Universal-KD: Attention-based Output-Grounded Intermediate Layer Knowledge Distillation

Yimeng Wu[1], **Mehdi Rezagholizadeh**[1], **Abbas Ghaddar**[1],
Md Akmal Haidar[1], **Ali Ghodsi**[2,3]
[1]Huawei Noah's Ark Lab
[2]David R. Cheriton School of Computer Science, University of Waterloo
[3]Department of Statistics and Actuarial Science, University of Waterloo
{yimeng.wu,mehdi.rezagholizadeh,abbas.ghaddar,md.akmal.haidar}@huawei.com
ali.ghodsi@uwaterloo.ca

Abstract

Intermediate layer matching is shown as an effective approach for improving knowledge distillation (KD). However, this technique applies matching in the hidden spaces of two different networks (i.e. student and teacher), which lacks clear interpretability. Moreover, intermediate layer KD cannot easily deal with other problems such as layer mapping search and architecture mismatch (i.e. it requires the teacher and student to be of the same model type). To tackle the aforementioned problems all together, we propose Universal-KD to match intermediate layers of the teacher and the student in the output space (by adding pseudo classifiers on intermediate layers) via the attention-based layer projection. By doing this, our unified approach has three merits: (i) it can be flexibly combined with current intermediate layer distillation techniques to improve their results (ii) the pseudo classifiers of the teacher can be deployed instead of extra expensive teacher assistant networks to address the capacity gap problem in KD which is a common issue when the gap between the size of the teacher and student networks becomes too large; (iii) it can be used in cross-architecture intermediate layer KD. We did comprehensive experiments in distilling BERT-base into BERT-4, RoBERTa-large into DistilRoBERTa and BERT-base into CNN and LSTM-based models. Results on the GLUE tasks show that our approach is able to outperform other KD techniques.

1 Introduction

Despite the great success of deep neural networks in different tasks such as computer vision (Huang et al., 2018; Lawrence et al., 1997) and natural language processing (NLP) (Liu et al., 2019; Devlin et al., 2019; Radford et al., 2018; Vaswani et al., 2017) due to huge over-parameterization, when it comes to deploying these models for the end user or on edge devices with limited memory and computational power, this over-parameterization can be very prohibitive. Therefore, different neural model compression techniques such as quantization (Gong et al., 2014; Prato et al., 2019), pruning (Han et al., 2015), layer decomposition (Mondelli and Montanari, 2019) and knowledge distillation (KD) (Buciluǎ et al., 2006; Hinton et al., 2015) aim at reducing the number of parameters of the models, improving their memory requirements or running efficiency.

KD is one of the most well-known neural model compression techniques. KD provides a way to compress a large model (so-called the teacher model) into a small model (i.e. the student model). KD has shown being reliable in reducing the number of parameters and computations while achieving competitive results on downstream tasks. Recently, KD has attracted more attention in the NLP field especially due to large pre-trained language models (PLMs) (Sanh et al., 2019; Sun et al., 2019b; Jiao et al., 2019). However, it is evident that the original KD is not performing very well in maintaining the performance of compressed PLMs and it is required to be equipped with other auxiliary training objectives (Jiao et al., 2019; Sun et al., 2019b). Jiao et al. (2019) refers to a few of these auxiliary techniques such as Intermediate Layer Distillation (ILD) and data augmentation. MobileBERT (Sun et al., 2020) deploys a progressive layer-wise training. Jafari et al. (2021) proposes a two-stage gradual training by using a dynamic temperature factor during training to address the common capacity gap problem (i.e. training with KD becomes more difficult when the capacity gap between the teacher and student networks become too large) in KD. Sun et al. (2019b) and Passban et al. (2020) show the importance of ILD and try to improve the search and skip problems in ILD, respectively.

Among all auxiliary techniques, the focus of this paper is on ILD as a promising approach in improving KD. Existing ILD techniques are established on

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7649–7661
November 7–11, 2021.

raw intermediate representations. We argue that the physical interpretation of these raw representations are not known. Currently in most of ILD techniques, distilling these representations from one network to another is done arbitrarily and without any particular reasoning. Performing ILD in this form leads to other weaknesses such as requiring layer mapping search and requiring the architecture of the two networks to be the same. To address these limitations, we provide a universal solution which maps the raw hidden representations of each layer to the output space using so-called intermediate pseudo classifiers. We demonstrate in this work that our Universal-KD technique leads to the following contributions:

1. Universal-KD provides an interpretable ILD technique which can be flexibly combined with other ILD methods to improve their results.

2. Universal-KD is extended to solve the capacity gap problem without requiring any extra teacher assistant network and outperforms corresponding state-of-the-art models.

3. Universal-KD is appropriate for intermediate layer cross-architecture distillation and to the best of our knowledge, it is the first time ILD is able to show improvement in this setting on NLP tasks.

2 Background

KD adds a new loss term (KD loss) to the regular Cross Entropy (CE) classification loss function. KD loss aims in pushing the output of the student model to follow that of the teacher.

$$\mathcal{L} = \alpha\mathcal{L}_{CE} + (1-\alpha)\mathcal{L}_{KD}$$
$$\mathcal{L}_{CE}(x, y; \theta) = CE(y, S(x; \theta))$$
$$\mathcal{L}_{KD}(x, y; \theta) = \mathcal{T}^2\ KL\Big(\sigma(\frac{z_t(x;\phi)}{\mathcal{T}}), \sigma(\frac{z_s(x;\theta)}{\mathcal{T}})\Big) \quad (1)$$

where α is a hyper-parameter to adjust the contribution of the KD loss vs. the CE loss, $\sigma(.)$ is the *softmax* function, z_t and z_s are the teacher and student logits, $S(x;\theta) = \sigma(z_s(x;\theta))$ shows the student output class probabilities, $\mathcal{T}$ indicates the temperature factor which controls the smoothness of the output probabilities of the two networks, θ and ϕ refer to the parameters of the teacher and student networks respectively, and $KL(\cdot,\cdot)$ is the KL divergence loss.

