MKD: a Multi-Task Knowledge Distillation Approach for Pretrained Language Models

Linqing Liu, [1] Huan Wang,[2] Jimmy Lin,[1]
Richard Socher,[2]

All work was done while the first author was a research intern at Salesforce Research.

Caiming Xiong[2]
[1] David R. Cheriton School of Computer Science, University of Waterloo
[2] Salesforce Research

{linqing.liu, jimmylin}@uwaterloo.ca, {huan.wang, rsocher, cxiong}@salesforce.com

Abstract

Pretrained language models have led to significant performance gains in many NLP tasks. However, the intensive computing resources to train such models remain an issue. Knowledge distillation alleviates this problem by learning a light-weight student model. So far the distillation approaches are all task-specific. In this paper, we explore knowledge distillation under the multi-task learning setting. The student is jointly distilled across different tasks. It acquires more general representation capacity through multi-tasking distillation and can be further fine-tuned to improve the model in the target domain. Unlike other BERT distillation methods which specifically designed for Transformer-based architectures, we provide a general learning framework. Our approach is model agnostic and can be easily applied on different future teacher model architectures. We evaluate our approach on a Transformer-based and LSTM based student model. Compared to a strong, similarly LSTM-based approach, we achieve better quality under the same computational constraints. Compared to the present state of the art, we reach comparable results with much faster inference speed.

1 Introduction

Pretrained language models learn highly effective language representations from large-scale unlabeled data. A few prominent examples include ELMo Peters et al. (2018), BERT Devlin et al. (2019), RoBERTa Liu et al. (2019c), and XLNet Yang et al. (2019), all of which have achieved state of the art in many natural language processing (NLP) tasks, such as natural language inference, sentiment classification, and semantic tex-

tual similarity. However, such models use dozens, if not hundreds, of millions of parameters, invariably leading to resource-intensive inference. The consensus is that we need to cut down the model size and reduce the computational cost while maintaining comparable quality.

One approach to address this problem is knowledge distillation (KD; Ba and Caruana, 2014; Hinton et al., 2015), where a large model functions as a *teacher* and transfers its knowledge to a small *student* model. Previous methods focus on task-specific KD, which transfers knowledge from a single-task teacher to its single-task student. Put it another way, the knowledge distillation process needs to be conducted all over again when performing on a new NLP task. The inference speed of the large-scale teacher model remains the bottleneck for various downstream tasks distillation.

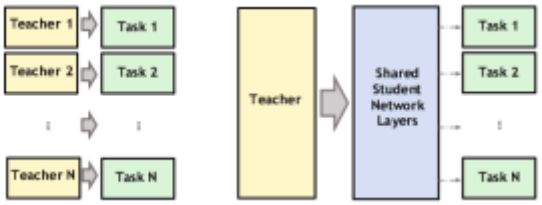

Figure 1: The left figure represents task-specific KD. The distillation process needs to be performed for each different task. The right figure represents our proposed multi-task KD. The student model consists of shared layers and task-specific layers.

Our goal is to find a *distill-once-fits-many* solution. In this paper, we explore the knowledge distillation method under the setting of multi-task learning (MTL; Caruana, 1997; Baxter, 2000). We propose to distill the student model from different tasks jointly. The overall framework is illustrated in Figure 1. The reason is twofold: first, the distilled model learns a more universal language representation by leveraging cross-task data. Second, the student model achieves both comparable quality and fast inference speed across multiple tasks. MTL is based on the idea Maurer et al. (2016) that tasks are related by means of a common low dimensional representation. We also provide an intuitive explanation on why using shared structure could possibly help by assuming some connections over the conditional distribution of different tasks.

We evaluate our approach on two different student model architectures. One uses three layers Transformers Vaswani et al. (2017), since most of the KD works Sun et al. (2019); Jiao et al. (2019) use Transformers as their students. Another is LSTM based network with bi-attention mechanism. Previously Tang et al. (2019) examine the representation capacity of a simple, single-layer Bi-LSTM only, so we are interested in whether

adding more previous effective modules, such as an attention mechanism, will further improve its effectiveness. It exemplifies that our approach is model agnostic, i.e., the choice of student model does not depend on the teacher model architecture; The teacher model can be easily switched to other powerful language models other than BERT.

We further study several important problems in knowledge distillation, such as the choice of modules in student model, the influence of different tokenization methods, and the influence of MTL in KD. We evaluate our approach on seven datasets across four different tasks. For LSTM based student, our approach keeps the advantage of inference speed while maintaining comparable performances as those specifically designed for Transformer methods. For our Transformer based student, it does provide a modest gain, and outperforms other KD methods without using external training data.

