IMPROVED KNOWLEDGE DISTILLATION BY UTILIZING BACKWARD PASS KNOWLEDGE IN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge distillation (KD) is one of the prominent techniques for model compression. In this method, the knowledge of a large network (teacher) is distilled into a model (student) with usually significantly fewer parameters. KD tries to better-match the output of the student model to that of the teacher model based on the knowledge extracts from the forward pass of the teacher network. Although conventional KD is effective for matching the two networks over the given data points, there is no guarantee that these models would match in other areas for which we do not have enough training samples. In this work, we address that problem by generating new auxiliary training samples based on extracting knowledge from the backward pass of the teacher in the areas where the student diverges greatly from the teacher. We compute the difference between the teacher and the student and generate new data samples that maximize the divergence. This is done by perturbing data samples in the direction of the gradient of the difference between the student and the teacher. Augmenting the training set by adding this auxiliary improves the performance of KD significantly and leads to a closer match between the student and the teacher. Using this approach, when data samples come from a discrete domain, such as applications of natural language processing (NLP) and language understanding, is not trivial. However, we show how this technique can be used successfully in such applications. We studied the effect of the proposed method on various tasks in different domains, including images and NLP tasks with considerably smaller student networks. The results of our experiments, when compared with the original KD, show 4% improvement on MNIST with a student network that is 160 times smaller, 1% improvement on a CIFAR-10 dataset with a student that is 9 times smaller, and an average 1.5% improvement on the GLUE benchmark with a distilroBERTa-base student.

1 INTRODUCTION

During the last few years, we faced the emerge of a huge number of cumbersome state-of-the-art deep neural network models in different fields of machine learning, including computer vision (Wong et al., 2019; Howard et al., 2017), natural language processing (Prato et al., 2019; Jiao et al., 2019; Lan et al., 2019; Brown et al., 2020) and speech processing (Bie et al., 2019; He et al., 2019). We need powerful servers to be able to deploy such large models. Running such large models on edge devices would be infeasible due to the limited memory and computational power of edge devices (Sun et al., 2020). On the other hand, considering users' privacy concerns, network reliability issues, and network delays increase the demand for offline machine learning solutions on edge devices. The field of neural model compression focuses on providing compression solutions such as quantization (Jacob et al., 2018), pruning (Wang et al., 2019), tensor decomposition (Tjandra et al., 2018) and knowledge distillation (KD) (Hinton et al., 2015) for large neural networks.

Knowledge distillation (KD) is one of the most prominent compression techniques in the literature. As its name implies, KD tries to transfer the learned knowledge from a large teacher network to a small student. The idea of KD was proposed by Buciluǎ et al. (2006) for the first time and later this idea generalized by Hinton et al. (2015) for deep neural nets. The original KD method concerns transferring knowledge from a teacher to a student network only by matching their forward pass outputs. Later on, several works in the literature suggested other sources of knowledge in the teacher network besides the logit outputs of the last layer. This includes using intermediate layer feature

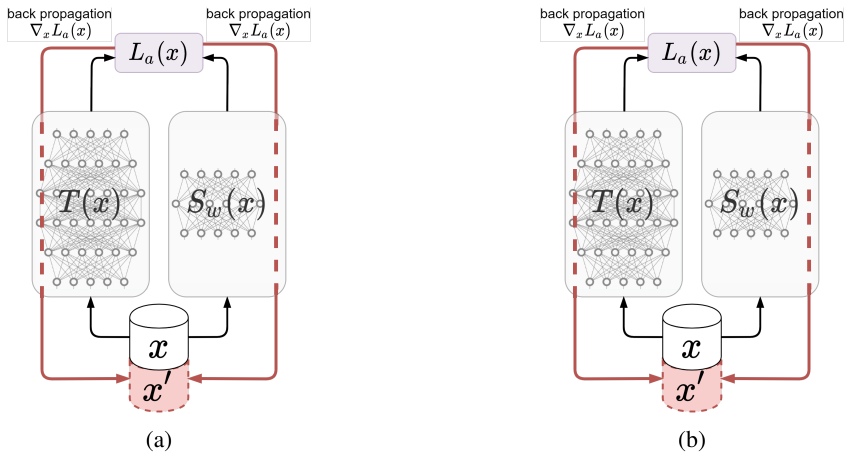

back propagation $\nabla_x L_a(x)$
back propagation $\nabla_x L_a(x)$
$L_a(x)$
$T(x)$
$S_w(x)$
x
x'
(a)
back propagation $\nabla_x L_a(x)$
back propagation $\nabla_x L_a(x)$
$L_a(x)$
$T(x)$
$S_w(x)$
x
x'
(b)