3 Related Work

Recent years have seen a wide array of methods that leverage intermediate layer matching (Wu et al., 2020; Passban et al., 2020; Ji et al., 2021), data augmentation (Fu et al., 2020; Jiao et al., 2019; Kamalloo et al., 2021), adversarial training (Zaharia et al., 2021; Rashid et al., 2020, 2021), lately loss terms re-weighting (Clark et al., 2019; Zhou et al., 2021; Jafari et al., 2021) in order to reduce the teacher-student performance gap. In this section, we introduce the related work to ILD, the capacity gap problem, and intermediate layer cross-architecture distillation. You can find a comprehensive comparison of Universal-KD and related works in Table 1.

3.1 Intermediate Layer KD

Sun et al. (2019b) found that apart from the teacher's final prediction, the student can also benefit from the internal components of the teacher. Therefore, they proposed PKD to extract such knowledge from the hidden layers at the fine-tuning stage. In Attention-based Layer Projection KD (ALP-KD), Passban et al. (2020) raised two problems in ILD techniques including PKD, when the number of teacher layers N are more than that of the student M: the *skip* problem and the *search* problem. The *skip* problem refers to the issue that when $N > M$, multiple layers of the teacher might be ignored in the ILD process which can lead to loss of information. The *search* problem indicates that for $N > M$, finding the best layer mapping requires a tedious search process. To address these problems, ALP-KD uses an attention mechanism between each layer of the student and all layers of the teacher. This way, no teacher layer is skipped during distillation and hence the skip problem and the search problem are solved. Aside from the above two problems, ILD is usually performed in the hidden space of intermediate layers which lacks clear interpretation. In other words, it is hard to assign physical meanings to these hidden representations which makes it difficult to justify how their distillation works. We refer to this problem as interpretability problem in ILD. Multi-head KD (MHKD) (Wang et al., 2020) proposed a solution to address this problem by adding auxiliary classifier heads to intermediate layers. Therefore, the matching between the teacher and student intermediate layers can be done in the output space which is more interpretable. However, MHKD suffers

KD Methods	Intermediate Layer KD (ILD)			Capacity Gap	Cross Architecture Distillation		Task
	Layer Mapping	Matching Space	Interpretable	Solution	ILD	Architectures	
PKD (Sun et al., 2019b)	Arbitrary (Search)	Hidden	✘	N/A	-	-	NLP
ALP (Passban et al., 2020)	Attention (No Search)	Hidden	✘	N/A	-	-	NLP
MHKD (Wang et al., 2020)	Arbitrary (Search)	Output	✓	N/A	-	-	CV
TAKD (Mirzadeh et al., 2020)	-	-	-	Multiple TAs	-	-	CV
Annealing KD (Jafari et al., 2021)	-	-	-	Annealing Loss	-	-	CV/NLP
DGKD (Son et al., 2020)	-	-	-	Multiple TAs	-	-	CV
DIH (Asadian and Salehi-Abari, 2021)	Arbitrary (Search)	Output	✓	ILD	-	-	CV
XtremeDistil (Mukherjee and Awadallah, 2020)	Arbitrary (Search)	Hidden	✘	N/A	✓	BERT → LSTM	NLP
Universal-KD (Ours)	Attention (No Search)	Output	✓	ILD	✓	BERT → LSTM / CNN	NLP

Table 1: Comparing Universal-KD with different related work which shows generality of our technique.

from the skip and search problem, and it is done only on computer vision tasks. We will explain the benefits of our Universal-KD over MHKD in the Section 4.1 in more detail.

3.2 Capacity Gap Problem in KD

It has been observed that larger teachers do not necessarily lead to better distillation results, especially when the size of the teacher is much larger than the size of the student (Mirzadeh et al., 2020). This problem is referred to as the *capacity gap* in KD (Jafari et al., 2021). Teacher assistants KD (TAKD) (Mirzadeh et al., 2020) alleviated this gap by incorporating some intermediate size TA networks (larger than the student but smaller than the teacher) in the distillation process. Son et al. (2020) pointed out that TAKD suffers from error accumulation when one of the TAs transfers wrong knowledge to the next component. Thus, they proposed Densely Guided KD (DGKD) (Son et al., 2020) to employ all TAs and the teacher to guide the model simultaneously. However, training multiple TA networks in above techniques can be prohibitive in the real world especially when dealing with PLMs. Jafari et al. (2021) came up with Annealing KD which does not need any extra TA for its training. They let the teacher gradually generate annealed soft targets at different temperatures and let the student follow the annealed output of the teacher in the meantime. Asadian and Salehi-Abari (2021) proposed another TA-Free Distillation via Intermediate Heads (DIH) to reduce the computational cost. They attach auxiliary classifier heads to some selected teacher layers, and the output of these classifiers are used to guide the final student classifier. However, in contrast to our model, the selection of which layers to add classifiers is arbitrary, each classifier contributes equally to the distillation, and they only explored computer vision tasks.

