Contrastive Distillation on Intermediate Representations for Language Model Compression

Siqi Sun, Zhe Gan, Yu Cheng, Yuwei Fang, Shuohang Wang, Jingjing Liu
Microsoft Dynamics 365 AI Research
{siqi.sun,zhe.gan,yu.cheng,yuwfan,shuohang.wang,jingjl}@microsoft.com

Abstract

Existing language model compression methods mostly use a simple L_2 loss to distill knowledge in the intermediate representations of a large BERT model to a smaller one. Although widely used, this objective by design assumes that all the dimensions of hidden representations are independent, failing to capture important *structural* knowledge in the intermediate layers of the teacher network. To achieve better distillation efficacy, we propose Contrastive Distillation on Intermediate Representations (CoDIR), a principled knowledge distillation framework where the student is trained to distill knowledge through intermediate layers of the teacher via a *contrastive objective*. By learning to distinguish positive sample from a large set of negative samples, CoDIR facilitates the student's exploitation of rich information in teacher's hidden layers. CoDIR can be readily applied to compress large-scale language models in both pre-training and finetuning stages, and achieves superb performance on the GLUE benchmark, outperforming state-of-the-art compression methods.[1]

1 Introduction

Large-scale pre-trained language models (LMs), such as BERT (Devlin et al., 2018), XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019), have brought revolutionary advancement to the NLP field (Wang et al., 2018). However, as new-generation LMs grow more and more into behemoth size, it becomes increasingly challenging to deploy them in resource-deprived environment. Naturally, there has been a surge of research interest in developing model compression methods (Sun et al., 2019; Sanh et al., 2019; Shen et al., 2019) to reduce network size in pre-trained LMs, while retaining comparable performance and efficiency.

PKD (Sun et al., 2019) was the first known effort in this expedition, an elegant and effective method that uses knowledge distillation (KD) for BERT model compression at finetuning stage. Later on, DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2019) and MobileBERT (Sun et al., 2020) carried on the torch and extended similar compression techniques to pre-training stage, allowing efficient training of task-agnostic compressed models. In addition to the conventional KL-divergence loss applied to the probabilistic output of the teacher and student networks, an L_2 loss measuring the difference between normalized hidden layers has proven to be highly effective in these methods. However, L_2 norm follows the assumption that all dimensions of the target representation are independent, which overlooks important structural information in the many hidden layers of BERT teacher.

Motivated by this, we propose **Co**ntrastive **D**istillation for **I**ntermediate **R**epresentations (CoDIR), which uses a *contrastive objective* to capture higher-order output dependencies between intermediate representations of BERT teacher and the student. Contrastive learning (Gutmann and Hyvärinen, 2010) aims to learn representations by enforcing similar elements to be equal and dissimilar elements further apart. Formulated in either supervised or unsupervised way, it has been successfully applied to diverse applications (Hjelm et al., 2018; He et al., 2019; Tian et al., 2019; Khosla et al., 2020). To the best of our knowledge, utilizing contrastive learning to compress large Transformer models is still an unexplored territory, which is the main focus of this paper.

A teacher network's hidden layers usually contain rich semantic and syntactic knowledge that can be instrumental if successfully passed on to the student (Tenney et al., 2019; Kovaleva et al.,

[1] Code will be released at https://github.com/intersun/CoDIR.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 498–508, November 16–20, 2020. ©2020 Association for Computational Linguistics

2019; Sun et al., 2019). Thus, instead of directly applying contrastive loss to the final output layer of the teacher, we apply contrastive learning to its intermediate layers, in addition to the use of KL-loss between the probabilistic outputs of the teacher and student. This casts a stronger regularization effect for student training by capturing more informative signals from intermediate representations. To maximize the exploitation of intermediate layers of the teacher, we also propose the use of mean-pooled representation as the distillation target, which is empirically more effective than commonly used special `[CLS]` token.

To realize constrastive distillation, we define a *congruent* pair $(\boldsymbol{h}_i^t, \boldsymbol{h}_i^s)$ as the pair of representations of the same data input from the teacher and student networks, as illustrated in Figure 1. *Incongruent* pair $(\boldsymbol{h}_i^t, \boldsymbol{h}_j^s)$ is defined as the pair of representations of two different data samples through the teacher and the student networks, respectively. The goal is to train the student network to distinguish the congruent pair from a large set of incongruent pairs, by minimizing the constrastive objective.

For efficient training, all data samples are stored in a memory bank (Wu et al., 2018; He et al., 2019). During finetuning, incongruent pairs can be selected by choosing sample pairs with different labels to maximize the distance. For pre-training, however, it is not straightforward to construct incongruent pairs this way as labels are unavailable. Thus, we randomly sample data points from the same mini-batch pair to form incongruent pairs, and construct a proxy congruent-incongruent sample pool to assimilate what is observed in the downstream tasks during finetuning stage. This and other designs in CoDIR make constrative learning possible for LM compression, and have demonstrated strong performance and high efficiency in experiments.

