Knowledge Distillation from Internal Representations

Gustavo Aguilar,[1] **Yuan Ling,**[2] **Yu Zhang,**[2] **Benjamin Yao,**[2] **Xing Fan,**[2] **Chenlei Guo**[2]
[1]Department of Computer Science, University of Houston, Houston, USA
[2]Alexa AI, Amazon, Seattle, USA
gaguilaralas@uh.edu, {yualing,yzzhan,banjamy,fanxing,guochenl}@amazon.com

Abstract

Knowledge distillation is typically conducted by training a small model (the student) to mimic a large and cumbersome model (the teacher). The idea is to compress the knowledge from the teacher by using its output probabilities as soft-labels to optimize the student. However, when the teacher is considerably large, there is no guarantee that the internal knowledge of the teacher will be transferred into the student; even if the student closely matches the soft-labels, its internal representations may be considerably different. This internal mismatch can undermine the generalization capabilities originally intended to be transferred from the teacher to the student. In this paper, we propose to distill the internal representations of a large model such as BERT into a simplified version of it. We formulate two ways to distill such representations and various algorithms to conduct the distillation. We experiment with datasets from the GLUE benchmark and consistently show that adding knowledge distillation from internal representations is a more powerful method than only using soft-label distillation.

Introduction

Transformer-based models have significantly advanced the field of natural language processing by establishing new state-of-the-art results in a large variety of tasks. Specifically, BERT (Devlin et al. 2018), GPT (Radford et al. 2018), GPT-2 (Radford et al. 2019), XLM (Lample and Conneau 2019), XLNet (Yang et al. 2019), and RoBERTa (Liu et al. 2019c) lead tasks such as text classification, sentiment analysis, semantic role labeling, question answering, among others. However, most of the models have hundreds of millions of parameters, which significantly slows down the training process and inference time. Besides, the large number of parameters demands a lot of memory consumption, making such models hard to adopt in production environments where computational resources are strictly limited.

Due to these limitations, many approaches have been proposed to reduce the size of the models while still providing similar performance. One of the most effective techniques is knowledge distillation (KD) in a teacher-student setting (Hinton, Vinyals, and Dean 2015), where a cumbersome already-optimized model (i.e., the teacher) produces output probabilities that are used to train a simplified model (i.e., the student). Unlike training with one-hot labels where the classes are mutually exclusive, using a probability distribution provides more information about the similarities of the samples, which is the key part of the teacher-student distillation.

Even though the student requires fewer parameters while still performing similar to the teacher, recent work shows the difficulty of distilling information from a huge model. Mirzadeh et al. (2019) state that, when the gap in between the teacher and the student is large (e.g., shallow vs. deep neural networks), the student struggles to approximate the teacher. They propose to use an intermediate teaching assistant (TA) model to distill the information from the teacher and then use the TA model to distill information towards the student. However, we argue that the abstraction captured by a large teacher is only exposed through the output probabilities, which makes the internal knowledge from the teacher (or the TA model) hard to infer by the student. This can potentially take the student to very different internal representations undermining the generalization capabilities initially intended to be transferred from the teacher.

In this paper, we propose to apply KD to internal representations. Our approach allows the student to internally behave as the teacher by effectively transferring its linguistic properties. We perform the distillation at different internal points across the teacher, which allows the student to learn and compress the abstraction in the hidden layers of the large model systematically. By including internal representations, we show that our student outperforms its homologous models trained on ground-truth labels, soft-labels, or both.

Related Work

Knowledge distillation has become one of the most effective and simple techniques to compress huge models into simpler and faster models. The versatility of this framework has allowed the extension of KD to scenarios where a set of expert models in different tasks distill their knowledge into a unified multi-task learning network (Clark et al. 2019b), as well as the opposite scenario where an ensemble of multi-task

models are distilled into a task-specific network (Liu et al. 2019a; 2019b). We extend the knowledge distillation framework with a different formulation by applying the same principle to internal representations.

Using internal representations to guide the training of a student model was initially explored by Romero et al. (2014). They proposed FITNET, a convolutional student network that is thinner and deeper than the teacher while using significantly fewer parameters. In their work, they establish a middle point in both the teacher and the student models to compare internal representations. Since the dimensionality between the teacher and the student differs, they use a convolutional regressor model to map such vectors into the same space, which adds a significant number of parameters to learn. Additionally, they mainly focus on providing a deeper student network than the teacher, exploiting the particular benefits of depth in convolutional networks. Our work differs from theirs in different aspects: 1) using a single point-wise loss on the middle layers has mainly a regularization effect, but it does not guarantee to transfer the internal knowledge from the teacher; 2) our distillation method is applied across all the student layers, which effectively compress groups of layers from the teacher into a single layer of the student; 3) we use the internal representations as-is instead of relying on additional parameters to perform the distillation; 4) we do not focus on deeper models than the teacher as this can slow down the inference time, and it is not necessarily an advantage on transformer-based models.