3.3 Intermediate Layer Cross-Architecture Distillation

The increase of computation resources required by Transformer-based (Vaswani et al., 2017) NLP models (Devlin et al., 2019; Liu et al., 2019) has motivated researchers to consider more efficient student architectures. This type of knowledge transfer is referred as *cross-architecture* KD. While we can find a few solutions for cross-architecture distillation in the literature (Tang et al., 2019; Kaliamoorthi et al., 2021), intermediate layer cross-architecture distillation has not yet been much explored due to the interpretability problem in ILD. We could only find (Mukherjee and Awadallah, 2020) on multilingual Named Entity Recognition KD which transfers knowledge from a BERT-base (Devlin et al., 2019) teacher to an LSTM student through ILD. However, their ILD is not performed in the output space (not interpretable), does not consider the search and skip problem, and does not show any improvement on the results using ILD. Therefore, their work might not be directly comparable to ours.

4 Methodology

Methodology of our work concerns introducing our Universal-KD solution and showing how this technique can address the main discussed problems in KD which are interpretability of ILD, the capacity gap issue and intermediate layer cross-architecture distillation.

4.1 Universal-KD

In the regular ILD (Sun et al., 2019b), layer matching is performed in the representation space (e.g. the CLS output representation of each layer for BERT-based models or the average representation of the entire sequence). This representation matching is usually done using the MSE loss over the

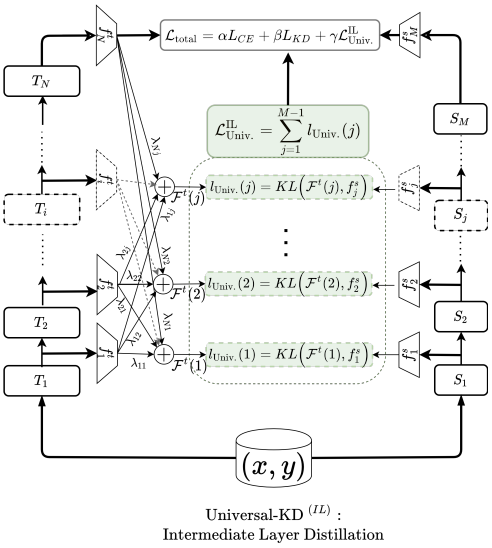

Figure 1: Our Universal-KD in the ILD setting with the pseudo classifiers and attention-based layer projection

normalized representations of each layer from the teacher and student networks:

$$\text{Sim}(h^t_i, h^s_j) = \left\| \frac{h^t_i}{||h^t_i||_2} - \frac{h^s_j}{||h^s_j||_2} \right\|^2_2 \qquad (2)$$

where $\text{Sim}(\cdot, \cdot)$ refers to the similarity function between two input vectors, h^t_i and h^s_j indicate the hidden representation vector of the i^{th} layer of the teacher and j^{th} layer of the student, respectively. However, the physical meaning of these hidden space representations of different layers (i.e. meaningful features of the input data which are encoded by hidden space representation of different layers) of each network might not be very clear, and consequently finding a proper layer distillation projection (which distils similar intermediate features) for two networks would be quite challenging.

To address this problem, we propose our Universal-KD based on layer matching in the output space rather than the hidden space level. In other words, we suggest mapping the intermediate hidden layer representations to the output space by applying some pseudo classifiers on the corresponding intermediate layers before performing distillation. After applying the pseudo classifiers, we can get the output distributions over classes and measure the similarity of the hidden space representations in the output space using the KL divergence as following:

$$\begin{aligned} &\text{Sim}_{\text{Univ.}}(f^t_i(h^t_i), f^s_j(h^s_j)) = \\ &KL\Big(\sigma(W^t_i h^t_i), \sigma(W^s_j h^s_j)\Big) \end{aligned} \qquad (3)$$

where $\text{Sim}_{\text{Univ.}}(\cdot, \cdot)$ is the universal similarity measure, $f^t_i(h^t_i) = \sigma(W^t_i h^t_i)$ and $f^s_j(h^s_j) = \sigma(W^s_j h^s_j)$ are the pseudo classifiers applied to the i^{th} layer of the teacher and j^{th} layer of the student respectively. Moreover, $\sigma(.)$ refers to the *softmax* function. W^t_i and W^s_j are the corresponding weight matrices of the pseudo classifiers applied to the i^{th} layer of the teacher and j^{th} layer of the student respectively. Bear in mind that W^t_i can be pre-trained in a warm-up phase with the ground-truth training data when the weights of the pre-trained teacher model are kept frozen. For the student, the weights of pseudo classifiers are trained during the Universal-KD training.

