MATE-KD: Masked Adversarial TExt, a Companion to Knowledge Distillation

Ahmad Rashid[1]*, **Vasileios Lioutas**[2]*†, **Mehdi Rezagholizadeh**[1]
[1]Huawei Noah's Ark Lab, [2]University of British Columbia
ahmad.rashid@huawei.com, contact@vlioutas.com,
mehdi.rezagholizadeh@huawei.com

Abstract

The advent of large pre-trained language models has given rise to rapid progress in the field of Natural Language Processing (NLP). While the performance of these models on standard benchmarks has scaled with size, compression techniques such as knowledge distillation have been key in making them practical. We present MATE-KD, a novel text-based adversarial training algorithm which improves the performance of knowledge distillation. MATE-KD first trains a masked language model-based generator to perturb text by maximizing the divergence between teacher and student logits. Then using knowledge distillation a student is trained on both the original and the perturbed training samples. We evaluate our algorithm, using BERT-based models, on the GLUE benchmark and demonstrate that MATE-KD outperforms competitive adversarial learning and data augmentation baselines. On the GLUE test set our 6 layer RoBERTa based model outperforms $\text{BERT}_{\text{LARGE}}$.

1 Introduction

Transformers (Vaswani et al., 2017) and transformer-based Pre-trained Language Models (PLMs) (Devlin et al., 2019) are ubiquitous in applications of NLP. They are highly parallelizable and their performance scales well with an increase in model parameters and data. Increasing model parameters depends on the availability of computational resources and PLMs are typically trained on unlabeled data which is cheaper to obtain.

Recently, the trillion parameter mark has been breached for PLMs (Fedus et al., 2021) amid serious environmental concerns (Strubell et al., 2019). However, without a change in our current training paradigm , training larger models may be unavoidable (Li et al., 2020). In order to deploy these models for practical applications such as for virtual personal assistants, recommendation systems, e-commerce platforms etc. model compression is necessary.

Knowledge Distillation (KD) (Buciluǎ et al., 2006; Hinton et al., 2015) is a simple, yet powerful knowledge transfer algorithm which is used for neural model compression (Jiao et al., 2019; Sanh et al., 2019), ensembling (Hinton et al., 2015) and multi-task learning (Clark et al., 2019). In NLP, KD for compression has received renewed interest in the last few years. It is one of the most widely researched algorithms for the compression of transformer-based PLMs (Rogers et al., 2020).

One key feature which makes KD attractive is that it only requires access to the teacher's output or logits and not the weights themselves. Therefore, if a trillion parameter model resides on the cloud, an API level access to the teacher's output is sufficient for KD. Consequently, the algorithm is architecture agnostic, i.e., it can work for any deep learning model and the student can be a different model from the teacher.

Recent works on KD for transfer learning with PLMs extend the algorithm in two main directions. The first is towards "model" distillation (Sun et al., 2019; Wang et al., 2020; Jiao et al., 2019) i.e. distilling the intermediate weights such as the attention weights or the intermediate layer output of transformers. The second direction is towards curriculum-based or progressive KD (Sun et al., 2020; Mirzadeh et al., 2019; Jafari et al., 2021) where the student learns one layer at a time or from an intermediary teacher, known as a teacher assistant. While these works have shown accuracy gains over standard KD, they have come at the cost of architectural assumptions, least of them a common architecture between student and teacher, and

* Equal Contribution
† Work done during an internship at Huawei Noah's Ark Lab.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 1062–1071
August 1–6, 2021.

greater access to teacher parameters and intermediate outputs. Another issue is that the decision to distill one teacher layer and to skip another is arbitrary. Still the teacher typically demonstrates better generalization

We are interested in KD for model compression and study the use of adversarial training (Goodfellow et al., 2014) to improve student accuracy using just the logits of the teacher as in standard KD. Specifically, our work makes the following contributions:

- We present a text-based adversarial algorithm, MATE-KD, which increases the accuracy of the student model using KD.
- Our algorithm only requires access to the teacher's logits and thus keeps the teacher and student architecture independent.
- We evaluate our algorithm on the GLUE (Wang et al., 2018) benchmark and demonstrate improvement over competitive baselines.
- On the GLUE test set, we achieve a score of 80.9, which is higher than $\text{BERT}_{\text{LARGE}}$
- We also demonstrate improvement on out-of-domain (OOD) evaluation.