Our contributions are summarized as follows. (i) We propose CoDIR, a principled framework to distill knowledge in the intermediate representations of large-scale language models via a contrastive objective, instead of a conventional L_2 loss. (ii) We propose effective sampling strategies to enable CoDIR in both pre-training and finetuning stages. (iii) Experimental results demonstrate that CoDIR can successfully train a half-size Transformer model that achieves competing performance to BERT-base on the GLUE benchmark (Wang et al., 2018), with half training time and GPU demand. Our pre-trained model checkpoint will be released for public access.

2 Related Work

Language Model Compression To reduce computational cost of training large-scale language models, many model compression techniques have been developed, such as quantization (Shen et al., 2019; Zafrir et al., 2019), pruning (Guo et al., 2019; Gordon et al., 2020; Michel et al., 2019), knowledge distillation (Tang et al., 2019; Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2019; Sun et al., 2020), and direct Transformer block modification (Kitaev et al., 2020; Wu et al., 2020).

Quantization refers to storing model parameters from 32- or 16-bit floating number to 8-bit or even lower. Directly truncating the parameter values will cause significant accuracy loss, hence quantization-aware training has been developed to maintain similar accuracy to the original model (Shen et al., 2019; Zafrir et al., 2019). Michel et al. (2019) found that even after most attention heads are removed, the model still retains similar accuracy, indicating there is high redundancy in the learned model weights. Later studies proposed different pruning-based methods. For example, Gordon et al. (2020) simply removed the model weights that are close to zero; while Guo et al. (2019) used re-weighted L_1 and proximal algorithm to prune weights to zero. Note that simple pruning does not improve inference speed, unless there is structure change such as removing the whole attention head.

There are also some efforts that try to improve the Transformer block directly. Typically, language models such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) can only handle a sequence of tokens in length up to 512. Kitaev et al. (2020) proposed to use reversible residual layers and locality-sensitive hashing to reduce memory usage to deal with extremely long sequences. Besides, Wu et al. (2020) proposed to use convolutional neural networks to capture short-range attention such that reducing the size of self-attention will not significantly hurt performance.

Another line of research on model compression is based on knowledge transfer, or knowledge distillation (KD) (Hinton et al., 2015), which is the main focus of this paper. Note that previously introduced model compression techniques are orthogonal to KD, and can be bundled for further speedup. Distilled BiLSTM (Tang et al., 2019) tried to dis-

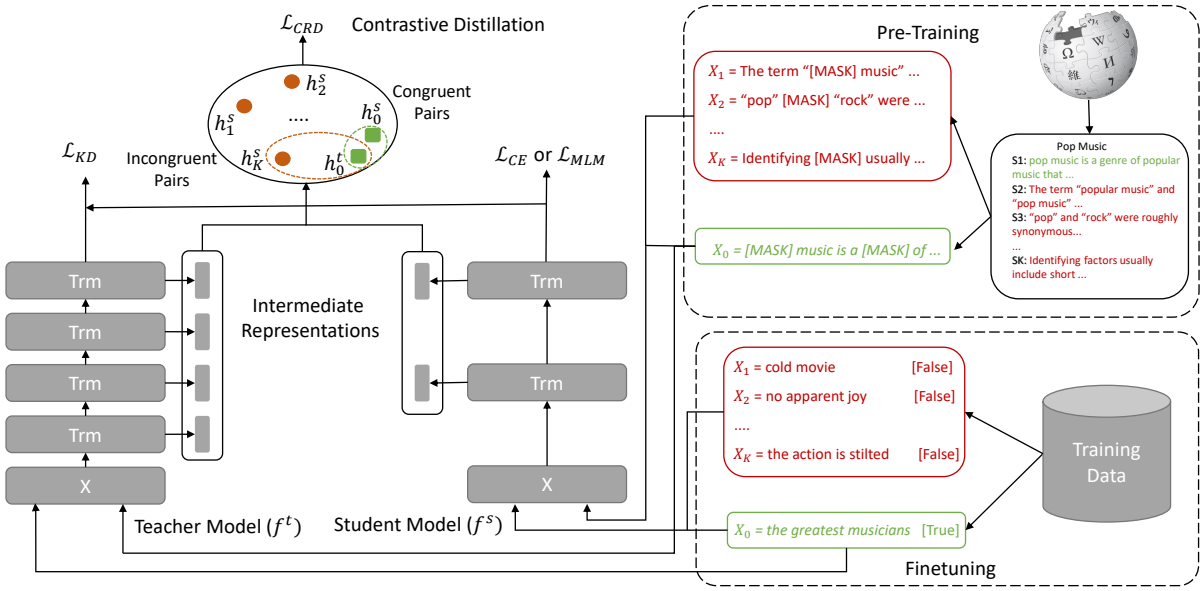

Figure 1: Overview of the proposed CoDIR framework for language model compression in both pre-training and finetuning stages. "Trm" represents a Transformer block, X are input tokens, f^t and f^s are teacher and student models, and $X_0, \{X_i\}_{i=1}^K$ represent one positive sample and a set of negative samples, respectively. The difference between CoDIR pre-training and finetuning mainly lies in the negative example sampling method.

