Continuation KD: Improved Knowledge Distillation through the Lens of Continuation Optimization

Aref Jafari[1] **Ivan Kobyzev**[2] **Mehdi Rezagholizadeh**[2]
Pascal Poupart[1] **Ali Ghodsi**[1]
[1]University of Waterloo
[2]Huawei Noah's Ark Lab
{aref.jafari, ppoupart, ali.ghodsi}@uwaterloo.ca
{ivan.kobyzev, mehdi.rezagholizadeh}@huawei.com

Abstract

Knowledge Distillation (KD) has been extensively used for natural language understanding (NLU) tasks to improve a small model's (a student) generalization by transferring the knowledge from a larger model (a teacher). Although KD methods achieve state-of-the-art performance in numerous settings, they suffer from several problems limiting their performance. It is shown in the literature that the capacity gap between the teacher and the student networks can make KD ineffective. Additionally, existing KD techniques do not mitigate the noise in the teacher's output: modeling the noisy behaviour of the teacher can distract the student from learning more useful features. We propose a new KD method that addresses these problems and facilitates the training compared to previous techniques. Inspired by continuation optimization, we design a training procedure that optimizes the highly non-convex KD objective by starting with the smoothed version of this objective and making it more complex as the training proceeds. Our method (Continuation-KD) achieves state-of-the-art performance across various compact architectures on NLU (GLUE benchmark) and computer vision tasks (CIFAR-10 and CIFAR-100).

1 Introduction

Deep neural networks have achieved great success in many challenging tasks including the ones in natural language processing (Vaswani et al., 2017; Brown et al., 2020a) and computer vision (Dosovitskiy et al., 2020). Most successful neural networks are usually large and overparametrised (Liu et al., 2019; Devlin et al., 2018). The big size of these models prevents them from being deployed on computers with low computational power such as edge devices. Compressing the models can empower us to provide a variety of machine learning-based services offline on low-resource machines. This issue is even more severe in models designed for NLP. Although, after the introduction of the Transformers (Vaswani et al., 2017) the performance of models in this field improved dramatically, the size of the models increased exponentially. Nowadays, some of these models have more than 100 billion parameters (Brown et al., 2020b), and they are still increasing.

One way to address the expensive computational complexity of deep networks and their overparameterization is neural model compression (Jacob et al., 2018; Tjandra et al., 2018; Bie et al., 2019). Among all of the compression methods, Knowledge Distillation (KD) (Hinton et al., 2015) is one of the prominent techniques which have been used to compress a variety of models in different deep learning applications such as Natural Language Processing (Clark et al., 2019; Sun et al.; Jiao et al., 2019; Rashid et al., 2020, 2021; Kamalloo et al., 2021; Haidar et al., 2021; Wu et al., 2021), speech processing (Yun et al., 2020; Chebotar and Waters, 2016), and computer vision (Mirzadeh et al., 2019; Guo et al., 2020). In KD, we have a large accurate network, which is called a teacher, and a small network, which is called a student, that we desire to train. The innate capacity gap between the student and the teacher was speculated (Lopez-Paz et al., 2015) to impede the training and was addressed by multiple works (Mirzadeh et al., 2019; Jafari et al., 2021). However, the existing solutions have several complications: for example, (Mirzadeh et al., 2019) requires an extra intermediate network to be trained, and (Jafari et al., 2021) has a rigid two-stage structure that requires the careful design for successful training. Moreover, none of these techniques is robust against the noisy data and noisy teacher's outputs, which can distract the student with the limited capacity from learning more useful features and make it overfit to the noise.

We propose a new solution, Continuation KD, inspired by the continuation method from optimiza-

tion and nonlinear equations. During the training, we gradually move from the smoothed objective function, which is robust to overfitting to the noise, to the original highly non-convex function. We conduct extensive experiments on the GLUE benchmark for DistilRoBERTA (6-layer) (Sanh et al., 2019) and BERT-small (4-layers) (Turc et al., 2019) core models and show a significant improvement over the previous baselines. Besides that, we demonstrate that in the computer vision setting continuation KD also outperforms its competitors. Overall, our contributions are the following:

1. We proposed a novel KD technique based on the Continuation method, which gradually increases the complexity of the loss function and provides a better optimization for all knowledge distillation scenarios in both computer vision and NLP.

2. We implement a loss function like hinge loss function which makes a student trained with our technique robust against the teacher's output noise.