2 Related Work

2.1 Knowledge Distillation

We can summarize the knowledge distillation loss, $\mathcal{L}$, as following:

$$\begin{aligned} \mathcal{L}_{CE} &= \mathcal{H}_{CE}\big(y, S(X)\big) \\ \mathcal{L}_{KD} &= \mathcal{T}^2 D_{KL}\Big(\sigma(\frac{z_t(X)}{\mathcal{T}}), \sigma(\frac{z_s(X)}{\mathcal{T}})\Big) \\ \mathcal{L} &= (1-\lambda)\mathcal{L}_{CE} + \lambda\mathcal{L}_{KD} \end{aligned} \quad (1)$$

where $\mathcal{H}_{CE}$ represents the cross entropy between the true label y and the student network prediction $S(X)$ for a given input X, D_{KL} is the KL divergence between the teacher and student predictions softened using the temperature parameter $\mathcal{T}$, $z(X)$ is the network output before the softmax layer (logits), and $\sigma(.)$ indicates the softmax function. The term λ in the above equation is a hyper-parameter which controls the amount of contribution from the cross entropy and KD loss.

Patient KD (Sun et al., 2019) introduces an additional loss to KD which distills the intermediate layer information onto the student network. Due to a difference in the number of student and teacher layers they propose either skipping alternate layers or distilling only the last few layers. TinyBERT (Jiao et al., 2019) applies embedding distillation and intermediate layer distillation which includes hidden state distillation and attention weight distillation. Although it achieves strong results on the GLUE benchmark, this approach is infeasible for very large teachers. MiniLM (Wang et al., 2020) proposed an interesting alternative whereby they distill the key, query and value matrices of the final layer of the teacher.

2.2 Adversarial Training

Adversarial examples are small perturbations to training samples indistinguishable to humans but enough to produce misclassifications by a trained neural network. Goodfellow et al. (2014) showed that adding these examples to the training set can make a neural network model robust to perturbations. Miyato et al. (2016) adapt adversarial training to text classification and improve performance on a few supervised and semi-supervised text classification tasks.

In NLP, adversarial training has surprisingly been shown to improve generalization as well (Cheng et al., 2019; Zhu et al., 2019). Cheng et al. (2019) study machine translation and propose making the model robust to both source and target perturbations, generated by swapping the embedding of a word with that of its synonym. They model small perturbations by considering word swaps which cause the smallest increase in the loss gradient. They achieve a higher BLEU score on Chinese-English and English-German translation compared to the baseline.

Zhu et al. (2019) propose a novel adversarial training algorithm, FreeLB, to make gradient-based adversarial training efficient by updating both embedding perturbations and model parameters simultaneously during the backward pass of training. They show improvements on multiple language models on the GLUE benchmark. Embedding perturbations are attractive because they produce stronger adversaries (Zhu et al., 2019) and keep the system end-to-end differentiable as the embeddings are continuous. The salient features of adversarial training for NLP are a) a *minimax* formulation where adversarial examples are generated to maximize a loss function and the model is trained to

minimize the loss function and b) a way of keeping the perturbations small such as a norm-bound on the gradient (Zhu et al., 2019) or replacing words by their synonyms (Cheng et al., 2019).

If these algorithms are adapted to KD one key challenge is the embedding mismatch between the teacher and student. Even if the embedding size is the same, the student embedding needs to be frozen to match the teacher embedding and freezing embeddings typically leads to lower performance. If we adapt adversarial training to KD, one key advantage is that access to the teacher distribution relaxes the requirement of generating label preserving perturbations. These considerations have prompted us to design an adversarial algorithm where we perturb the actual text instead of the embedding. Rashid et al. (2020) also propose a text-based adversarial algorithm for the problem of zero-shot KD (where the teacher's training data is unavailable), but their generator instead of perturbing text generates new samples and requires additional losses and pre-training to work well.