till knowledge from BERT into a simple LSTM. Though achieving more than 400 times speedup compared to BERT-large, it suffers from significant performance loss due to the shallow network architecture. DistilBERT (Sanh et al., 2019) proposed to distill predicted logits from the teacher model into a student model with 6 Transformer blocks. BERT-PKD (Sun et al., 2019) proposed to not only distill the logits, but also the representation of `[CLS]` tokens from the intermediate layers of the teacher model. TinyBERT (Jiao et al., 2019), MobileBERT (Sun et al., 2020) and SID (Aguilar et al., 2019) further proposed to improve BERT-PKD by distilling more internal representations to the student, such as embedding layers and attention weights. These existing methods can be generally divided into two categories: (i) task-specific, and (ii) task-agnostic. Task-specific methods, such as Distilled BiLSTM, BERT-PKD and SID, require the training of individual teacher model for each downstream task; while task-agnostic methods such as DistilBERT, TinyBERT and MobileBERT use KD to pre-train a model that can be applied to all downstream tasks by standard finetuning.

Contrastive Representation Learning Contrastive learning (Gutmann and Hyvärinen, 2010; Arora et al., 2019) is a popular research area that has been successfully applied to density estimation and representation learning, especially in self-supervised setting (He et al., 2019; Chen et al., 2020). It has been shown that the contrastive objective can be interpreted as maximizing the lower bound of mutual information between different views of the data (Hjelm et al., 2018; Oord et al., 2018; Bachman et al., 2019; Hénaff et al., 2019). However, it is unclear whether the success is determined by mutual information or by the specific form of the contrastive loss (Tschannen et al., 2019). Recently, it has been extended to knowledge distillation and cross-modal transfer for image classification tasks (Tian et al., 2019). Different from prior work, we propose the use of contrastive objective for Transformer-based model compression and focus on language understanding tasks.

3 CoDIR for Model Compression

In this section, we first provide an overview of the proposed method in Sec. 3.1, then describe the details of contrastive distillation in Sec. 3.2. Its adaptation to pre-training and finetuning is further discussed in Sec. 3.3.

3.1 Framework Overview

We use RoBERTa-base (Liu et al., 2019) as the teacher network, denoted as f^t, which has 12 layers with 768-dimension hidden representations. We aim to transfer the knowledge of f^t into a student network f^s, where f^s is a 6-layer Trans-

former (Vaswani et al., 2017) to mimic the behavior of f^t (Hinton et al., 2015). Denote a training sample as (X, y), where $X = (x_0, \ldots, x_{L-1})$ is a sequence of tokens in length L, and y is the corresponding label (if available). The word embedding matrix of X is represented as $\mathbf{X} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{L-1})$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional vector, and $\mathbf{X} \in \mathbb{R}^{L \times d}$. In addition, the intermediate representations at each layer for the teacher and student are denoted as $\mathbf{H}^t = (\mathbf{H}_1^t, \ldots, \mathbf{H}_{12}^t)$ and $\mathbf{H}^s = (\mathbf{H}_1^s, \ldots, \mathbf{H}_6^s)$, respectively, where $\mathbf{H}_i^t, \mathbf{H}_i^s \in \mathbb{R}^{L \times d}$ contains all the hidden states in one layer. And $\boldsymbol{z}^t, \boldsymbol{z}^s \in \mathbb{R}^k$ are the logit representations (before the softmax layer) of the teacher and student, respectively, where k is the number of classes.

As illustrated in Figure 1, our distillation objective consists of three components: (i) original *training loss* from the target task; (ii) conventional *KL-divergence-based loss* to distill the knowledge of $\boldsymbol{z}^t$ into $\boldsymbol{z}^s$; (iii) proposed *contrastive loss* to distill the knowledge of $\mathbf{H}^t$ into $\mathbf{H}^s$. The final training objective can be written as:

$$\begin{aligned}\mathcal{L}_{\text{CoDIR}}(\boldsymbol{\theta}) &= \mathcal{L}_{\text{CE}}(\boldsymbol{z}^s, y; \boldsymbol{\theta}) + \alpha_1 \mathcal{L}_{\text{KD}}(\boldsymbol{z}^t, \boldsymbol{z}^s; \boldsymbol{\theta}) \\ &\quad + \alpha_2 \mathcal{L}_{\text{CRD}}(\mathbf{H}^t, \mathbf{H}^s; \boldsymbol{\theta}) \,, \end{aligned} \tag{1}$$

where $\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KD}}$ and $\mathcal{L}_{\text{CRD}}$ correspond to the original loss, KD loss and contrastive loss, respectively. $\boldsymbol{\theta}$ denotes all the learnable parameters in the student f^s, while the teacher network is pre-trained and kept fixed. α_1, α_2 are two hyper-parameters to balance the loss terms.