3. Our proposed method is simple and, unlike its competitors, does not have several stages. This feature makes our method stable and efficient.

2 Related Works

2.1 Knowledge Distillation (KD)

Knowledge Distillation (Hinton et al., 2015) is a well-known neural model compression method. Despite the success of the original method, it has been shown in the literature (Lopez-Paz et al., 2015; Mirzadeh et al., 2019) that the large gap between the size of the student and the teacher networks makes KD ineffective. To address this *capacity gap* problem, Mirzadeh et al. (2019) proposed the teacher assistant knowledge distillation (TAKD) method.However, this technique is computationally expensive as it requires training multiple TA networks for a task. Moreover, the errors of the TAs can accumulate and transfer to the student. To alleviate these problems, Jafari et al. (2021) proposed Annealing-KD that achieved state-of-the-art performance on NLU and computer vision tasks. Although this method could handle the capacity gap problem, it is still not robust against the noisy data and noisy teacher's outputs. Also, two phases of the training require some *a priori* decisions on when to switch from the first phase to the second one.

2.2 Continuation Optimization

Continuation method was first proposed as a numerical method for solving nonlinear equations (Davidenko, 1953) and then it was adopted as a heuristic for nonconvex optimization (Watson, 2000). The main intuition of the continuation method is to include the problem we are trying to solve in a continuous family of problems, such that one of the members of this family is easy to solve and its solution could be pulled over to give an approximate solution of the original problem.

Machine Learning community applied the continuation idea for training neural networks. Mobahi and Fisher III (2015) proposed the first theoretical analysis of the bound on the approximate solution given by the continuation optimization. Gulcehre et al. (2017) suggested optimizing highly non-convex neural networks by starting with a smoothed objective function and making it more complex over the training. We adapted this general method for Knowledge Distillation.

3 Background

Vanilla-KD The original knowledge distillation (Hinton et al., 2015) trains a small network (a student) by using two guiding signals: the hard labels coming from the training dataset, and the predictions of a large network pre-trained on the same task (a teacher) which is known as soft labels. To achieve this goal, Vanilla-KD utilizes a particular loss function which is a linear combination of two losses. The first one is a cross-entropy between the softmax output of the student and hard labels, and the second one is a KL-divergence between the softened version of the softmax outputs of the student and the teacher networks. Equation 1 explains this loss function in details:

$$\mathcal{L}_{KD} = \lambda CE(y, \sigma\big(z_S(x)\big) + (1-\lambda) KL(\sigma(\frac{z_T(x)}{\tau}), \sigma(\frac{z_S(x)}{\tau})) \quad (1)$$

Here $CE(.)$ is the cross-entropy function, $KL(.)$ is the KL-divergence function, $z_T(.)$ and $z_S(.)$ are the teacher and student logits, $\sigma(.)$ is the softmax function, and τ is the softening parameter. Also, λ is a hyper-parameter between $[0, 1]$ which indicates the amount of contribution of each loss function. Minimizing the above loss function decreases the

distance between both underlying function and the teacher model. In Vanilla-KD usually we assume that the teacher is a good approximation of the underlying function.

TAKD In TAKD method (Mirzadeh et al., 2019) the teacher model first trains an intermediate model with a slightly smaller capacity, which is called teacher assistant (TA), by utilizing Vanilla-KD. Then the TA model trains the student model with a small capacity by using Vanilla-KD again. TAKD tries to fill the capacity gap between the teacher and the student models by introducing the TA model but this gap can be still large. As mentioned in (Mirzadeh et al., 2019), a better idea is to use hierarchical TAs to have a smoother knowledge transfer from the teacher to the student.

Annealing-KD For controlling the complexity of the teacher model, instead of using multiple TA networks, Annealing-KD (Jafari et al., 2021) adds an annealed dynamic temperature factor to the output of the teacher. By using this factor, Annealing-KD reduces the sharpness of the teacher at the beginning of the training process. Then it increases the sharpness of the teacher gradually during the training. Therefore, since the complexity of the teacher increases gradually during the training time, the teacher knowledge transfers much more smoothly to the student than in TAKD.