2.3 Data Augmentation

One of the first works on BERT compression (Tang et al., 2019) used KD and proposed data augmentation using heuristics such as part-of-speech guided word replacement. They demonstrated improvement on three GLUE tasks. One limitation of this approach is that the heuristics are task specific. Jiao et al. (2019) present an ablation study in their work whereby they demonstrate a strong contribution of data augmentation to their KD algorithm performance. They augment the data by randomly selecting a few words of a training sentence and replacing them with words with the closest embedding under cosine distance. Our adversarial learning algorithm can be interpreted as a data augmentation algorithm, but instead of a heuristic approach we propose a principled end-to-end differentiable augmentation method based on adversarial learning.

Khashabi et al. (2020) presented a data augmentation technique for question answering whereby they took seed questions and asked humans to perturb only a few tokens to generate new ones. The human annotators could modify the label if needed. They demonstrated improved generalization and robustness with the augmented data. We will demonstrate that our algorithm is built on similar principles but does not require humans in the loop. Instead of human annotators to modify the labels we use the teacher.

3 Methodology

We propose an algorithm that involves co-training and deploy an adversarial text generator while training a student network using KD. Figure 1 gives an illustration of our architecture.

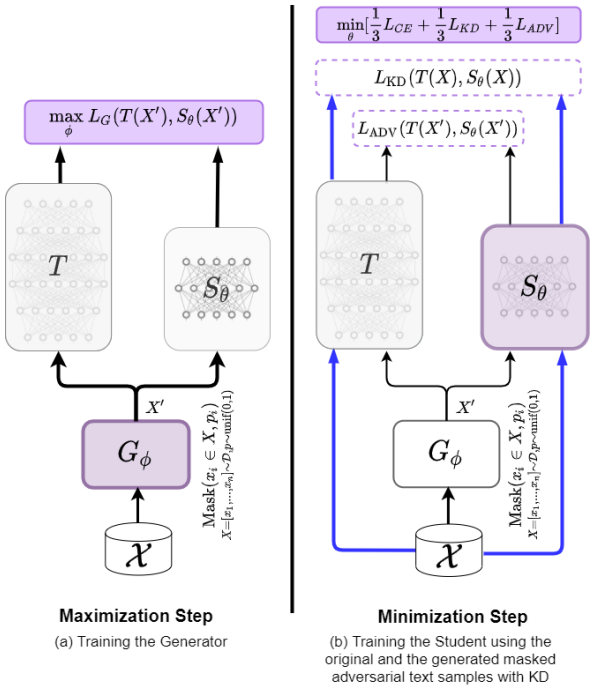

Figure 1: Illustration of the maximization and minimization steps of MATE-KD

3.1 Generator

The text generator is simply a pre-trained masked language model which is trained to perturb training samples adversarially. We can frame our technique in a *minimax* regime such that in the maximization step of each iteration, we feed the generator with a training sample with few of the tokens replaced by masks. We fix the rest of the sentence and replace the masked tokens with the generator output to construct a pseudo training sample X'. This pseudo sample is fed to both the teacher and the student models and the generator is trained to maximize the divergence between the teacher and the student. We present an example of the masked generation process in Figure 2. The student is trained during the minimization step.

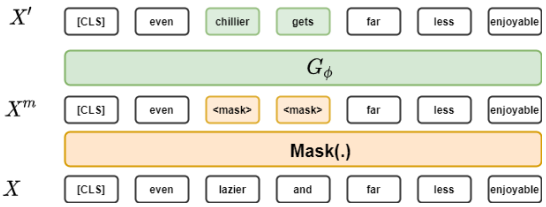

Figure 2: This figure illustrates how a training sample will be randomly masked and then fed to the text generator G_ϕ to get the pseudo training sample.

