REWEIGHTING AUGMENTED SAMPLES BY MINIMIZING THE MAXIMAL EXPECTED LOSS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data augmentation is an effective technique to improve the generalization of deep neural networks. However, previous data augmentation methods usually treat the augmented samples equally without considering their individual impacts on the model. To address this, for the augmented samples from the same training example, we propose to assign different weights to them. We construct the maximal expected loss which is the supremum over any reweighted loss on augmented samples. Inspired by adversarial training, we minimize this maximal expected loss (MMEL) and obtain a simple and interpretable closed-form solution: more attention should be paid to augmented samples with large loss values (i.e., harder examples). Minimizing this maximal expected loss enables the model to perform well under any reweighting strategy. The proposed method can generally be applied on top of any data augmentation methods. Experiments are conducted on both natural language understanding tasks with token-level data augmentation, and image classification tasks with commonly-used image augmentation techniques like random crop and horizontal flip. Empirical results show that the proposed method improves the generalization performance of the model.

1 INTRODUCTION

Deep neural networks have achieved state-of-the-art results in various tasks in natural language processing (NLP) tasks (Sutskever et al., 2014; Vaswani et al., 2017; Devlin et al., 2019) and computer vision (CV) tasks (He et al., 2016; Goodfellow et al., 2016). One approach to improve the generalization performance of deep neural networks is data augmentation (Xie et al., 2019; Jiao et al., 2019; Cheng et al., 2019; 2020). However, there are some problems if we directly incorporate these augmented samples into the training set. Minimizing the average loss on all these samples means treating them equally, without considering their different implicit impacts on the loss.

To address this, we propose to minimize a reweighted loss on these augmented samples to make the model utilize them in a cleverer way. Example reweighting has previously been explored extensively in curriculum learning (Bengio et al., 2009; Jiang et al., 2014), boosting algorithms (Freund & Schapire, 1999), focal loss (Lin et al., 2017) and importance sampling (Csiba & Richtárik, 2018). However, none of them focus on the reweighting of augmented samples instead of the original training samples. A recent work (Jiang et al., 2020a) also assigns different weights on augmented samples. But weights in their model are predicted by a mentor network while we obtain the weights from the closed-form solution by minimizing the maximal expected loss (MMEL). In addition, they focus on image samples with noisy labels, while our method can generally be applied to also textual data as well as image data. Tran et al. (2017) propose to minimize the loss on the augmented samples under the framework of Expectation-Maximization algorithm. But they mainly focus on the generation of augmented samples.

Unfortunately, in practise there is no way to directly access the optimal reweighting strategy. Thus, inspired by adversarial training (Madry et al., 2018), we propose to minimize the maximal expected loss (MMEL) on augmented samples from the same training example. Since the maximal expected loss is the supremum over any possible reweighting strategy on augmented samples’ losses, minimizing this supremum makes the model perform well under any reweighting strategy. More importantly, we derive a closed-form solution of the weights, where augmented samples with larger training losses
have larger weights. Intuitively, MMEL allows the model to keep focusing on augmented samples that are harder to train.

The procedure of our method is summarized as follows. We first generate the augmented samples with commonly-used data augmentation technique, e.g., lexical substitution for textual input (Jiao et al., 2019), random crop and horizontal flip for image data (Krizhevsky et al., 2012). Then we explicitly derive the closed-form solution of the weights on each of the augmented samples. After that, we update the model parameters with respect to the reweighted loss. The proposed method can generally be applied above any data augmentation methods in various domains like natural language processing and computer vision. Empirical results on both natural language understanding tasks and image classification tasks show that the proposed reweighting strategy consistently outperforms the counterpart of without using it, as well as other reweighting strategies like uniform reweighting.