2 Related Work

Language model pretraining. Given a sequence of tokens, pretrained language models encode each token as a general language representational embedding. A large body of literature has explored this area. Traditional pretrained word representations Turian et al. (2010) presume singular word meanings and thus adapt poorly to multiple contexts—for some notable examples, see word2vec Mikolov et al. (2013), GloVe Pennington et al. (2014), and FastText Bojanowski et al. (2017). For more flexible word representations, a few advancements exist: Neelakantan et al. (2015) learn multiple embeddings per word type; context2vec Melamud et al. (2016) uses bidirectional LSTM to encode contexts around target words; CoVe McCann et al. (2017) trains LSTM encoders on some machine translation datasets, showing that these encoders are well-transferable to other tasks. Prominently, ELMo Peters et al. (2018) learns deep word representations using a bidirectional language model. It can be easily added to an existing model and boost performance across six challenging NLP tasks.

Fine-tuning approaches are mostly employed in more recent work. They pretrain the language model on a large-scale unlabeled corpus and then fine-tune it with in-domain labeled data for a supervised downstream task Dai and Le (2015); Howard and Ruder (2018). BERT Devlin et al. (2019), GPT Radford et al. (2018) and GPT-2 Radford et al. are some of the prominent examples. Following BERT, XLNet Yang et al. (2019) proposes a generalized autoregressive pretraining method and RoBERTa Liu et al. (2019c) optimizes BERT pretraining approach. These pretrained models are large in size and contain millions of parameters. We target the BERT model and aim to address this problem through knowledge distillation. Our approach can be easily applied to other models as well.

Knowledge distillation. Knowledge distillation Ba and Caruana (2014); Hinton et al. (2015) transfers knowledge from a large *teacher* model to a smaller *student* model. Since the distillation only matches the output distribution, the student model architecture can be completely different from that of the teacher model. There are already many efforts trying to distill BERT into smaller models. BERT-PKD Sun et al. (2019) extracts knowledge not only from the last layer of the teacher, but also from previous layers. TinyBERT Jiao et al. (2019) introduces a two-stage learning framework which performs transformer distillation at both pretraining and task-specific stages. Zhao et al. (2019) train a student model with smaller vocabulary and lower hidden states dimensions. DistilBERT Sanh et al. (2019) reduces the layers of BERT and uses this small version of BERT as its student model. All the aforementioned distillation methods are performed on a single task, specifically designed for the transformer-based teacher architecture, resulting in poor generalizability to other type of models. Our objective is to invent a general distillation framework, applicable to either transformer-based models or other architectures as well. Tang et al. (2019) distill BERT into a single-layer BiLSTM. In our paper, we hope to extract more knowledge from BERT through multi-task learning, while keeping the student model simple.

Multi-task learning. Multi-task learning (MTL) has been successfully applied on different applications Collobert and Weston (2008); Deng et al. (2013); Girshick (2015). MTL helps the pretrained language models learn more generalized text representation by sharing the domain-specific information contained in each related task training signal Caruana (1997). Liu et al. (2019b, 2015) propose a multi-task deep neural network (MT-DNN) for learning representations across multiple tasks. Clark et al. (2019) propose to use knowledge distillation so that single task models can teach a multi-task model. Liu et al. (2019a) train an ensemble of large DNNs and then distill their knowledge to a single DNN via multi-task learning to ensemble its teacher performance.

3 Model Architecture

In this section, we introduce the teacher model and student model for our distillation approach. We explored two different student architectures: a traditional bidirectional long short-term memory network (BiLSTM) with bi-attention mechanism in 3.2, and the popular Transformer in 3.3.

3.1 Multi-Task Refined Teacher Model

We argue that multi-task learning can leverage the regularization of different natural language understanding tasks. Under this setting, language models can be more effective in learning universal language representations. To this end, we consider the bidirectional transformer language model (BERT; Devlin et al., 2019) as bottom shared text encoding layers, and fine-tune the task-specific top layers for each type of NLU task.

There are mainly two stages for the training procedure: pretraining the shared layer and multi-task refining.

Shared layer pretraining. Following Devlin et al. (2019), the input token is first encoded as the the summation of its corresponding token embeddings, segmentation embeddings and position embeddings. The input embeddings are then mapped into contextual embeddings C through a multi-layer bidirectional transformer encoder. The pretraining of these shared layers use the cloze task and next sentence prediction task. We use the pretrained $BERT_{LARGE}$ to initialize these shared layers.