3.2 Maximization Step

The generator is trained to generate pseudo samples by maximizing the following loss function:

$$\max_{\phi} \mathcal{L}_G(\phi) = D_{KL}\Big(T\big(G_\phi(X^m)\big), S_\theta\big(G_\phi(X^m)\big)\Big), \tag{2}$$

where D_{KL} is the KL divergence, $G_\phi(.)$ is the text generator network with parameters ϕ, $T(\cdot)$ and $S_\theta(\cdot)$ are the teacher and student networks respectively, and X^m is a randomly masked version of the input $X = [x_1, x_2, ..., x_n]$ with n tokens.

$$\begin{aligned} &\forall x_i \in X = [x_1, ..., x_i, ..., x_n] \sim \mathcal{D}, \\ &x_i^m = \underset{p\sim \text{unif}(0,1)}{\text{Mask}}(x_i \in X, p_i) \\ &= \begin{cases} x_i, & p_i \geq \rho \\ < \text{mask} >, & \text{o.w.} \end{cases} \end{aligned} \tag{3}$$

where $\text{unif}(0, 1)$ represents the uniform distribution, and the $\text{Mask}(\cdot)$ function masks the tokens of inputs sampled from the data distribution $\mathcal{D}$ with the probability of ρ. The term ρ can be treated as a hyper-parameter in our technique. In summary, for each training sample, we randomly mask some tokens according to the samples derived from the uniform distribution and the threshold value of ρ.

Then in the forward pass, the masked sample, X^m, is fed to the generator to obtain the output pseudo text based on the generator predictions of the mask tokens. The generator needs to output a one-hot representation but using an *argmax* inside the generator would lead to non-differentiability. Instead we apply the Gumbel-Softmax (Jang et al., 2016), which, is an approximation to sampling from the *argmax*. Using the straight through estimator (Bengio et al., 2013) we can still apply *argmax* in the forward pass and can obtain text, X' from the network outputs:

$$X' = \underset{\text{FORWARD}}{G_\phi(X^m)} = \text{argmax}\big(\sigma_{\text{Gumbel}}(z_\phi(X^m))\big) \tag{4}$$

where

$$\sigma_{\text{Gumbel}}(z_i) = \frac{\exp\Big(\big(\log(z_i) + g_i\big)/\tau\Big)}{\Sigma_{j=1}^{K} \exp\Big(\big(\log(z_j) + g_j\big)/\tau\Big)} \tag{5}$$

$g_i \sim \text{Gumbel}(0, 1)$ and $z_\phi(.)$ returns the logits produced by the generator for a given input. τ is the temperature in equation 5.

In the backward pass, the generator simply applies the gradients from the Gumbel-Softmax without the *argmax* :

$$\underset{\text{BACKWARD}}{G_\phi(X^m)} = \sigma_{\text{Gumbel}}(z_\phi(X^m)) \tag{6}$$

3.3 Minimization Step

In the minimization step, the student network is trained to minimize the gap between the teacher and student predictions and match the hard labels from the training data by minimizing the following loss equation:

$$\min_{\theta} \mathcal{L}_{\text{MATE-KD}}(\theta) = \frac{1}{3}\mathcal{L}_{CE}(\theta) + \frac{1}{3}\mathcal{L}_{KD}(\theta) + \frac{1}{3}\mathcal{L}_{ADV}(\theta) \tag{7}$$

where

$$\mathcal{L}_{ADV}(\theta) = D_{KL}\Big(T(X'), S_\theta(X')\Big) \tag{8}$$

In Equation 7, the terms L_{KD} and L_{CE} are the same as Equation 1, $L_{KD}(\theta)$ and $L_{ADV}(\theta)$ are used to match the student with the teacher, and $L_{CE}(\theta)$ is used for the student to follow the ground-truth labels y.

Bear in mind that our $\mathcal{L}_{\text{MATE-KD}}(\theta)$ loss is different from the regular KD loss in two aspects: first, it has the additional adversarial loss, $\mathcal{L}_{ADV}$ to minimize the gap between the predictions of the student and the teacher with respect to the generated masked adversarial text samples, X', in the maximization step; second, we do not have the weight term λ form KD in our technique any more (i.e. we consider equal weights for the three loss terms in $\mathcal{L}_{\text{MATE-KD}}$).

