.75	78.51	57.42	207	45.69
Teacher Model	$\text{BERT}_{\text{large}}$	81.47	79.10	80.90	90.30	68.23	16	333.58
	$\text{BERT}_{\text{large}}$ ensemble	81.66	79.57	81.39	90.91	70.75	16/3	333.58*3
Traditional Distillation Model	Bi-LSTM (1-o-1)	71.69	59.39	69.59	69.12	56.31	207	50.44
	Bi-LSTM (1_{avg}-o-1)	71.93	59.60	70.04	69.53	57.35	207	50.44
	Bi-LSTM (m-o-m)	72.04	61.71	72.89	69.89	58.12	207/3	50.44*3
	BERT-3 (1-o-1)	77.35	71.07	78.62	77.65	55.23	217	45.69
	BERT-3 (1_{avg}-o-1)	77.63	70.63	78.64	78.20	58.12	217	45.69
	BERT-3 (m-o-m)	77.44	71.28	78.71	77.90	57.40	217/3	45.69*3
Our Distillation Model	Bi-LSTM ($\text{TMKD}_{\text{base}}$)	74.73	61.68	71.71	69.99	62.74	207	50.45
	*$\text{TMKD}_{\text{base}}$	79.93	71.29	78.35	83.53	66.64	217	45.70
	*$\text{TMKD}_{\text{large}}$	**80.43**	**73.93**	**79.48**	**86.44**	**67.50**	**217**	**45.70**

* These two models are BERT-3 based models.

4.3 Baselines

We compare our model with several strong baselines to verify the effectiveness of our approach.

- **BERT-3**: a student model without any knowledge distillation but instead trained as a small version of BERT/GPT, which initilized by the bottom 3-layer weight of BERT.
- **$\text{BERT}_{\text{Large}}$ [5]**: We use the $\text{BERT}_{\text{large}}$ fine-tuning model (24-layer transformer blocks, 1024 hidden size, and 16 heads) as another strong baseline.
- **$\text{BERT}_{\text{Large}}$ Ensemble**: We use $\text{BERT}_{\text{large}}$ fine-tuning model ensemble as another strong baseline (the output probability distribution decided by the average probability distributions of all models).
- **Single Student Model (1-o-1 and 1_{avg}-o-1) [11]**: Student model learns from one single teacher model using knowledge distillation. For teacher model selection, we have two strategies. Firstly, we pick the best model selected from **Original BERT** teacher models to distill one single model (called 1-o-1). Secondly, we pick the average score of teacher models as another special teacher to distill one single student (called 1_{avg}-o-1). We implement this method under two architectures: BERT-3 model and Bi-LSTM model. In the following sections, where we do not clarify the basic model is BERT-3 model.
- **Student Model Ensemble (m-o-m)**: For each teacher model, 1-o-1 is used to train a single student model. Based on this method, 3 separate student models are trained based on 3 different teacher models. Finally an ensemble aggregation is used by simply averaging the output scores to form the final results. We also implement it under BERT-3 base model and Bi-LSTM model.

4.4 Parameter Settings

All teacher models are trained using $\text{BERT}_{\text{large}}$ with batch size of 128 for 10 epochs, and max sequence length as 150. On each dataset, we train three different teacher models with different learning rates in $\{2, 3, 5\} \times 10^{-5}$. For BERT-3 student model, we optimize the student model using a learning rate of 1×10^{-4}, and all BERT-based models are initialized using pre-trained BERT model weights. For all BERT based models, we implement on top of the PyTorch implementation of BERT[4].

For all Bi-LSTM based models, we set the LSTM hidden units as 256, LSTM layer count as 2, and word embedding dimension as 300. Top 15 thousands of words are selected as vocabulary and 300 dimension Glove is used for embedding weight initialization. Words not in Glove vocabulary are randomly initialized with normal distribution. The parameters are optimized using Adam optimizer with learning rate as 1×10^{-3}.