2 RELATED WORK

Data augmentation. Data augmentation is proven to be an effective technique to improve the generalization ability of various tasks, e.g., natural language processing (Xie et al., 2019; Zhu et al., 2020; Jiao et al., 2019), computer vision (Krizhevsky et al., 2014), and speech recognition (Park et al., 2019). For image data, baseline augmentation methods like random crop, flip, scaling, and color augmentation (Krizhevsky et al., 2012) have been widely used. Other heuristic data augmentation techniques like Cutout (DeVries & Taylor, 2017) which masks image patches and Mixup (Zhang et al., 2018) which combines pairs of examples and their labels, are later proposed. Automatically searching for augmentation policies (Cubuk et al., 2018; Lim et al., 2019) have recently proposed to improve the performance further. For textual data, Zhang et al. (2015), Wei & Zou (2019) and Wang (2015) respectively use lexical substitution based on the embedding space. Jiao et al. (2019); Cheng et al. (2019); Kumar et al. (2020) generate augmented samples with a pre-trained language model. Some other techniques like back translation (Xie et al., 2019), random noise injection (Xie et al., 2017) and data mixup (Guo et al., 2019; Cheng et al., 2020) are also proven to be useful.

Adversarial training. Adversarial learning is used to enhance the robustness of model (Madry et al., 2018), which dynamically constructs the augmented adversarial samples by projected gradient descent across training. Although adversarial training hurts the generalization of model on the task of image classification (Raghunathan et al., 2019), it is shown that adversarial training can be used as data augmentation to help generalization in neural machine translation (Cheng et al., 2019; 2020) and natural language understanding (Zhu et al., 2020; Jiang et al., 2020b). Our proposed method differs from adversarial training in that we adversarially decide the weight on each augmented sample, while traditional adversarial training adversarially generates augmented input samples.

In (Behpour et al., 2019), adversarial learning is used as data augmentation in object detection. The adversarial samples (i.e., bounding boxes that are maximally different from the ground truth) are reweighted to form the underlying annotation distribution. A similar method is also proposed in dynamic network prediction (Li et al., 2018). However, besides the difference in the model and task, their training objective and the resultant solution are also different from ours.

Sample reweighting. Minimizing a reweighted loss on training samples has been widely explored in literature. Curriculum learning (Bengio et al., 2009; Jiang et al., 2014) feeds first easier and then harder data into the model to accelerate training. Zhao & Zhang (2014); Needell et al. (2014); Csiba & Richtárik (2018); Katharopoulos & Fleuret (2018) use importance sampling to reduce the variance of stochastic gradients to achieve faster convergence rate. Boosting algorithms (Freund & Schapire, 1999) choose harder examples to train subsequent classifiers. Similarly, hard example mining (Malisiewicz et al., 2011) downsamples the majority class and exploits the most difficult examples. Focal loss (Lin et al., 2017; Goyal & He, 2018) focuses on harder examples by reshaping the standard cross-entropy loss in object detection. Ren et al. (2018); Jiang et al. (2018); Shu et al. (2019) use meta-learning method to reweight examples to handle the noisy label problem. Unlike all these existing methods, in this work, we reweight the augmented samples’ losses instead of training samples.
3 MINIMIZE THE MAXIMAL EXPECTED LOSS

In this section, we derive our reweighting strategy on augmented samples from the perspective of maximal expected loss. We first give a derivation of the closed-form solution of the weights on augmented samples. Then we describe two kinds of loss under this formulation. Finally, we give the implementation details using the natural language understanding task as an example.

3.1 WHY MAXIMAL EXPECTED LOSS

Consider a classification task with \(N\) training samples. For the \(i\)-th training sample \(x_i\), its label is denoted as \(y_{x_i}\). Let \(f_\theta(\cdot)\) be the model with parameter \(\theta\) which outputs the classification probabilities. \(\ell(\cdot, \cdot)\) denotes the loss function, e.g. the cross-entropy loss between outputs \(f_\theta(x_i)\) and the ground-truth label \(y_{x_i}\). Given an original training sample \(x_i\), the set of augmented samples generated by some method is \(B(x_i)\). Without loss of generality, we assume \(x_i \in B(x_i)\). The conventional training objective is to minimize the loss on every augmented sample \(z\) in \(B(x_i)\) as

\[
\min_\theta \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{1}{|B(x_i)|} \sum_{(z, y_z) \in B(x_i)} \ell(f_\theta(z), y_z) \right],
\]

(1)

where \(y_z\) is the label of \(z \in B(x_i)\), and can be different with \(y_{x_i}\). \(|B(x_i)|\) is the number of augmented samples in \(B(x_i)\), which is assumed to be finite.