3.4 Rationale Behind the Masked Adversarial Text Generation for KD

The rationale behind generating partially masked adversarial texts instead of generating adversarial texts from scratch (that is equivalent to masking the input of the text generator entirely) is three-fold:

1. Partial masking is able to generate more realistic sentences compared to generating them from scratch when trained only to increase teacher and student divergence. We present a few generated sentences in section 4.6

2. Generating text from scratch increases the chance of generating OOD data. Feeding OOD data to the KD algorithm leads to matching the teacher and student functions across input domains that the teacher is not trained on.

3. By masking and changing only a few tokens of the original text, we constrain the amount of perturbation as is required for adversarial training.

In our MATE-KD technique, we can tweak the ρ to control our divergence from the data distribution and find the sweet spot which gives rise to maximum improvement for KD. We also present an ablation on the effect of this parameter on downstream performance in section 4.5.

4 Experiments

We evaluated MATE-KD on all nine datasets of the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmark which include classification and regression. These datasets can be broadly divided into 3 families of problems. Single set tasks which include linguistic acceptability (CoLA) and sentiment analysis (SST-2). Similarity and paraphrasing tasks which include paraphrasing (MRPC and QQP) and a regression task (STS-B). Inference tasks which include Natural Language Inference (MNLI, WNLI, RTE) and Question Answering (QNLI).

4.1 Experimental Setup

We evaluate our algorithm on two different setups. On the first the teacher model is $\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019) and the student is initialized with the weights of DistillRoBERTa (Sanh et al., 2019). $\text{RoBERTa}_{\text{LARGE}}$ consists of 24 layers with a hidden dimension of 1024 and 16 attention heads and a total of 355 million parameters. We use the pre-trained model from Huggingface (Wolf et al., 2019). The student consists of 6 layers, 768 hidden dimension, 8 attention heads and 82 million parameters. Both models have a vocabulary size of 50,265 extracted using the Byte Pair Encoding (BPE) (Sennrich et al., 2016) tokenization method.

On our second setup, the teacher model is $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) and the student model is initialized with the weights of DistilBERT which consists of 6 layers with a hidden dimension of 768 and 8 attention heads. The pre-trained models are taken from the authors' release. The teacher and the student are 110M and 66M parameters respectively with a vocabulary size of 30,522 extracted using BPE.

Hyper-parameters We fine-tuned the RoBERTa student model and picked the best checkpoint that gave the highest score on the dev set of GLUE. These hyper-parameters were fixed for the GLUE test submissions as well as the BERT experiments.

We used the AdamW (Loshchilov and Hutter, 2017) optimizer with the default values. In addition, we used a linear decay learning rate scheduler with no warmup steps. We set the masking probability p to be 0.3. Additionally, we set the value n_G to 10 and n_S to 100. The learning rate, number of epochs, and other hyper-parameters are presented on table 8 of Appendix A.

Hardware Details We trained all models using a single NVIDIA V100 GPU. We used mixed-precision training (Micikevicius et al., 2018) to expedite the training procedure. All experiments were run using the PyTorch[1] framework.

4.2 Results

Table 1 presents the results of MATE-KD on the GLUE dev set. Even though the datasets have different evaluation metrics, we present the average of all scores as well, which is used to rank the submissions to GLUE. Our first baseline is the fine-tuned DistilRoBERTa and then we compare with KD, FreeLB, FreeLB plus KD, and TinyBERT (Jiao et al., 2019) data augmentation plus KD.