Multi-task refining. The contextual embeddings C are then passed through the upper task-specific layers. Following Liu et al. (2019b), our current NLU training tasks on GLUE Wang et al. (2018) can be classified into four categories: single-sentence classification (CoLA and SST-2), pairwise text classification (RTE, MNLI, WNLI, QQP, and MRPC), pairwise text similarity (STS-B), and relevance ranking (QNLI). Each category corresponds to its own output layer.

Here we take the text similarity task as an example to demonstrate the implementation details. Following Devlin et al. (2019), we consider the contextual embedding of the special `[CLS]` token as the semantic representation of the input sentence pair (X_1, X_2). The similarity score can be predicted by the similarity ranking layer:

$$\text{Sim}(X_1, X_2) = W_{STS}^{\top} x \tag{1}$$

where W_{STS} is a task-specific learnable weight vector and x is the contextual embedding of the `[CLS]` token.

In the multi-task refining stage, all the model parameters, including bottom shared layers and task-specific layers, are updated through mini-batch stochastic gradient descent Li et al. (2014). The training data are packed into mini-batches and each mini-batch only contains samples from one task. Running all the mini-batches in each epoch approximately optimizes the sum all of all multi-task objectives. In each epoch, the model is updated according to the selected mini-batch and its task-specific objective. We still take the text similarity task as an example, where each pair of sentences is labeled with a real-value similarity score y. We use the mean-squared error loss as our objective function:

$$\|y - \text{Sim}(X_1, X_2)\|_2^2 \tag{2}$$

For text classification task, we use the cross-entropy loss as the objective function. For relevance ranking task, we minimize the negative log likelihood of the positive examples Liu et al. (2019b). We can also easily add other tasks by adding its own task-specific layer.

3.2 LSTM-based Student Model

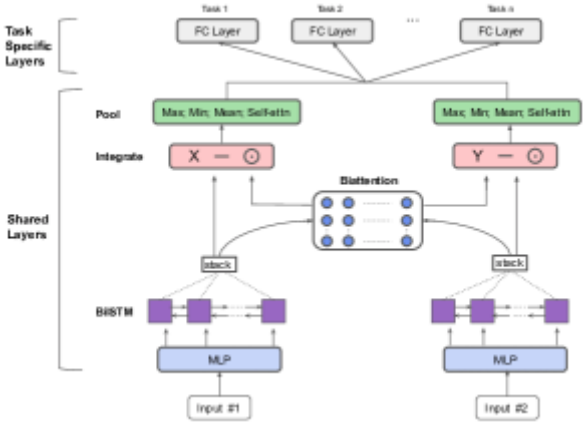

Figure 2: Architecture for the bi-attentive student neural network.

We're interested in exploring whether a simple architecture, such as LSTM, has enough representation capacity to transfer knowledge from the teacher model. We also incorporate bi-attention module since it's widely used between pairs of sentences Peters et al. (2018); Wang et al. (2018). And the inputs in our experiments are mostly two sentences. Our LSTM-based bi-attentive student model is depicted in Figure 2. For equation representations, the embedding vectors of input sequences are denoted as w^x and w^y. For single-sentence input tasks, w^y is the same as w^x. $\oplus$ represents vectors concatenation.

w^x and w^y are first converted into $\widehat{w}^x$ and $\widehat{w}^y$ through a feedforward network with ReLU activation Nair and Hinton (2010) function. For each token in $\widehat{w}^x$ and $\widehat{w}^y$, we then use a bi-directional LSTM encoder to compute its hidden states and stack them over time axis to form matrices X and Y separately.

Next, we apply the biattention mechanism Xiong et al. (2016); Seo et al. (2016) to compute the attention contexts $A = XY^\top$ of the input sequences. The attention weight A_x and A_y is extracted through a column-wise normalization for each sequence. The context vectors C_x and C_y for each token is computed as the multiplication of its corresponding representation and attention weight:

$$A_x = \text{softmax}(A) \quad C_x = A_x^\top X \tag{3}$$

Same as McCann et al. (2017), we concatenate three different computations between original representations and context vector to reinforce their relationships. The con-

catenated vectors are then passed through one single-layer BiLSTM:

$$\begin{aligned} X_y &= \text{BiLSTM}([X \oplus X - C_y \oplus X \odot C_y]) \\ Y_x &= \text{BiLSTM}([Y \oplus Y - C_x \oplus Y \odot C_x]) \end{aligned} \tag{4}$$

The pooling operations are then applied on the outputs of BiLSTM. We use max, mean, and self-attentive pooling to extract features. These three pooled representations are then concatenated to get one context representation. We feed this context representation through a fully-connected layer to get final output.