$\mathcal{L}_{\text{CE}}$ is typically implemented as a cross-entropy loss for classification problems, and $\mathcal{L}_{\text{KD}}$ can be written as

$$\mathcal{L}_{\text{KD}}(\boldsymbol{z}^t, \boldsymbol{z}^s; \boldsymbol{\theta}) = \text{KL}\left(g(\boldsymbol{z}^t/\rho) \| g(\boldsymbol{z}^s/\rho)\right) \,, \tag{2}$$

where $g(\cdot)$ denotes the softmax function, and ρ is the temperature. $\mathcal{L}_{\text{KD}}$ encourages the student network to produce distributionally-similar outputs to the teacher network.

Only relying on the final logit output for distillation discards the rich information hidden in the intermediate layers of BERT. Recent work (Sun et al., 2019; Jiao et al., 2019) has found that distilling the knowledge from intermediate representations with L_2 loss can further enhance the performance. Following the same intuition, our proposed method also aims to achieve this goal, with a more principled contrastive objective as detailed below.

3.2 Contrastive Distillation

First, we describe how to summarize intermediate representations into a concise feature vector. Based on this, we detail how to perform contrastive distillation (Tian et al., 2019) for model compression.

Intermediate Representation Directly using $\mathbf{H}^t$ and $\mathbf{H}^s$ for distillation is infeasible, as the total feature dimension is $|\mathbf{H}^s| = 6 \times 512 \times 768 \approx 2.4$ million for a sentence in full length (*i.e.*, $L = 512$). Therefore, we propose to first perform mean-pooling over $\mathbf{H}^t$ and $\mathbf{H}^s$ to obtain a layer-wise sentence embedding. Note that the embedding of the `[CLS]` token can also be used directly for this purpose; however, in practice we found that mean-pooling performs better. Specifically, we conduct row-wise average over $\mathbf{H}_i^t$ and $\mathbf{H}_i^s$:

$$\bar{\boldsymbol{h}}_i^t = \text{Pool}(\mathbf{H}_i^t), \;\; \bar{\boldsymbol{h}}_i^s = \text{Pool}(\mathbf{H}_i^s) \,, \tag{3}$$

where $\bar{\boldsymbol{h}}_i^t, \bar{\boldsymbol{h}}_i^s \in \mathbb{R}^d$ are the sentence embedding for layer i of the teacher and student model, respectively. Therefore, the student's intermediate representation can be summarized as $\bar{\boldsymbol{h}}^s = [\bar{\boldsymbol{h}}_1^s; \ldots; \bar{\boldsymbol{h}}_6^s] \in \mathbb{R}^{6d}$, where $[;]$ denotes vector concatenation. Similarly, the teacher's intermediate representation can be summarized as $\bar{\boldsymbol{h}}^t = [\bar{\boldsymbol{h}}_1^t; \ldots; \bar{\boldsymbol{h}}_{12}^t] \in \mathbb{R}^{12d}$. Two linear mappings $\phi^s : \mathbb{R}^{6d} \to \mathbb{R}^m$ and $\phi^t : \mathbb{R}^{12d} \to \mathbb{R}^m$ are further applied to project $\bar{\boldsymbol{h}}^t$ and $\bar{\boldsymbol{h}}^s$ into the same low-dimensional space, yielding $\boldsymbol{h}^t, \boldsymbol{h}^s \in \mathbb{R}^m$, which are used for calculating the contrastive loss.

Contrastive Objective Given a training sample (X_0, y_0), we first randomly select K negative samples with different labels, denoted as $\{(X_i, y_i)\}_{i=1}^K$. Following the above process, we can obtain a summarized intermediate representation $\boldsymbol{h}_0^t, \boldsymbol{h}_0^s \in \mathbb{R}^m$ by sending X_0 to both the teacher and student network. Similarly, for negative samples, we can obtain $\{\boldsymbol{h}_i^s\}_{i=1}^K$.