We observe that FreeLB (Zhu et al., 2019) significantly improves the fine-tuned student by around 1.2 points on average. However, when we apply both FreeLB + KD, we do not see any further improvement whereas applying KD alone improves

[1] https://pytorch.org/

Method	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Score
$\text{RoBERTa}_{\text{Large}}$ (teacher)	68.1	96.4	91.9	92.3	91.5	90.2	94.6	86.3	85.28
DistilRoBERTa (student)	56.6	92.7	89.5	87.2	90.8	84.1	91.3	65.7	78.78
Student + FreeLB	58.1	93.1	90.1	88.8	90.9	84.0	91.0	67.8	80.01
Student + FreeLB + KD	58.1	93.2	90.5	88.6	91.2	83.7	90.8	68.2	80.06
Student + KD	60.9	92.5	90.2	89.0	91.6	84.1	91.3	71.1	80.77
Student + TinyBERT Aug + KD	61.3	93.3	90.4	88.6	91.7	84.4	91.6	72.5	81.12
Student + MATE-KD (Ours)	65.9	94.1	91.9	90.4	91.9	85.8	92.5	75.0	**82.64**

Table 1: Dev Set results using DistilRoBERTa as the student on the GLUE benchmark. The score for the WNLI task is 56.3 for all models.

the score by about 2 points. This is so because FreeLB relies on the model (student) output rather than the teacher output to generate adversarial perturbation and therefore cannot benefit from KD. As previously discussed, FreeLB relies on embedding perturbation and in order to generate the teacher output on the perturbed student, both the embeddings need to be tied together, which is infeasible due to the size and training requirements.

We also compared against the data augmentation algorithm of TinyBERT. We ran their code to generate the augmented data offline. Although they augment the data about 20 times depending on the GLUE task, we observed poor results if we use all this data to fine-tune with KD. We only generated 1x augmented data and saw an average improvement of 0.35 score over KD. MATE-KD achieves the best result among the student models on all GLUE tasks and achieves an average improvement of 1.87 over just KD. We also generated the same number of adversarial samples as the training data.

We present the results on the test set of GLUE on Table 2. We list the number of parameters for each model. The results of $\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2019), TinyBERT and MobileBERT (Sun et al., 2020) are taken from the GLUE leaderboard[2]. The KD models have $\text{RoBERTa}_{\text{Large}}$, fine-tuned without ensembling as the teacher.

TinyBERT and MobileBERT are the current state-of-the-art 6 layer transformer models on the GLUE leaderboard. We include them in this comparison although their teacher is $\text{BERT}_{\text{BASE}}$ as opposed to $\text{RoBERTa}_{\text{Large}}$. We make the case that one reason we can train with a larger and more powerful teacher is that we only require the logits of the teacher while training. Most of the works in the literature proposing intermediate layer distillation (Jiao et al., 2019; Sun et al., 2020, 2019) are trained on 12 layer BERT teachers. As PLMs get bigger in size, feasible approaches to KD will involve algorithms which rely on only minimal access to teachers.

We apply a standard trick to boost the performance of STS-B and RTE, i.e., we initialize these models with the trained checkpoint of MNLI (Liu et al., 2019). This was not done for the dev results. The WNLI score is the same for all the models and although, not displayed on the table, is part of the average score. We make a few observations from this table. Firstly, using KD a student with a powerful teacher can overcome a significant difference in parameters between competitive models. Secondly, our algorithm significantly improves KD with an average 2 point increase on the unseen GLUE testset. Our model is able to achieve state-of-the-art results for a 6 layer transformer model on the GLUE leaderboard.

We also evaluate our algorithm using $\text{BERT}_{\text{BASE}}$ as teacher and DistilBERT as student on GLUE benchmark. WNLI results are the same for all and they are used to calculate the average. We compare against the teacher, student, and KD plus TinyBERT augmentation. Here, remarkably MATE-KD can beat the teacher performance on average. On the two largest datasets in GLUE, QQP and MNLI, we beat and match the teacher performance respectively.

We observe that MATE-KD outperforms its competitors when both the teacher is twice the size and four times the size of the student. This may be because the algorithm generates adversarial examples based on the teacher's distribution. A well designed adversarial algorithm can help us probe parts of the teacher's distribution not spanned by the training data leading to better generalization.