Figure 1: (a) Minimization Step: Using the teacher model knowledge for training the student in KD (utilizing forward knowledge) (b) Maximization Step: Augmenting the input dataset x with auxiliary data samples x' which is generated by the back propagation of gradient through both networks (utilizing backward knowledge)

maps (Sun et al., 2019; Sun et al.; Jiao et al., 2019), gradients of the network outputs w.r.t the inputs (Czarnecki et al., 2017; Srinivas & Fleuret, 2018)), and matching decision boundaries for classification tasks (Heo et al., 2019). using this additional information might be useful to get the student network performance closer to that of the teacher.

In this work, we focus on identifying regions of the input space of the teacher and student networks in which the two functions diverge the most from each other. Moreover, we highlight the importance of incorporating backward knowledge of the teacher and student networks in the knowledge distillation process. Our proposed iterative backward KD approach is comprised of: first, a maximization step in which a new set of auxiliary training samples is generated by pushing training samples towards maximum divergence regions of the two functions; second, a minimization step in which the student network is trained using the regular KD approach over its training data together with the generated auxiliary samples from the first step. Bear in mind that despite the minimization and maximization steps in our technique, our approach can be distinguished from adversarial zero-shot knowledge distillation (ZSKD) techniques such as (Nayak et al., 2019; Micaelli & Storkey, 2019) from different point of views. Frist, ZSKD techniques do not use any of teacher's training data and usually generate data from scratch; however, our technique is data-aware and model-aware. Second, mostly ZSKD work based on continuous adversarial perturbations in image classification tasks, and deploying them in NLP tasks is challenging due to the discrete nature of text. However, our model is able to deal with image and text inputs. Third, some ZSKD approaches such as (Micaelli & Storkey, 2019) are based on an extra generator network; however, our technique does not add any parameter to the network.

We show the success of our backward KD technique in improving KD on both classification and regression tasks over the image and textual data and also in the few-sample KD scenario. We summarize the main contributions of this paper in the following:

- Our technique extracts knowledge from both the forward and backward passes of the teacher and student networks in order to identify the maximum divergence regions between the two functions and generate auxiliary data samples around those regions.
- We provide a solution on how to address the non-differentiability of discrete tokens in NLP applications.
- Our approach is generic and is applicable to any improved KD approach.
- The results of our experiments, show 4% improvement on MNIST with a student network that is 160 times smaller, 1% improvement on the CIFAR-10 dataset with a student that is 9 times smaller, and an average 1.5% improvement on the GLUE benchmark with a distilroBERTa-base student.

2 Related Works

2.1 Knowledge Distillation

In the original KD, the process of transferring knowledge from a teacher to a student model accomplishes by minimizing a loss function between the logits of student and teacher networks. This loss function has been used in addition to the regular training loss function for the student network. In other words, we have an additional loss term in the KD loss function between the softmax outputs of teacher and student networks which is softened by a temperature term.

$$\mathcal{L}_{KD} = \alpha\mathcal{L}\Bigg(\text{Softmax}\big(S(x)\big), y\Bigg) + (1-\alpha)\mathcal{L}\Bigg(\text{Softmax}\bigg(\frac{S(x)}{\tau}\bigg), \text{Softmax}\bigg(\frac{T(x)}{\tau}\bigg)\Bigg) \quad (1)$$

where $S(x)$ and $T(x)$ are student and teacher networks respectively. τ is the temperature parameter and α is a coefficient between $[0, 1]$. This loss function is a linear combination of two loss functions. The first loss function minimizes the difference between the output of the student model and the given true label. The second loss function minimizes the difference between the outputs of the student model and the teacher model. Therefore the whole loss function minimizes the distance between the student and both underlying and teacher functions. Since the teacher network is assumed to be a good approximation of the underlying function, it should be close enough to the underlying function of data samples. Fig. 2-(a) shows a simple example with three data points, an underlying function, a trained teacher and a potential student function that satisfies the KD loss function in eq. 1. However, the problem is, even though the student satisfies the KD objective function and intersects the teacher function close to the training data samples, there is no guarantee that it would fit the teacher network in other regions of the input space as well. In this work, we try to address this problem by deploying the backward gradient information w.r.t the input (we refer to as backward knowledge) in the two networks.

