Reinforced Multi-Teacher Selection for Knowledge Distillation

Fei Yuan, Linjun Shou, Jian Pei, Wutao Lin, Ming Gong, Yan Fu, Daxin Jiang

1 University of Electronic Science and Technology of China
2 Microsoft STCA NLP Group
3 School of Computing Science, Simon Fraser University

feiyuan@std.uestc.edu.cn, {lisho, wutlin, migon, djiang}@microsoft.com
jpei@cs.sfu.ca, fuyan@uestc.edu.cn

Abstract
In natural language processing (NLP) tasks, slow inference speed and huge footprints in GPU usage remain the bottleneck of applying pre-trained deep models in production. As a popular method for model compression, knowledge distillation transfers knowledge from one or multiple large (teacher) models to a small (student) model. When multiple teacher models are available in distillation, the state-of-the-art methods assign a fixed weight to a teacher model in the whole distillation. Furthermore, most of the existing methods allocate an equal weight to every teacher model. In this paper, we observe that, due to the complexity of training examples and the differences in student model capability, learning differentially from teacher models can lead to better performance of student models distilled. We systematically develop a reinforced method to dynamically assign weights to teacher models for different training instances and optimize the performance of student model. Our extensive experimental results on several NLP tasks clearly verify the feasibility and effectiveness of our approach.

Introduction
Deep pre-trained models, such as BERT (Devlin et al. 2018), XLNet (Yang et al. 2019), RoBERTa (Liu et al. 2019) and ALBERT (Lan et al. 2019), have proved effective on many NLP tasks by establishing record-breaking state-of-the-art results. However, due to huge amounts of model parameters, typically at the magnitude of hundreds of millions or even billions, the bottleneck for applying those pre-trained models in production is the slow inference speed and the huge footprints in using GPUs. To save the computation cost and speed up the inference process, knowledge distillation (KD) (Hinton, Vinyals, and Dean 2015) as an effective approach to compress large models into smaller ones stands out and becomes the de facto best choice among other alternatives, such as pruning (Han, Mao, and Dally 2015) and quantization (Gong et al. 2014).

The knowledge distillation approach is based on a teacher-student learning paradigm, where a teacher model, which is often large, is taken and the output of the teacher model is integrated as a soft target in the loss function to train a student model, which is often small. The teacher-student learning paradigm demonstrates excellent performance on various NLP tasks (Kim and Rush 2016; Tang et al. 2019). Knowledge distillation methods start from a single teacher. Some recent methods employ multiple teachers and show great promises in further boosting student model performance effectively.

Most of the existing methods using multiple teachers simply assign an equal weight to all teacher models during the whole distillation process. The uniform distribution of weights among all teachers keeps the coordination, management and implementation of the multi-teacher framework simple. At the same time, the indifference to strengths and weaknesses of various teacher models leaves a huge untouched space for better knowledge distillation. To make a multi-teacher approach work well, teacher models have to be diverse. Exploiting the diverse strengths of various teacher models can bring in huge advantages.

Individual teacher models may perform differently on various instances. Different models may vary in hypothesis space, optimization strategy, parameter initialization and many other factors, which result in different performance among different cases. Ideally, we want to assign different weights to different teacher models for different training instances according to their performance in individual cases.

Differentiating among teacher models is far from trivial. Surprisingly, a stronger teacher model may not necessarily lead to a better student model. As shown in Table 1 (Sun et al. 2019), the RoBERTa-Base model performs better than BERT-Base. However, the student model distilled from BERT-Base, the weaker model, performs better than the same student model distilled from RoBERTa-Base, the stronger teacher model.

Table 1: RoBERTa-Base performs better than BERT-Base. However, the student model distilled from BERT-Base, the weaker model, performs better than the same student model distilled from RoBERTa-Base, the stronger teacher model.
the BERT-Base model on the MRPC and MNLI-mm tasks. However, the student model using three-layer transformer BERT distilled from the weaker teacher model performs better on the same tasks than the same student model distilled from the stronger teacher model. One possible reason is that the effectiveness of distillation may be bounded by the capability of the student model. A simple student model with fewer parameters may not be able to approximate a very complex teacher model, since the complex teacher model may capture finer-grained patterns in data and cause the student model to overfit in some parts of the data and under some other parts. To achieve good distillation, we have to choose teacher models matching capacities of student models.