Those teacher models used for TMKD and m-o-m training are identical for fair comparison. The only difference between $\text{TMKD}_{\text{base}}$ and $\text{TMKD}_{\text{large}}$ is the training data in the distillation pre-training stage. To be more specific, $\text{TMKD}_{\text{base}}$ leverages CommQA-Unlabeled base corpus for pre-training while $\text{TMKD}_{\text{large}}$ is pre-trained using CommQA-Unlabeled large corpus.

4.5 Comparison Against Baselines

In this section, we conduct experiments to compare TMKD with baselines in terms of three dimensions, i.e. inference speed, parameter size and performance on task specific test set. From the results shown in Table 4, it is intuitive to have the following observations:

- It is not surprising that original BERT teacher model shows the best performance due to its sheer amount of parameters (340M), but inference speed is super slow and memory consumption is huge for production usage.
- 1-o-1 and 1_{avg}-o-1 (BERT-3 and Bi-LSTM) obtain pretty good results regarding inference speed and memory capacity. However there are still some gaps compared to the original BERT model in terms of ACC metric.
- m-o-m performs better than 1-o-1. However, the inference speed and memory consumption increase in proportion to the number of student models used for ensemble.
- Compared with 1-o-1, 1_{avg}-o-1 and m-o-m, TMKD achieves optimum in all three dimensions. In terms of memory, TMKD only needs small amount of additional memory consumption since the majority of parameters are shared across different distillation tasks compared with the 1-o-1. In addition,

[4] github.com/huggingface/pytorch-pretrained-BERT.

TMKD performs significant better than BERT-3, which further proves the effective of our model.

To conclude, TMKD performs better in three dimensions than several strong baseline compressed models with knowledge distillation (i.e. 1-o-1, 1_{avg}-o-1, m-o-m) on all the evaluation datasets, and also further decreases performance gap with the original BERT model, which verifies the effectiveness of TMKD.

5 ABLATION STUDIES

TMKD consists of multiple teacher distillation pre-training stage and distillation fine-tuning stage. In this section, we further conduct several experiments to analyze the contribution of each factor in TMKD, in order to obtain a better understanding of the proposed approach.

5.1 Impact of Different Training Stages

5.1.1 Impact of Distillation Pre-training Stage. One advantage of TMKD is to introduce a multi-teacher distillation task for student model pre-training to boost model performance. We analyze the impact of pre-training stage by evaluating two new models:

TKD: A 3-layer $BERT_{base}$ model which is firstly trained using 1-o-1 distillation pre-training on CommQA-Unlabeled large-scale dataset (i.e. 40M <Question, Passage> pairs), then fine-tuned on task specific corpus with golden label and single soft label (i.e. by only one teacher) of each task.

KD (1-o-1): Another 3-layer $BERT_{base}$ model which is fine-tuned on task specific corpus with golden label and single soft label of each task but without distillation pre-training stage.

From the results in Table 5, we have the following observations: (1) On DeepQA dataset, TKD shows significant gains by leveraging large-scale unsupervised Q&A pairs for distillation pre-training. (2) Although Q&A task is different with GLUE tasks, the student model of GLUE tasks still benefit a lot from the distillation pre-training stage leveraging Q&A task. This proves the effect of the distillation pre-training stage leveraging Q&A large corpus.

Table 5: Comparison between KD and TKD

Model	Performance (ACC)				
	DeepQA	MNLI	SNLI	QNLI	RTE
KD (1-o-1)	77.35	71.07	78.62	77.65	55.23
TKD	**80.12**	**72.34**	**78.23**	**85.89**	**67.35**

5.1.2 Impact of Multi-teacher Distillation vs Single-teacher Distillation. Another advantage of TMKD is designing a unified framework to jointly learn from multiple teachers. We analyze the impact of multi-teacher versus single-teacher knowledge distillation by the following three models:

MKD: A 3-layer $BERT_{base}$ model trained by Multi-teacher distillation (m-o-1) without pre-training stage.

