Reinforced Multi-Teacher Selection for Knowledge Distillation

Fei Yuan[1*] **Linjun Shou**[2] **Jian Pei**[3] **Wutao Lin**[2] **Ming Gong**[2] **Yan Fu**[1] **Daxin Jiang**[2†]
[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; Han et al. 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.

Teacher	Student	MRPC	MNLI-mm
		Acc	Acc
BERT-Base	N / A	83.3	83.8
RoBERTa-Base	N / A	88.5	86.8
BERT-Base	$BERT_3$	74.0	75.3
RoBERTa-Base	$BERT_3$	73.0	71.9

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.

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

[*] Work is done during internship at Microsoft STCA NLP Group.
[†] Corresponding author.

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 Buciluǎ, 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 (Tang 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 hyper-parameters 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 (Buciluǎ, 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 $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ be the training set with N training examples, where $\mathbf{x}_i$ is the i-th input, such as a single sentence or a pair of sentences, and $\mathbf{y}_i$ 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 $\mathcal{M}_k$ ($1 \leq k \leq K$) the k-th teacher model, and by Θ_k^t the set of model parameters of $\mathcal{M}_k$. The output of teacher model $\mathcal{M}_k$ for a given input $\mathbf{x}_i$ is written as $\tilde{\mathbf{y}}_{i,k}^t = \langle \tilde{y}_{i,k,1}^t, \ldots, \tilde{y}_{i,k,C}^t \rangle$, where $\tilde{y}_{i,k,c}^t = P^t(\mathbf{y}_i = c|\mathbf{x}_i; \Theta^t)$ is the probability of $\mathbf{x}_i$ belonging to class c ($1 \leq c \leq C$) computed by model $\mathcal{M}_k$.

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

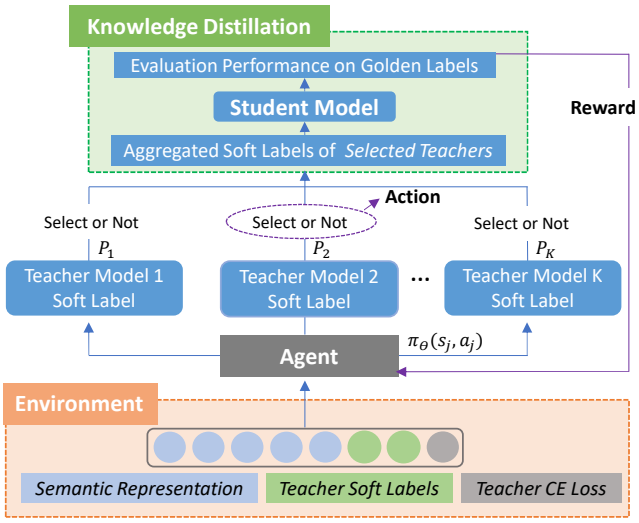

Figure 1: Overview of RL-KD, Reinforced Multi-Teacher Selection for Knowledge Distillation.

and that of the student model, that is,

$$\mathcal{L}_{DL} = -\frac{1}{K}\sum_{i,k,c}\left[\tilde{y}^t_{i,k,c} \cdot \log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^s)\right] \quad (1)$$

where Θ^s is the set of parameters for the student model and $P^s(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^s)$ is the probability of example $\mathbf{x}_i$ belonging to class c computed by the student model.

Besides the distillation loss, a ground-truth loss is also introduced in the training of student model.

$$\mathcal{L}_{CE} = -\sum_{i,c}\mathbb{I}\left[\mathbf{y}_i = c\right] \cdot \log P^s(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^s) \quad (2)$$

where $\mathbb{I}\left[\mathbf{y}_i = c\right]$ is the one-hot vector from the ground truth label. The final objective of KD is

$$\mathcal{L}_{KD} = \alpha\mathcal{L}_{DL} + (1-\alpha)\mathcal{L}_{CE} \quad (3)$$

where α is a hyper-parameter to balance between the soft teacher label $\mathcal{L}_{DL}$ and the hard ground-truth label $\mathcal{L}_{CE}$.