Annealing-KD has two stages of training where in the first stage, a student learns from a teacher for k epochs. During this stage, Annealing-KD matches the student's logits to the teacher's logits by using a mean square error loss function. At the beginning of the training the temperature factor sets to a high value to apply the maximum smoothing to the output of the teacher and then it decreases gradually until there is no smoothing effect remains on the output of the teacher. Formally, one first defines a monotonically increasing function $\phi : \mathbb{N} \rightarrow [0, 1]$, going from zero at the beginning of the training to one at the end. Then, the student loss for stage 1 is:

$$\mathcal{L}(i) = ||z_S(x) - \phi(i) z_T(x)||_2^2, \tag{2}$$

where i is the training epoch, $z_S(x)$ and $z_T(x)$ are logit outputs of the student and the teacher respectively. In the second stage, Annealing-KD gets the best checkpoint of the first stage and finetune it on the hard labels from the dataset by using a cross-entropy loss for m epochs. The hyperparameters k and m must be chosen before the training.

4 Methodology

In this section, we describe our Continuation-KD technique, which addresses both the capacity gap problem and the lack of robustness against the noise in the teacher's output. Continuation-KD uses a loss function with two objectives (Eq. 3). The first objective $\mathcal{L}_{CE}$ is a cross-entropy loss term that trains the student based on the given hard labels. The second objective $\mathcal{L}_{KD}^{CNT}$ is our proposed annealed hinge loss function that gradually trains the student to mimic the behaviour of the teacher. Inspired by the continuation method (Gulcehre et al., 2017), Continuation-KD starts with an easy objective to train at the beginning of training. As the training proceeds, the whole objective function becomes more and more complex. Formally, the loss function of the Continuation-KD is defined as follows:

$$\mathcal{L} = (\psi(i))\mathcal{L}_{CE} + (1 - \psi(i))\mathcal{L}_{KD}^{CNT} \tag{3}$$

where $1 \leq i \leq n$ indicates the epoch index with the maximum number of epochs n. The $0 \leq \psi(i) \leq 1$ is an increasing function between 0 and 1 where $\psi(1) = 0$ at the beginning and it increases during the training. L_{KD}^{CNT} defines as:

$$\mathcal{L}_{KD}^{CNT} = \max\{0, \|z_S - \phi(\mathcal{T}_i) z_T\|_2^2 - m\ \phi(\mathcal{T}_i)\} \tag{4}$$

where z_S and z_T are the output logits of the student and teacher networks, respectively; m is the margin factor; $1 \leq \mathcal{T}_i \leq \mathcal{T}_{max}$ is the temperature factor, $\mathcal{T}_{max}$ is the maximum temperature, and $0 \leq \phi(\mathcal{T}_i) \leq 1$ is an increasing function. We define this function as:

$$\phi(\mathcal{T}_i) = 1 - \frac{\mathcal{T}_i - 1}{\mathcal{T}_{\text{max}}}, 1 \leq \mathcal{T}_i \leq \mathcal{T}_{\text{max}}, \mathcal{T}_i \in \mathbb{N}. \tag{5}$$

Note that in Eq. 4, $\|z_S - \phi(\mathcal{T}_i) z_T\|_2^2$ is a mean square loss between the student's logits and the annealed version of the teacher's logits. Also, $\max\{0, \|z_S - \phi(\mathcal{T}_i) z_T\|_2^2 - \phi(\mathcal{T}_i) m\}$ is a hinge loss with an annealed margin $\phi(\mathcal{T}_i) m$. This loss function avoids penalizing negligible differences (the ones less than $m\ \phi(\mathcal{T}_i)$) between the outputs of the student and teacher. This feature helps the student to learn a meaningful behaviour of the teacher rather than focusing on higher frequency fluctuations.