Based on the above insights, in this paper, we systematically study how to coordinate teacher models and student models in knowledge distillation. Specifically, we investigate how to assign appropriate weights to different teacher models on various training samples. To the best of our knowledge, we are the first to treat teacher models deferentially at instance level in knowledge distillation.

We formulate the teacher model selection problem under a reinforcement learning framework: the decision is made based on the characteristics of training examples and the outputs of teacher models, while the policy is learned towards maximizing the student performance as the return.

To verify the effectiveness of our approach in NLP, we conduct extensive experiments on several important tasks from the GLUE benchmark (Wang et al. 2019), including sentiment analysis, paraphrase similarity matching and natural language inference. Our experimental results clearly show that our reinforced multi-teacher selection strategy boosts the performance of student models substantially. Our method is not only principled but also practical in major NLP tasks.

Our contributions are twofold. First, this is the first systematic study developing sample instance-based weighting for multiple teacher models. This is a concrete and novel contribution to knowledge distillation and machine learning. Our idea is general and can be used in many applications of knowledge distillation. Second, we apply our novel idea of reinforced multi-teacher selection to a series of important NLP tasks. This is a novel contribution to the NLP domain.

Related Work

To achieve model compression, Hinton, Vinyals, and Dean (2015) propose a knowledge distillation (KD) approach based on a teacher-student framework, which substantially extends the method by Bucilă, Caruana, and Niculescu-Mizil (2006). The KD approach has been widely adopted in many applications. For example, Kim and Rush (2016) demonstrate that standard knowledge distillation is effective for neural machine translation. Recent studies (Yang et al. 2019; Sun et al. 2019; Jiao et al. 2019) distill knowledge from BERT (Devlin et al. 2018) into small student models and achieve competent results. All those methods distill knowledge from one teacher model.

To improve the performance of student models that employ deep neural networks, some recent studies leverage multiple teacher models in knowledge distillation. Chebotar and Waters (2016) simply leverage the weighted average of teacher models to distill student model, where weights are hyperparameters and fixed during training. Fukuda et al. (2017) examine two strategies to leverage labels from multiple teacher models in training student models. The first strategy updates the parameters of the student model by combining the soft labels from the teacher models with fixed weights. The other strategy randomly selects one teacher model at the mini-batch level to provide soft target labels for training student models. Wu, Chiu, and Wu (2019) also assign a fixed weight to each teacher model and use the weighted average of the probability distributions from multiple teacher models to train a student model. Yang et al. (2020) propose a two-stage multi-teacher KD method, which first pre-trains a student model for the Q&A distillation task and then fine-tunes the pre-trained student model with multi-teacher distillation on downstream tasks. Teacher models are assigned with equal weights during distillation.

All previous knowledge distillation methods using multi-teacher models fix the same weight for a teacher model on all training examples. In this paper, we learn a policy to dynamically assign weights to teacher models based on individual examples.

Our Approach

In this section, we first recall the preliminaries of knowledge distillation and teacher models/student models. Then, we present our reinforced teacher selection method. Last, we introduce the model training algorithm.

Preliminaries

In general, the knowledge distillation approach (Bucilă, Caruana, and Niculescu-Mizil 2006; Hinton, Vinyals, and Dean 2015) uses the soft output (logits) of one or multiple large models as the knowledge and transfers the knowledge to a small student model. In this paper, we mainly target at NLP tasks and thus use the state-of-the-art NLP models as examples to illustrate our approach. Our approach in general can be applied to all knowledge distillation tasks using teacher/student models.

Given a NLP task, let $D = \{(x_i, y_i)\}_{i=1}^N$ be the training set with $N$ training examples, where $x_i$ is the $i$-th input, such as a single sentence or a pair of sentences, and $y_i$ is the corresponding ground truth label. Without loss of generality, we assume that a class label $c$ is an integer between 1 and $C$, where $C$ is the number of classes in the data set.

We assume $K$ teacher models. For example, we can fine-tune the pre-trained models with 12-layer transformers, such as BERT (Devlin et al. 2018), XLNet (Yang et al. 2019), RoBERTa (Liu et al. 2019) or ALBERT (Lan et al. 2019), on the training data as possible teacher models. Denote by $M_k \ (1 \leq k \leq K)$ the $k$-th teacher model, and by $\Theta_k$ the set of model parameters of $M_k$. The output of the teacher model $M_k$ for a given input $x_i$ is written as $\hat{y}_{i,k} = \langle \hat{y}_{i,k,1}, \ldots, \hat{y}_{i,k,C} \rangle$, where $\hat{y}_{i,k,c} = P^k(y_i = c | x_i; \Theta_k)$ is the probability of $x_i$ belonging to class $c$ ($1 \leq c \leq C$) computed by model $M_k$.