RL based Teacher Selector (TS)

The original KD methods (Buciluǎ, Caruana, and Niculescu-Mizil 2006; Hinton, Vinyals, and Dean 2015) assumes only one single teacher model. Although several recent studies employ multiple teacher models in distillation, as reviewed in Related Work, those methods assign a fixed weight to each teacher model. Inspired by the insights discussed in Introduction, we propose to dynamically assign weights to teacher models at instance level using a reinforced approach (Sutton 1992). The weights are implemented as the sampling probabilities calculated by the reinforcement learning policy function, as to be explained in Equation 6.

Figure 1 shows the overview of our Reinforcement Learning based Knowledge Distillation approach (RL-KD for short). In each iteration, an agent interacts with the environment and receives the representation of a training instance. The agent then applies the teacher selection policy and samples the teachers to participate in the knowledge distillation step. The outputs of the sampled teachers are integrated into a soft label to train the student model. After an episode of training examples, the performance of the trained student model is used as the reward to update the policy parameters. 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 $\mathbf{F}(s_j)$, which includes the concatenation of three features.

The first feature is a **vector representation** $\mathcal{R}(\mathbf{x}_i) \in \mathbb{R}^d$ of an input instance $\mathbf{x}_i$. In general, any sentence-encoding network can be applied here to embed a sentence in a 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 $\mathcal{R}$ here is different from a BERT-Base teacher model $\mathcal{M}_k$. $\mathcal{R}$ is a pre-trained BERT-Based model aiming to effectively represent the content of input instances. A teacher model $\mathcal{M}_k$ 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** $\{P^t_k(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^t_k)\}^C_{c=1}$ on all classes $1,\ldots,C$ predicted by the teacher model $\mathcal{M}_k$ with parameters Θ^t_k on the input instance $\mathbf{x}_i$. In practice, the probability is often derived from a softmax function on the hidden representation of the input, that is,

$$P^t_k(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^t_k) = [\text{softmax}(\mathbf{W_k} \cdot \mathcal{M}_k(\mathbf{x_i}))]_c \quad (4)$$

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

The third feature is the **cross entropy loss** $\mathcal{L}_k(\mathbf{x}_i)$ on the input instance $\mathbf{x}_i$ of the teacher model $\mathcal{M}_k$ with parameters Θ^t_k, which is the ground-truth loss of teacher model for $\mathbf{x}_i$.

$$\mathcal{L}_k(\mathbf{x}_i) = -\sum_c \mathbb{I}\left[\mathbf{y}_i = c\right] \cdot \log P^t_k(\mathbf{y}_i = c|\mathbf{x}_i;\Theta^t_k) \quad (5)$$

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 $\mathcal{M}_k$ 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 θ the trainable parameters in the policy function. In this paper, we adopt a simple logistic function as

the policy model.

$$\begin{aligned}\pi_\theta(s_j, a_j) =& P_\theta(a_j|s_j) \\ =& a_j\sigma(\mathbf{AF}(s_j) + \mathbf{b}) \\ &+ (1 - a_j)(1 - \sigma(\mathbf{AF}(s_j) + \mathbf{b}))\end{aligned} \tag{6}$$

where $\mathbf{F}(s_j) \in \mathbb{R}^{d+(C+1)*K}$ is the state vector and $\sigma(\cdot)$ is the sigmoid function with trainable parameter $\theta = \{\mathbf{A} \in \mathbb{R}^{d+(C+1)*K}, \mathbf{b} \in \mathbb{R}^1\}$. The probabilities defined by the above policy function are the weights assigned to the teacher models.