Inspired by ALP-KD (Passban et al., 2020) , Universal-KD applies attention-based layer projection on top of pseudo classifier outputs. For the intermediate KD setup, we apply pseudo classifiers on top of all layers of the student and teacher networks. Then, for each pseudo classifier of the student we find weighted prediction of teacher's pseudo classifiers, $\mathcal{F}^t(j)$, to be used for distillation to the j^{th} pseudo classifier of the student. The weights of this summation are derived from the attention weights calculated according to the similarity values between the outputs of each pseudo classifiers of the student and all pseudo classifiers of the teacher:

$$\lambda_{ij} = \frac{\exp(f^t_i(h^t_i).f^s_j(h^s_j))}{\sum_{i=1}^{N} \exp(f^t_i(h^t_i).f^s_j(h^s_j))} \qquad (4)$$

where λ_{ij} describes the attention weight of the j^{th} student layer to the i^{th} teacher layer. Then, the aggregated prediction of teacher's pseudo classifiers, $\mathcal{F}^t(j)$, to be used for distillation to the j^{th} layer of the student can be calculated as:

$$\mathcal{F}^t(j) = \sum_{i=1}^{N} \lambda_{ij} f^t_i(h^t_i). \qquad (5)$$

where N is the total number of layers of the teacher. The Universal-KD loss for each layer of the student can be calculated as following:

$$\begin{aligned} l_{\text{Univ.}}(j) &= \text{Sim}_{\text{Univ.}}(\mathcal{F}^t(j), f^s_j(h^s_j)) \\ &= KL\Big(\mathcal{F}^t(j), f^s_j(h^s_j)\Big) \end{aligned} \qquad (6)$$

In the following, we explain how our Universal-KD loss can be set up for ILD, capacity gap problem and cross-architecture problems.

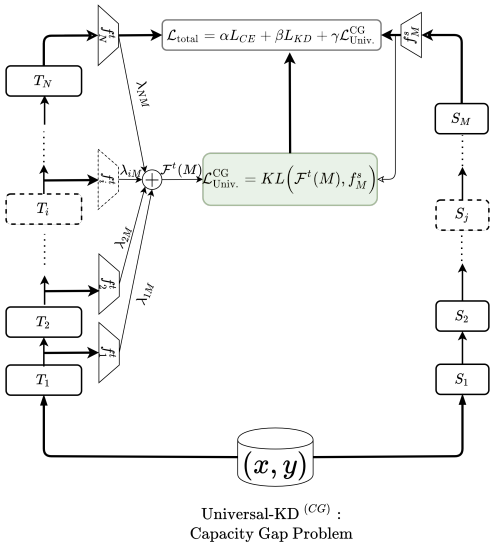

Figure 2: Universal-KD to solve the capacity gap problem. Each pseudo classifier acts as a pseudo TA to fill the capacity gap between the two networks.

Intermediate Layer Distillation In ILD, we have an additional loss term in KD training for matching intermediate representations of the two networks. In our Universal-KD we add this intermediate loss in the output space and we calculate it for $M-1$ intermediate layers of the student as follows (see Fig. 1):

$$\mathcal{L}^{\text{IL}}_{\text{Univ.}} = \sum_{j=1}^{M-1} l_{\text{Univ.}}(j). \tag{7}$$

Bear in mind that our work can be distinguished from MHKD (Wang et al., 2020) in different points: first, in contrast to MHKD which only applies pseudo classifiers to some arbitrarily selected layers of the teacher and skip the others, Universal-KD applies pseudo classifiers to all the teacher and student layers to avoid the loss of information and to avoid the burden of selecting the best layer mapping strategy; second, we have an attention mechanism in the process of intermediate pseudo classifiers distillation; third, unlike MHKD which applies CE loss to the training of student's pseudo classifiers, our Universal-KD avoids using the CE loss for training the student pseudo classifiers to prevent them from over-fitting ; fourth, MHKD only investigates the ILD setting in computer vision tasks, but our Universal-KD deploys the output-grounded ILD to solve a more comprehensive set of problems in KD such as the capacity gap problem and intermediate layer cross architecture distillation in NLP.

Capacity Gap Problem We can apply Universal-KD to address the capacity gap problem. In this regard, instead of training excessive TA networks, which can be really prohibitive for large PLMs, we create pseudo TAs by applying pseudo classifier heads to the intermediate layers of the teacher. Each teacher's pseudo classifier will play the role of a TA (which we refer to as a pseudo TA) to fill the capacity gap for us. Then, we can distill the aggregated pseudo classifiers of the teacher into the last layer of the student:

$$\mathcal{L}^{\text{CG}}_{\text{Univ.}} = l_{\text{Univ.}}(M) \tag{8}$$

This setting is equivalent to distilling from multiple TAs at the same time (see Fig. 2). But unlike DIH (Asadian and Salehi-Abari, 2021), we do not have a uniform distillation from the pseudo TAs. Instead, it is the output layer of the student determines how much we should attend to each pseudo TA.

Cross-Architecture Distillation In the cross-architecture setting, the building blocks of the teacher and student network are different. In this case, since the output of the pseudo classifiers are not depending to the architecture, the Universal-KD loss is similar to the ILD scenario:

$$\mathcal{L}^{\text{CA}}_{\text{Univ.}} = \sum_{j=1}^{M-1} l_{\text{Univ.}}(j) \tag{9}$$

We need to add a remark here that without the pseudo classifiers, the intermediate hidden representations of two architecturally different networks are not grounded, and hence matching them would not be meaningful.

4.2 Training Process

By combining the Universal loss with the KD loss and CE loss which are described in Eq. 1, we can obtain the final objective function for distilling the teacher into the student model:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{CE} + \beta \mathcal{L}_{KD} + \gamma \mathcal{L}_{\text{Univ.}} \tag{10}$$

where α, β and γ are hyper-parameters. It is worth mentioning that we apply a two-stage training to avoid extensive hyper-parameter tuning. In the first stage, the model is only guided by the Universal-KD loss and KD loss. α is set to 0, β is selected from $\{0.2, 0.5, 0.7\}$, and γ is set to 1-β in the first

stage. For the second stage, we only train the model on the CE loss so the α is set to 1 and other two hyper-parameters are set to 0 in stage two.