Contrastive learning aims to map the student's representation $\boldsymbol{h}_0^s$ close to the teacher's representation $\boldsymbol{h}_0^t$, while the negative samples' representations $\{\boldsymbol{h}_i^s\}_{i=1}^K$ far apart from $\boldsymbol{h}_0^t$. To achieve this, we use the following InfoNCE loss (Oord et al., 2018) for model training:

$$\mathcal{L}_{\text{CRD}}(\boldsymbol{\theta}) = -\log \frac{\exp\left(\langle \boldsymbol{h}_0^t, \boldsymbol{h}_0^s \rangle / \tau\right)}{\sum_{j=0}^K \exp\left(\langle \boldsymbol{h}_0^t, \boldsymbol{h}_j^s \rangle / \tau\right)} \,, \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the cosine similarity between two feature vectors, and τ is the temperature that

controls the concentration level. As demonstrated, contrastive distillation is implemented as a $(K+1)$-way classification task, which is interpreted as maximizing the lower bound of mutual information between $\boldsymbol{h}_0^t$ and $\boldsymbol{h}_0^s$ (Oord et al., 2018; Tian et al., 2019).

3.3 Pre-training and Finetuning Adaptation

Memory Bank For a positive pair $(\boldsymbol{h}_0^t, \boldsymbol{h}_0^s)$, one needs to compute the intermediate representations for all the negative samples, *i.e.*, $\{\boldsymbol{h}_i^s\}_{i=1}^K$, which requires $K+1$ times computation compared to normal training. A large number of negative samples is required to ensure performance (Arora et al., 2019), which renders large-scale contrastive distillation infeasible for practical use. To address this issue, we follow Wu et al. (2018) and use a memory bank $\mathbf{M} \in \mathbb{R}^{N \times m}$ to store the intermediate representation of all N training examples, and the representation is only updated for positive samples in each forward propagation. Therefore, the training cost is roughly the same as in normal training. Specifically, assume the mini-batch size is 1, then at each training step, $\mathbf{M}$ is updated as:

$$\boldsymbol{m}_0 = \beta \cdot \boldsymbol{m}_0 + (1-\beta) \cdot \boldsymbol{h}_0^s\,, \qquad (5)$$

where $\boldsymbol{m}_0$ is the retrieved representation from memory bank $\mathbf{M}$ that corresponds to $\boldsymbol{h}_0^s$, and $\beta \in (0,1)$ is a hyper-parameter that controls how aggressively the memory bank is updated.

Finetuning Since task-specific label supervision is available in finetuning stage, applying CoDIR to finetuning is relatively straightforward. When selecting negative samples from the memory bank, we make sure the selected samples have different labels from the positive sample.

Pre-training For pre-training, the target task becomes masked language modeling (MLM) (Devlin et al., 2018). Therefore, we replace the $\mathcal{L}_{\text{CE}}$ loss in Eqn. (1) with $\mathcal{L}_{\text{MLM}}$. Following Liu et al. (2019); Lan et al. (2019), we did not include the next-sentence-prediction task for pre-training, as it does not improve performance on downstream tasks. Since task-specific label supervision is unavailable during pre-training, we propose an effective method to select negative samples from the memory bank. Specifically, we sample negative examples randomly from the same mini-batch each time, as they have closer semantic meaning as some of them are from the same article, especially for Bookcorpus (Zhu et al., 2015). Then, we use the sampled negative examples to retrieve representations from the memory bank. Intuitively, negative examples sampled in this way serve as "hard" negatives, compared to randomly sampling from the whole training corpus; otherwise, the $\mathcal{L}_{\text{CRD}}$ loss could easily drop to zero if the task is too easy.

4 Experiments

In this section, we present comprehensive experiments on a wide range of downstream tasks and provide detailed ablation studies, to demonstrate the effectiveness of the proposed approach to large-scale LM compression.

4.1 Datasets

We evaluate the proposed approach on sentence classification tasks from the General Language Understanding Evaluate (GLUE) benchmark (Wang et al., 2018), as our finetuning framework is designed for classification, and we only exclude the STS-B dataset (Cer et al., 2017). Following other works (Sun et al., 2019; Jiao et al., 2019; Sun et al., 2020), we also do not run experiments on WNLI dataset (Levesque et al., 2012), as it is very difficult and even majority voting outperforms most benchmarks.[2]

CoLA Corpus of Linguistic Acceptability (Warstadt et al., 2019) contains a collection of 8.5k sentences drawn from books or journal articles. The goal is to predict if the given sequence of words is grammatically correct. Mattthews correlation coefficient is used as the evaluation metric.

SST-2 Stanford Sentiment Treebank (Socher et al., 2013) consists of 67k human-annotated movie reviews. The goal is to predict whether each review is positive or negative. Accuracy is used as the evaluation metric.

MRPC Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) consists of 3.7k sentence pairs extracted from online news, and the goal to predict if each pair of sentences is semantically equivalent. F1 score from GLUE server is reported as the metric.

QQP The Quora Question Pairs[3] task consists of 393k question pairs from Quora webiste. The

[2] Please refer to https://gluebenchmark.com/leaderboard.