Reward The reward function is correlated with the performance of the student model trained from the distillation of the selected teacher models. We define an episode ξ_b as one batch of training instances, that is, $\mathcal{D}_b = \{\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+m-1}\}$, where b denotes the batch ID and m is the batch size. For each instance $\mathbf{x}_j$ ($i \leq j \leq i+m-1$), we construct the state vector $\mathbf{s}_{j_k}$ for each teacher model $\mathcal{M}_k$ and sample the actions a_{j_k} according to policy $\pi_\theta(s_{j_k}, a_{j_k})$ (Equation 6). For all a_{j_k} sampled, we integrate the average of the loss $\mathcal{L}_{DL}$ (Equation 1) into the loss $\mathcal{L}_{KD}$ (Equation 3) to train the student model.

We propose three reward calculation methods. The first is the minus of ground truth loss $\mathcal{L}_{CE}$ (Equation 2) of the student model. The second reward function combines both the minus of ground-truth loss and the distillation loss to measure the student model. Last, in order to encourage better generalization, we further take the accuracy metric of student model on the development set* $\mathcal{D}'$ as the reward. Other metrics generated on the development set could also be applicable. In summary, we have

$$\begin{aligned}reward_1 =& -\mathcal{L}_{CE} \\ reward_2 =& -\mathcal{L}_{CE} - \mathcal{L}_{DL} \\ reward_3 =& \gamma * (-\mathcal{L}_{CE} - \mathcal{L}_{DL}) \\ &+ (1-\gamma) * Accuracy\, on\, \mathcal{D}'\end{aligned} \tag{7}$$

where γ is a hyper-parameter to balance the reward from the training set and the development set. Please note that the reward is not given immediately after each step is taken. Instead, it is delayed until the training of the whole batch is completed.

Optimization We follow the standard policy gradient method to optimize the parameters, that is,

$$\theta \leftarrow \theta + \beta \sum_j r \sum_k \bigtriangledown_\theta \pi_\theta(s_{j_k}, a_{j_k}) \tag{8}$$

where r is defined by Equation 7 and β is the learning rate.

Model Training

Algorithm 1 shows the overall RL-KD approach. At the beginning, we initialize the student model Θ_0^s by employing all teacher models in the knowledge distillation, that is, a_{j_k} is set to 1 for all $\mathcal{M}_k$ and for all instances $\mathbf{x}_i$. We then initialize parameter θ for the policy function using the same setting under Θ_0^s. After initialization, we iteratively conduct knowledge distillation and teacher selection in turn.

As described in Algorithm 2, in the KD process, we fix the teacher selection policy θ and learn the student model Θ^s. In the TS process, we fix Θ^s to calculate the return and optimize the teacher selection policy θ. The iteration continues for L epochs. For more implementation details, please refer to Section "Implementation Details".

Algorithm 1: Overall Training Procedure

1. Pre-train the student model Θ_0^s using knowledge distillation from the average of all teacher models by maximizing $\mathcal{L}_{KD} = \alpha\mathcal{L}_{DL} + (1-\alpha)\mathcal{L}_{CE}$.
2. Pre-train the TS policy θ_0 by calculating the return under Θ_0^s 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 $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_M\}$. A KD model and a TS model initialized as $\Theta^s = \Theta_0^s$ and $\theta = \theta_0$.

for *epoch* $l = 1$ *to* L **do**
- Shuffle $\mathcal{D}$ to obtain a new training sequence.
- **for** *each batch* $\mathcal{D}_b \in \mathcal{D}$ **do**
  - TS sample actions for each instance $\mathbf{x}_i \in \mathcal{D}_b$ with θ to get selected teachers $\mathcal{K}$ by: $a_j \sim \pi_\theta(s_j, a_j)$.
  - Stored (a_j, s_j) to the episode history $\mathcal{H}$.
  - Compute the average of soft labels of the selected teachers: $\frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} P_k^t(\mathbf{y}_i = c|\mathbf{x}_i; \Theta_k^t)$
  - Update the parameter Θ^s of KD by: $\mathcal{L}_{KD} = \alpha\mathcal{L}_{DL} + (1-\alpha)\mathcal{L}_{CE}$
- **end**
- **for** *each* $(a_j, s_j) \in \mathcal{H}$ **do**
  - Compute delayed reward following Equation 7.
  - Update the parameter θ of TS following Equation 8.
- **end**

end

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.