In equation (1), for each \(x_i\), the weights on each of its augmented samples are the same (i.e., \(1/|B(x_i)|\)). However, different samples have different implicit impacts on the loss, and we can assign different weights on them to facilitate training. Note that computing the weighted sum of losses of each augmented sample in \(B(x_i)\) can be viewed as taking expectation of loss on augmented samples \(z \in B(x_i)\) under a certain distribution. When the augmented samples generated from the same training sample are drawn from a uniform distribution, the loss in equation (1) can be rewritten as

\[
\min_\theta R_\theta(P_U) = \min_\theta \frac{1}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{z \sim P_U(\cdot | x_i)} [\ell(f_\theta(z), y_z)] - \lambda P \text{KL}(P_U(\cdot | x_i) \parallel P_U(\cdot | x_i)) \right],
\]

(2)

where the Kullback–Leibler (KL) divergence \(\text{KL}(P_U(\cdot | x_i) \parallel P_U(\cdot | x_i))\) equals zero. Here \(P_U(\cdot | x_i)\) denotes the uniform distribution on \(B(x_i)\). When the augmented samples are drawn from a more general distribution \(P_B(\cdot | x_i)\) instead of the uniform distribution, we can generalize \(P_U(\cdot | \cdot)\) here to some other conditional distribution \(P_B\).

\[
\min_\theta R_\theta(P_B) = \min_\theta \frac{1}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{z \sim P_B(\cdot | x_i)} [\ell(f_\theta(z), y_z)] - \lambda P \text{KL}(P_B(\cdot | x_i) \parallel P_U(\cdot | x_i)) \right].
\]

(3)

Remark 1. When \(P_B(\cdot | x_i)\) reduces to the uniform distribution \(P_U(\cdot | x_i)\) for any \(x_i\), since \(\text{KL}(P_U(\cdot | x_i) \parallel P_U(\cdot | x_i)) = 0\), the objective in equation (3) reduces to the one in equation (1).

The KL divergence term in equation (3) is used as a regularizer to encourage \(P_B\) close to \(P_U\) (see Remark 2). From equation (3), the conditional distribution \(P_B\) determines the weights of each augmented sample in \(B(x_i)\). There may exist an optimal formulation of \(P_B\) in some regime, e.g. corresponding to the optimal generalization ability of model. Unfortunately, we can not explicitly characterize such an unknown optimal \(P_B\). To address this, we borrow the idea from adversarial training (Madry et al.[2018]) and minimize the maximal reweighted loss on augmented samples. Then, the model is guaranteed to perform well under any reweighting strategy, including the underlying optimal one. Specifically, let the conditional distribution \(P_B\) be \(P_B^* = \text{arg sup}_{P_B} R_\theta(P_B)\). Our objective is to minimize the following reweighted loss

\[
\min_\theta R_\theta(P_B^*) = \min_\theta \sup_{P_B} R_\theta(P_B).
\]

(4)

The following Remark 2 discusses about the KL divergence term in equation (3).

Remark 2. Since we take a supremum over \(P_B\) in equation (4), the regularizer \(\text{KL}(P_B \parallel P_U)\) encourages \(P_B\) to be close to \(P_U\) because it reaches the minimal value zero when \(P_B = P_U\). Thus the regularizer controls the diversity among the augmented samples by constraining the discrepancy between \(P_B\) and uniform distribution \(P_U\), e.g., a larger \(\lambda\) promotes a larger diversity among the augmented samples.

\footnote{In the following, we simplify \(P_B(\cdot | \cdot)\) as \(P_B\) if there is no obfuscation.}
The following Theorem\cite{1} gives the explicit formulation of $R_\theta(P_B^a)$.