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

Model	CoLA (8.5k)	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m/-mm (393k)	QNLI (108k)	RTE (2.5k)	Ave.
RoBERTa-base (Ours)	60.3	95.3	91.0	89.6	87.7/86.8	93.5	71.7	84.5
BERT-base (Google)	52.1	93.5	88.9	89.2	84.6/83.4	90.5	66.4	81.7
DistilBERT	32.8	91.4	82.4	88.5	78.9/78.0	85.2	54.1	73.9
SID	41.4	-	83.8	89.1	-	-	62.2	-
BERT_6-PKD	24.8	92.0	86.4	88.9	81.5/81.0	89.0	65.5	76.0
$\text{TinyBERT}_4^\star$	43.3	92.6	86.4	**89.2***	82.5/81.8	87.7	62.9	76.7
TinyBERT_6	51.1*	93.1	87.3	89.1	**84.6/83.2**	**90.4**	66.0	80.6
MLM-Pre + Fine	50.6	93.0	88.7	**89.2**	82.9/82.0	89.6	62.1	79.8
CoDIR-Fine	53.6	93.6	89.4	89.1	83.6/82.8	**90.4**	65.6	81.0
CoDIR-Pre	**53.7**	**94.1**	89.3	89.1	83.7/82.6	**90.4**	66.8	**81.2**
CoDIR-Pre + CoDIR-Fine	**53.7**	93.6	**89.6**	89.1	83.5/82.7	90.1	**67.1**	**81.2**

Table 1: Results on GLUE Benchmark. (*) indicates those numbers are unavailable in the original papers and were obtained by us through submission to the official leaderboard using their codebases. Other results are obtained from published papers. ($\star$) indicates those methods with fewer Transformer blocks, and may not be fair comparison.

task is to predict whether a pair of questions is semantically equivalent. Accuracy is used as the evaluation metric.

NLI Multi-Genre Natural Language Inference Corpus **(MNLI)** (Williams et al., 2017), Question-answering NLI **(QNLI)** (Rajpurkar et al., 2016) and Recognizing Textual Entailment **(RTE)**[4] are all natural language inference (NLI) tasks, which consist of 393k/108k/2.5k pairs of premise and hypothesis. The goal is to predict if the premise entails the hypothesis, or contradicts it, or neither. Accuracy is used as the evaluation metric. Besides, MNLI test set is further divided into two splits: matched (MNLI-m, in-domain) and mismatched (MNLI-mm, cross-domain), accuracy for both are reported.

4.2 Implementation Details

We mostly follow the pre-training setting from Liu et al. (2019), and use the fairseq implementation (Ott et al., 2019). Specifically, we truncate raw text into sentences with maximum length of 512 tokens, and randomly mask 15% of tokens as `[MASK]`. For model architecture, we use a randomly initialized 6-layer Transformer model as the student, and RoBERTa-base with 12-layer Transformer as the teacher. The student model was first trained by using Adam optimizer with learning rate 0.0007 and batch size 8192 for 35,000 steps. For computational efficiency, this model serves as the initialization for the second-stage pre-training with the teacher. Then, the student model is further trained for another 10,000 steps with KD and the proposed contrastive objective, with learning rate set to 0.0001. We denote this model as CoDIR-Pre. For ablation purposes, we also train two baseline models with only MLM loss or KD loss, using the same learning rate and number of steps. Similarly, these two models are denoted as MLM-Pre and KD-Pre, respectively. For other hyper-parameters, we use $\alpha_1 = \alpha_2 = 0.1$ for both $\mathcal{L}_{\text{KD}}$ and $\mathcal{L}_{\text{CRD}}$.

Due to high computational cost for pre-training, all the hyper-parameters are set empirically without tuning. As there exist many combinations of pre-training loss (MLM, KD, and CRD) and finetuning strategies (standard finetuning with cross-entropy loss, and finetuning with additional CRD loss), a grid search of all the hyper-parameters is infeasible. Thus, for standard finetuning, we search learning rate from {1e-5, 2e-5} and batch size from {16, 32}. The combination with the highest score on dev set is reported for ablation studies, and is kept *fixed* for future experiments. We then fix the hyper-parameters in KD as $\rho = 2, \alpha_1 = 0.7$, and search weight of the CRD loss α_2 from $\{0.1, 0.5, 1\}$, and the number of negative samples from {100, 500, 1000}. Results with the highest dev scores were submitted to the official GLUE server to obtain the final results. For fair comparison with other baseline methods, all the results are based on single-model performance.

4.3 Experimental Results

Results of different methods from the official GLUE server are summarized in Table 1. For sim-

[4]Collections of series of annual textual entailment challenges.