*Following the common practice in building machine learning models, we assume that data used are divided into training data, validation/development data, and testing data. Development set is used to tune hyper-parameters.

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

Table 2: Statistics of the datasets for experiment.

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), 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 (Rajpurkar et al. 2016), 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) (Wang et al. 2018).

Baseline Models

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 (Sun et al. 2019), 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 (Hinton, Vinyals, and Dean 2015) 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 Yang et al. (2020). 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 Fukuda et al. (2017) 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 Chebotar and Waters (2016); Fukuda et al. (2017); Wu, Chiu, and Wu (2019). 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 $P(\mathbf{y}_i = c|\mathbf{x}_i) = \sum_k w_k P_k^t(\mathbf{y}_i = c|\mathbf{x}_i; \Theta_k^t)$. 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 (Sun et al. 2019)[†]. 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 α to {0.2, 0.5, 0.7}. We choose the best model based on the performance on the development set. The γ 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

[†] https://github.com/intersun/PKD-for-BERT-Model-Compression.

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

Teacher Model	QQP	MRPC	MNLI-mm	RTE	MNLI-m	QNLI	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.

Monte-Carlo based policy gradient method (Williams 1992) 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 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, QNLI), ALBERT shows higher accuracies on tasks MRPC and RTE, while XL-Net 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 3 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 performs 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

Teacher	Student	Strategy	QQP	MRPC	MNLI-(mm/m)	RTE	QNLI	SST-2	AVG.
-	$BERT_3$	FT	88.3	72.1	74.6 / 74.8	60.7	83.3	85.9	77.1
$BERT_{12}$	$BERT_3$	V-KD	88.4	74.0	75.3 / 75.6	56.7	83.7	87.5	77.3
$Robert_{12}$	$BERT_3$	V-KD	85.3	73.0	71.9 / 71.7	55.2	81.9	86.0	75.0
$XLNet_{12}$	$BERT_3$	V-KD	85.4	74.3	72.0 / 71.6	55.2	81.4	87.4	75.3
$ALBERT_{12}$	$BERT_3$	V-KD	88.4	74.0	76.4 / 75.6	56.7	83.7	87.5	77.5
Rand-Single-Ensemble	$BERT_3$	V-KD	87.3	70.6	75.0 / 74.5	51.6	83.7	86.0	75.5
W-Ensemble	$BERT_3$	V-KD	85.3	74.8	72.2 / 72.0	56.7	82.4	87.5	75.8
LR-Dev-Ensemble	$BERT_3$	V-KD	88.6	71.8	75.4 / 75.4	54.9	84.2	86.8	76.7
Best-Single-Ensemble	$BERT_3$	V-KD	88.6	**77.2**	75.1 / 74.7	56.0	84.1	87.0	77.5
Our Method ($reward_1$)	$BERT_3$	RL-KD	**89.1**	76.0	76.9 / **76.8**	61.4	**85.4**	**89.1**	79.2
Our Method ($reward_2$)	$BERT_3$	RL-KD	**89.1**	76.2	**77.4** / 76.3	63.5	84.8	88.5	79.4
Our Method ($reward_3$)	$BERT_3$	RL-KD	89.0	76.7	76.7 / 75.7	**64.6**	85.3	**89.1**	**79.6**
-	$BERT_6$	FT	90.0	82.1	80.7 / 80.3	63.2	87.1	90.1	81.9
$BERT_{12}$	$BERT_6$	V-KD	90.1	82.6	80.5 / 81.2	63.5	88.4	90.8	82.4
$Robert_{12}$	$BERT_6$	V-KD	87.2	80.2	78.0 / 77.0	57.4	84.5	90.5	79.3
$XLNet_{12}$	$BERT_6$	V-KD	87.4	80.6	77.5 / 76.7	62.1	84.1	91.1	79.9
$ALBERT_{12}$	$BERT_6$	V-KD	90.2	79.7	82.3 / 81.1	65.3	88.3	89.9	82.4
Rand-Single-Ensemble	$BERT_6$	V-KD	89.7	77.7	81.0 / 80.7	61.7	87.6	90.6	81.3
W-Ensemble	$BERT_6$	V-KD	87.3	81.1	77.6 / 77.2	62.1	84.8	90.6	80.1
LR-Dev-Ensemble	$BERT_6$	V-KD	90.3	80.6	81.3 / 81.1	64.6	88.3	90.8	82.4
Best-Single-Ensemble	$BERT_6$	V-KD	90.1	80.4	80.7 / 80.5	66.1	87.1	90.3	82.2
Our Method ($reward_1$)	$BERT_6$	RL-KD	**90.7**	82.8	**82.3** / 82.0	67.1	88.7	91.7	83.6
Our Method ($reward_2$)	$BERT_6$	RL-KD	90.6	82.1	**82.3** / 82.1	67.2	88.8	91.4	83.5
Our Method ($reward_3$)	$BERT_6$	RL-KD	90.5	**83.3**	82.2 / 81.6	**68.2**	**88.8**	**92.3**	**83.8**