KD (1_{avg}-o-1): A 3-layer $BERT_{base}$ model trained by Single-teacher distillation (1_{avg}-o-1) without pre-training stage, which is to learn from the average score of teacher models.

From Table 6, MKD outperforms KD (1_{avg}-o-1) on the majority of tasks, which demonstrates that multi-teacher distillation approach (m-o-1) is able to help student model learn more generalized knowledge by fusing knowledge from different teachers.

Table 6: Comparison Between KD (1_{avg}-o-1) and MKD

Model	Performance (ACC)				
	DeepQA	MNLI	SNLI	QNLI	RTE
KD (1_{avg}-o-1)	77.63	70.63	78.64	**78.20**	58.12
MKD	**78.21**	**71.98**	**78.80**	77.80	**59.92**

5.1.3 Dual-Impact of Two Stages. Finally, TKD, MKD and TMKD are compared altogether. From Figure 2, TMKD significantly outperforms TKD and MKD in all datasets, which verifies the complementary impact of the two stages (distillation pre-training & m-o-1 fine-tuning) for the best results.

5.1.4 Extensive Experiments: Multi-teacher Ensemble or Multi-teacher Distillation? TMKD leverage multi-teacher distillation in both pre-training and task specific fine-tuning stages. This multi-teacher mechanism actually introduces multi-source information from different teachers. A common approach to introduce multi-source information is *ensemble* (e.g. average score of the prediction outputs from multiple models). Compared with the common multi-teacher ensemble approach, are there extra benefits from multi-teacher distillation? We conduct further experiments to explore this question.

For clear comparisons, we apply some degradation operations to TMKD. We remove the multi-teacher distillation mechanism from TMKD, and then use ensemble teacher score (the average score of soft labels by multiple teachers) and single teacher score (from the best teacher) to train two new models with a two-stage setting respectively, which are denoted as TKD_{base} (1_{avg}-o-1) and TKD_{base} (1-o-1). Experiments using both BERT-3 and Bi-LSTM as the student model architecture are conducted, as shown in Table 7.

Table 7: Comparison between TKD and TMKD

Model	Dataset				
	DeepQA	MNLI	SNLI	QNLI	RTE
Bi-LSTM (TKD_{base} (1-o-1))	74.26	61.43	71.54	69.2	59.56
Bi-LSTM (TKD_{base} (1_{avg}-o-1))	74.38	61.55	71.7	69.08	61.01
Bi-LSTM ($TMKD_{base}$)	**74.73**	**61.68**	**71.71**	**69.99**	**62.74**
*TKD_{base} (1-o-1)	79.5	71.07	77.66	82.79	63.89
*TKD_{base} (1_{avg}-o-1)	79.73	71.21	77.70	83.40	**67.10**
*$TMKD_{base}$	**79.93**	**71.29**	**78.35**	**83.53**	66.64

* These three models are BERT-3 based models.

From the results, we have the following observations: (1) For both BERT-3 ad Bi-LSTM based models, the TKD_{base}(1_{avg}-o-1) performs better than TKD_{base}(1-o-1). This demonstrates that ensemble of teacher models is able to provide more robust knowledge than single teacher model when distill the student model. (2) Compared with TKD_{base}(1-o-1) and TKD_{base}(1_{avg}-o-1), $TMKD_{base}$ obtains the best performance no matter using Bi-LSTM or BERT-3. It because that the multi-source information was diluted by the average score. TMKD introduces the differences when training, the multi-source information can be adpative at the training stage.