5 Experiments

5.1 Datasets

We evaluate our proposed approach on eight GLUE datasets (Wang et al., 2018) which include both classification and regression tasks. These datasets are used to validate the performance of three tasks: single sentence, similarity and paraphrasing, and inference. Single sentence tasks refer to linguistic acceptability (CoLA (Warstadt et al., 2018)) and sentiment analysis (SST-2 (Socher et al., 2013)). Similarity and paraphrasing tasks consist of paraphrasing (MRPC (Dolan and Brockett, 2005) and QQP (Chen et al., 2018)) and a regression task STS-B (Cer et al., 2017). Inference tasks compose of Natural Language Inference (MNLI (Williams et al., 2018), RTE (Bentivogli et al., 2009)) and Question Answering (QNLI (Rajpurkar et al., 2016)).

5.2 Experimental Setup

We first test our method Universal-KD$^{(IL)}$ for intermediate layer knowledge distillation. Our teacher is a standard 12-layer BERT-base model (Devlin et al., 2019) with 12 heads and the hidden and feed-forward dimensions are 768 and 3072 respectively. Our students are also BERT models with 4 layers, while other configurations are the same as the BERT-base model. To make fair comparisons with PKD and ALP, we utilize the first 4 layers of the pre-trained BERT-base teacher to initialize the students. During training, we run a grid search over batch size ($\{8, 16, 32\}$), learning rate ($\{2e-5, 5e-5\}$) as well as β ($\{0.2, 0.5, 0.7\}$) for each task to find the best hyper-parameters.

In the capacity gap experiment, we want to verify whether our proposed solution Universal-KD$^{(CG)}$ for the capacity gap problem is effective especially when the size of the teacher and the student differs a lot. Thus, we choose Roberta-large (Liu et al., 2019) as our teacher and Distilroberta as our student model. Our teacher has 24 layers with a hidden dimension of 1024, 16 attention heads and 355M parameters in total. Our student is a 6-layer model with a hidden dimension of 768, 8 attention heads and a total of 82M parameters. The selection of hyper-parameters and the hyper-parameter search strategy is exactly the same with the first experiment.

For all experiments discussed in Section 5.2, after the teacher is trained on the task, we train the pseudo classifiers all together through the cross entropy loss while the teacher model is frozen. It is a common technique to train pseudo classifiers (Asadian and Salehi-Abari, 2021; Xin et al., 2020). The student is first trained for 20 epochs in stage 1 for all datasets except for CoLA. CoLA needs 50 epochs to get high-quality results. Then, the model is further trained for 10 epochs in stage 2.

We also conduct two more experiments under the cross-architecture setting. Our teacher is a strong BERT-base model (Devlin et al., 2019) while the student is Bi-LSTM and Gated CNN, respectively. In this experiment, we want to verify if our method is still helpful when the student and teacher are of different types. We set without KD and Vanilla KD as baselines for all experiments. For all dev set results, we report Matthew's Correlations for CoLA, Pearson correlations for STS-B and accuracy scores for all other datasets.

5.3 Results of Intermediate Layers KD

We compare our Universal-KD$^{(IL)}$ with three intermediate layer KD methods that are described in Section 3.1: PKD, MHKD and ALP. All the results of 4-layer BERT-base students on the GLUE dev set and test set are summarized in Table 2 and 3. We observe that Universal-KD$^{(IL)}$ outperforms all other ILD methods on both the dev set and test set, which shows the superiority of incorporating the attention mechanism with the output space distillation. Moreover, compared with MHKD and ALP, the attention mechanism and the output space matching in Universal-KD$^{(IL)}$ gives 0.7% and 0.5% performance improvement respectively on average.

5.4 Capacity Gap

Here, we evaluate Universal-KD$^{(CG)}$ compared to the three baselines of the capacity gap problem: TAKD, DIH and Annealing KD. Considering the computational cost, the TA used in TAKD is a single RoBERTa-base model with 12 layers. We present the DistilRoBERTa student results on GLUE dev set in Table 4. Surprisingly, We find that TAKD has a similar performance to vanilla KD, which means that without multiple well-designed TA networks, TAKD can hardly fill the gap between the teacher and the student. DIH performs better than TAKD with an acceptable margin, which is expected since DIH distills from multiple interme-

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
BERT-base	57.3	83.4	86.8	91.3	91	68.2	92.7	88.9	82.4
BERT-4	31	76.8	77.7	85.1	89	61.7	88.2	87.3	74.6
Vanilla KD	29.2	79.3	79.4	86.8	90.3	65.3	90.4	87.5	76
PKD†	32.1	79.3	80.2	86.6	90.2	65.7	90.1	87.3	76.4
MHKD⋆	32.8	79.4	80.6	86.8	90.1	66.4	90.5	87.5	76.8
ALP†	33.1	**79.6**	80.7	87	90.5	67.2	90.4	87.6	77
Universal-KD$^{(IL)}$	**34.2**	**79.6**	**81**	**87.1**	**90.7**	**67.9**	**90.6**	**87.9**	**77.4**

Table 2: 4-layer BERT-base student results on GLUE dev set. † denotes the results are taken from (Passban et al., 2020). ⋆ denotes we reproduce this baseline since it is original proposed for CV tasks.