Concurrent to this work, similar transformer-based distillation techniques have been studied. Sanh et al. (2019) propose DistilBERT, which compresses BERT during pre-training to provide a smaller general-purpose model. They pre-train their model using a masked language modeling loss, a cosine embedding loss at the hidden states, and the teacher-student distillation loss. Conversely, Sun et al. (2019) distill their model during task-specific fine-tuning using a MSE loss at the hidden states and cross-entropy losses from soft- and hard-labels. While our work is similar to theirs, the most relevant differences are 1) the use of KL-divergence loss at the self-attention probabilities, which have been shown to capture linguistic knowledge (Clark et al. 2019a), and 2) the introduction of new algorithms to distill the internal knowledge from the teacher (i.e., progressive and stacked knowledge distillation).

Curriculum learning (CL) (Bengio 2009) is another line of research that focuses on teaching complex tasks by building upon simple concepts. Although the goal is similar to ours, CL is conducted by stages focusing on simple tasks first and progressively moving to more complicated tasks. However, this method requires annotations among the preliminary tasks, and they have to be carefully picked so that the order and relation among the build-up tasks are helpful for the model. Unlike CL, we focus on teaching the internal representations of an optimized complex model, which are assumed to have the preliminary build-up knowledge for the task of interest.

Other model compression techniques include quantization (Hubara et al. 2017; He et al. 2016; Courbariaux et al. 2016) and weights pruning (Han, Mao, and Dally 2015). The first one focuses on approximating a large model into a smaller one by reducing the precision of each of the parameters. The second one focuses on removing weights in the network that do not have a substantial impact on model performance. These techniques are complementary to the method we propose in this paper, which can potentially lead to a more effective overall compression approach.

Methodology

In this section, we detail the process of distilling knowledge from internal representations. First, we describe the standard KD framework (Hinton, Vinyals, and Dean 2015), which is an essential part of our method. Then, we formalize the objective functions to distill the internal knowledge of transformer-based models. Lastly, we propose various algorithms to conduct the internal distillation process.

Knowledge Distillation

Hinton, Vinyals, and Dean (2015) proposed knowledge distillation (KD) as a framework to compress a large model into a simplified model that achieves similar results. The framework uses a teacher-student setting where the student learns from both the ground-truth labels (if available) and the soft-labels provided by the teacher. The probability mass associated with each class in the soft-labels allows the student to learn more information about the label similarities for a given sample. The formulation of KD considering both soft and hard labels is given as follows:

$$\mathcal{L}_{KD} = -\frac{1}{N}\sum_{i}^{N} p(y_i|x_i,\theta_T) log(\hat{y}_i) - \lambda \frac{1}{N}\sum_{i}^{N} y_i log(\hat{y}_i)) \tag{1}$$

where θ_T represents the parameters of the teacher, and $p(y_i|x_i,\theta_T)$ are its soft-labels; $\hat{y}_i$ is the student prediction given by $p(y_i|x_i,\theta_S)$ where θ_S denotes its parameters, and λ is a small scalar that weights down the hard-label loss. Since the soft-labels often present high entropy, the gradient tends to be smaller than the one from the hard-labels. Thus, λ balances the terms by reducing the impact of the hard loss.

Matching Internal Representations

In order to make the student model behave as the teacher model, the student is optimized by the soft-labels from teacher's output. In addition, the student also acquires the abstraction hidden in the teacher by matching its internal representations. That is, we want to teach the student how to internally behave by compressing the knowledge of multiple layers from the teacher into a single layer of the student. Figure 1 shows a teacher with twice the number of layers of the student, where the colored boxes denote the layers where the student is taught the internal representation of the teacher. In this case, the student compresses two layers into one while preserving the linguistic behavior across the teacher layers.

We study the internal KD of transformer-based models, specifically the case of BERT and simplified versions of it (i.e., fewer transformer layers). We define the internal KD by using two terms in the loss function. Given a pair of

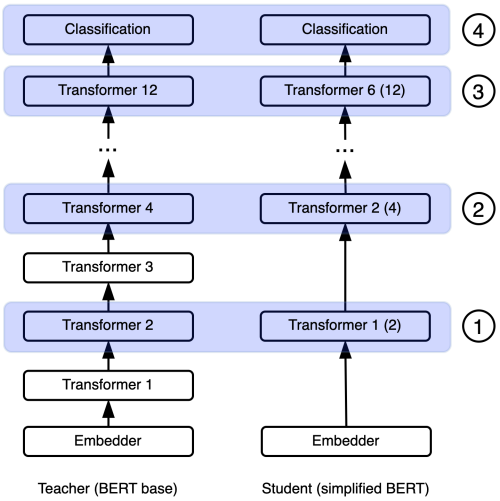

Figure 1: Knowledge distillation from internal representations. We show the internal layers that the teacher (left) distills into the student (right).

transformer layers to match (see Figure 1), we calculate (1) the Kullback-Leibler (KL) divergence loss across the self-attention probabilities of all the transformer heads[1], and (2) the cosine similarity loss between the `[CLS]` activation vectors for the given layers.