**Theorem 1.** Let $R_\theta(P_B)$ and $R_\theta(P_B^a)$ be defined in equation (1) and (4), then we have

$$R_\theta(P_B^a) = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{z \in B(x_i)} \mathbb{P}_\theta^a(z \mid x_i) \ell(f_\theta(z), y_a) - \lambda_P \mathbb{P}_\theta^a(z \mid x_i) \log (\mathbb{P}(B(x_i)) \mathbb{P}_\theta^a(z \mid x_i)) \right], \quad (5)$$

where

$$\mathbb{P}_\theta^a(z \mid x_i) = \frac{\exp \left( \frac{1}{\lambda_P} \ell(f_\theta(z), y_a) \right)}{\sum_{z \in B(x_i)} \exp \left( \frac{1}{\lambda_P} \ell(f_\theta(z), y_a) \right)} = \text{Softmax}_z \left( \frac{1}{\lambda_P} \ell(f_\theta(B(x_i)), y_B(x_i)) \right), \quad (6)$$

where $\text{Softmax}_z(\frac{1}{\lambda_P} \ell(f_\theta(B(x_i)), y_B(x_i)))$ represents the output probability of $z$ for vector $(\frac{1}{\lambda_P} \ell(f_\theta(z_1), y_a), \ldots, \frac{1}{\lambda_P} \ell(f_\theta(z|B(x_i)), y|B(x_i)))$.

**Remark 3.** If we ignore the KL divergence term in equation (5), due to the equivalence of minimizing cross-entropy loss and MLE loss\cite{Martens2019}, the proposed MMEL also falls into the generalized Expectation-Maximization (GEM) framework\cite{Dempster1977}. Specifically, given a training example, the augmented samples of it can be viewed as latent variable, and any reweighting on these augmented samples corresponds to a specific conditional distribution of these augmented samples given the training sample. In the expectation step (E-step), we explicitly derive the closed-form solution of the weights on each of these augmented samples according to (5). In the maximization step, since there is no analytical solution for deep neural networks, following\cite{Tran2017}, we update the model parameters with respect to the reweighted loss by one step of gradient descent.

The proof of this theorem can be found in Appendix A\cite{1}. From Theorem 1, the loss of it decides the weight on each augmented sample $z \in B(x_i)$, and the weight is normalized by Softmax over all augmented samples in $B(x_i)$. The reweighting strategy allows more attention paid to augmented samples with higher loss values. The strategy is similar to those in\cite{Lin2017, Zhao2014} but they apply it on training samples.

### 3.2 Two Types of Loss

For augmented sample $z \in B(x_i)$, instead of computing the discrepancy between the output probability $f_\theta(z)$ and the hard label $y_a$ as in equation (5), one can also compute the discrepancy between $f_\theta(z)$ and the “soft” probability $f_\theta(x_i)$ in the absence of ground-truth label on augmented samples as in\cite{Xie2019}. In the following, We use superscript “hard” for the loss in equation (5) as

$$R_\theta^{\text{hard}}(P_\theta^a, x_i) = \sum_{z \in B(x_i)} \mathbb{P}_\theta^a(z \mid x_i) \ell(f_\theta(z), y_a)) - \lambda_P \mathbb{P}_\theta^a(z \mid x_i) \log (\mathbb{P}(B(x_i)) \mathbb{P}_\theta^a(z \mid x_i)), \quad (7)$$

to distinguish with the following objective which uses the “soft probability”:

$$R_\theta^{\text{soft}}(P_\theta^a, x_i) = \ell(f_\theta(x_i), y_a) + \lambda_T \sum_{z \in B(x_i), z \neq x_i} \mathbb{P}_\theta^a(z \mid x_i) \ell(f_\theta(z), f_\theta(x_i))$$

$$- \lambda_T \mathbb{P}_\theta^a(z \mid x_i) \log (\mathbb{P}(B(x_i)) - 1) \mathbb{P}_\theta^a(z \mid x_i)), \quad (8)$$

The two terms in $R_\theta^{\text{soft}}(P_\theta^a, x_i)$ respectively correspond to the loss on original training samples $x_i$ and the reweighted loss on the augmented samples. The reweighted loss promotes a small discrepancy between the augmented samples and the original training sample. $\lambda_T > 0$ is the coefficient used to balance the two loss terms, and $\mathbb{P}_\theta^a(z \mid x_i)$ is defined similar to (5) as

$$\mathbb{P}_\theta^a(z \mid x_i) = \frac{\exp \left( \frac{1}{\lambda_P} \ell(f_\theta(z), f_\theta(x_i)) \right)}{\sum_{z \in B(x_i), z \neq x_i} \exp \left( \frac{1}{\lambda_P} \ell(f_\theta(z), f_\theta(x_i)) \right)} \cdot \quad (9)$$