Pre-training Loss	Finetuning Method	CoLA (8.5k)	SST-2 (67k)	MRPC (3.7k)	QNLI (108k)	RTE (2.5k)	Ave.
$\mathcal{L}_{\text{MLM}}$	Standard	55.4	92.0	88.0	90.2	67.1	78.5
$\mathcal{L}_{\text{MLM}}$	KD	57.8	92.4	89.5	90.3	68.2	79.4
$\mathcal{L}_{\text{MLM}}$	CoDIR-Fine	59.3	92.7	90.0	90.7	70.8	80.7
$\mathcal{L}_{\text{MLM}}$	Standard	55.4	92.0	88.0	90.2	67.1	78.5
$\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{KD}}$	Standard	56.6	92.7	89.0	90.5	69.0	79.6
CoDIR-Pre	Standard	57.6	92.8	90.4	90.8	71.5	80.6

Table 2: Ablation study on different combination of pre-trained models and finetuning approach. The results are based on GLUE dev set.

Model	#Trm Layers	#Params	#Params (Emb Layer)	Inference Time (ms/seq)	Speed-up
BERT-base	12	109.5M	23.8M	2.60	1.00×
RoBERTa-base	12	125.2M	39.0M	2.53	1.03×
DistilBERT	6	67.0M	23.8M	1.27	2.05×
CoDIR-Pre	6	82.7M	39.0M	1.25	2.08×

Table 3: Inference speed comparison between teacher and students. Inference time is measured on MNLI dev set. Speed up is measured against BERT-base, which is the teacher model for other baseline methods.

plicity, we denote our baseline approach without using any teacher supervision as "MLM-Pre + Fine": pre-trained by using MLM loss first, then finetuning using standard cross-entropy loss. Our baseline already achieves high average score across 8 tasks, and outperforms task-specific model compression methods (such as SID (Aguilar et al., 2019) and BERT-PKD (Sun et al., 2019)) as well as DistilBERT (Sanh et al., 2019) by a large margin.

After adding contrastive loss at the finetuning stage (denoted as CoDIR-Fine), the model outperforms the state-of-the-art compression method, TinyBERT with 6-layer Transformer, on average GLUE score. Especially on datasets with fewer data samples, such as CoLA and MRPC, the improved margin is large (+2.5% and +2.1%, respectively). Compared to our MLM-Pre + Fine baseline, CoDIR-Fine achieves significant performance gain on almost all tasks (+1.2% absolute improvement on average score), demonstrating the effectiveness of the proposed approach. The only exception is QQP (-0.1%) with more than 360k training examples. In such case, standard finetuning may already bring in enough performance boost with this large-scale labeled dataset, and the gap between the teacher and student networks is already small (89.6 vs 89.2).

We further test the effectiveness of CoDIR for pre-training (CoDIR-Pre), by applying standard finetuning on model pre-trained with additional contrastive loss. Again, compared to the MLM-Pre + Fine baseline, this improves the model performance on almost all the tasks (except QQP), with a significant lift on the average score (+1.4%). We notice that this model performs similarly to the contrastive-finetuning only approach (CoDIR-Fine) on almost all tasks. However, CoDIR-Pre is preferred because it utilizes the teacher's knowledge in the pre-training stage, thus no task-specific teacher is needed for finetuning downstream tasks. Finally, we experiment with the combination of CoDIR-Pre and CoDIR-Fine, and our observation is that adding constrastive loss for finetuning is not bringing in much improvement after already using constrastive loss in pre-training. Our hypothesis is that the model's ability to identify negative examples is already well learned during pre-training.

Inference Speed We compare the inference speed of the proposed CoDIR with the teacher network and other baselines. Statistics of Transformer layers and parameters are presented in Table 3. The statistics for BERT_6-PKD and TinyBERT_6 are omitted as they share the same model architecture as DistilBERT. To test the inference speed, we ran each algorithm on MNLI dev set for 3 times, with batch size 32 and maximum sequence length 128 under the same hardware configuration. The average running time with 3 different random seeds is reported as the final inference speed. Though our RoBERTa teacher has almost 16 million more parameters, it shares almost the same inference

Model	CoLA (8.5k)	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m/-mm (393k)	QNLI (108k)	RTE (2.5k)	Ave.
CoDIR-Fine ([CLS])	57.6	92.9	89.2	91.3	84.0/84.0	90.8	70.0	82.5
CoDIR-Fine (Mean Pool)	59.3	92.7	90.0	91.3	84.2/84.2	90.7	70.8	**83.0**
CoDIR-Fine (100-neg)	57.3	92.1	88.2	91.3	84.0/84.0	90.4	69.3	82.1
CoDIR-Fine (500-neg)	58.2	92.5	89.7	91.2	84.0/84.0	90.6	70.8	82.6
CoDIR-Fine (1000-neg)	59.3	92.7	90.0	91.3	84.2/84.2	90.7	70.4	**83.0**

Table 4: Ablation study on the use of [CLS] and Mean-Pooling as sentence embedding (upper part) and effect of number of negative examples (neg) for CoDIR-Fine (bottom part). The results are based on GLUE dev set.