For student models, we consider transformer models with fewer layers, such as those with only 3 or 6 layers. To train a student model, the distillation loss is defined as the cross-entropy loss between the predictions of the teacher model

$$
\text{CrossEntropy}(y_i, \hat{y}_{i,k}) = -\sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,k,c})
$$

where $y_{i,c}$ is the true label and $\hat{y}_{i,k,c}$ is the predicted probability. The distillation loss is defined as the cross-entropy loss between the predictions of the teacher model and the student model, weighted by the teacher model's output:

$$
\text{DistillationLoss}(y_i, \hat{y}_{i,k}) = \sum_{k=1}^{K} w_k \text{CrossEntropy}(y_i, \hat{y}_{i,k})
$$

where $w_k$ is the weight assigned to the $k$-th teacher model. In our experiments, we use $w_k = 1$ for all $k$.

Our reinforced teacher selection method leverages this idea by assigning weights to each teacher model based on the characteristics of training examples. We formulate this problem as a reinforcement learning task: given a training example $x_i$, we need to select the weights $w_k$ for each teacher model $M_k$ to minimize the distillation loss $\text{DistillationLoss}(y_i, \hat{y}_{i,k})$. We design a policy function $\pi(x_i)$ that maps each training example to a vector of weights $w_k$ for each teacher model $M_k$.

We use a reinforcement learning framework to learn this policy function. The state $s_i$ for each training example $x_i$ is a representation of the characteristics of the example, such as its length or its distribution of tokens. The action $a_i$ is a vector of weights $w_k$ for each teacher model $M_k$. The reward $r_i$ is the decrease in the distillation loss $\text{DistillationLoss}(y_i, \hat{y}_{i,k})$ after selecting the weights $w_k$ for the teacher models.

In our setup, we use a simple policy function $\pi(x_i) = \frac{1}{K}$ for all $k$, which assigns equal weight to each teacher model. This can be seen as a baseline for our method. We also use a more sophisticated policy function that learns from the characteristics of training examples. We use a neural network to learn the policy function $\pi(x_i)$, where the input is the state $s_i$ and the output is the action $a_i$.

In summary, we formulate the teacher model selection problem under a reinforcement learning framework: the decision is made based on the characteristics of training examples and the outputs of teacher models, while the policy is learned towards maximizing the student performance as the return. This way, we can distill knowledge from multiple teacher models and assign appropriate weights to each teacher model based on individual examples.
This process iterates on episodes until the performance of the student model converges.

In general, a reinforcement learning approach involves elements in the form of \((state, action, reward)\). The elements in our method are as follows.

**State** Our reinforcement learning method maintains a series of environment states \(s_1, s_2, \ldots\) that summarize the characteristics of the input instances as well as the candidate teacher models, so that judicious decisions can be made accordingly. We design a state \(s_j\) as a vector of real-values \(F(s_j)\), which includes the concatenation of three features.

The first feature is a vector representation \(R(x_i) \in \mathbb{R}^d\) of an input instance \(x_i\). In general, any sentence-encoding network can be applied here to embed a sentence in an NLP task into a vector. In this paper, we use the embedding for the [CLS] token in the last layer of the pre-trained BERT-Base model (Devlin et al. 2018).

The representation model \(R\) here is different from a BERT-Base teacher model \(M_k\). \(R\) is a pre-trained BERT-Based model aiming to effectively represent the content of input instances. A teacher model \(M_k\) is fine-tuned for a specific task, such as predicting the class label distribution given an input instance.

The second feature is the probability vector \(P_k^c(y_i = c|x_i; \Theta_k^c)\) on all classes \(1, \ldots, C\) predicted by the teacher model \(M_k\) with parameters \(\Theta_k^c\) on the input instance \(x_i\). In practice, the probability is often derived from a softmax function on the hidden representation of the input, that is,

\[
P_k^c(y_i = c|x_i; \Theta_k^c) = \frac{\text{softmax}(W_k \cdot M_k(x_i))}{c} \tag{4}
\]

where \([\cdot]_c\) refers to the \(c\)-th element in a vector, \(W_k \in \mathbb{R}^{C \times d}\) is a trainable weight matrix, and \(M_k(x_i)\) is the embedding of the [CLS] token in the last layer of \(M_k\).