The two losses are shown in Figure\cite{1}. Summing over all the training samples, we get the two kinds of reweighted training objectives.
3.3 Example: MMEL Implementation on Natural Language Understanding Tasks

In this section, we elaborate on implementing the proposed method using textual data in natural language understanding tasks as an example. Our method is separated into two phases. In the first phase, we generate augmented samples. Then in the second phase, with these augmented samples, we update the model parameters under these augmented samples with respect to the hard reweighted one word in a sentence can drastically change its meaning. For instance, suppose the original sentence methods like random crop and horizontal flip of an image do not change its label, substituting even can be found in Algorithm 2 in Appendix B.

Various methods have been proposed to generate augmented samples for textual data. Recently, large-scale pre-trained language models like BERT (Devlin et al. 2019) and GPT-2 (Radford et al. 2019) learn contextualized representations and have been used widely in generating high-quality augmented sentences (Jiao et al. 2019, Kumar et al. 2020). In this paper, we use a pre-trained BERT trained from masked language modeling to generate augmented samples. For each original input sentence, we randomly mask $k$ tokens. Then we do a forward propagation of the BERT to predict the tokens in those masked positions by greedy search. Details can be found in Algorithm 2 in Appendix B.

**Mismatches Label.** For $R_{\text{hard}}^T$ in equation (7), the loss term $\ell(f_\theta(z), y_z)$ on augmented sample $z \in B(x_i)$ for some $x_i$ relies on its label $y_z$. Unlike image data, where conventional augmentation methods like random crop and horizontal flip of an image do not change its label, substituting even one word in a sentence can drastically change its meaning. For instance, suppose the original sentence is “She is my daughter”, and the word “She” is masked. The top 5 words predicted by the pre-trained BERT are “This, She, That, It, He”. Apparently, for the task of linguistic acceptability, replacing “She” with “He” can change the label from linguistically “acceptable” to “non-acceptable”. Thus for

![Diagram](image-url)
textual input, for the term \( \ell(f_\theta(z), y_z) \) in hard loss (7), instead of directly setting \( y_z \) as \( y_{x_i} \) (Zhu et al. 2020), we replace \( y_z \) with the output probability of a trained teacher model. On the other hand, for the soft loss in equation (8), if an augmented sample \( z \in B(x_i) \) is predicted to a different class from \( x_i \) by the teacher model, it is unreasonable to still minimize the discrepancy between \( f_\theta(z) \) and \( f_\theta(x_i) \). In this case, we replace \( f_\theta(x_i) \) in the loss term \( \lambda_T \sum_{z \in B(x_i); z \neq x_i} P_\theta(z | x_i) \ell(f_\theta(z), f_\theta(x_i)) \) with the output probability from the teacher model.

4 EXPERIMENTS

In this section, we evaluate the efficacy of the proposed MMEL algorithm with both hard loss (MMEL-H) and soft loss (MMEL-S). Experiments are conducted on both the image classification tasks CIFAR-10 and CIFAR-100 (Krizhevsky et al. 2014) with the ResNet Model (He et al. 2016), and the General Language Understanding Evaluation (GLUE) tasks (Wang et al. 2019) with the BERT model (Devlin et al. 2019).

4.1 EXPERIMENTS ON IMAGE CLASSIFICATION TASKS.

Data. Krizhevsky et al. 2014 is a benchmark data set for image classification. We use both CIFAR-10 and CIFAR-100 in our experiments, which are colorful images with 50000 training samples and 10000 validation samples, but from 10 and 100 object classes, respectively.