Model	CoLA	MNLI-m/mm	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
PKD	25.4	79.1/78.5	82.1/74.9	86	**69.2/88.6**	61.4	**90.2**	82/80.4	72.6
MHKD	23.4	79.3/78.7	83.8/76.6	**86.5**	68.8/88.5	**63.3**	89.8	82/80.3	72.8
ALP	**27.2**	79.2/78.7	83.2/76.2	86.1	69.1/88.5	61.5	89.3	83/81.8	73
Universal-KD$^{(IL)}$	27	**79.3/78.9**	**84.1/77.2**	86.3	68.9/88.5	62.6	**90.2**	**83.7/82.8**	**73.5**

Table 3: 4-layer BERT-base student results on GLUE test set.

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
RoBERTa-large	68.1	90.2	89	94.6	91.5	86.3	96.4	92.3	88.6
DistilRoBERTa	59.3	84	84.3	90.8	90.9	67.9	92.5	88.5	82.3
Vanilla KD	61	84.2	86.5	91.4	91.7	71.1	92.54	88.9	83.4
TAKD†	61.2	83.9	85.8	91.3	**91.7**	71.8	92.5	89	83.4
DIH⋆	61.5	84.5	86.8	91.5	91.1	72.6	92	88.7	83.6
Annealing KD†	61.7	**85.3**	87.3	**91.6**	91.5	73.6	93.1	89	84.2
Universal-KD$^{(CG)}$	**63**	83.9	**87.8**	91.6	**91.7**	**74**	**93.5**	**89.9**	**84.4**

Table 4: 6-layer DistilRoBERTa student results on GLUE dev set. † denotes the results are taken from (Jafari et al., 2021). ⋆ denotes we reproduce this baseline since it is original proposed for CV tasks.

	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
Teacher	BERT-base	57.3	83.4	86.8	91.3	91.0	68.2	92.7	88.8	82.4
BiLSTM	w/o KD	22	68.6	**70.8**	65.1	84.4	55.6	86.5	18.5	58.9
	Vanilla KD	24.5	69.1	69.4	64.5	84.8	56.3	86.1	20.2	59.4
	Universal-KD$^{(CA)}$	**25.7**	**69.3**	70.6	**65.2**	**85**	**56.5**	**86.9**	**22.1**	**60.2**
Gated-CNN	w/o KD	21.3	58.4	70.6	61.9	81.7	53.4	88	21.8	57.1
	Vanilla KD	20.6	58.7	69.6	62.2	82	54.1	88.8	21.6	57.2
	Universal-KD$^{(CA)}$	**24.5**	**59.1**	**70.8**	**62.4**	**82.2**	**56**	**89**	**24.9**	**58.6**

Table 5: Performances of Bi-LSTM and Gated-CNN students on GLUE dev sets when BERT-base is used as teacher. For each student architecture, we report the scores of models without KD, with vanilla KD, and with Universal-KD.

diate classifiers at the same time. However, DIH just employs a uniform distillation without considering the contribution of each teacher layer to the final distillation. This explains why DIH has worse results compared to Universal-KD$^{(CG)}$. Annealing KD is the current state-of-the-art method for solving the capacity gap problem, and it works better than the two previous baselines. It is worth noting that our Universal-KD can outperform Annealing KD by 0.2% without any temperature adjustment.

5.5 Cross-Architecture KD

In this section, we test the validity of our approach in the intermediate layer cross-architecture KD (Mukherjee and Awadallah, 2020; Kaliamoorthi et al., 2021) setting. We experiment with Bi-LSTM (Hochreiter and Schmidhuber, 1997) and Gated CNN (Dauphin et al., 2017; Ghaddar and Langlais, 2019) as students, while the teacher is a BERT-base model. The encoder of the LSTM and CNN models have 3 and 4 layers, respectively, with a hidden size of 768 and both models have simi-

lar capacity of around 25M trainable parameters. At the output layer, the attention sum (Bahdanau et al., 2015) mechanism is used to reduce the encoded sequence into a single vector, which in turn is projected to the output space. Hyper-parameter details are listed in Appendix A. Table 5 reports the performance on GLUE dev sets for Bi-LSTM and Gated-CNN student models without KD, with vanilla KD and Universal-KD$^{(CA)}$. First, we notice that students without KD perform roughly 20% less compared to their counterparts in Table 2. This is primarily due to the superiority of pre-trained and fine-tuning approach over feature-based one (Devlin et al., 2019; Sun et al., 2019a; Ghaddar et al., 2021b,a), and partially due to the use of the single encoder for sentence pair tasks like STS-B, QNLI and MNLI (For more detail please refer to Appendix). Second, we observe that LSTM students outperform CNN ones by roughly 1.6% on average. It is worth mentioning that in our experiments, CNN models were 3 times faster to train due to their parallelism ability (Strubell et al., 2017).

Expectedly, vanilla KD improves the performance of both Bi-LSTM and Gated-CNN models by an average of 0.5% and 0.1%, respectively. Our Universal-KD significantly improves the performances of Vanilla-KD by 0.8% and 1.4% for Bi-LSTM and Gated-CNN, respectively.

5.6 Analysis

Attention on logits v.s. attention on losses .