KL-divergence loss. Consider A as the self-attention matrix that contains row-wise probability distributions per token in a sequence given by $A = \text{softmax}(d_a^{-0.5}QK^T)$ (Vaswani et al. 2017). For a given head in a transformer layer, we use the KL-divergence loss as follows:

$$\mathcal{L}_{kl} = \frac{1}{L}\sum_{i}^{L} A_{T_i} log \frac{A_{T_i}}{A_{S_i}} \quad (2)$$

where L is the length of a sequence, A_{T_i} and A_{S_i} describe the i-th row of the self-attention matrix for the teacher and student, respectively. The motivation of applying this loss function to the self-attention matrices comes from recent research that documents the linguistic patterns captured by the attention probabilities of BERT (Clark et al. 2019a). Forcing the divergence between the self-attention probability distributions to be as small as possible preserves the linguistic behavior in the student.

Cosine similarity loss. For the second term of our internal distillation loss, we use cosine similarity[2] as follows:

$$\mathcal{L}_{cos} = 1 - cos(h_T, h_S) \quad (3)$$

where h_T and h_S are the hidden vector representations for the `[CLS]` token for the teacher and student, respectively. We include this term in our internal KD formulation to consider a similar behavior in the activation going through the network. That is, while KL-divergence focuses on the self-attention matrix, it is the weighted hidden vectors that finally pass to the upper layers, not the probabilities. Even if we force the self-attention probabilities to be similar, there is no guarantee that the final activation passed to the upper layers is similar. Thus, using this extra term, we can regularize the context representation of the sample to be similar to the one from the teacher.[3]

Algorithm 1 Stacked Internal Distillation (SID)

procedure HEADLOSS(TS, $batch, layer_t, layer_s$)
  for $sample \leftarrow batch$; **init** $\mathcal{L} \leftarrow 0$; **do**
    P $\leftarrow$ CONCATHEADS(TS.teacher, $sample, layer_t$)
    Q $\leftarrow$ CONCATHEADS(TS.student, $sample, layer_s$)
    $\mathcal{L} \leftarrow \mathcal{L} + mean(sum(\text{P} * log(\text{P/Q}), \text{axis=2}))$
  return $\mathcal{L}/size(batch)$
procedure STACKINTDISTILL(TS, $batch$)
  $\mathcal{L}_{cos}, \mathcal{L}_{kl} \leftarrow 0, 0$
  for $layer_t, layer_s \leftarrow$ MATCH($0 \ldots$ TS.nextLocked)) **do**
    $\text{CLS}_t \leftarrow$ GETCLS(TS.teacher, $batch, layer_t$)
    $\text{CLS}_s \leftarrow$ GETCLS(TS.student, $batch, layer_s$)
    $\mathcal{L}_{cos} \leftarrow \mathcal{L}_{cos} + mean(1 - cos(\text{CLS}_t, \text{CLS}_s))$
    $\mathcal{L}_{kl} \leftarrow \mathcal{L}_{kl}$ + HEADLOSS(TS, $batch, layer_t, layer_s$)
  return $\mathcal{L}_{cos}, \mathcal{L}_{kl}$
TS $\leftarrow$ INITIALIZETSMODEL()
repeat:
  for $e \leftarrow 0, epochs$ **do**
    if TS.nxtLockedLay $<$ TS.student.nLayers **then**
      ▷ Perform internal distillation
      for $batch \leftarrow data$; **init** $\tau \leftarrow 0$; **do**
        $\mathcal{L}_{cos}, \mathcal{L}_{kl} \leftarrow$ STACKINTDISTILL(TS, $batch$)
        $backprop$(TS, $\mathcal{L}_{cos} + \mathcal{L}_{kl}$)
        $\tau \leftarrow \tau + \mathcal{L}_{cos}$ ▷ Accumulate for threshold
      if $\tau < \mathcal{T}$ OR $e \geq lim$(TS.nxtLockedLay, e) **then**
        TS.nextLockedLay $\leftarrow$ TS.nxtLockedLay $+ 1$
    else
      ▷ Perform standard distillation
      for $batch \leftarrow data$ **do**
        $backprop$(TS, $xentropy$(TS, $batch$))
until convergence

How to Distill the Internal Knowledge?