3.3 Transformer-based Student Model

Most of the pre-trained language models, which can be employed as teachers, are built with Transformers. Transformer Vaswani et al. (2017) now is an ubiquitous model architecture. It draws global dependencies between input and output entirely relying on an self-attention mechanism. Our student model uses three layers of Transformers. Same as BERT Devlin et al. (2019), [CLS] is added in front of every input example, and [SEP] is added between two input sentences. We use the average [CLS] representation from each layer as the final output.

4 Multi-Task Distillation

The parameters of student models, introduced in Section 3.2 and 3.3, are shared across all tasks. Each task has its individual layer on top of it. We begin by describing the task-specific layers: for each task, the hidden representations are first fed to a fully connected layer with rectified linear units (ReLU), whose outputs are passed to another linear transformation to get logits $z = Wh$. During multi-task training, the parameters from both the bottom student network and upper task-specific layers are jointly updated.

Considering one text classification problem, denoted as task t, a softmax layer will perform the following operations on the i^{th} dimension of z to get the predicted probability for the i^{th} class:

$$\text{softmax}(z_i^t) = \frac{\exp\{z_i^t\}}{\sum_j \exp\{z_j^t\}} \tag{5}$$

According to Ba and Caruana (2014), training the student network on logits will make learning easier. There might be information loss from transferring logits into probability space, so it follows that the teacher model's logits provides more information about the internal model behaviour than its predicted one-hot labels. Then, our distillation objective is to minimize the mean-squared error (MSE) between the student network

logits z_S^t and the teacher's logits z_T^t:

$$L_{distill}^t = \left\| z_T^t - z_S^t \right\|_2^2 \tag{6}$$

The training samples are selected from each dataset and packed into task-specific batches. For task t, we denote the current selected batch as b_t. For each epoch, the model running through all the batches equals to attending over all the tasks:

$$L_{distill} = L_{distill}^1 + L_{distill}^2 + ... + L_{distill}^t \tag{7}$$

During training, the teacher model first uses the pretrained BERT model Devlin et al. (2019) to initialize its parameters of shared layers. It then follows the multi-task refining procedure described in Section 3.1 to update both the bottom shared-layers and upper task-specific layers.

For student model, the shared parameters are randomly initialized. During training, for each batch, the teacher model first predicts teacher logits. The student model then updates both the bottom shared layer and the upper task-specific layers according to the teacher logits. The complete procedure is summarized in Algorithm 1.

Algorithm 1 Multi-task Distillation

Initialize the shared layers with BERT$_{Large}$ then multi-task refine the teacher model
Randomly initialize the student model parameters
Set the max number of epoch: $epoch_{max}$
// Pack the data for T Tasks into batches
for $t \leftarrow 1$ to T **do**
  1. Generate augmented data: t_{aug}
  2. Pack the dataset t and t_{aug} into batch D_t
end for
// Train the student model
for $epoch \leftarrow 1$ to $epoch_{max}$ **do**
  1. Merge all datasets:
    $D = D_1 \cup D_2 \ldots \cup D_T$
  2. Shuffle D
  for b_t in D **do**
    3. Predict logits z^T from teacher model
    4. Predict logits z^S from student model
    5. Compute loss $L_{distill}(\theta)$
    6. Update student model:
      $\theta = \theta - \alpha \nabla_\theta L_{distill}$
  end for
end for

5 An Intuitive Explanation

In this section we give an intuitive explanation on why using some shared structure during the multi-task training could possibly help. Suppose the samples of the task T are independent and identically distributed $x^T, y^T \sim \mathcal{P}^T_{\mathcal{X}\mathcal{Y}}$, where x^T, y^T are the feature and labels of the samples in task T respectively. The joint density can be decomposed as $p^T(x, y) = p^T(x)p^T(y \mid x)$. During the discriminative learning process, one tries to estimate the conditional distribution $p^T(\cdot \mid x)$. For different tasks, $p^T(\cdot \mid X)$ could be very different. Indeed if there is no connections in $p^T(\cdot \mid X)$ for different tasks, then it is hard to believe training on one task may help another. However if we assume some smoothness over $p^T(\cdot \mid X)$, then some connections can be built across tasks.