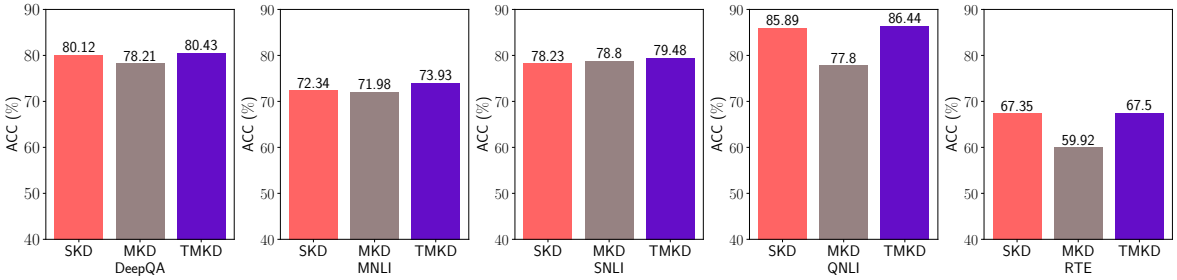

Figure 2: Performance comparison of TKD, MKD and TMKD on different datasets

5.2 Impact of Training Data Size

To further evaluate the potential of TMKD, we conduct extensive experiments on CommQA-Unlabeled extremely large-scale corpus data (0.1 billion unlabeled Q&A pairs) and CommQA-Labeled (12M labeled Q&A labeled pairs). Four separate teacher models (T_1-T_4) are trained with batch size of 128 and learning rate with $\{2, 3, 4, 5\} * e^{-5}$. Max sequence length is set as 200, and number of epochs as 4. The settings of KD, MKD, and TMKD keep the same as Section 5.1. The results are shown in Table 8. Interestingly, on this extremely large Q&A dataset, TMKD even exceeds the performance of its teacher model (ACC: 79.22 vs 77.00), which further verifies the effectiveness of our approach.

Table 8: Extremely large Q&A dataset results.

Performance (ACC)			
$\text{BERT}_{\text{large}}$	KD	MKD	TMKD
77.00	73.22	77.32	**79.22**

5.3 Impact of Transformer Layer Count

In this section, we discuss the impact of transformer layer count n for TMKD[5] with $n \in \{1, 3, 5\}$. As observed from Table 9: (1) With n increasing, ACC increases as well but inference speed decreases, which aligns with our intuition. (2) With n increasing, the performance gain between two consecutive trials decreases. That say, when n increases from 1 to 3, the ACC gains of the 5 datasets are (3.87, 9.90, 7.46, 11.44, 11.19) which are very big jump; while n increases from 3 to 5, gains decrease to (1.08, 1.63, 0.53, 2.89, 0.37), without decent add-on value compared with the significantly decreased QPS.

Table 9: Compare different number of transformer layer.

Dataset	Metrics	Layer Number		
		1	3	5
DeepQA	ACC	74.59	78.46	79.54
MNLI	ACC	61.23	71.13	72.76
SNLI	ACC	70.21	77.67	78.20
QNLI	ACC	70.60	82.04	84.94
RTE	ACC	54.51	65.70	66.07
	QPS	511	217	141

Based on the above results, we set n as 3 since it has the highest performance/QPS ratio for web Question Answering System. In real production scenarios, we need to balance between performance and latency.

[5]In order to save experimental costs, we choose $\text{TMKD}_{\text{base}}$ for experimentation.

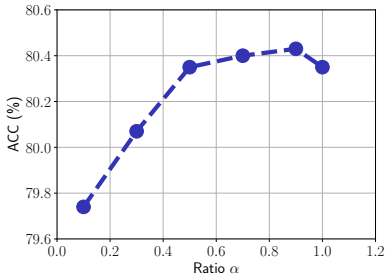

Figure 3: The impact of different loss weighted ratio.