Different layers across the teacher capture different linguistic concepts. Recent research shows that BERT builds linguistic properties that become more complex as we move from the bottom to the top of the network (Clark et al. 2019a). Since the model builds upon bottom representations, in addition to distilling all the internal layers simultaneously, we also consider distilling knowledge progressively matching internal representation in a bottom-up fashion. More specifically, we consider the following scenarios:

1. **Internal distillation of all layers**. All the layers of the student are optimized to match the ones from the teacher in every epoch. In Figure 1, the distillation simultaneously occurs on the circled numbers ①, ②, ③, and ④.
2. **Progressive internal distillation (PID)**. We distill the knowledge from lower layers first (close to the input) and

[1] We are interested in a loss function that considers the probability distribution as a whole, and not point-wise errors.

[2] L_2 loss could be used as well without impacting generality.

[3] We only use the context vector instead of all the hidden token vectors to avoid over-regularizing the model (Romero et al. 2014).

progressively move to upper layers until the model focuses only on the classification distillation. Only one layer is optimized at a time. In Figure 1, the loss will be given by the transition ① → ② → ③ → ④.

3. **Stacked internal distillation (SID)**. We distill the knowledge from lower layers first, but instead of moving from one layer to another exclusively, we keep the loss produced by previous layers stacking them as we move to the top. Once at the top, we only perform classification (see Algorithm 1). In Figure 1, the loss is determined by the transition ① → ① + ② → ① + ② + ③ → ④.

For the last two scenarios, to move to upper layers, the student either reaches a limited number of epochs per layer or a cosine loss threshold, whatever happens first (see line 24 in Algorithm 1). Additionally, these two scenarios can be combined with the classification loss at all times, not only until the model reaches the top layer.

Experiments and Results

Datasets

We conduct experiments on four datasets of the GLUE benchmark (Wang et al. 2018), which we describe briefly:

1. **CoLA**. The Corpus of Linguistic Acceptability (Warstadt, Singh, and Bowman 2018) is part of the single sentence tasks, and it requires to determine whether an English text is grammatically correct. It uses the Matthews Correlation Coefficient (MCC) to measure the performance.
2. **QQP**. The Quora Question Pairs[4] is a semantic similarity dataset, where the task is to determine whether two questions are semantically equivalent or not. It uses accuracy and F1 as metrics.
3. **MRPC**. The Microsoft Research Paraphrase Corpus (Dolan and Brockett 2005) contains pairs of sentences whose annotations describe whether the sentences are semantically equivalent or not. Similar to QQP, it uses accuracy and F1 as metrics.
4. **RTE**. The Recognizing Textual Entailment (Wang et al. 2018) has a collection of sentence pairs whose annotations describe entitlement or not entitlement between the sentences (formerly annotated with labels entitlement, contradiction or neutral). It uses accuracy as a metric.

For the MRPC and QQP datasets, the metrics are accuracy and F1, but we optimize the models on F1 only.

Parameter Initialization

We experiment with $\text{BERT}_{\text{base}}$ (Devlin et al. 2018) and simplified versions of it. In the case of BERT with 6 transformer layers, we initialize the parameters using different layers of the original $\text{BERT}_{\text{base}}$ model, which has 12 transformer layers. Since our goal is to compress the behavior of a subset of layers into one layer, we initialize a layer of the simplified BERT model with the upper layer of the subset. For example, Figure 1 shows the compression of groups of two layers into one layer, hence, the first layer of the student model is initialized with the parameters of the second layer of the $\text{BERT}_{\text{base}}$ model.[5]

Experimental Setup

Table 1 shows the results on the development set across four datasets. We define the experiments as follows:

- **Exp1.0: $\text{BERT}_{\text{base}}$**. This is the standard $\text{BERT}_{\text{base}}$ model that is fine-tuned on task-specific data without any KD technique. Once optimized, we use this model as a teacher for the KD experiments.
- **Exp1.1: BERT_{6}**. This is a simplified version of $\text{BERT}_{\text{base}}$, where we use 6 transformer layers instead of 12. The layer selection for initialization is described in the previous section. We do not use any KD for this experiment. The KD experiments described below use this architecture as the student model.
- **Exp2.0: BERT_{6} soft**. The model is trained with soft-labels produced by the fine-tuned $\text{BERT}_{\text{base}}$ teacher from experiment 1.0. This scenario corresponds to Equation 1 with $\lambda = 0$ to ignore the one-hot loss.
- **Exp3.0: BERT_{6} soft + kl**. The model uses both the soft-label and the KL-divergence losses from Equations 1 and 2. The KL-divergence loss is averaged across all the self-attention matrices from the student (i.e., 12 attention heads per transformer layer per 12 transformer layers).
- **Exp3.1: BERT_{6} soft + cos**. The model uses both the soft-label and the cosine similarity losses from Equations 1 and 3. The cosine similarity loss is computed from the `[CLS]` vector from all matching layers.
- **Exp3.2: BERT_{6} soft + kl + cos**. The model uses all the losses from all layers every epoch. This experiment combines experiments 3.0 and 3.1.
- **Exp3.3: BERT_{6} [PID] kl + cos → soft**. The model only uses *progressive internal distillation* until it reaches the classification layer. Once there, only soft-labels are used.
- **Exp3.4: BERT_{6} [SID] kl + cos → soft**. The model uses *stacked internal distillation* until it reaches the classification layer. Once there, only soft-labels are used.
- **Exp3.5: BERT_{6} [SID] kl + cos + soft**. The model uses *stacked internal distillation* and soft-labels distillation all the time during training.
- **Exp3.6: BERT_{6} [SID] kl + cos + soft + hard**. Same as Exp3.5, but it includes the hard-labels in the Equation 1 with $\lambda = 0.1$.