[2] https://gluebenchmark.com/leaderboard

Model (Param.)	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE	Score
TinyBERT (66M)	51.1	93.1	87.3/82.6	85.0/83.7	71.6/89.1	84.6/83.2	90.4	70.0	78.1
$\text{BERT}_{\text{BASE}}$ (110M)	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	78.3
MobileBERT (66M)	51.1	92.6	88.8/84.5	86.2/84.8	70.5/88.3	84.3/83.4	91.6	70.4	78.5
DistilRoB. + KD (82M)	54.3	93.1	86.0/80.8	85.7/84.9	71.9/89.5	83.6/82.9	90.8	74.1	78.9
$\text{BERT}_{\text{LARGE}}$ (340M)	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	80.5
MATE-KD (82M)	56.0	94.9	91.7/88.7	88.3/87.7	72.6/89.7	85.5/84.8	92.1	75.0	**80.9**

Table 2: Leaderboard test results of experiments on GLUE tasks. The score for the WNLI task is 65.1 for all models.

Method	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Score
$\text{BERT}_{\text{BASE}}$ (teacher)	59.5	93.1	86.7	88.4	91.0	84.6	91.5	68.2	79.9
DistilBERT (student)	51.3	91.3	87.5	86.9	88.5	82.1	89.2	59.9	77.0
Student + TinyBERT Aug. + KD	55.2	91.9	87.0	87.8	89.5	82.1	89.7	68.6	78.7
Student + MATE-KD (Ours)	60.4	92.2	88.0	88.5	91.4	84.5	91.2	70.0	**80.3**

Table 3: Dev results on the GLUE benchmark using DistilBERT as the student model. WNLI results are 56.3 for all models.

4.3 OOD Evaluation

It has been shown that strong NLU models tend to learn spurious surface level patterns from the dataset (Poliak et al., 2018; Gururangan et al., 2018) and may perform poorly on carefully constructed OOD datasets. In Table 4 we present the evaluation of MATE-KD (RoBERTa-based) trained on MNLI and QQP on the HANS (McCoy et al., 2019) and the PAWS (Zhang et al., 2019) evaluation sets respectively.

Model	HANS	PAWS
DistilRoBERTa	58.9	36.5
Mate-KD	66.6	38.3

Table 4: Model Performance on OOD evaluation sets HANS and PAWS for MNLI and QQP respectively

We use the same model checkpoint as the one presented in Table 1 and compare against DistilRoBERTa. We observe that MATE-KD improves the baseline performance on both evaluation datasets. The performance increase on HANS is larger. We can conclude that the algorithm improvements are not due to learning spurious correlations and biases in the dataset.

4.4 Ablation Study

Table 5 presents the contribution of the generator and adversarial learning to MATE-KD. We first present the result of MATE-KD on all the GLUE datasets (except WNLI) and compare against the effect of removing the adversarial training and then the generator altogether. When we remove the adversarial training, we essentially remove the maximization step and do not train the generator. The generator in this setting is a pre-trained masked language model. In the minimization step, we still generate pseudo samples and apply all losses. The setting where we remove the generator is akin to a simple KD.

We observe that the generator improves KD by an average of 1.3 and the adversarial training increases the score further by 0.6.

4.5 Sensitivity Analysis

Our algorithm does not require the loss interpolation weight of KD but instead relies on one additional parameter, ρ, which is the probability of masking a given token. We present the effect of changing ρ in Table 7 on MNLI and RTE dev set results fixing all other hyper-parameters. We selected MNLI and RTE because they are part of Natural Language Inference, which is one of the hardest tasks on GLUE. Moreover, in the RoBERTa experiments we see the largest drop in student scores for these two datasets. We can observe that for MNLI the best result is for 30% followed by 20% and for RTE the best choice is 40% followed by 30%. This corresponds to the heuristic based data augmentation works where they typically modify tokens with a 30% to 40% probability. We set this parameter to 30% for all the experiments and did not tune this for each dataset or each architecture.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Score
MATE-KD	**65.9**	**94.1**	**91.9**	**90.4**	**91.9**	**85.8**	92.5	**75.0**	**82.64**
- Adv train	64.7	93.1	90.0	90.3	91.8	85.3	**92.8**	74.0	82.03
- Generator	60.9	92.5	90.2	89.0	91.6	84.1	91.3	71.1	80.77

Table 5: The ablation of MATE-KD on four datasets from the GLUE benchmark. We present the result of MATE-KD, a version of the algorithm without training the generator and a version of the algorithm without the generator. Results are on the dev set.