Attention weights could either be used to aggregate teacher classifier outputs (our Universal-KD$^{(CG)}$ setting) or weigh the loss function. For the latter, the loss function $L^2_{\text{Univ.}}$ could be expressed as $L^2_{\text{Univ.}} = \sum_{i=1}^{M} \lambda_{iM} KL(f_i^t(h_i^t), f_M^s(h_M^s))$. We conduct one experiment on two small datasets (MRPC, RTE), one medium dataset (SST-2) and one large dataset (QQP) to verify which way of utilizing attention weights is more beneficial. All the settings are the same with Universal-KD$^{(CG)}$ except we use $L^2_{\text{Univ.}}$ rather than $L^{\text{CG}}_{\text{Univ.}}$. We call this variant as Universal-KD$^{\blacktriangle}$. Table 6 shows the dev results. To justify the effect of attention weights, We also include the results of DIH which does not weigh losses. First, we observe that Universal-KD$^{\blacktriangle}$ has better results than DIH, which indicates that different teacher layers contribute differently, and the attention score reflects the importance of each teacher layer. Second, on all datasets, our original Universal-KD$^{(CG)}$ outperforms the weighted average of losses by attention weights. This suggests that adding the attention model on logits gains extra benefits.

Model	MRPC	RTE	SST-2	QQP
DIH	86.8	72.6	92	91.1
Universal-KD$^{\blacktriangle}$	87.1	72.6	92.4	91.4
Universal-KD$^{(\text{CG})}$	**87.8**	**74**	**93.5**	**91.7**

Table 6: Dev results for attention on logits and attention on losses.

Output space v.s. hidden space To verify if the attention layer projection works better in the output space or the hidden space, we visualize the attention weights of ALP and Universal-KD$^{(IL)}$. Since the concrete importance degree of each teacher layer is hard to measure, we treat the Euclidean distance between real labels and classifier outputs as a reference measure for our student models. We randomly select 10 samples from the RTE dataset, and demonstrate the importance levels of each layer of the BERT-base teacher in the left figure of Fig. 3. The x and y axes represent 12 teacher layers and 10 samples respectively. It is clearly shown that the last two layers can generate more accurate predictions close to the true label, thus later layers of the student are expected to focus more on these layers of the teacher. In the middle and right figures of Fig. 3, we show attention weights between the third (penultimate) layer of the 4-layer Bert-base student (i.e. trained by ALP and Universal-KD$^{(IL)}$, respectively) and all 12 teacher layers. From Fig. 3, we observe that the penultimate student layer in Universal-KD$^{(IL)}$ attends to more important teacher layers, compared to the same student layer in ALP, which mostly focuses on first teacher layers.

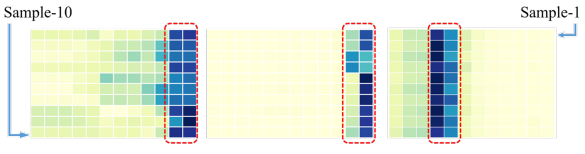

Figure 3: Visualizing (left) the Euclidean distance between real labels and classifier outputs of all 12 teacher layers, (middle) attention weights between the third layer of the 4-layer student trained by Universal-KD$^{(IL)}$ with all 12 teacher layers, (right) attention weights between the third layer of the 4-layer student trained by ALP with all 12 teacher layers for 10 samples from RTE.

6 Conclusion

In this paper, we introduced the Universal-KD approach, which employs attention-based layer projection to match the intermediate layers of the teacher and the student in the output space, thereby solving the dimension mismatching, layer mapping search and same-architecture limitation issues in the conventional intermediate layer KD approaches. Moreover, Universal-KD could also address the capacity gap problem without utilizing sophisticated intermediate TA networks. Extensive experiments on GLUE benchmark reflect that Universal-KD works for both intermediate KD, capacity gap solving as well as intermediate layer cross-architecture distillation.

Acknowledgments

We thank Mindspore[1] for the partial support of this work, which is a new deep learning computing framework.

References

Aryan Asadian and Amirali Salehi-Abari. 2021. Distilling knowledge via intermediate classifier heads. *arXiv preprint arXiv:2103.00497*.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6:737–744.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

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*.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs. *University of Waterloo*.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc Le. 2019. Bam! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

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.

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

Jie Fu, Xue Geng, Zhijian Duan, Bohan Zhuang, Xingdi Yuan, Adam Trischler, Jie Lin, Chris Pal, and Hao Dong. 2020. Role-wise data augmentation for knowledge distillation. *arXiv preprint arXiv:2004.08861*.

Abbas Ghaddar and Philippe Langlais. 2019. Contextualized word representations from distant supervision with and for ner. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 101–108.

Abbas Ghaddar, Philippe Langlais, Ahmad Rashid, and Mehdi Rezagholizadeh. 2021a. Context-aware adversarial training for name regularity bias in named entity recognition. *Transactions of the Association for Computational Linguistics*, 9:586–604.

Abbas Ghaddar, Phillippe Langlais, Mehdi Rezagholizadeh, and Ahmad Rashid. 2021b. End-to-end self-debiasing framework for robust NLU training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1923–1929, Online. Association for Computational Linguistics.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Francis Grégoire and Philippe Langlais. 2018. Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1442–1453.

[1] https://www.mindspore.cn/

Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. 2018. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2503–2510. IEEE.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. 2021. Annealing knowledge distillation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2493–2504.

Mingi Ji, Byeongho Heo, and Sungrae Park. 2021. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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*.

Prabhu Kaliamoorthi, Aditya Siddhant, Edward Li, and Melvin Johnson. 2021. Distilling large language models into tiny and effective students using pqrnn. *arXiv preprint arXiv:2101.08890*.

Ehsan Kamalloo, Mehdi Rezagholizadeh, Peyman Passban, and Ali Ghodsi. 2021. Not far away, not so close: Sample efficient nearest neighbour data augmentation via minimax. *arXiv preprint arXiv:2105.13608*.