The third feature is the cross entropy loss \(L_k(x_i)\) on the input instance \(x_i\) of the teacher model \(M_k\) with parameters \(\Theta_k^c\) which is the ground-truth loss of teacher model for \(x_i\).

\[
L_k(x_i) = -\sum_c \mathbb{1}[y_i = c] \cdot \log P_k^c(y_i = c|x_i; \Theta_k^c) \tag{5}
\]

For the second and third features, we also concatenate the predictions of other teacher models as state features for the policy agent to better select teachers.

**Action** Each teacher model \(M_k\) is associated with one agent. An agent chooses between two possible actions, selecting the teacher model or not for the current instance. A policy function \(\pi_\theta(s_j, a_j)\) determines the distribution over the states, from which the action value of \(a_j \in \{0, 1\}\) is sampled. Denote by \(\theta\) the trainable parameters in the policy function. In this paper, we adopt a simple logistic function as the policy model.

\[
\pi_\theta(s_j, a_j) = P_\theta(a_j|s_j) = a_j \sigma(AF(s_j) + b) + (1 - a_j)(1 - \sigma(AF(s_j) + b)) \tag{6}
\]

where \(F(s_j) \in \mathbb{R}^{d+(C+1) \times K}\) is the state vector and \(\sigma(\cdot)\) is the sigmoid function with trainable parameter \(\theta = \{A \in \mathbb{R}^{d \times (C+1) \times K}, b \in \mathbb{R}^{1 \times (C+1) \times K}\}\).
Algorithm 1: Overall Training Procedure

1. Pre-train the student model $Θ_{0}$ using knowledge distillation from the average of all teacher models by maximizing $L_{KD} = αL_{DL} + (1−α)L_{CE}$.
2. Pre-train the TS policy $θ_{0}$ by calculating the return under $Θ_{0}$ with all teacher models selected.
3. Run Algorithm 2 to iteratively train KD and TS in turn until convergence.

Algorithm 2: Joint Training of TS and KD

Input: Epoch number $L$. Training data $D = \{D_{1}, D_{2}, ..., D_{M}\}$. A KD model and a TS model initialized as $Θ^{*} = Θ^{0}$ and $θ = θ_{0}$.

for epoch $l = 1$ to $L$ do

| for each batch $D_{b} \in D$ do |
| --- |
| TS sample actions for each instance $x_{i} \in D_{b}$ with $θ$ to get selected teachers $K$ by: $a_{j} \sim π_{θ}(s_{j}, a_{j})$. Stored $(a_{j}, s_{j})$ to the episode history $H$.
| Compute the average of soft labels of the selected teachers: $\bar{p}_{i}^{j} = \frac{1}{|K|} \sum_{k \in K} p^{j}_{i}$.
| Update the parameter $Θ^{*}$ of KD by: $L_{KD} = αL_{DL} + (1−α)L_{CE}$.
| end |

| for each $(a_{j}, s_{j}) \in H$ do |
| --- |
| Compute delayed reward following Equation 7.
| end |

| Update the parameter $θ$ of TS following Equation 8. |
| end |

end

the teacher selection policy $θ$. The iteration continues for $L$ epochs. For more implementation details, please refer to Section “Implementation Details”.

Experiments

In this section, we first describe our experimental setup, and then introduce the baseline methods and implementation details of our method. Last, we report the experimental results.

Data Sets and Evaluation Metric

Following Patient KD (Sun et al. 2019), we evaluate our proposed approach on three different NLP tasks from the GLUE benchmark (Wang et al. 2019), namely Sentiment Classification (SC), Paraphrase Similarity Matching (PSM) and Natural Language Inference (NLI). The statistics of the data sets are shown in Table 2. We use prediction accuracy as the metric in evaluation.

For SC, we experiment on Stanford Sentiment Treebank (SST-2) (Socher et al. 2013). The target is to predict the sentiment of a given sentence.

For PSM, we use Microsoft Research Paragraph Corpus (MRPC) (Dolan and Brockett 2005) and Quora Question Pairs (QQP) (Wang et al. 2019). The goal on both data sets is to decide whether two sentences of questions are semantically equivalent.