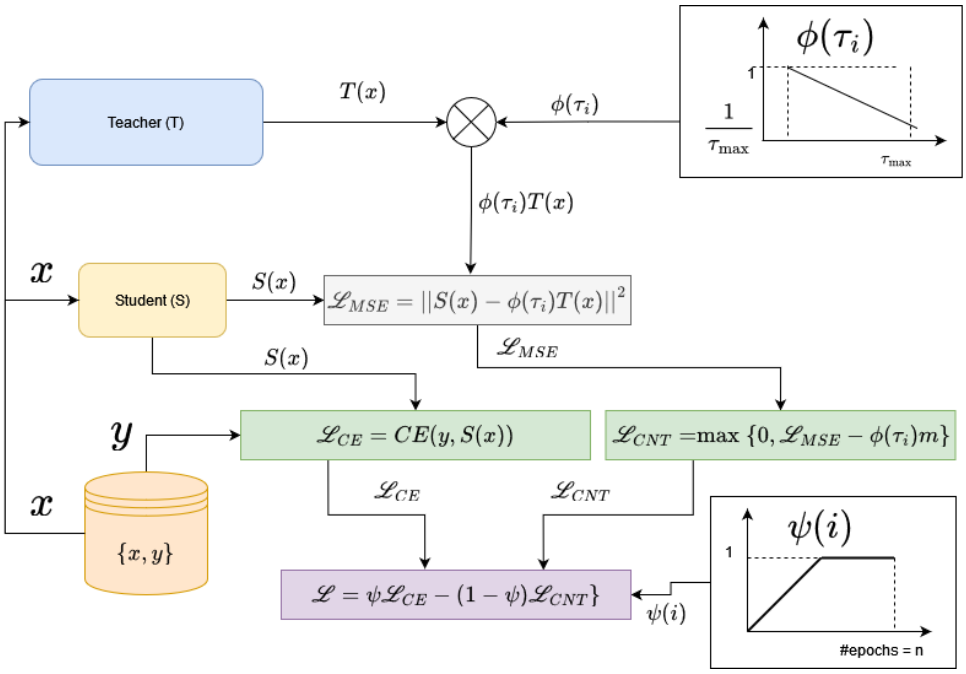


Figure 1: Principle diagram illustrating different components of Continuation-KD. The main loss function (purple box) is a composition of two losses - cross entropy loss $\mathcal{L}_{CE}$ and continuation loss $\mathcal{L}_{CNT}$ (green boxes). Because of the smoothing in $\mathcal{L}_{CNT}$ with the dynamic factor ϕ, it is an easier objective to optimize than $\mathcal{L}_{CE}$. During the training, Continuation KD gradually moves from the easier objective to more complex objective with aid of the dynamic factor ψ.

At the beginning of training, we set $\mathcal{T}_1 = \mathcal{T}_{max}$ which leads to the most softened version of the teacher's output ($\phi(\mathcal{T}_1) = \frac{1}{\mathcal{T}_{max}}$). Since we have $\psi(1) = 0$, the student only learns the behaviour of the teacher's smoothest version, which is an easy target to learn. Then, during training, we decrease the temperature, At this phase, functions $\psi(i)$ and $\phi(\mathcal{T}_i)$ are both increasing which in turn leads to increasing the sharpness of the teacher and smoothly shifting from the hinge loss to the cross-entropy loss. Both of these operations increase the complexity of the whole loss function. Note that at the function $\phi(\mathcal{T}_i)$ also anneals margin m. Its reason is that smoothing the teacher with $\phi(\mathcal{T}_i)$ damps its noise as well. Therefore we damp the margin m with $\phi(\mathcal{T}_i)$ to apply a margin proportional to the amount of noise in the smoothed version of the teacher. Figure 1 visualize different components of continuation-KD.

Also note that, if we set $m = 0$ and $\psi(i)$ to the step function in Eq 6, then Continuation-KD becomes identical to Annealing-KD, where k is the number of epochs in the first stage and $n - k$ is the number of epochs in the second stage. This fact shows that the Annealing-KD is actually a special case of our Continuation-KD.

$$\psi(i) = \begin{cases} 0, & 1 \leq i \leq k \\ 1, & k < i \leq n \end{cases} \tag{6}$$

Algorithm 1 demonstrates the details of Continuation-KD. It requires a student S, a teacher T, a dataset D, max temperature $\mathcal{T}_{max}$, number of epochs n, an increasing function ψ, and margin m as inputs and returns the trained student at the output. At the beginning, it sets variables $\mathcal{T} = \mathcal{T}_{max}$ to get the maximum smoothness of the teacher. Also, variable k indicates the number of epochs before updating $\mathcal{T}$during training. Φ and Ψ are the output values of $\phi(i)$ and $\psi(i)$. Function GET-MINI-BATCH(D) retrieves a mini-batch (X, Y) from the dataset D. Then these data samples feed into loss functions in the next lines to get the outputs of $\mathcal{L}_{CE}$ and $\mathcal{L}_{KD}^{CNT}$. Then the linear combination of line 14 combines these two losses to get the continuation loss $\mathcal{L}$. Finally, $\mathcal{L}$ is fed into OPTIMIZATION-BACK-PROPAGATION$(.)$ function to optimize the student network based on the back propagation of the gradient of this loss function and update the weights of the student. This part of the training is identical to regular training of the neural networks.