Table 4: Student model performance on test set. We compare three training strategies for student models: (1) FT: The student model is directly fine-tuned on task label data without any soft target labels from teacher models. (2) V-KD: vanilla KD from single teacher, teacher ensemble, or the best single teacher for each instance. (3) RL-KD: our RL method to select from multiple teacher models at instance level. For RL-KD, we consider all single model teachers ($BERT_{12}$, $Robert_{12}$, $XLNet_{12}$, $ALBERT_{12}$). For our method, we compare three different reward functions: (1) $reward_1$: we use the minus of ground-truth loss (CE) of student model as the reward function for teacher model selection. (2) $reward_2$: Besides the minus of ground-truth loss (CE), we also introduce the minus of knowledge distillation loss (DL) into the reward function. (3) $reward_3$: Besides $reward_2$, we also take the Accuracy metric of student model on development set into account for better generalization.

	RTE	Mean	Stdev
W-Ensemble	56.7, 57.0, 57.0, 57.8, 58.1	57.32	0.597
LR-Ensemble	53.8, 56.0, 57.8, 58.1, 58.5	56.84	1.950
RL-KD^{1,2}	61.7, 63.2, 63.9, 64.6, 64.6	63.60	1.083
	MRPC	**Mean**	**Stdev**
W-Ensemble	71.6, 72.3, 72.5, 74.7, 74.8	73.18	1.472
LR-Ensemble	72.1, 72.1, 73.8, 74.3, 74.8	73.42	1.256
RL-KD^{1,2}	74.7, 75.0, 75.5, 75.7, 76.7	75.52	0.688

Table 5: Mean and Standard deviation values and statistically significance T-test (p-Value) across five different runs on RTE and MRPC. 1 and 2 denote statistically significant improvements over W-Ensemble and LR-Ensemble.

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 of data can already yield a reasonable model, which in turn provides reliable reward as the feedback for policy training. That explains why student model $BERT_6$ achieves better performance than the other KD methods even on relatively small data sets.

Variance Analysis

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* .

Buciluǎ, 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* .

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355* .

Sutton, R. S. 1992. Reinforcement learning architectures. *Proceedings ISKIT* 92.

Tang, R.; Lu, Y.; Liu, L.; Mou, L.; Vechtomova, O.; and Lin, J. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *arXiv preprint arXiv:1903.12136* .

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* .

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 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*.

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

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Wu, M.-C.; Chiu, C.-T.; and Wu, K.-H. 2019. Multi-teacher Knowledge Distillation for Compressed Video Action Recognition on Deep Neural Networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2202–2206. IEEE.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; and Le, Q. V. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* .

Yang, Z.; Shou, L.; Gong, M.; Lin, W.; and Jiang, D. 2020. Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System. *Proceedings of the 13th International Conference on Web Search and Data Mining* .