For NLI, we evaluate on Multi-Genre Natural Language Inference (MNLI) (Williams, Nangia, and Bowman 2017).
We fine-tune the pre-trained models BERT_{12}, RoBERTa_{12}, ALBERT_{12} and XLNet_{12} on task specific training data to get four candidate teacher models, where the subscript 12 means 12 layers of transformers. Similar to Patient KD \cite{sun2019patient}, we consider the BERT models with 3 layers and 6 layers of transformers as two student models, denoted by BERT_{3} and BERT_{6}, respectively.

To validate the effectiveness of our proposed method, we compare with various baselines reviewed in Related work.

**Single Teacher KD** \cite{hinton2015distilling} is applied to train student models from BERT_{12}, RoBERTa_{12}, ALBERT_{12} and XLNet_{12} individually, which is referred to as Vanilla KD (V-KD) in the rest of this section.

**U-Ensemble Teacher** is our implementation of the equal weight method by \cite{yang2020warm}. Every teacher model is assigned an equal weight in KD, and the student model learns from an aggregated distribution by averaging the outputs of all teacher models.

**Rand-Single-Ensemble Teacher** uses the strategy of \cite{fukuda2017concrete} to randomly select one teacher from the teacher model candidates at mini-batch level to provide soft-targets for training student model.

**W-Ensemble Teacher** is a weighted assemble method following \cite{Chebotar2016, fukuda2017concrete, wu2019KD}. We assign a different weight to each teacher model. The weights are fixed during the whole distillation.

Besides these existing baseline methods, we further propose two strong baselines for comparison.

**LR-Ensemble Teacher** uses Logistic Regression (LR) model to model the best weights for each teacher candidate, instead of setting weights using heuristics like U-Ensemble/W-Ensemble. Specifically, let the aggregated distribution \( P(y_i = c | x_i) = \sum_k w_k P_k(y_i = c | x_i; \Theta_k) \). We learn \( w_k \) by maximizing the performance of the weighted ensemble teacher model. Since the training of the best weights can be conducted on either the training set or the development set, we denote by LR-Train-Ensemble and LR-Dev-Ensemble the corresponding LR models, respectively.

All the above baseline methods either use a single teacher model or assign fixed weights to ensemble multiple teacher models. As the last baseline, **Best-Single-Ensemble Teacher** considers an instance-level teacher model selection method, where for each instance in the training set (where ground-truth is available), the best performing single teacher model, that is, the one achieving the lowest cross entropy loss, is selected to train student model. We can treat this model as the upper bound of selecting best teacher model. Please note that this model cannot be evaluated on any test set since we are not supposed to use the ground-truth labels there for tuning models.

### Implementation Details

The training code is built on top of the code repository of Patient KD \cite{sun2019patient}. All tasks in our experiments can be treated as classification problems where the input is either a single sentence or a pair sentences. For the tasks whose inputs are individual sentences, the model input has the form of \([CLS] sentence_1 [SEP] \). For tasks whose input is sentence pairs, the input form is \([CLS] sentence_1 [SEP] sentence_2 [SEP] \).

To fine-tune the teacher models, we adopt the open-sourced pre-trained weights for BERT_{12}, RoBERTa_{12}, ALBERT_{12} and XLNet_{12} as initialization. The learning rate is set to \{1e-5, 2e-5, 5e-5\}. The batch size is set to 32. The maximum sequence length is set to 128. The number of epochs is set to 4. The best model is selected according to the accuracies on the development set.

The student models, BERT_{3} and BERT_{6}, are initialized by the bottom 3 and 6 layers of BERT-Base respectively. Meanwhile, we set the batch size to 32, the number of epochs to 4, the maximum length of sequence to 128, the learning rate to \{1e-5, 2e-5, 5e-5\}, the distillation temperature \( T \) to \{5, 10, 20\}, and the loss equilibrium coefficient \( \alpha \) to \{0.2, 0.5, 0.7\}. We choose the best model based on the performance on the development set. The \( \gamma \) in the experiments ranges from \{0.3, 0.5, 0.7, 0.9\}, which is selected based on development set performance.