SAVE-BEST-CHECKPOINT(.) function checks the performance of the current student model on a validation dataset. If it is better than the previous checkpoints, it saves the checkpoint. In the end, we load the best checkpoint and return it.

In the next section, we report the experimental results of Continuation-KD method.

5 Experiments

This section demonstrates our evaluation results comparing Continuation-KD with other baselines for natural language processing and computer vision tasks. We compare our method with state-of-the-art techniques such as annealing-KD (Jafari et al., 2021), TAKD (Mirzadeh et al., 2019) and other baselines like Vanilla-KD (Hinton et al., 2015) and training the student only with hard labels. In the following sub-sections, we will discuss each in more detail.

5.1 Hardware Details

We trained all our baselines using a single NVIDIA V100 GPU. All experiments were run using the PyTorch framework[1] and for NLP experiments we used HuggingFace[2] API.

5.2 Image Classification

For the image classification tasks, we used CIFAR-10 and CIFAR-100 datasets with 10 and 100 classes respectively. Both of these datasets have 60,000 data samples, and each of them is a 32×32 pixel color image. Also, both of these datasets have 50,000 train and 10,000 test samples.

In all of our computer vision (CV) experiments, ResNet-8 and ResNet-110 models are used as the student and the teacher respectively. For the TAKD baseline, the ResNet-20 model is used as the TA model. Continuation-KD is trained for 200 epochs with a maximum temperature of 20, a learning rate of 0.2, and batch size 32. Also, the following $\psi(\cdot)$ function used in these experiments:

$$\psi(i) = \begin{cases} \frac{i}{150}, & 1 \leq i \leq 150 \\ 1, & i \geq 150 \end{cases} \tag{7}$$

The CIFAR-10 and CIFAR-100 experiments results are reported in Tables 1 and 2. We can see that Continuation-KD clearly outperforms other baselines. Also, Annealing-KD achieved second-best results in these experiments. However, other baselines showed almost similar performances.

Table 1: Comparing the test accuracy of Continuation-KD, Annealing-KD, TAKD, Vanilla-KD, and finetuning on CIFAR-10 dataset with ResNet model

Type	Training method	Accuracy
Teacher(110)	from scratch	93.8
TA(20)	KD	92.39
Student(8)	from scratch	88.44
Student(8)	KD	88.45
Student(8)	TAKD	88.47
Student(8)	Annealing-KD	89.44
student(8)	**Continuation-KD**	**90.21**

Table 2: Comparing the test accuracy of Continuation-KD, Annealing-KD, TAKD, Vanilla-KD, and finetuning on CIFAR-100 dataset with ResNet model

Type	Training method	Accuracy
teacher(110)	from scratch	71.92
TA(20)	KD	67.6
student(8)	from scratch	61.37
student(8)	KD	61.41
student(8)	TAKD	61.82
student(8)	Annealing-KD	63.1
student(8)	**Continuation-KD**	**64.2**

5.3 Natural Language Understanding

For the NLU experiments, we use the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). The benchmark contains several tasks, including textual entailment (RTE and MNLI), question-answer entailment (QNLI), paraphrase (MRPC), question paraphrase (QQP), sentiment (SST-2), textual similarity (STS-B), linguistic acceptability (CoLA), and Winograd Schema (WNLI).

Two types of students with different capacities are used in these experiments. In our first experiment, we used distilRoBERTa with 6 layers as our student and RoBERTa-Large (24 layers) as our teacher. Also, for the TAKD baseline, we used RoBERTa-base (12 layers) as the teacher assistant model. For this experiment, the maximum temperature was 10, learning rate was 2e-5, and batch size was 64. Also, we used the pre-trained distilRoBERTa as our student and we fine-tuned it for 30 epochs. The ψ function for this experiment is