Model	CoLA (8.5k)	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m (393k)	QNLI (108k)	RTE (2.5k)
Median	56.4	92.4	87.9	91.2	83.9	90.7	66.3
Maximum	57.8	93.0	90.3	91.3	84.2	91.0	70.2
Standard Deviation	1.46	0.28	1.66	0.06	0.43	0.18	1.41

Table 5: Analysis of model variance on GLUE dev set. Statistical results (median, maximum, and standard deviation) are based on 8 runs with the same hyper-parameters.

speed as BERT-base, because its computational cost mainly comes from the embedding layer with 50k vocabulary size that does not affect inference speed. By reducing the number of Transformer layers to 6, our proposed student model achieves 2 times speed up compared to the teacher, and achieves state-of-the-art performance among all models with similar inference time.

4.4 Ablation Studies

Sentence Embedding We also conduct experiments to evaluate the effectiveness of using different sentence embedding strategies. More detailed, based on the same model pre-trained on $\mathcal{L}_{\text{MLM}}$ alone, we run finetuning experiments with contrastive loss on the GLUE dataset by using: (i) [CLS] as sentence embedding; and (ii) mean-pooling as sentence embedding. The results on GLUE dev set are presented in top rows of Table 4, showing that mean-pooling yields better results than [CLS] (83.0 vs. 82.5 on average). As a result, we use mean pooling as our chosen sentence embedding for all our experiments.

Negative Examples As we mentioned in Section 4.2, the experiments are conducted using 100, 500 and 1000 negative examples. We then evaluate the effect of number of negative examples by comparing their results on GLUE dev set, and the results are presented in the bottom part of Table 4. Obviously, for most dataset the accuracy increases as a larger number of negative examples are used during training. Similar observations were also reported in Tian et al. (2019), and a theoretical analysis is provided in Arora et al. (2019). The only two exceptions are QQP and RTE. As discussed in Section 4.3, our CoDIR method seems also not work well on QQP due to the small gap between teacher and student. As for RTE, due to the small number of training examples, the results are quite volatile, which may make the results inconsistent. Besides, the number of negative examples is close to the number of examples per class (1.25k) for RTE, which can also result in the contrastive loss close to 0.

Contrastive Loss We first evaluate the effectiveness of the proposed CRD loss for finetuning on a subset of GLUE dev set, using the following settings: (i) finetuning with cross-entropy loss only; (ii) finetuning with additional KD loss; and (iii) finetuning with additional KD loss and CRD loss. Results in Table 2 (upper part) show that using KD improves over standard finetuning by 0.9% on average, and using CRD loss further improves another 1.0%, demonstrating the advantage of using contrastive learning for finetuning.

To further validate performance improvement of using contrastive loss on pre-training, we apply standard finetuning to three different pre-trained models: (i) model pre-trained by $\mathcal{L}_{\text{MLM}}$ (MLM-Pre); (ii) model pre-trained by $\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{KD}}$ (KD-Pre); and (iii) model pre-trained by $\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{CRD}}$ (CoDIR-Pre). Results are summarized in Ta-

ble 2 (bottom part). Similar trend can be observed that the model pre-trained with additional CRD loss performs the best, outperforming MLM-Pre and KD-Pre by 1.9% and 1.0% on average, respectively.

Model Variance Since different random seeds can exhibit different generalization behaviors, especially for tasks with a small training set (*e.g.,* CoLA), we examine the median, maximum and standard deviation of model performance on the dev set of each GLUE task, and present the results in Table 5. As expected, the models are more stable on larger datasets (SST-2, QQP, MNLI, and QNLI), where all standard deviations are lower than 0.5. However, the model is sensitive to the random seeds on smaller datasets (CoLA, MRPC, and RTE) with the standard deviation around 1.5. These analysis results provide potential references for future work on language model compression.

5 Conclusion

In this paper, we present CoDIR, a novel approach to large-scale language model compression via the use of contrastive loss. CoDIR utilizes information from both teacher's output layer and its intermediate layers for student model training. Extensive experiments demonstrate that CoDIR is highly effective in both finetuning and pre-training stages, and achieves state-of-the-art performance on GLUE benchmark compared to existing models with a similar size. All existing work either use BERT-base or RoBERTa-base as teacher. For future work, we plan to investigate the use of a more powerful language model, such as Megatron-LM (Shoeybi et al., 2019), as the teacher; and different strategies for choosing hard negatives to further boost the performance.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Edward Guo. 2019. Knowledge distillation from internal representations. *arXiv preprint arXiv:1910.03723*.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*.

Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In *NeurIPS*.

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*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. 2019. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. 2019. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*.

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

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

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*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

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

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. Q-bert: Hessian based ultra low precision quantization of bert. *arXiv preprint arXiv:1909.05840*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

S. Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *EMNLP/IJCNLP*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

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*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*.

Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. 2019. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

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 *NeurIPS*.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *TACL*.

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

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*.

Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.