Without loss of generality, we investigate the case of two tasks. For task T_1 and T_2, let's assume there exist some common domain of representations $\mathcal{H}$, and two functions: $h^{T_1}(x), h^{T_2}(x) : \mathcal{X} \mapsto \mathcal{H}$, such that

$$p^{T_1}(\cdot \mid x) = g^{T_1} \circ h^{T_1}(x), \tag{8}$$

$$p^{T_2}(\cdot \mid x) = g^{T_2} \circ h^{T_2}(x), \tag{9}$$

$$\forall\, x_1, x_2,\ \|h^{T_1}(x_1) - h^{T_2}(x_2)\| \leq \eta \|x_1 - x_2\|, \tag{10}$$

where $g^T : \mathcal{H} \mapsto \mathcal{Y}^T$ is a function that maps from the common domain $\mathcal{H}$ to the task labels $\mathcal{Y}^T$ for task T, $\circ$ denotes function composition, and η is a smoothness constant.

The Lipschitz-ish inequality (10) suggests the hidden representation h^{T_1} on task T_1 may help the estimation of h^{T_2}, since $h^{T_2}(x_2)$ will be close to $h^{T_1}(x_1)$ if x_1 and x_2 are close enough. This is implicitly captured if we use one common network to model both h^{T_1} and h^{T_2} since the neural network with ReLU activation is Lipschitz.

6 Experimental Setup

6.1 Datasets

We conduct the experiments on seven most widely used datasets in the General Language Understanding Evaluation (GLUE) benchmark Wang et al. (2018): one sentiment dataset SST-2 Socher et al. (2013), two paraphrase identification datasets QQP [1]

[1]https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

and MRPC Dolan and Brockett (2005), one text similarity dataset STS-B Cer et al. (2017), and three natural language inference datasets MNLI Williams et al. (2018), QNLI Rajpurkar et al. (2016) and RTE. For the QNLI dataset, version 1 expired on January 30,

2019; the result is evaluated on QNLI version 2.

6.2 Implementation Details

We use the released MT-DNN model[2]

[2]https://github.com/namisan/mt-dnn

to initialize our teacher model. We further refine the model against the multi-task learning objective for 1 epoch with learning rate set to 5e-4. The performance of our refined MT-DNN is lower than reported results in Liu et al. (2019b).

The LSTM based student model (*MKD-LSTM*) is initialized randomly. For multi-task distillation, We use the Adam optimizer Kingma and Ba (2014) with learning rates of 5e-4. The batch size is set to 128, and the maximum epoch is 16. We clip the gradient norm within 1 to avoid gradient exploding. The number of BiLSTM hidden units in student model are all set to 256. The output feature size of task-specific linear layers is 512. The Transformer-based student model (*MKD-Transformer*) consists of three layers of Transformers. Following the settings of BERT-PKD, it is initialized with the first three layers parameters from pre-trained BERT-base.

We also fine-tune the multi-task distilled student model for each task. During fine-tuning, the parameters of both shared layers and upper task-specific layers are updated. The learning rate is chosen from $\{1, 1.5, 5\} \times 10^{-5}$ according to the validation set loss on each task. Other parameters remain the same as above. For both teacher and student models, we use WordPiece embeddings Wu et al. (2016) with a 30522 token vocabulary.

Data augmentation. The training data for typical natural language understanding tasks is usually very limited. Larger amounts of data are desirable for the teacher model to fully express its knowledge. Tang et al. (2019) proposes two methods for text data augmentation: masking and POS-guided word replacement. We employ the only first masking technique which randomly replaces a word in the sentence with `[MASK]`, because, as shown in both Tang et al. (2019) and our own experiments, POS-guided word replacement does *not* lead to consistent improvements in quality across most of the tasks. Following their strategies, for each word in a sentence, we perform masking with probability $p_{mask} = 0.1$. We use the combination of original corpus and augmentation data in distillation procedure. For smaller datasets STS-B, MRPC and RTE, the size of the augmented dataset is 40 times the sizes of the original corpus; 10 times for other larger datasets.

6.3 Methods and Baselines

Model	**Size # Param**	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
		Acc	F_1/Acc	r/ρ	F_1/Acc	Acc	Acc	Acc
MTL-BERT (Teacher)	**303.9M**	94.7	84.7/79.7	84.0/83.3	72.3/89.6	85.9/85.7	90.5	77.7
OpenAI GPT	**116.5M**	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0
ELMo	**93.6M**	90.4	84.4/78.0	74.2/72.3	63.1/84.3	74.1/74.5	79.8	58.9
Distilled BiLSTM	**1.59M**	91.6	82.7/75.6	79.6/78.2	68.5/88.4	72.5/72.4	-	-
BERT-PKD	**21.3M**	87.5	80.7/72.5	-	68.1/87.8	76.7/76.3	84.7	58.2
TinyBERT	**5.0M**	92.6	86.4/81.2	81.2/79.9	71.3/89.2	82.5/81.8	87.7	62.9
$BERT_{EXTREME}$	**19.2M**	88.4	84.9/78.5	-	-	78.2/77.7	-	
MKD-LSTM	**10.2M**	91.0	85.4/79.7	80.9/80.9	70.7/88.6	78.6/78.4	85.4	**67.3**
MKD-Transformer	**21.3M**	90.1	86.2/79.8	**81.5/81.5**	71.1/**89.4**	79.2/78.5	83.5	67.0