[1] https://pytorch.org
[2] https://huggingface.co

Algorithm 1

function CONTINUATION-KD(S,T,D, $\mathcal{T}_{max}$, n, $\psi(\cdot)$, m)
  $\mathcal{T} = \mathcal{T}_{max}$
  $k = \lfloor \frac{n}{\mathcal{T}_{max}} \rfloor$
  for $i = 1$ to n **do**
    if $i \bmod k = 0$ **then**
      $\mathcal{T} = \mathcal{T} - 1$
    end if
    $\Phi \leftarrow 1 - \frac{\mathcal{T}-1}{\mathcal{T}_{\text{max}}}$,$\Psi \leftarrow \psi(i)$
    $X, Y \leftarrow$ GET-MINI-BATCH(D)
    $\mathcal{L}_{CE} \leftarrow CE(\sigma(S(X)), Y)$
    $\mathcal{L}_{KD}^{CNT} \leftarrow \max\{0, \|S(X) - \Phi T(X)\|_2^2 - m\ \Phi\}$
    $\mathcal{L} = \Psi \mathcal{L}_{CE} + (1 - \Psi)\mathcal{L}_{KD}^{CNT}$
    OPTIMIZATION-BACK-PROPAGATION($\mathcal{L}$)
    SAVE-BEST-CHECKPOINT(S)
  end for
  $S \leftarrow$ LOAD-BEST-CHECKPOINT( )
  return S
end function

defined as following:

$$\psi(i) = \begin{cases} \frac{i}{40}, & 1 \leq i \leq 20 \\ 1, & 20 < i \leq 30 \end{cases} \tag{8}$$

Tables 3 and 4 show the results of this experiment on the dev set and test set. As shown in these tables, Continuation-KD achieved superior results in most of the tasks and it improves the previous baselines with a good overall average score.

For the second experiment, we utilize BERT-large (24-layers) as the teacher, BERT-Small (4-layers) as the student and BERT-base (12-layers) as the teacher-assistant for TAKD. We train the teacher for 30 epochs, and for the student, we use the same hyperparameters as the first experiment. We report the results in Tables 5 and 6. Here, Continuation-KD also outperforms other baselines and gets better overall performance than its competitors.

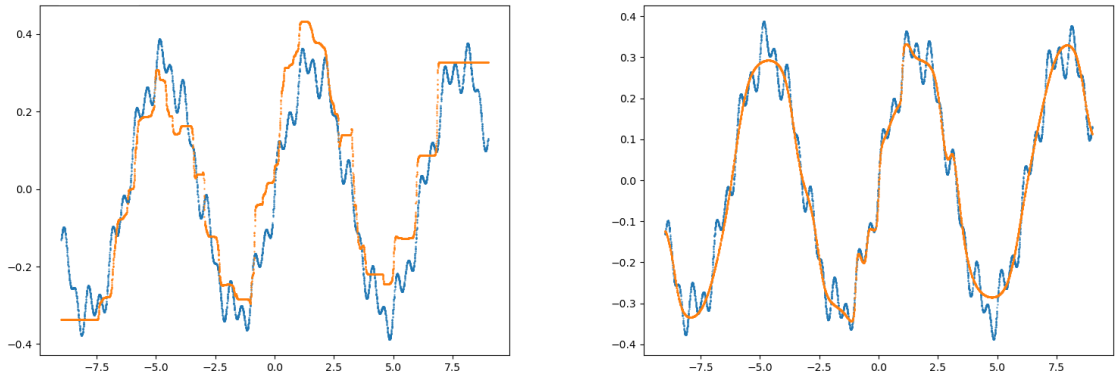
All presented results in our experiments show that continuation-KD performs better than TAKD and Annealing-KD, which indicates the effectiveness of this method. The experimental results support the claim that our proposed Continuation-KD can provide a better generalization than the other methods.

6 Analysis

To investigate how Continuation-KD works, we did two ablation studies explaining different aspects of this technique: effects of the dynamically changing factors and noise mitigation.

Effects of Dynamic Factors ϕ and ψ. In the first ablation, we scrutinize the effect of each dynamically changing component of the Continuation-KD loss function. It basically considers the effect of functions: $\psi(i)$ from Eq. 3, $\phi(\mathcal{T}_i)$ when it anneals the outputs of the teacher in Eq. 4, and $\phi(\mathcal{T}_i)$ when it anneals the margin of the hinge loss in Eq. 4. To investigate the effects of these components, we repeat the NLU experiments with distilRoBERTa model described in section 5.3 on MRPC and RTE datasets three times. In each trial, we fixed two of the components, and we only let one of them dynamically change during training. Table 7 reports the performance of each of these experiments. The first three columns of this table show the performance of distilRoBERTa on each of the datasets when only one of the three dynamical components changes and two others are fixed. The last column shows the performance of the model on these datasets in a regular training when all components dynamically change. In the first column, the $\phi = 1$ is fixed for both teacher and margin. In the second column, the margin's coefficient is fixed to 1 and $\psi = 0.5$. Also, in the third column, the teacher's coefficient is fixed to 1 and $\psi = 0.5$.