Original	Generated
the new insomnia is a surprisingly faithful remake of its chilly predecessor, and	sinister new insomnia shows a surprisingly terrible remake of its hilarious predecessor, and
beautifully shot, delicately scored and powered by a set of heartfelt performances	beautifully sublime, delicately scored, powered by great dozens of heartfelt performances
a perfectly pleasant if slightly pokey comedy	a 10 pleasant if slightly pokey comedy
that appeals to me	Federal appeals punished me
good news to anyone who's fallen under the sweet, melancholy spell of this unique director's previous films	good news for anyone who's fallen under the sweet, melancholy spell of this unique director's previous mistakes

Table 6: Examples of original and adversarially generated samples during training for the SST-2 dataset

Task	p Hyperparameter				
	10%	20%	30%	40%	50 %
MNLI	85.4	85.5	85.8	84.7	84.6
RTE	74.0	74.8	75.0	75.4	74.6

Table 7: ρ value sensitivity analysis on two GLUE tasks.

4.6 Generated Samples

We present a few selected samples that our generator produced during training for the SST-2 dataset on table 6. SST-2 is a binary sentiment analysis dataset. The data consist of movie reviews and is both at the phrase and sentence level.

We observe that we only modify a few tokens in the generated text. However, one of three things happens if the text is semantically plausible. Either the generated sentence keeps the same sentiment as in Examples 2 and 3, or it changes the sentiment as in Examples 1 and 4 or the text has ambiguous sentiment as in Example 5. We can use all of these for training since we do not rely on the original label but obtain the teacher's output.

5 Discussion and Future Work

We have presented MATE-KD, a novel text-based adversarial training algorithm which improves the student model in KD by generating adversarial examples while accessing the logits of the teacher only. This approach is architecture agnostic and can be easily adapted to other applications of KD such as model ensembling and multi-task learning.

We demonstrate the need for an adversarial training algorithm for KD based on text rather than embedding perturbation. Moreover, we demonstrate the importance of masking for our algorithm.

One key theme that we have presented in this work is that as PLMs inevitably increase in size and number of parameters, techniques that rely on access to the various layers and intermediate parameters of the teacher will be more difficult to train. In contrast, algorithms which are well-motivated and require minimal access to the teacher may learn from more powerful teachers and would be more useful. An example of such an algorithm is the KD algorithm itself.

Future work will consider a) using label information and a measure of semantic quality to filter the generated sentences b) exploring the application of our algorithm to continuous data such as speech and images and c) exploring other applications of KD.

Acknowledgement

We thank MindSpore [3], a new deep learning computing framework, for the partial support of this work

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

Impact Statement

Our research primarily deals with deploying high quality NLP applications to a wide audience around the globe. We contend that these technologies can simplify many of our mundane tasks and free up our time to pursue more pleasurable work.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

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.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*.

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. *arXiv preprint arXiv:1907.04829*.

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 Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint arXiv:2101.03961*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

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

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.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

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*.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020. More bang for your buck: Natural perturbation for robust question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*.

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*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. In *International Conference on Learning Representations*.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2019. Improved knowledge distillation via teacher assistant. *arXiv preprint arXiv:1902.03393*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018.

Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*.

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*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

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

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 Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. 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 1298–1308.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Thomas Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*.

A Training Details

We present the details of the learning rate, number of epochs, and the batch size we use for each training set of GLUE for both the BERT and the RoBERTa settings.

	Batch size	LR	Epochs
CoLA	8	2e-5	50
SST-2	32	2e-5	50
MRPC	8	3e-5	100
STS-B	32	2e-5	100
QQP	32	2e-5	30
MNLI	32	2e-5	30
QNLI	32	2e-5	50
RTE	16	7e-6	50
WNLI	8	7e-5	50

Table 8: Hyper-parameter values for the GLUE datasets. LR is the learning rate.