5.4 Impact of Loss Weighted Ratio

We also conducts several experiments to analyze the impact of the loss weighted ratio α defined in Section 3.3, where $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. Specially, when set the ratio as 1.0, we only use the soft label headers to calculate the final output result. The results of TMKD against different α values are shown in Figure 3. We can observe: (1) The larger value the ratio is, the better performance is obtained (except when α is 1.0). (2) Without the golden label supervision (i.e. α is 1.0), the performance decreases. The intuition is just like the knowledge learning process of human beings. We learn knowledge not only from teachers but also through reading books which can provide us a comprehensive way to master knowledge with less bias.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel Two-stage Multi-teacher Knowledge Distillation (**TMKD**) approach for model compression. Firstly a Q&A multi-teacher distillation task is proposed for student model pre-training, then a multi-teacher paradigm is designed to jointly learn from multiple teacher models (m-o-1) for more generalized knowledge distillation on downstream specific tasks. Experiment results show that our proposed method outperforms the baseline state-of-art methods by great margin and even achieves comparable results with the original teacher models, along with significant speedup of model inference. The compressed Q&A model with TMKD has already been applied to real commercial scenarios which brings significant gains.

In the future, we will extend our methods to more NLU tasks, such as sequence labelling, machine reading comprehension, etc. On the other hand, we will explore how to select teacher models more effectively for better student model distillation.

REFERENCES

[1] Philipp Cimiano, Christina Unger, and John McCrae. 2014. Ontology-based interpretation of natural language. *Synthesis Lectures on Human Language Technologies* 7, 2 (2014), 1–178.

[2] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 5931–5937.

[3] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167. https://doi.org/10.1145/1390156.1390177

[4] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 1269–1277.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019), 4171–4186.

[6] Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V Chawla. 2014. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 15–24. https://doi.org/10.1145/2623330.2623703

[7] Murhaf Fares, Stephan Oepen, and Erik Velldal. 2018. Transfer and Multi-Task Learning for Noun-Noun Compound Interpretation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. 1488–1498.

[8] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[9] Babak Hassibi and David G. Stork. 1993. Second order derivatives for network pruning: Optimal Brain Surgeon. In *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles (Eds.). Morgan-Kaufmann, 164–171.

[10] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 1398–1406. https://doi.org/10.1109/ICCV.2017.155

[11] Geoffrey E Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv: Machine Learning* (2015).

[12] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. 328–339. https://doi.org/10.18653/v1/P18-1031

[13] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866* (2014).

[14] Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1317–1327.

[15] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1744–1753.

[16] Yann LeCun, John S. Denker, and Sara A. Solla. 1989. Optimal Brain Damage. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. 598–605.

[17] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David J. Crandall, and Dhruv Batra. 2015. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks. *CoRR* abs/1511.06314 (2015). arXiv:1511.06314 http://arxiv.org/abs/1511.06314

[18] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding. *CoRR* abs/1904.09482 (2019).

[19] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 4487–4496.

[20] Lili Mou, Ran Jia, Yan Xu, Ge Li, Lu Zhang, and Zhi Jin. 2016. Distilling Word Embeddings: An Encoding Approach. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. 1977–1980. https://doi.org/10.1145/2983323.2983888

[21] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[22] Anastasia Pentina and Christoph H Lampert. 2017. Multi-Task Learning with Labeled and Unlabeled Tasks. *stat* 1050 (2017), 1.

[23] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. 2227–2237.

[24] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *CoRR* abs/1802.05668 (2018).

[25] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.

[26] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[27] Meng-Chieh Wu, Ching-Te Chiu, and Kun-Hsuan Wu. 2019. Multi-teacher Knowledge Distillation for Compressed Video Action Recognition on Deep Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. 2202–2206.

[28] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. 2015. Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 676–684. https://doi.org/10.1109/CVPR.2015.7298667

[29] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. 2017. Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. 1285–1294.

[30] Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Reducing BERT Pre-Training Time from 3 Days to 76 Minutes. *CoRR* abs/1904.00962 (2019).

[31] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. 2015. Efficient and accurate approximations of nonlinear convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 1984–1992. https://doi.org/10.1109/CVPR.2015.7298809