Table 1: Results from the GLUE test server. The first group contains large-scale pretrained language models. The second group lists previous knowledge distillation methods for BERT. Our MKD results based on LSTM and Transformer student model architectures are listed in the last group. The number of parameters doesn't include embedding layer.

Results on test data reported by the official GLUE evaluation server are summarized in Table 1. Each entry in the table is briefly introduced below:

MTL-BERT. We use the multi-task refined BERT (described in Section 3.1) as our teacher model. We tried to replicate the results of the released MT-DNN Liu et al. (2019b) model.

OpenAI GPT. A generative pre-trained Transformer-based language model Radford et al. (2018). In contrast to BERT, GPT is auto-regressive, only trained to encode uni-directional context.

ELMo. Peters et al. (2018) learns word representations from the concatenation of independently trained left-to-right and right-to-left LSTMs. We report the results of a BiLSTM-based model with bi-attention baseline Wang et al. (2018) trained on top of ELMo.

Distilled BiLSTM. Tang et al. (2019) distill BERT into a simple BiLSTM. They use different models for single and pair sentences tasks.

BERT-PKD. The Patient-KD-Skip approach Sun et al. (2019) which student model patiently learns from multiple intermediate layers of the teacher model. We use their student model consisting of three layers of Transformers.

TinyBERT Jiao et al. (2019) propose a knowledge distillation method specially designed for transformer-based models. It requires a general distillation step which is performed on a large-scale English Wikipedia (2,500 M words) corpus.

BERT$_{\text{EXTREME}}$. Zhao et al. (2019) aims to train a student model with smaller vocabulary and lower hidden state dimensions. Similar to BERT-PKD, they use the same training corpus to train BERT to perform KD.

#	Model	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
1	**Biatt LSTM**	85.8	80.4/69.9	12.24/11.33	81.1/86.5	73.0/73.7	80.3	53.1
2	**Single Task Distilled Biatt LSTM**	89.2	82.5/72.1	20.2/20.0	84.6/88.4	74.7/75.0	82.0	52.0
3	**BiLSTM$_{\text{MTL}}$**	87.5	83.2/72.8	71.6/72.6	81.6/87.0	70.2/71.3	75.4	56.3
4	**MKD-LSTM *Word-level Tokenizer***	87.3	84.2/75.7	72.2/72.6	71.1/79.3	69.4/70.9	75.1	54.9
5	**MKD-LSTM**	89.3	86.8/81.1	84.5/84.5	85.2/89.0	78.4/79.2	83.0	67.9

Table 2: Ablation studies on GLUE dev set of different training procedures. All models are not fine-tuned. Line 1 is our bi-attentive LSTM student model trained without distillation. Line 2 is our bi-attentive LSTM student distilled from single task. Line 3 is the Multi-task distilled BiLSTM. Line 4 is the Multi-task distilled model using word-level tokenizer.

Model	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
Sentiment Task	✓						
MKD-LSTM	89.9	81.4/70.8	51.2/49.9	84.9/88.3	74.3/74.7	83.2	50.9
PI Tasks		✓		✓			
MKD-LSTM	89.3	85.2/77.2	83.4/83.3	84.9/88.7	73.2/73.9	83.8	59.6
NLI Tasks					✓	✓	✓
MKD-LSTM	90.4	87.9/82.1	84.1/84.1	84.8/88.4	77.1/78.1	84.5	66.8
All Tasks	✓	✓	✓	✓	✓	✓	✓
MKD-LSTM	90.5	86.9/80.2	85.0/84.8	84.8/89.0	77.4/78.3	84.9	68.2

Table 3: Ablation experiments on the dev set use different training tasks in multi-task distillation. The results are reported with the original corpus, without augmentation data. The model is fine-tuned on each individual task.

7 Result and Discussions

The results of our model are listed as MKD-LSTM and MKD-Transformer in the tables.