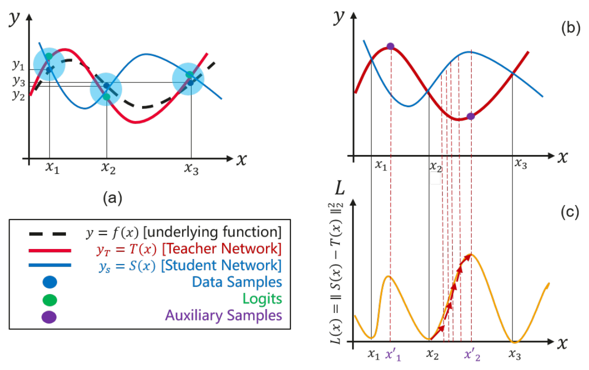
2.2 Sobolev Training for KD

As we mentioned in 2.1 (see Fig. 2.), the KD loss cannot guarantee the student and teacher functions to match over the entire input space. The reason is training two networks based on the original KD loss function would only match their output values on the training samples and not their gradients. There are some works in the literature to address this issue by matching the gradients of the two networks at given training samples during training (Czarnecki et al., 2017; Srinivas & Fleuret, 2018). However, since we usually deal with networks with multidimensional inputs and outputs, the gradients of output vectors w.r.t input vectors give rise to large Jacobin matrices. Matching these Jacobian matrices is not computationally efficient and is not practical in real-world problems.

Sobolev training (Czarnecki et al., 2017) proposes a solution to avoid large Jacobian matrices: instead of directly matching the gradients of the two networks, one can match the projection of the gradients onto a random vector v which is sampled uniformly from the unit sphere. Although this approach can reduce the computational complexity of matching gradients during the training, still computing Jacobian matrices before this projection can be very computationally expensive (especially for NLP applications that deal with large vocabulary sizes). To tackle this problem in our work, we define a new scalar loss function based on an l_2 norm to measure the distance between the teacher and student networks (see Fig. 2-(c)). Gradients of this scalar loss function is a vector with the same size as the input vector x and can be used as a proxy for the network gradients introduced in (Czarnecki et al., 2017; Srinivas & Fleuret, 2018).

3 Methodology: Improving Knowledge Distillation using Backward Pass Knowledge

In this section, we propose our improved KD method based on generating new out of sample points around the areas of the input domain where the student output diverges greatly from the teacher. This approach identifies the areas of the input space $\mathcal{X}$ around which the two functions have maximum distance. Then we generate out of sample points $X' \subset \mathcal{X}$ from the existing training set $X \subset \mathcal{X}$ over those regions. These new generated samples X' can be labelled by the teacher and then $X \leftarrow X \cup X'$



(a)
(b)
(c)
$y = f(x)$ [underlying function]
$y_T = T(x)$ [Teacher Network]
$y_s = S(x)$ [Student Network]
Data Samples
Logits
Auxiliary Samples
$L(x) = \| S(x) - T(x) \|_2^2$

Figure 2: Visualizing the data insufficiency issue for the original KD algorithm. (a) behaviour of the teacher and the student function after training with KD loss. (b) divergence areas between the teacher and the student networks. (c) behaviour of $l_2 - norm$ loss function between teacher and the student and the idea of obtaining auxiliary data samples.

be deployed in the KD's training process to match the student better to the teacher over a broader range in the input space (see Fig. 2). We show that augmenting the training set by adding this auxiliary set improves the performance of KD significantly and leads to a closer match between the student and teacher. Our improved KD approach follows a procedure similar to the $minimax$ principle (Bratko & Gams, 1982) : first, in the maximization step we generate auxiliary data samples; second, in the minimization step we apply regular KD on the union of existing X and generated auxiliary data X'.

To have a better understanding of how this can be cast as an instance of minimax estimator, assume that we are given the data samples $\{x_i, T(x_i))\}_{i=1}^{N}$. The goal is to estimate $T(x)$ by $S(x)$. We may seek an estimator $S(x)$ attaining the $minimax$ principle. In minimax principle, where θ is an estimand and δ is an estimator, we evaluate all estimators according to its maximum risk $R(\theta, \delta)$. An estimator δ_0 , then, is said to be $minimax$ if:

$$\sup_{\theta} R(\theta, \delta_0) = \inf_{\delta \in C} \sup_{\theta \in \Theta} R(\theta, \delta) \tag{2}$$

That is we chose the estimator for the situation that the worst divergence between θ and δ is smallest. We follow a similar insight: i.e. the maximization step computes X', where there is the worst divergence between the teacher and the student. The minimization step finds the weights of the student network such that the difference between the student and teacher for this worst scenario is the smallest.