As shown in Table 7, for both datasets, the performance in the first three columns is almost similar, which indicates the equal contribution of each dy-



(a) (b)

Figure 2: (a) Illustrates the behaviour of the student model after training with a noisy teacher without using continuation-KD. (b) Illustrates the behaviour of the student model after training with a noisy teacher by using continuation-KD for training. Blue points are the samples from the noisy teacher. Orange points are the samples of the trained student in each scenario.

Table 3: DistilRoBERTa results for Continuation-KD on dev set. F1 scores are reported for MRPC, pearson correlations for STB-B, and accuracy scores for all other tasks.

KD Method	CoLA	RTE	MRPC	STS-B	SST-2	QNLI	QQP	MNLI	score
Teacher	68.1	86.3	91.9	92.3	96.4	94.6	91.5	90.22	88.91
Finetune	59.3	67.9	88.6	88.5	92.5	90.8	90.9	84	82.81
Vanilla-KD	60.97	71.11	90.2	88.86	92.54	91.37	91.64	84.18	83.85
TAKD	61.15	71.84	89.91	88.94	92.54	91.32	**91.7**	83.89	83.91
Annealing KD	61.67	73.64	90.6	89.01	93.11	91.64	91.5	**85.34**	84.56
Continuation-KD	**63.75**	**77.62**	**91.93**	**89.71**	**93.58**	**91.76**	91.64	85.16	**85.64**

Table 4: Performance of DistilRoBERTa trained by Continuation-KD on the GLUE leaderboard compared with Vanilla-KD, TAKD, and annealing KD.

KD Method	CoLA	RTE	MRPC	STS-B	SST-2	QNLI	QQP	MNLI	score
Finetune	52.97	74.56	87.93	84.9	93.1	90.57	83.4	88.77	82.02
Vanilla-KD	54.3	74.1	86	85.7	93.1	83.6	90.8	89.5	82.14
TAKD	53.2	74.2	86.7	85.6	93.2	83.8	91	89.4	82.14
Annealing KD	54	73.7	88	**87.0**	93.6	83.8	90.8	89.7	82.58
Continuation-KD	**54.5**	74.2	**90**	**87.0**	**93.8**	**84.7**	**91.6**	**90.1**	**83.24**

Table 5: BERT-Small results for Continuation-KD on dev set. F1 scores are reported for MRPC, pearson correlations for STS-B, and accuracy scores for all other tasks.

KD Method	CoLA	RTE	MRPC	STS-B	SST-2	QNLI	QQP	MNLI	score
Teacher	65.8	71.48	91.38	89.2	92.77	92.82	91.45	86.3	85.15
Finetune	41.7	64.98	83.75	87.41	88.3	86.49	88.43	78.42	77.44
Vanilla-KD	**41.89**	64.98	86	85.95	88.76	86.75	88.24	78.62	77.65
TAKD	40.2	65.7	85.23	86.44	88.88	86.78	88.4	78.78	77.55
Annealing-KD	41.36	64.9	87.93	87.04	89.56	86.99	88.58	78.66	78.13
Continuation-KD	41.51	**65.26**	**89.44**	**87.81**	**90.21**	**87.52**	**90.61**	**79.27**	**78.95**

Table 6: BERT-Small results for Continuation-KD on test set. F1 scores are reported for MRPC, pearson correlations for STS-B, and accuracy scores for all other tasks.

KD Method	CoLA	RTE	MRPC	STS-B	SST-2	QNLI	QQP	MNLI	score
Finetune	38.1	61.8	83.4	78.8	89.7	86.4	78.1	77.6	74.24
Vanilla-kd	37.3	63.4	80.6	78.2	90.2	86.5	78.3	78.3	74.09
TAKD	38.5	62.3	80.5	79.3	89.7	86.7	78	78.2	74.14
Annealing-KD	38.3	63.3	81.9	80.6	89.8	86.8	78.4	79.3	74.78
Continuation-KD	38.5	**64.8**	**84.6**	**82.5**	**90.6**	**87.2**	**80.1**	**79.5**	**75.98**

namic component in the improvement of the results. Also, the last column shows the dramatic improvement in the performance. Hence, we can conclude from this experiment that all three dynamic components are necessary for achieving better performance.