We optimize our models using Adam with an initial learning rate of 2e-5 and a learning rate scheduler as described by Devlin et al. (2018). We fine-tune $\text{BERT}_{\text{base}}$ for 10 epochs, and the simplified BERT models for 50 epochs both with a batch size of 32 samples and a maximum sequence length of 64 tokens. We evaluate the statistical significant of our models using t-tests as described by Dror et al. (2018). All

[4] data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[5] Note that the initialization does not take the parameters of the fine-tuned teacher. Instead, we use the parameters of the general-purpose $\text{BERT}_{\text{base}}$ model.

Experiment	Description	CoLA [8.5k] MCC	QQP [364k] Acuracy / F1	MRPC [3.7k] Acuracy / F1	RTE [2.5k] Acuracy
Fine-tuning $BERT_{base}$ and $BERT_6$ without KD					
Exp1.0	$BERT_{base}$	60.16	91.44 / 91.45	83.09 / 82.96	67.51
Exp1.1	$BERT_6$	44.56	90.58 / 90.62	76.23 / 73.72	59.93
Fine-tuning $BERT_6$ with different KD techniques using $BERT_{base}$ (Exp1.0) as teacher					
Exp2.0	$BERT_6$ soft	41.72	90.61 / 90.65	77.21 / 75.74	62.46
Exp3.0	$BERT_6$ soft + kl	43.70	91.32 / 91.32	**83.58 / 82.46**	67.15
Exp3.1	$BERT_6$ soft + cos	42.64	91.08 / 91.10	79.66 / 78.35	57.04
Exp3.2	$BERT_6$ soft + kl + cos	42.07	**91.37 / 91.38**	83.09 / 81.39	66.43
Exp3.3	$BERT_6$ [PID] kl + cos $\rightarrow$ soft	45.54	91.22 / 91.24	81.62 / 80.12	64.98
Exp3.4	$BERT_6$ [SID] kl + cos $\rightarrow$ soft	**46.09**	91.25 / 91.27	82.35 / 81.39	64.62
Exp3.5	$BERT_6$ [SID] kl + cos + soft	43.93	91.21 / 91.22	81.37 / 79.16	66.43
Exp3.6	$BERT_6$ [SID] kl + cos + soft + hard	42.55	91.20 / 91.21	70.10 / 69.68	**67.51**

Table 1: The development results across four datasets. Experiments 1.0 and 1.1 are trained without any distillation method, whereas experiments 2.0 and 3.X use a different combination of algorithms to distill information. Experiment 2.0 only uses standard knowledge distillation, and it can be considered as baseline.

Exp.	CoLA MCC	QQP Acc. / F1	MRPC Acc. / F1	RTE Acc.
Exp1.0	51.4	71.3 / 89.2	84.9 / 79.9	66.4
Exp2.0	38.3	69.1 / 88.0	81.6 / 73.9	59.7
Exp3.X	41.4	70.9 / 89.1	83.8 / 77.1	62.2

Table 2: The test results from the best models according to the development set. We add Exp1.0 ($BERT_{base}$) for reference. Exp2.0 uses $BERT_6$ with standard distillation (soft-labels only), and Exp3.X uses the best internal KD technique with $BERT_6$ as student according to the development set.

the internal KD results have shown statistical significance with a p-value less than 1e-3 with respect to the standard KD method across the datasets.

Development and Evaluation Results

As shown in Table 1, we perform extensive experiments for $BERT_6$ as a student, where we evaluate different training techniques with or without knowledge distillation. In general, the first thing to notice is that the distillation techniques outperforms $BERT_6$ trained without distillation (Exp1.1). While it is not always the case for standard distillation (Exp1.1 vs. Exp2.0 for CoLA), the internal distillation method proposed in this work consistently outperforms both Exp1.1 and Exp2.0 across all datasets. Nevertheless, the gap between the results substantially depends on the size of the data. Intuitively, this is expected behavior since the more data we provide to the teacher, the more knowledge is exposed, and hence, the student reaches a more accurate approximation of the teacher.