Setup. The model we used is ResNet (He et al. 2016) with different depths. We use random crop and horizontal flip (Krizhevsky et al. 2012) to augment the original training images. Since these operations do not change the augmented sample label, we directly adopt the original training sample for each \( x_i \). We also compare with two other baselines that also use 10 times more training data: (i) naive training with data augmentation but with 10 times more training epochs compared with MMEL (abbreviated as “Baseline(DA+Long)”); (ii) training with data augmentation under the framework of MMEL but with uniform weights on the augmented samples (abbreviated as “Baseline(DA+UNI)”).

Main Results. The results are shown in Table 1. As can be seen, for both CIFAR-10 and CIFAR-100, MMEL-H and MMEL-S significantly outperform the Baseline(DA), with over 0.5 points higher accuracy on all four architectures. Compared to Baseline(DA+Long), the proposed MMEL-H and MMEL-S also have comparable or better performance, while being much more efficient in training. This is because our backward pass only computes the gradient of the weighted loss instead of the separate loss of each example. Compared to Baseline(DA+UNI) which has the same computational cost as MMEL-H and MMEL-S, the proposed methods also have better performance. This indicates the efficacy of the proposed maximal expected loss based reweighting strategy.

Table 1: Performance of ResNet on CIFAR-10 and CIFAR-100. The time is the training time measured on a single NVIDIA V100 GPU. The results expect for “Baseline(DA + Long)” are obtained on the average of five independent runs with “mean (±std)” reported.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Model</th>
<th>Baseline(DA)</th>
<th>Baseline(DA+Long)</th>
<th>Baseline(DA+UNI)</th>
<th>MMEL-H</th>
<th>MMEL-S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acc</td>
<td>time</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>92.53±0.10</td>
<td>0.7h</td>
<td>93.27±0.6h</td>
<td>93.00±0.10h</td>
<td>93.16±0.03h</td>
<td>2.8h</td>
</tr>
<tr>
<td></td>
<td>93.46±0.21</td>
<td>0.7h</td>
<td>94.43±0.7h</td>
<td>94.11±0.33h</td>
<td>94.31±0.07h</td>
<td>4.3h</td>
</tr>
<tr>
<td></td>
<td>93.92±0.10</td>
<td>0.8h</td>
<td>94.11±0.8h</td>
<td>94.30±0.18h</td>
<td>94.70±0.14h</td>
<td>5.7h</td>
</tr>
<tr>
<td></td>
<td>93.96±0.20</td>
<td>1.1h</td>
<td>94.12±1.0h</td>
<td>94.62±0.18h</td>
<td>94.85±0.15h</td>
<td>7.0h</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>69.45±0.3h</td>
<td>0.3h</td>
<td>71.98±0.7h</td>
<td>71.59±0.10h</td>
<td>72.51±0.07h</td>
<td>4.3h</td>
</tr>
<tr>
<td></td>
<td>70.66±0.16</td>
<td>0.7h</td>
<td>71.99±0.2h</td>
<td>71.59±0.10h</td>
<td>72.51±0.07h</td>
<td>4.3h</td>
</tr>
<tr>
<td></td>
<td>71.43±0.30</td>
<td>0.8h</td>
<td>72.83±0.3h</td>
<td>72.30±0.38h</td>
<td>73.18±0.31</td>
<td>5.7h</td>
</tr>
<tr>
<td></td>
<td>72.22±0.26</td>
<td>1.1h</td>
<td>73.09±0.6h</td>
<td>73.44±0.13h</td>
<td>74.20±0.24</td>
<td>7.0h</td>
</tr>
</tbody>
</table>
Varying the Number of Augmented Samples. One hyperparameter of the proposed method is the number of augmented samples $|B(x)|$. In Table 2, we evaluate the effect of $|B(x)|$ on the CIFAR data set. We vary $|B(x)|$ in $\{2, 5, 10, 20\}$ for both MMEL-H and MMEL-S with other settings unchanged. As can be seen, the performance of MMEL improves with more augmented samples for small $|B(x)|$. However, the performance gain begins to saturate when $|B(x)|$ reaches 5 or 10 for some cases. Since a larger $|B(x)|$ also brings more training cost, we should choose a proper number of augmented samples rather than continually increasing it.