$$\min_{w} \max_{x} R(T_x, S_{x,w}) \tag{3}$$

3.1 Maximization Step: Generating Auxilary Data based on Backward-KD Loss

In the maximization step of our technique, we define a new loss function (we refer to as the backward KD loss or BKD throughout this paper) to measure the distance between the output of the teacher and the student networks:

$$\mathcal{L}_{\mathcal{BKD}} = ||S(x) - T(x)||_2^2 \tag{4}$$

Here the main idea is that by taking the gradient of $\mathcal{L}_{\mathcal{BKD}}$ loss function in eq. 4 w.r.t the input samples, we can perturb the training samples along the directions of their gradients to increase the loss between two networks. Using this process, we can generate new auxiliary training samples for which the student and the teacher networks are in maximum distance. To obtain these auxiliary data samples, we can consider the following optimization problem.

$$x' = \max_{x \in \mathcal{X}} ||S(x) - T(x)||_2^2 \tag{5}$$

We can solve this problem using stochastic gradient ascent method. Therefore our perturbation formula for each data sample will be:

$$x^{i+1} = x^i + \eta \, \nabla_x \, ||S(x) - T(x)||_2^2 \tag{6}$$

where in this formula η is the perturbation rate. This is an iterative algorithm and i is the iteration index. x^i is a training sample at i^{th} iteration. Each time, we perturb x^i by adding a portion of the gradient of loss to this sample. In general, if we continue the number of iterations until the convergence, there can be a risk for generating out of distribution samples. To avoid this issue, in practice we do the perturbation steps for a limited number of iterations to keep the generated auxiliary samples in data distributions. That is because the data manifold is smooth (manifold assumption) and if we have limited number of data perturbation epochs, the auxiliary samples will stay on a locally linear patch of the manifold. Consider section B in the Appendix for more detail about this algorithm.

Fig. 2 demonstrates our idea using a simple example more clearly. Fig. 2-(a) shows a trained teacher and student functions given the training samples (x_1,y_1), (x_2,y_2), (x_3,y_3). Fig. 2-(c) constructs the $\mathcal{L}_{BKD}$ between these two networks. $\mathcal{L}_{BKD}$ shows where the two networks diverge in the original space. Bear in mind that $\mathcal{L}_{BKD}$ gives a scalar for each input. Hence, the gradient of $\mathcal{L}_{BKD}$ with respect to input variable x will be a vector with the same size as the variable x. Therefore, it does not need to deal with the large dimensionality issue of the Jacobian matrix as described in (Czarnecki et al., 2017). Fig. 2-(c) also illustrates an example of generating one auxiliary sample from the training sample x_2. If we apply eq. 6 to this sample, after several iterations, we will reach to a new auxiliary data point (x_2'). It is evident in Fig. 2-(a) that, as expected, there is a large divergence between the teacher and student networks in (x_2') point.

3.2 Minimization Step: Improving KD with Generated Auxiliary Data

We can apply the maximization step to all given training data to generate their corresponding auxiliary samples. Then by adding the auxiliary samples X' into the training dataset $X \leftarrow X' \cup X$, we can train the student network again based on the original KD algorithm over the updated training set in order to obtain a better output match between the student and teacher networks. Inspired by Mirzadeh et al. (2019), we have used the following KD loss function in our work:

$$\mathcal{L}_{KD} = (1-\lambda)\, H\Big(\sigma\big(S(x)\big), y\Big) + \tau^2\, \lambda\, KL\Bigg(\sigma\Big(\frac{S(x)}{\tau}\Big), \sigma\Big(\frac{T(x)}{\tau}\Big)\Bigg) \tag{7}$$

where $\sigma(.)$ is the softmax function, $H(.)$ is the cross-entropy loss function, $KL(.)$ is the Kullback Leibler divergence, λ is a hyper parameter, τ is the temperature parameter, and y is the true labels.

The intuition behind expecting to get a better KD performance using the updated training data is as follows. Now given the auxiliary data samples which point toward the regions of the input space where the student and teacher have maximum divergence, these regions of input space are no t dark for the original KD algorithm anymore. Therefore, it is expected from the KD algorithm to be able to match the student to the teacher network over a larger input space (see Fig. 4). Moreover, it is worth mentioning that the maximization and minimization steps can be taken multiple times. In this regard, for each maximization step, we need to construct the auxiliary set X' from scratch and we do not need the previously generated auxiliary sets. However, in our few-sample training scenarios where the number of data samples is small, we can keep the auxiliary samples. The maximization steps happen along with the regular KD training. For a better explanation, suppose regular KD needs $n = e \times (h + 2)$ epochs to train the student network. First we perform the minimization step for e epochs. Then, after each minimization step, we perform the maximization step h time in order to generate the auxiliary samples, and enrich the training dataset to achieve a better match between the teacher and student models. These steps happen h times in the algorithm. Also, to pay more attention to the original data samples rather than the auxiliary data samples, at the end of the training, we fine-tune the student model with only the original data samples for e epochs.