Additionally, our internal distillation results are consistently better than the standard soft-label distillation in the test set, as described in Table .

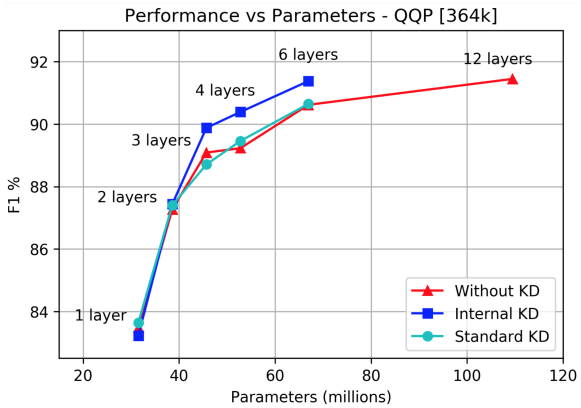

Figure 2: Performance vs. parameters trade-off. The points along the lines denote the number of layers used in BERT, which is reflected by the number of parameters in the x-axis.

Analysis

This section provides more insights into our algorithm based on parameter reduction, data size impact, model convergence, self-attention behavior, and error analysis.

Performance vs. Parameters

We analyze the parameter reduction capabilities of our method. Figure 2 shows that $BERT_6$ can easily achieve similar results than the original $BERT_{base}$ model with 12 transformer layers. Note that $BERT_{base}$ has around 109.4M parameters, which can be broken down into 23.8M parameters related to embeddings and around 85.6M parameters related to transformer layers. The $BERT_6$ student, however, has 43.1M parameters in the transformer layers, which means that the parameter reduction is about 50%, while still performing very similar to the teacher (91.38 F1 vs. 91.45 F1 for QQP, see Table 1). Also, note that the 0.73% F1 drop

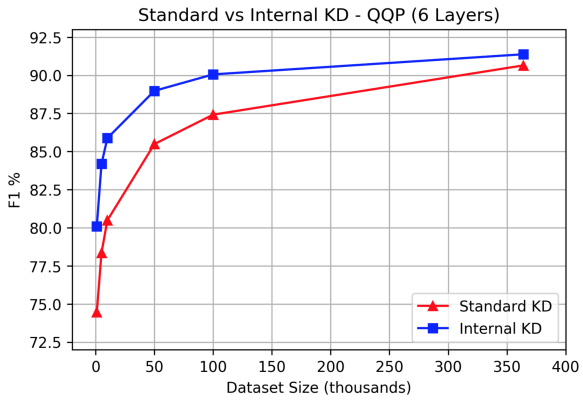

Figure 3: The impact of training size for standard vs. internal KD. We experiment with sizes between 1K and +350K.

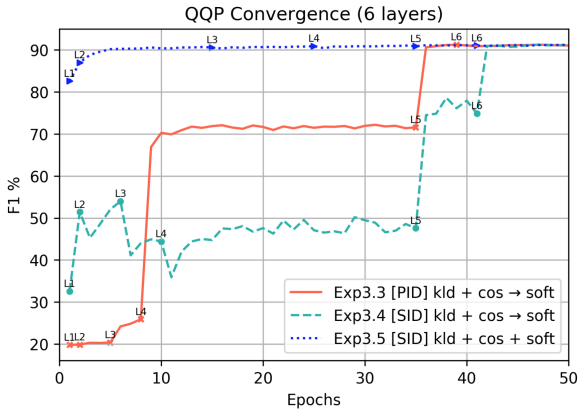

Figure 4: Comparing algorithm convergences across epochs. The annotations along the lines denote the layers that have been completely optimized. After the L6 point, only the classification layer is trained.

is statistical significant between the student only trained on soft-labels and the student trained with our method.

Moreover, if we keep reducing the number of layers, the performance decays for both student models (see Figure 2). However, the internal distillation method is more resilient to keep a higher performance. Eventually, with one transformer layer to distill internally, the compression rate is too high for the model to account for an additional boost when we compare BERT_1 students with standard and internal distillation methods.

The Impact of Data Size

We also evaluate the impact of the data size. For this analysis, we fix the student architecture to the BERT_6, and we only modify the size of the training data. We compare the standard and the internal distillation techniques for the QQP dataset, as shown in Figure 3. Consistently, the internal distillation outperforms the soft-label KD method. However, the gap between the two methods is small when the data size is large, but it tends to increase in favor of the internal KD method when the data size decreases.