7.1 Model Quality Analysis

Comparison with GPT / ELMo. Our model has better or comparable performance compared with ELMo and OpenAI GPT. MKD-LSTM has higher performance than ELMo over all seven datasets: notably 8.4 points for RTE, 8.6 points in Spearman's ρ for STS-B, 7.6 points in F-1 measure for QQP, and 0.6 to 5.6 points higher for other datasets. Compared with OpenAI GPT, MKD-LSTM is 11.3 points higher for RTE and 4 points higher for MRPC.

Comparison with Distilled BiLSTM / BERT-PKD. While using the same Transformer layers and same amount of parameters, MKD-Tranformer significantly outperforms BERT-PKD by a range of $0.4 \sim 9.1$ points. MKD-LSTM leads to significant performance gains than BERT-PKD while using far less parameters, and compensate for the effectiveness loss of Distlled BiLSTM.

Comparison with TinyBERT / BERT$_{\text{EXTREME}}$. These two approaches both use the large-scale unsupervised text corpus, same as the ones to train the teacher model, to execute their distillation process. However, we only use the data within downstream tasks. There are two caveats for their methods: (1) Due to massive training data, KD still requires intensive computing resources, e.g. BERT$_{\text{EXTREME}}$ takes 4 days on 32 TPU cores to train their student model. (2) The text corpus to train the teacher model is not always

available due to data privacy. Under some conditions we can only access to the pre-trained models and their approach are not applicable.

While not resorting to external training data, our model has the best performance across the state-of-the-art KD baselines (i.e., BERT-PKD). It also achieves comparable performance compared to intensively trained KD methods (i.e, $\text{BERT}_{\text{EXTREME}}$) on external large corpus.

7.2 Ablation Study

We conduct ablation studies to investigate the contributions of: (1) the different training procedures (in Table 2); (2) Different training tasks in multi-task distillation (in Table 3). We also compare the inference speed of our models and previous distillation approach. (in Table 4). The ablation studies are all conducted on LSTM-based student model since it has the advantage of model size and inference speed compared to Transformers.

Do we need attention in the student model? Yes. Tang et al. (2019) distill BERT into a simple BiLSTM network. Results in Table 1 demonstrates that our model is better than Distilled BiLSTM and achieves an improvement range of $2.2 \sim 6.1$ points across six datasets. To make fair comparison, we also list the results of multi-task distilled BiLSTM in Line 3 in Table 2. It's obvious that Line 5, which is the model with bi-attentive mechanism, significantly outperform Line 3. We surmise that the attention module is an integral part of the student model for sequence modeling.

Better vocabulary choices? WordPiece works better than the word-level tokenizers in our experiments. The WordPiece-tokenized vocabulary size is 30522, while the word-level tokenized vocabulary size is much larger, along with more unknown tokens. WordPiece effectively reduces the vocabulary size and improves rare-word handling. The comparison between Line 4 and Line 5 in Table 2 demonstrates that the method of tokenization influences all the tasks.

The influence of MTL in KD? The single-task distilled results are represented in Line 2 of Table 2. Compared with Line 5, all the tasks benefit from information sharing through multi-task distillation. Especially for STS-B, the only regression task, greatly benefit from the joint learning from other classification tasks.

We also illustrate the influence of different number of tasks for training. In Table 3, the training set incorporates tasks of the same type individually. Even for the tasks which are in the training sets, they still perform better in the all tasks training setting. For example, for RTE, the *All Tasks* setting increases 1.4 points than *NLI Tasks* setting. For other training settings which RTE is excluded from training set, *All Tasks* leads to better

performance.

	Distilled BiLSTM	BERT-PKD	TinyBERT	MKD-LSTM
Inf. Time	1.36	8.41	3.68	2.93

Table 4: The inference time (in seconds) for baselines and our model. The inference is performed on QNLI training set and on a single NVIDIA V100 GPU.

7.3 Inference Efficiency

To test the model efficiency, we ran the experiments on QNLI training set. We perform the inference on a single NVIDIA V100 GPU with batch size of 128, maximum sequence length of 128. The reported inference time is the total running time of 100 batches.

From Table 4, the inference time for our model is 2.93s. We re-implemented Distilled BiLSTM from Tang et al. (2019) and their inference time is 1.36s. For fair comparison, we also ran inference procedure using the released BERT-PKD and TinyBERT model on the same machine. Our model significantly outperforms Distilled BiLSTM with same magnitude speed. It also achieves comparable results but is faster in efficiency compared with other distillation models.