3.3 Backward KD for NLP Applications

It is not trivial how to deploy the introduced backward KD approach (i.e. calculating $\nabla_x \mathcal{L}_{BKD}$ for discrete inputs) when data samples come from a discrete domain, such as NLP applications. Here, we propose a solution to how this technique can be adapted for the NLP domain. For neural NLP models, first, we pass the one-hot vectors of the input tokens to the so-called embedding layer of neural networks. Then, these one-hot vectors are converted into embedding vectors (see Fig. 3). The key for our solution is that embedding vectors of input tokens are not discrete and we can take the gradient of loss function w.r.t the embedding vectors z. But the problem is that, in the KD algorithm,

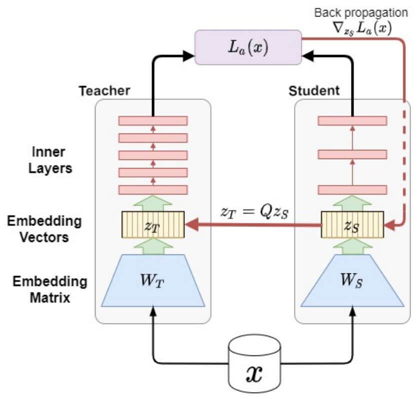


Figure 3: General procedure of utilizing auxiliary samples in NLP models. Here x is the one-hot vector of input tokens, W is the embedding matrix, and z is the embedding vector of x.

we have two networks with different embedding sizes (see Fig. 3). To address this issue, we can take the gradient of the loss function w.r.t one of the embedding vectors (here student embedding vector z_S). However, then we need a transformation matrix like Q to be able to derive the corresponding embedding vector z_T for the teacher network form z_S.

$$z_T = Qz_S \tag{8}$$

We can show that the transform matrix Q is equal to the following equation:

$$Q = W_T W_S^T (W_S W_S^T)^{-1} \tag{9}$$

where in this equation $W_S^T (W_S W_S^T)^{-1}$ is the pseudo inverse of W_S embedding matrix. We refer you to the Appendix to see the proof of this derivation. Therefore, to obtain the auxiliary samples, we can take the gradient of the $\mathcal{L}_{BKD}$ loss function w.r.t the student embedding vector z_S. Then by using equations 10 and 9, we can re-construct z_T during the steps of data perturbation as following.

$$\begin{aligned} z_S^{i+1} &= z_S^i + \eta \nabla_{zS} \mathcal{L}_{BKD} \\ z_T^{i+1} &= W_T W_S^T (W_S W_S^T)^{-1} z_S^{i+1} \end{aligned} \tag{10}$$

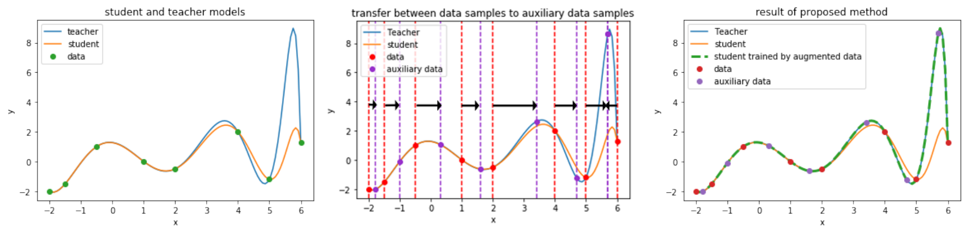
4 Experiments and Results

We designed five experiments to evaluate our proposed method.[1] The first experiment is designed based on synthetic data to visualize the idea behind this technique. The second and third experiments are on the image classification tasks and the last two experiments are in NLP. We followed the general procedure explained in the algorithms section in the Appendix for all of these experiments. For NLP experiments, we applied the method explained in section 3.3 (see section B.2 in the Appendix for more details). The next sections explain the details of these experiments.