Student Convergence

We analyze the convergence behavior during training by comparing the performance of the internal distillation algorithms across epochs. We conduct the experiments on the QQP dataset as described in Figure 4. We control over the student architecture, which is BERT_6, and exclusively experiment with different internal KD algorithms. The figure shows three experiments: *progressive internal distillation* (Exp3.3), *stacked internal distillation* (Exp3.4), and *stacked internal distillation* using soft-labels all the time (Exp3.5). Importantly, note that Exp3.3 and Exp3.4 do not update the classification layer until around epoch 40 when all the transformer layers have been optimized. Nevertheless, the internal distillation by itself allows the students to reach higher performance across epochs eventually. In fact, Exp3.3 reaches its highest value when the 6th transformer layer is being optimized while the classification layer remains as it was initialized (see epoch 38 in Figure 4). This

Method	**Teacher Right (36,967)**		**Teacher Wrong (3,463)**	
Standard KD (Exp2.0)	35,401 ✓	1,566 ✗	1,232 ✓	2,231 ✗
Internal KD (Exp3.2)	36,191 ✓	776 ✗	750 ✓	2,713 ✗

Table 3: Right and wrong predictions on the QQP development dataset. Based on the teacher results, we show the number of right (✓) and wrong (✗) predictions by the students from standard KD (Exp2.0) and internal KD (Exp3.2).

serves as strong evidence that the internal knowledge of the model can be taught and compress without even considering the classification layer.

Inspecting the Attention Behavior

We inspect the internal representations learned by the students from standard and internal KD and compare their behaviors against the ones from the teacher. The goal of this experiment is to get a sense of how much the student can compress from the teacher, and how different such representations are from a student trained on soft-label in a standard KD setting. For this experiment, we use the QQP dataset and BERT_6 as a student. The internally-distilled student corresponds to experiment 3.2, and the soft-label student comes from experiment 2.0 (see Table 1). Figure 5 shows the compression effectiveness of the internally distilled student with respect to the teacher. Even though the model is skipping one layer for every two layers of the teacher, the student is still able to replicate the behavior taught from the teacher. While the internal representations from the student with standard KD mainly serve to a general-purpose (i.e., attending to the separation token while spotting connections with the word college), the representations are not the ones intended to be transferred from the teacher. This means that the original goal of compressing a model does not hold entirely since its internal behavior is quite different than the one from the teacher (see Figure 5 for the KL divergence on each student).

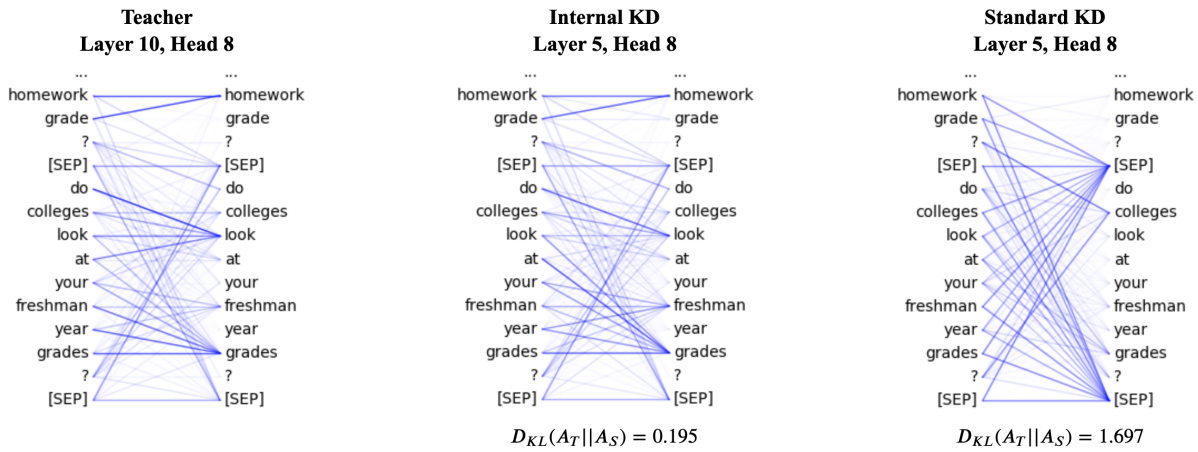

Figure 5: Attention comparison for head 8 in layer 5, each student with its corresponding head KL-divergence loss. The KL-divergence loss for the given example across all matching layers between the students and the teacher is 2.229 and 0.085 for the standard KD and internal KD students, respectively.