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

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. 1997. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

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*.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198.

Marco Mondelli and Andrea Montanari. 2019. On the connection between learning two-layer neural networks and tensor decomposition. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1051–1060. PMLR.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Xtremedistil: Multi-stage distillation for massive multilingual models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2221–2234.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2020. Alp-kd: Attention-based layer projection for knowledge distillation. *arXiv preprint arXiv:2012.14022*.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2019. Fully quantized transformer for machine translation. *arXiv preprint arXiv:1910.10485*.

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

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

Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2020. Towards zero-shot knowledge distillation for natural language processing. *arXiv preprint arXiv:2012.15495*.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. Mate-kd: Masked adversarial text, a companion to knowledge distillation.

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*.

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.

Wonchul Son, Jaemin Na, and Wonjun Hwang. 2020. Densely guided knowledge distillation using multiple teacher assistants. *arXiv preprint arXiv:2009.08825*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019a. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019b. 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)*, pages 4314–4323.

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*.

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.

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. In *BlackboxNLP@EMNLP*, pages 353–355. Association for Computational Linguistics.

Huan Wang, Suhas Lohit, Michael Jones, and Yun Fu. 2020. Multi-head knowledge distillation for model compression. *arXiv preprint arXiv:2012.02911*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

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.

Yimeng Wu, Peyman Passban, Mehdi Rezagholizadeh, and Qun Liu. 2020. Why skip if you can combine: A simple knowledge distillation technique for intermediate layers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1016–1021.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*.

George-Eduard Zaharia, Andrei-Marius Avram, Dumitru-Clementin Cercel, and Traian Rebedea. 2021. Dialect identification through adversarial learning and knowledge distillation on romanian bert. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 113–119.

Helong Zhou, Liangchen Song, Jiajie Chen, Ye Zhou, Guoli Wang, Junsong Yuan, and Qian Zhang. 2021. Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective. *arXiv preprint arXiv:2102.00650*.

A Hyper-parameter Tunning

A.1 Hyper-parameter Tunning on Transformer-based student

In this section, we include more implementation details of Universal-KD$^{(IL)}$ and Universal-KD$^{(CG)}$. For Universal-KD$^{(IL)}$, our teacher is a 12-layer BERT-base model while the student is a 4-layer BERT-base model. The student model is initialized by the first 4 layers of the pre-trained BERT-base teacher. For Universal-KD$^{(CG)}$ experiments, our teacher is a 24-layer RoBERTa-large model while the student is a 6-layer DistilRoBERTa model.

We apply the same training configuration for both BERT-4 and DistilRoBERTa student. During training, grid search is performed over batch size ($\{8, 16, 32\}$), learning rate ($\{2e-5, 5e-5\}$) and β ($\{0.2, 0.5, 0.7\}$). We set $\mathcal{T}$ to 1 for all experiments. The student is first trained for 20 epochs in stage 1 for all datasets except for CoLA. CoLA needs 50 epochs to get high-quality results. Then the model is further trained for 10 epochs in stage 2. The best hyper-parameters values we used for BERT-4 and DistilRoBERTa students are summarized in Table 7 and Table 8, respectively.

A.2 Hyper-parameter Tunning on Bi-LSTM and CNN students

We adopt the Adam (Kingma and Ba, 2014) optimization algorithm, and varied learning rate between $1e-4$ and $5e-5$, batch size ($\{8, 16, 32, 64\}$), dropout ($[0.1 - 0.8]$), and β ($\{0.2, 0.5, 0.7\}$). We select the best performing hyper-parameters independently for each model on each task. Table 9 shows the hyper-parameters values we used for Bi-LSTM and Gated-CNN experiments.

It is worth mentioning that we do not adopt a siamese architecture (Bromley et al., 1993) [2] for sentence pair classification tasks, although it performs much better in these tasks (Grégoire and Langlais, 2018; Tang et al., 2019). We prefer to keep our models simple by using a single sequence encoder in all experiments. We do so to avoid having a mismatch between teacher's and student's inputs during distillation, which is an interesting problem we want to explore in future works.

[2] Encoding sentences independently and then concatenating their representations.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
learning rate	5e-5	5e-5	2e-5	5e-5	2e-5	5e-5	2e-5	5e-5
batch size	32	32	32	32	32	16	32	32
β	0.2	0.5	0.5	0.5	0.7	0.5	0.5	0.2

Table 7: Hyperparameters values for 4-layer BERT student.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
learning rate	2e-5	2e-5	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
batch size	32	32	32	32	32	16	16	32
β	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Table 8: Hyperparameters values for DistilRoBERTa student.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
Bi-LSTM								
learning rate	2e-4	2e-5	3e-5	2e-5	3e-5	1e-5	5e-5	1e-5
batch size	8	64	64	32	64	8	32	8
dropout	0.4	0.1	0.4	0.2	0.1	0.4	0.3	0.4
β	0.5	0.5	0.7	0.5	0.5	0.5	0.7	0.2
Gated-CNN								
learning rate	2e-4	2e-5	3e-5	2e-5	3e-5	1e-5	5e-5	1e-5
batch size	64	64	16	32	32	16	64	8
dropout	0.4	0.1	0.4	0.2	0.1	0.4	0.3	0.4
β	0.5	0.5	0.7	0.5	0.5	0.5	0.7	0.2

Table 9: Hyperparameters values for Bi-LSTM and Gated-CNN cross-architecture experiments.