4.1 Synthetic data experiment

For visualizing our technique and showing the intuition behind it, we designed a very simple experiment to show how the proposed method works over a synthetic setting. In this experiment, we consider a polynomial function of degree 20 as the trained teacher function. Then, we considered 8 data points on its surface as our data samples to train a student network which is a polynomial function from degree 15 (see Fig. 4-(a)). As it is depicted in this figure, although the student model perfectly fits the given data points, it diverges from the teacher model in some areas between the given points. After applying the backward KD method, we can generate some auxiliary samples in the diverged areas between the teacher and student models in Fig. 4-(b). Then, we augmented

[1]We used PyTorch (https://pytorch.org/) framework (Paszke et al., 2019) for implementing all experiments and Huggingface (https://huggingface.co/) framework (Wolf et al., 2019) in the implementations of NLP experiments.



(a) (b) (c)

Figure 4: Visualizing the generation of auxiliary samples and their utilization in training the student model.

the training data samples with the generated auxiliary samples and trained the student model based on this new augmented dataset. The resulting student model has achieved a much better fit on the teacher model as it is evident in Fig. 4-(c).

4.2 MNIST CLASSIFICATION:

In this experiment, one of our goals was testing the performance of the proposed method in the scenario of extremely small student networks. Because of that, we considered two fully connected neural networks as student and teacher networks for the MNIST dataset classification task. The teacher network consists of only one hidden layer with 800 neurons which leads in 636010 trainable parameters. The student network was an extremely simplified version of the same network with only 5 neurons in the hidden layer. This network has only 3985 trainable parameters, which is 160x smaller than the teacher network. The student network is trained in three different ways: a) from scratch with only training data, b) based on the original KD approach with training data samples augmented by random noise, and c) based on the proposed method. As it is illustrated in table 1, the student network which is trained by using the proposed method achieves much better results in comparison with two other trained networks.

Table 1: Results of experiment on the MNIST dataset

Model	method	#parameters	accuracy on test set
teacher	from scratch	636010	98.14
student	from scratch	3985	87.62
student	original KD	3985	88.04
student	**proposed method**	3985	**91.45**

4.3 CIFAR-10 CLASSIFICATION

The second experiment is conducted on the CIFAR10 dataset with two popular network structures as the teacher and the student networks. In this experiment, we used the inception v3 (Szegedy et al., 2016) network as the teacher and mobileNet v2 (Sandler et al., 2018) as the student. The teacher is approximately 9 times bigger than the student. We repeated the previous experiment on CIFAR10 by using these two networks. Table 2 shows the results of this experiment.

Table 2: Results of experiment on CIFAR10 dataset

Model	method	#parameters	accuracy on test set
inception v3 (teacher)	from scratch	21638954	95.41%
mobilenet (student)	from scratch	2236682	91.17%
mobilenet (student)	original KD	2236682	91.74%
mobilenet (student)	**proposed method**	2236682	**92.60%**

4.4 GLUE TASKS

The third experiment is designed based on General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) and roBERTa family language models (Liu et al., 2019; Sanh et al., 2019). The GLUE benchmark is a set of nine language understanding tasks, which are designed to evaluate the performance of natural language understanding systems. roBERTa models (roBERTa-large, roBERTa-base, and distilroBERTa) are BERT (Devlin et al., 2018) based language understanding pre-trained models where roBERTa-large and roBERTa-base are the cumbersome versions which are proposed in (Liu et al., 2019) and have 24 and 12 transformer layers respectively. distilroBERTa is the compressed version of these models with 6 transformer layers and has been trained based on KD procedure proposed in (Sanh et al., 2019) with utilizing the roBERTa-base as the teacher. The general procedure in GLUE tasks is fine-tuning the pre-trained models for its downstream tasks and the average performance score. Here, we fine-tuned the distilroBERTa model based on the proposed method by utilizing the fine-tuned roBERTa-large teacher for each of these tasks. As it is shown in table 3, the proposed method could improve the distilroBERTa performance on most of these tasks.

Table 3: Results of experiment on GLUE tasks

Model (Network)	ColA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	Score
roBERTa-large (Teacher)	60.56	96.33	89.95	91.75	91.01	89.11	93.08	79.06	56.33	85.82
DistilroBERTa (Student)	56.61	**92.77**	84.06	87.28	90.8	84.14	**91.36**	65.70	56.33	78.78
Our DistilroBERTa (Student)	**60.49**	92.51	**87.25**	**87.56**	**91.21**	**85.1**	91.19	**71.11**	56.33	**80.30**

4.5 GLUE TASKS WITH FEW SAMPLE POINTS

In this experiment, we modified the previous experiment slightly to investigate the performance of the proposed method in the few data sample scenario. Here we randomly select a small portion of samples in each data set and fine-tuned the distilroBERTa based on these samples. For CoLA, MRPC, STS-B, QNLI, RTE, and WNLI, 10% of data samples and for SST-2, QQP, and MNLI 5% of them in the dataset are used for fine-tuning the student model.