No.	QQP Development Samples	Class	Teacher	Std KD	Int KD
1	Q1: if donald trump loses the general election, will he attempt to seize power by force claiming the election was fraudulent? Q2: how will donald trump react if and when he loses the election?	1	1 (0.9999)	1 (0.9999)	0 (0.4221)
2	Q1: can depression lead to insanity? Q2: does stress or depression lead to mental illness?	0	0 (0.0429)	0 (1.2e-4)	1 (0.9987)
3	Q1: can i make money by uploading videos on youtube (if i have subscribers)? Q2: how do youtube channels make money?	1	1 (0.9998)	0 (0.0017)	1 (0.8868)
4	Q1: what are narendra modi's educational qualifications? Q2: why is pmo hiding narendra modi's educational qualifications?	0	0 (0.0203)	1 (0.9999)	0 (0.2158)

Table 4: Samples where the teacher predictions are right and only one of the students is wrong. We show the predicted label along with the probability for such prediction in parenthesis. We also provide the ground-truth label in the class column.

Error Analysis

In our internal KD method, the generalization capabilities of the teacher are replicated in the student model. This also implies that the student will potentially make the mistakes of the teacher. In fact, when we compare a student only trained on soft-labels (Exp2.0) against a student trained with our method (Exp3.2), we can see in Table 3 that the numbers of the latter align better with the teacher numbers for both wrong and right predictions. For instance, when the teacher is right (36,967), our method is right 97.9% of the same samples (36,191), whereas the standard distillation provides a rate of 95.7% (35,401) with more than twice the number of mistakes than our method (1,566 vs. 776). On the other hand, when the teacher is wrong (3,463), the student in our method makes more mistakes and provides less correct predictions than the student from standard KD. Nevertheless, the overall score of the student in our method significantly exceeds the score from the student trained in a standard KD setting.

We also inspect the samples where the teacher and only one of the students are right. The QQP samples 1 and 2 in Table 4 show wrong predictions by the internally-distilled student (Exp3.2) that are not consistent with the teacher. For sample 1, although the prediction is 0, the probability output (0.4221) is very close to the threshold (0.5). Our intuition is that the internal distillation method had a regularization effect on the student such that, considering that question 2 is much more specific than question 1, it does not allow the student to tell whether is similar or not confidently. Also, it is worth noting that standard KD student is extremely confident about the prediction (0.9999), which may not be ideal since this can be a sign of over-fitting or memorization. For sample 2, although the internally-distilled student is wrong (according to ground-truth annotation and the teacher), the questions are actually related which suggests that the student model is capable of disagreeing with the teacher while still generalizing well. Samples 3 and 4 show successful cases for the internally-distilled student, while the standard KD student fails.

Conclusions

We propose a new extension of the KD method that effectively compresses a large model into a smaller one, while still preserving a similar performance from the original

model. Unlike the standard KD method, where a student only learns from the output probabilities of the teacher, we teach our smaller models by also revealing the internal representations of the teacher. Besides preserving a similar performance, our method effectively compresses the internal behavior of the teacher into the student. This is not guaranteed in the standard KD method, which can potentially affect the generalization capabilities initially intended to be transferred from the teacher. Finally, we validate the effectiveness of our method by consistently outperforming the standard KD technique in four datasets of the GLUE benchmark.

References

Bengio, Y. 2009. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning* 2(1):1–127.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019a. What Does BERT Look At? An Analysis of BERT's Attention. *CoRR* abs/1906.04341.

Clark, K.; Luong, M.; Khandelwal, U.; Manning, C. D.; and Le, Q. V. 2019b. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. *CoRR* abs/1907.04829.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or 1. *arXiv preprint arXiv:1602.02830*.

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)*.

Dror, R.; Baumer, G.; Shlomov, S.; and Reichart, R. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1383–1392. Association for Computational Linguistics.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149*.

He, Q.; Wen, H.; Zhou, S.; Wu, Y.; Yao, C.; Zhou, X.; and Zou, Y. 2016. Effective Quantization Methods for Recurrent Neural Networks. *CoRR* abs/1611.10176.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *The Journal of Machine Learning Research* 18(1):6869–6898.

Lample, G., and Conneau, A. 2019. Cross-lingual Language Model Pretraining. *CoRR* abs/1901.07291.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019a. Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding. *arXiv preprint arXiv:1904.09482*.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019b. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4487–4496. Florence, Italy: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019c. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692.

Mirzadeh, S.-I.; Farajtabar, M.; Li, A.; and Ghasemzadeh, H. 2019. Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. *arXiv preprint arXiv:1902.03393*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving Language Understanding by Generative Pre-Training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* 1(8).

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. FitNets: Hints for Thin Deep Nets. *arXiv preprint arXiv:1412.6550*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4314–4323. Hong Kong, China: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *CoRR* abs/1706.03762.

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. *CoRR* abs/1804.07461.

Warstadt, A.; Singh, A.; and Bowman, S. R. 2018. Neural Network Acceptability Judgments. *CoRR* abs/1805.12471.

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*.