Table 7: Performance of distilRoBERTa on MRPC and RTE datasets for different dynamically changing components of continuation-KD loss

dataset	ψ	ϕ (teacher)	ϕ (margin)	all
MRPC	91.09	91.21	90.90	91.93
RTE	72.56	73.64	71.48	77.62

Noise Robustness In the second ablation study, we visualize the effect of Continuation-KD with a noisy teacher. For this purpose, we consider a sinusoidal function with low frequency, and then we add to it a sinusoidal noise with high frequency (blue curves in Figure 2). Then, we take a fully connected network with one-dimensional input and output and two hidden layers with 128 neurons in each layer as a student model. We sample 3,000 points from the graph of the noisy sinusoidal function and train the student model once with Vanilla-KD and once with Continuation-KD (orange curves in Figure 2). As Figure 2 shows, the model trained with Continuation-KD has a much smoother curve in comparison with the model trained with vanilla-KD and could learn the main behaviour of the teacher function rather than learning noise.

7 Conclusion

In this work, we present Continuation-KD, a novel KD method inspired by Continuation optimization. Our Continuation-KD technique starts optimizing a smoothed version of the objective function and gradually increases the complexity of the loss towards the original, highly non-convex one. We demonstrated that our method alleviates the capacity gap problem: an innate problem of KD resulting from the different capacities of a student's and a teacher's networks which detriments the performance. Besides that, we show that the method can lessen the student's overfitting to noise in the teacher's output. Our technique is stable because it doesn't require two stages in the training and is efficient. It outperforms its competitor KD methods for different backbone models' architectures in both computer vision and NLP.

For this investigation, we proposed to implement Continuation method by smoothing the objective with the hinge loss. However, it can potentially be done differently. Investigating other realizations of Continuation Optimization for improving small models' performance is an interesting next step.

Limitation

One of the advantages of our proposed method is that it mitigates the noise in the teacher's output and prevents the student from overfitting to the noise. We empirically demonstrate this claim, but we don't have rigorous theoretical proof of how continuation method achieves this robustness.

Acknowledgments

We thank Mindspore[3] for the partial support of this work. We thank the anonymous reviewers for their insightful comments.

References

Alex Bie, Bharat Venkitesh, Joao Monteiro, Md Haidar, Mehdi Rezagholizadeh, et al. 2019. A simplified fully quantized transformer for end-to-end speech recognition. *arXiv preprint arXiv:1911.03604*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child,

[3] A new deep learning computing framework https://www.mindspore.cn/

Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Yevgen Chebotar and Austin Waters. 2016. Distilling knowledge from ensembles of neural networks for speech recognition. In *Interspeech*, pages 3439–3443.

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

D.F. Davidenko. 1953. On a new method of numerical solution of systems of nonlinear equations. *Dokl. Akad. Nauk SSSR*, 88:601–602.

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*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Caglar Gulcehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. 2017. Mollifying networks. In *5th International Conference on Learning Representations*.

Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. 2020. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11020–11029.

Md Akmal Haidar, Nithin Anchuri, Mehdi Rezagholizadeh, Abbas Ghaddar, Philippe Langlais, and Pascal Poupart. 2021. Rail-kd: Random intermediate layer mapping for knowledge distillation. *arXiv preprint arXiv:2109.10164*.

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

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

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.

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*.

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*.

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*.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. 2019. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*.

Hossein Mobahi and John Fisher III. 2015. A theoretical analysis of optimization by gaussian continuation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

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. *arXiv preprint arXiv:2105.05912*.

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*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: Task-agnostic compression of bert for resource limited devices.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2018. Tensor decomposition for compressing recurrent neural network. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

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. *arXiv preprint arXiv:1804.07461*.

L.T. Watson. 2000. Theory of globally convergent probability-one homotopies for nonlinear programming. *SIAM Journal on Optimization*, 11(3):761–780.

Yimeng Wu, Mehdi Rezagholizadeh, Abbas Ghaddar, Md Akmal Haidar, and Ali Ghodsi. 2021. Universal-kd: Attention-based output-grounded intermediate layer knowledge distillation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7649–7661.

Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. 2020. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13876–13885.