Table 4: Results of few sample experiment on GLUE tasks

Model (Network)	ColA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	Score
roBERTa-large (Teacher)	60.56	96.33	89.95	91.75	91.01	89.11	93.08	79.06	56.33	85.82
DistilroBERTa (Student)	43.82	91.05	76.96	81.51	84.92	75.88	83.94	52.07	56.33	71.90
Our DistilroBERTa (Student)	**44.11**	**91.74**	**77.20**	**82.82**	**85.32**	**76.75**	**84.34**	**56.31**	56.33	**72.76**

5 CONCLUSION

In this paper, we have introduced the backward KD method and showed how we can use the backward knowledge of teacher model to train the student model. Based on this method, we could easily locate the diverge areas between teacher and student model in order to acquire auxiliary samples at those areas with utilizing the gradient of the networks and use these samples in the training procedure of the student model. We showed that our proposal can be efficiently applied to the KD procedure to improve its performance. Also, we introduced an efficient way to apply backward KD on discrete domain applications such as NLP tasks. In addition to the synthetic experiment which is performed to visualize the mechanism of our method, we tested its performance on several image and NLP tasks. Also, we examined the extremely small student and the few sample scenarios in two of these experiments. We showed that the backward KD can improve the performance of the trained student network in all of these practices. We believe that all auxiliary samples do not have the same contribution to improving the performance of the student model. Also perturbing all data samples can be computationally expensive in large datasets.

REFERENCES

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

Ivan Bratko and Matjaz Gams. Error analysis of the minimax principle. In *Advances in computer chess*, pp. 1–15. Elsevier, 1982.

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

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

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev Training for Neural Networks. *arXiv e-prints*, art. arXiv:1706.04859, June 2017.

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

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6381–6385. IEEE, 2019.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3771–3778, 2019.

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

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

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

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

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

Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, pp. 9547–9557, 2019.

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

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

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

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

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

Suraj Srinivas and Francois Fleuret. Knowledge Transfer with Jacobian Matching. *arXiv e-prints*, art. arXiv:1803.00443, March 2018.

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

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

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

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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

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

Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. *arXiv preprint arXiv:1909.12579*, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Alexander Wong, Mahmoud Famuori, Mohammad Javad Shafiee, Francis Li, Brendan Chwyl, and Jonathan Chung. Yolo nano: a highly compact you only look once convolutional neural network for object detection. *arXiv preprint arXiv:1910.01271*, 2019.

APPENDIX

A TRANSFORM MATRIX BETWEEN STUDENT AND TEACHER EMBEDDING

If $W_S \in \mathbb{R}^{d_1 \times |V|}$ be the embedding matrix of the student network and $W_T \in \mathbb{R}^{d_2 \times |V|}$ be the embedding matrix of the teacher network, where $|V|$ is the vocabulary size, d_1 and d_2 are the embedding vector size of the student and the teacher networks respectively. If $x \in \{0,1\}^{|v|}$ be the one-hot vector of a token in a text document and if $z_S = W_S x$ and $z_T = W_T x$ be the student and teacher embedding vectors of x, then there exists a transform matrix $Q \in \mathbb{R}^{d_2 \times d_1}$ such that:

$$z_T = Q z_S \tag{11}$$

Proof:

$$z_T = W_T x \tag{12}$$

$$z_S = W_S x \tag{13}$$

We want to find a transform matrix Q such that:

$$W_T = Q W_S \tag{14}$$

For this purpose we can solve the following optimization problem by using list square method:

$$\min_Q ||W_T - Q W_S||^2 \tag{15}$$

By solving the above optimization problem using the least squares method, we achieves the following solution for Q:

$$Q = W_T W_s^T (W_s W_s^T)^{-1} \tag{16}$$

Now, from Eq. 14 we have:

$$W_T = Q W_s \tag{17}$$

$$W_T x = Q W_s x \tag{18}$$

$$z_T = Q z_s \tag{19}$$

B ALGORITHMS

This section explains the details of the proposed algorithm. First, we will see the general procedure of the algorithm. Then in section A.1, the pseudo-code of the algorithm will be explained; and finally, in section A.2, we will see the modified version of the algorithm for NLP tasks. If we consider for training the student model, vanilla KD needs n epochs, here the idea is dividing these epochs into $h+2$ sections where each section consists of e epochs ($n = (h+2)e$) Then we have:

Pre-training Step (e epochs): We train the student network based on the original KD procedure for (e epochs). In this step, the student network will get close to the teacher network around the given training samples and will diverge from the teacher in some other areas.