For the teacher model selector, our policy function is a simple logistic regression model. After feeding the input sequence encoded by the original pre-trained BERT-Base model into the teacher model selector, we adopt a standard Monte-Carlo based policy gradient method \cite{williams1992simple} to optimize the parameters of the policy model.

**KD Pretraining** Our KD model is initialized using the pre-trained BERT model weights. Then, we use the distillation task in question to pre-train the KD model to learn from the average ensemble teacher model (i.e., average of all teacher model predictions). We set the batch size to 32 and max epochs to 4. The other hyper-parameters are kept the same as

Table 2: Statistics of the datasets for experiment.

| Dataset    | #Train | #Dev  | #Test  |
|------------|--------|-------|--------|
| RTE        | 2,490  | 277   | 3,000  |
| MRPC       | 3,668  | 408   | 1,725  |
| SST-2      | 67,349 | 872   | 1,821  |
| QNLI       | 104,743| 5,463 | 5,463  |
| MNLI-mm    | 392,702| 9,832 | 9,847  |
| MNLI-m     | 392,702| 9,815 | 9,796  |
| QQP        | 363,849| 40,430| 390,965|

*Question-Answering Natural Language Inference (QNLI) and Recognizing Textual Entailment (RTE). MNLI is a corpus of sentence pairs extracted from multiple domains, which are further divided into two splits, in-domain (MNLI-m) and cross-domain (MNLI-mm), to evaluate the model generality. QNLI is a data set converted from a question-answering data set SQuAD \cite{rajpurkar2016squad}, and is designed to predict whether the context sentence contains an answer to the question. RTE is based on a series of textual entailment challenges from General Language Understanding Evaluation (GLUE) \cite{wang2018glue}.*

---

\cite{https://github.com/intersun/PKD-for-BERT-Model-Compression} \cite{https://github.com/google-research/bert}
the normal KD training mentioned above.

**TS Pretraining** Pretraining is recommended by many reinforcement learning methods (Bahdanau et al. 2016; Feng et al. 2018). For TS pretraining, to avoid the instability of initial student model and speed up the training process, we leverage the performance of the average ensemble of the selected teacher models as the reward to train the TS Model.

**Iterative Training** After the pretraining stage, the KD and TS models are jointly trained alternatively in a batch-wise way. At batch #1, KD is trained while keeping TS fixed. At batch #2, TS is trained while keeping KD fixed, so on and so forth. If TS selects no teacher, this instance will not receive any reward from KD.

### Experimental Results

#### Results on Teacher Models

We start with the performance of individual teacher models and ensemble teacher models shown in Table 3. Individual teacher models perform differently in different tasks. No one single model wins on all tasks. RoBERTa achieves better results than the other models in three tasks (MNLI-mm, MNLI-m, QNL). ALBERT shows higher accuracies on tasks MRPC and RTE, while XLNet scores the highest on SST-2. This suggests that different models may learn different local optimum and bear different biases.

Moreover, ensemble of teacher models can lead to better performance. All ensemble teacher models achieve better results than a single teacher model. This verifies that model ensemble is effective to mitigate the biases in various teacher models. By comparing different ensemble strategies, the weighted ensemble methods including W-Ensemble and LR-Ensembles outperform U-Ensemble. It indicates that smartly assigning weights to different models can lead to better results. Among all the ensemble models, LR-Dev-Ensemble achieves the best performance. Intuitively, training the weights based on the development set may help this model gain better generalization capability. Therefore, in the next experiments where we compare student models, we pick this best teacher model as the strongest baseline to represent ensemble methods.

#### Results on Student Models

The performance of student model is shown in Table 4. A stronger teacher model may not necessarily lead to a better student. Table 5 shows that ensemble teacher models are better than single teacher models. However, the corresponding student models distilled from those stronger teacher models are not always stronger. For example, the BERT$_3$ student model distilled from ALBERT$_{12}$ is better than that from LR-Dev-Ensemble on MNLI-mm, MNLI-m and SST-2. The BERT$_6$ student model distilled from XLNET$_{12}$ is also better than or on par with that from LR-Dev-Ensemble on MRPC and SST-2.

Even in the extreme case where Best-Single-Ensemble is used as the teacher model and we always choose the best performing teacher whose output is the closest to the ground-truth label, the corresponding student model does not always perform the best. Take BERT$_6$ as example. The student model from the stronger teacher, Best-Single-Ensemble, is inferior to that from LR-Dev-Ensemble on almost all data sets. This observation indeed motivates our RL-KD method, that is, the best teacher model may not necessarily perform the best in distillation. Otherwise, we just use the ground truth label as the teacher and do not need KD in the first place.