8 Conclusion

In this paper, we propose a general framework for multi-task knowledge distillation. The student is jointly distilled across different tasks from a multi-task refined BERT model (teacher model). We evaluate our approach on Transformer-based and LSTM-based student model. Compared with previous KD methods using only data within tasks, our approach achieves better performance. In contrast to other KD methods using large-scale external text corpus, our approach balances the problem of computational resources, inference speed, performance gains and availability of training data.

References

Ba and Caruana (2014)

Jimmy Ba and Rich Caruana. 2014.
Do deep nets really need to be deep?

In *Advances in neural information processing systems*, pages 2654–2662.

Baxter (2000)

Jonathan Baxter. 2000.
A model of inductive bias learning.
Journal of artificial intelligence research, 12:149–198.

Bojanowski et al. (2017)

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017.
Enriching word vectors with subword information.
Transactions of the Association for Computational Linguistics, 5(1):135–146.

Caruana (1997)

Rich Caruana. 1997.
Multitask learning.
Machine learning, 28(1):41–75.

Cer et al. (2017)

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017.
Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation.
arXiv preprint arXiv:1708.00055.

Clark et al. (2019)

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.
arXiv preprint arXiv:1907.04829.

Collobert and Weston (2008)

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*, pages 160–167. ACM.

Dai and Le (2015)

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

In *Advances in neural information processing systems*, pages 3079–3087.

Deng et al. (2013) — Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603. IEEE.

Devlin et al. (2019) — Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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

Girshick (2015) — Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

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

Howard and Ruder (2018) — 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 (Volume 1: Long Papers)*, pages 328–339.

Jiao et al. (2019) — Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Kingma and Ba (2014) — Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li et al. (2014) — Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. 2014. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM.

Liu et al. (2015) — Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Liu et al. (2019a) — Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding.

arXiv preprint arXiv:1904.09482.

Liu et al. (2019b)

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b.
Multi-task deep neural networks for natural language understanding.
arXiv preprint arXiv:1901.11504.

Liu et al. (2019c)

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

Maurer et al. (2016)

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2016.
The benefit of multitask representation learning.
The Journal of Machine Learning Research, 17(1):2853–2884.

McCann et al. (2017)

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017.
Learned in translation: Contextualized word vectors.
In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Melamud et al. (2016)

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016.
context2vec: Learning generic context embedding with bidirectional lstm.
In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61.

Mikolov et al. (2013)

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781.

Nair and Hinton (2010)
Vinod Nair and Geoffrey E Hinton. 2010.
Rectified linear units improve restricted boltzmann machines.
In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Neelakantan et al. (2015)
Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015.
Efficient non-parametric estimation of multiple embeddings per word in vector space.
arXiv preprint arXiv:1504.06654.

Pennington et al. (2014)
Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014.
Glove: Global vectors for word representation.
In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters et al. (2018)
Matthew 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, Volume 1 (Long Papers)*, pages 2227–2237.

Radford et al. (2018)
Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018.
Improving language understanding by generative pre-training.

URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/ languageunsupervised/language understanding paper. pdf.

(31) Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever.
Language models are unsupervised multitask learners.

Rajpurkar et al. (2016) Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016.
Squad: 100,000+ questions for machine comprehension of text.
In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sanh et al. (2019) Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019.
Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.
arXiv preprint arXiv:1910.01108.

Seo et al. (2016) Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016.
Bidirectional attention flow for machine comprehension.
arXiv preprint arXiv:1611.01603.

Socher et al. (2013) Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013.
Recursive deep models for semantic compositionality over a sentiment treebank.
In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sun et al. (2019)
Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019.
Patient knowledge distillation for bert model compression.
arXiv preprint arXiv:1908.09355.

Tang et al. (2019)
Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019.
Distilling task-specific knowledge from bert into simple neural networks.
arXiv preprint arXiv:1903.12136.

Turian et al. (2010)
Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010.
Word representations: a simple and general method for semi-supervised learning.
In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Vaswani et al. (2017)
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017.
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008.

Wang et al. (2018)
Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018.
Glue: A multi-task benchmark and analysis platform for natural language understanding.
In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Williams et al. (2018)
Adina Williams, Nikita Nangia, and Samuel Bowman. 2018.

A broad-coverage challenge corpus for sentence understanding through inference.
In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Wu et al. (2016)
Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016.
Google's neural machine translation system: Bridging the gap between human and machine translation.
arXiv preprint arXiv:1609.08144.

Xiong et al. (2016)
Caiming Xiong, Victor Zhong, and Richard Socher. 2016.
Dynamic coattention networks for question answering.
arXiv preprint arXiv:1611.01604.

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

Zhao et al. (2019)
Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. 2019.
Extreme language model compression with optimal subwords and shared projections.
arXiv preprint arXiv:1909.11687.