Iterative Min Max Steps ($h \times e$ epochs): We do the following two steps iteratively h times:
1) Using the partially trained student network and the trained teacher network, we use the proposed maximization step in 3.2 for generating an auxiliary dataset.
2) Combine the auxiliary data with the training dataset and train the student network based on the augmented dataset using the original KD procedure for e epochs.

Fine-tuning Step (e epochs): Finally, fine-tune the student network using original KD only based on the train samples for e epochs to pay more attention to the original data samples.

B.1 Algorithm 1 (general pseudo-code)

Algorithm 1 explains the details of the proposed method in previous section and section 3 of the paper.We pass the student network $S(.)$, the teacher network $T(.)$, the input dataset X, the number of training epochs e, the number of maximization steps h, and the number of sample perturbing steps l to the proposed KD function. This algorithm assumes that the teacher network $T(.)$ has been trained, and will be used to train the student network $S(.)$. Also, we assume X' is the set of the augmented data samples. We first initialize X' with data set X in line 3 of the algorithm. The basic idea is that each time we train the student network using the Vanilla-KD function for a few training epochs e in the outer loop of line 4. Then, in line 6 first, we re-initialize X' with dataset X, and in lines 7 to 11, we perturb data samples in X' using the gradient of the loss between teacher and student iteratively to generate new auxiliary samples. In line 12, we replace X with the union of X and X' sets. In the next iteration of the loop in line 4, the Vanilla-KD function will be fed with the augmented data samples X'. Note that just in the first iteration, Vanilla-KD function is fed with the original data set X which is identical to pre-training step of the previous section.

Algorithm 1

```
1:  function PROPOSED-KD($S$,$T$,$X$, $e$, $h$, $l$)
2:      $X' \leftarrow X$
3:      for $i = 1$ to $h + 1$ do
4:          VANILLA-KD($S$,$T$,$X'$,$e$)
5:          $X' \leftarrow X$
6:          for $x'$ in $X'$ do
7:              for $j = 1$ to $l$ do
8:                  $x' \leftarrow x' + \eta \nabla_x ||S(x') - T(x')||_2^2$
9:              end for
10:         end for
11:         $X' \leftarrow X' \cup X$
12:     end for
13:     VANILLA-KD($S$,$T$,$X$,$e$)
14:     return $S$
15: end function
```

B.2 Algorithm 2 (NLP task's version)

Algorithm 2 explains how to apply the proposed method in NLP tasks. This algorithm is almost similar to algorithm 1. The only main difference is in the way we feed the networks. Here instead of considering the one-hot index vectors of tokens in the text documents, we consider the embedding vectors z_S and z_T of the input vector x (see lines 5 and 6 in the algorithm). Then we feed each of the teacher and the student networks separately using their own embedding vectors. In line 16, we use the transform method proposed in section 3.2 of the paper to transform student's perturbed embedding vectors into teacher's embedding vectors.

Algorithm 2

```
1:  function Proposed-KD($S$,$T$,$X$, $e$, $h$, $l$)
2:      $W_T \leftarrow$ Embedding-matrix($T$)
3:      $W_S \leftarrow$ Embedding-matrix($S$)
4:      $Z_T \leftarrow W_T X$
5:      $Z_S \leftarrow W_S X$
6:      $Z'_T \leftarrow Z_T$
7:      $Z'_S \leftarrow Z_S$
8:      for $i = 1$ to $h + 1$ do
9:          Vanilla-KD($S$,$T$,$Z'_T$, $Z'_S$,$e$)
10:         $Z'_T \leftarrow Z_T$
11:         $Z'_S \leftarrow Z_S$
12:         for $(z'_S, z'_T)$ in $(Z'_S, Z'_T)$ do
13:             for $j = 1$ to $l$ do
14:                 $z'_S \leftarrow z'_S + \eta \nabla_{z_S} ||S(z'_S) - T(z'_S)||_2^2$
15:                 $z'_T \leftarrow W_T W_S (W_S W_S^T)^{-1} z'_S$
16:             end for
17:         end for
18:         $Z'_S \leftarrow Z'_S \cup Z_S$
19:         $Z'_T \leftarrow Z'_T \cup Z_T$
20:     end for
21:     Vanilla-KD($S$,$T$,$Z_T$, $Z_S$,$e$)
22:     return $S$
23: end function
```