**RL-KD consistently outperforms the other methods**, including those strong baselines with ensemble teacher models. The student models (BERT$_3$ and BERT$_6$) trained by our proposed RL-KD method show consistently better results on most cases, while the performance of the student models distilled from different baseline KD methods varies on different data sets. This verifies that our proposed RL-KD method learns to adapt to the capability of the student models and dynamically selects the “most suitable teachers” in the training of student models. Among the three reward functions proposed, reward$_3$ performs the best. This is consistent with our intuition that this reward prefers the student that not only fits the training set well, but also is able to generalize to validation/development set by adding the acc on development set as reward.

The only exception is on the MRPC set for BERT$_3$, where the size of training data for this task is relatively small (see Table 2). For a weak student model such as BERT$_3$, when the training size is small, the reward computed on top of this model may not be robust, which is also indicated by the relatively lower accuracy on the MRPC sets. Consequently, the policy for teacher model selection may be compromised. As a comparison, for student model BERT$_6$, the model generalization ability is stronger. Therefore, even a small size

| Teacher Model | QQP    | MRPC   | MNLI-mm | RTE    | MNLI-m | QNL    | SST-2 | AVG. |
|---------------|--------|--------|---------|--------|--------|--------|-------|------|
| BERT$_{12}$   | 90.9   | 83.3   | 83.8    | 63.9   | 83.6   | 91.3   | 92.1  | 84.1 |
| RoBERTa$_{12}$| 91.3   | 88.5   | 86.8    | 70.8   | 86.5   | 91.9   | 93.7  | 87.1 |
| XLNet$_{12}$  | 91.0   | 87.5   | 85.8    | 71.8   | 85.3   | 91.5   | 93.8  | 86.7 |
| ALBERT$_{12}$ | 90.6   | 88.7   | 84.3    | 74.0   | 83.9   | 91.6   | 91.2  | 86.3 |
| U-Ensemble   | **92.3** | 89.2 | **87.5** | 72.6   | 87.3   | **93.0** | 93.7  | 87.9 |
| W-Ensemble   | **92.3** | 89.5 | **87.5** | 72.9   | **87.4** | **93.0** | 93.6  | 88.0 |
| Rand-Single-Ensemble | 90.7 | 88.5 | 85.4 | 72.6 | 84.4 | 91.3 | 92.7 | 86.5 |
| LR-Train-Ensemble | 92.1 | 89.5 | 86.7 | 71.8 | 86.5 | 92.5 | 92.9 | 87.4 |
| LR-Dev-Ensemble | **92.3** | **90.0** | 87.3 | **74.4** | 87.1 | **93.0** | **94.0** | **88.3** |

Table 3: Performance of various teacher models on test sets, including both individual models and ensemble models by various ensemble strategies.
We conduct 5 runs of models training and calculate the mean and standard deviation (Stdev) values. Besides, we also conduct a two-sided statistically significant t-test (p-Value) with threshold 0.05 comparing baseline methods with our RL-KD method. The experimental results are listed in Table 5. Results shows that (1) the variance of our approach is similar to baseline KD. (2) our method outperforms baselines with statistical significance.

### Conclusions

In this paper, we tackle the problem of teacher model selection in knowledge distillation when multiple teacher models are available. We propose a novel RL-based approach, which dynamically assigns weights to teacher models at instance level to better adapt to the strengths of teacher models as well as the capability of student models. The extensive experiments on several NLP tasks verify the effectiveness of our proposed approach.

### Acknowledgments

Jian Pei’s research is supported in part by the NSERC Discovery Grant program. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

### References

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; and Bengio, Y. 2016. An actor-
critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Buciluă, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535–541. ACM.

Chebotar, Y.; and Waters, A. 2016. Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition. In *Interspeech*, 3439–3443.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Fukuda, T.; Suzuki, M.; Kurata, G.; Thomas, S.; Cui, J.; and Ramabhadran, B. 2017. Efficient Knowledge Distillation from an Ensemble of Teachers. In *Interspeech*, 3697–3701.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Han, S.; Mao, H.; and Dally, W. 2015. Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 1135–1143.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Kim, Y.; and Rush, A. M. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.