| Data set     | Model     | $B(x|)| = 2$ | $B(x)| = 5$ | $B(x)| = 10$ | $B(x)| = 20$ |
|--------------|-----------|--------------|-------------|--------------|--------------|
| CIFAR-10     | Baseline/DA| 92.94 ± 0.14 | 93.58 ± 0.16 | 94.02 ± 0.14 | 94.48 ± 0.14 |
|              | MMEL-H    | 94.78 ± 0.12 | 95.37 ± 0.12 | 95.85 ± 0.12 | 96.34 ± 0.12 |
|              | MMEL-S    | 94.78 ± 0.12 | 95.37 ± 0.12 | 95.85 ± 0.12 | 96.34 ± 0.12 |
| CIFAR-100    | Baseline/DA| 92.94 ± 0.14 | 93.58 ± 0.16 | 94.02 ± 0.14 | 94.48 ± 0.14 |
|              | MMEL-H    | 94.78 ± 0.12 | 95.37 ± 0.12 | 95.85 ± 0.12 | 96.34 ± 0.12 |
|              | MMEL-S    | 94.78 ± 0.12 | 95.37 ± 0.12 | 95.85 ± 0.12 | 96.34 ± 0.12 |

4.2 RESULTS ON NATURAL LANGUAGE UNDERSTANDING TASKS

Data. GLUE is a benchmark containing various natural language understanding tasks, including textual entailment (RTE and MNLI), question answering (QNLI), similarity and paraphrase (MRPC, QQP, STS-B), sentiment analysis (ST-S2) and linguistic acceptability (CoLA). Among them, STS-B is a regression task, CoLA and SST-2 are single sentence classification tasks, while the rest are sentence-pair classification tasks. Following Devlin et al. (2019), for the development set, we report Spearman correlation for STS-B, Matthews correlation for CoLA and accuracy for the other tasks. For the test set for QQP and MRPC, we report “F1”.

Setup. The backbone model is BERT BASE (Devlin et al., 2019). We use the method in Section 3.3 to generate augmented samples. For the problem of mismatching label (RTE and MNLI), question answering (QNLI), similarity and paraphrase (MRPC, QQP, STS-B), sentiment analysis (ST-S2) and linguistic acceptability (CoLA). Among them, STS-B is a regression task, CoLA and SST-2 are single sentence classification tasks, while the rest are sentence-pair classification tasks. Following Devlin et al. (2019), for the development set, we report Spearman correlation for STS-B, Matthews correlation for CoLA and accuracy for the other tasks. For the test set for QQP and MRPC, we report “F1”.

The derivation of MMEL in Section 3 is based on the classification task, while STS-B is a regression task. Hence, we generalize our loss function accordingly for regression tasks as follows. For the hard loss in equation (7), we directly replace $y \in \mathbb{R}$ with the prediction of teacher model on $z$. For the soft loss (8), for each entry of $f_0(x_i)$ in loss term $\lambda_T \sum_{z \in B(x_i), z \neq x_i} P_0(z | x_i) \mathcal{MSE}(f_0(z), f_0(x_i))$, we replace it with the prediction of teacher model if the difference between them is larger than 0.5.

Similar to Section 4.1. We compare with three baselines. However, we change the first baseline to naive training without data augmentation (abbreviated as “Baseline”) since data augmentation is not used by default in NLP tasks. The other two baselines are similar to those in Section 4.1: (i) “Baseline(DA+Long)” which fine-tunes BERT with data augmentation with the same batch size; and (ii) “Baseline(DA+UNI)” which fine-tunes BERT with augmented samples by using average loss. We also compare with another recent data augmentation technique SMART (Jiang et al., 2020b).

Main Results. The development and test set results on the GLUE benchmark are shown in Table 3. The development set results for the BERT baseline are from our re-implementation, which is comparable or better than the reported results in the original paper (Devlin et al., 2019). The results for SMART are taken from Jiang et al. (2020b), and there are no test set results in Jiang et al. (2020b). As can be seen, data augmentation significantly improves the generalization of GLUE tasks. Compared to the baseline without data augmentation (Baseline), MMEL-H or MMEL-S consistently achieves better performance, especially on small data sets like CoLA and RTE. Similar to the observation in the image classification task in Section 4.1, the proposed MMEL-H and MMEL-S are more efficient and have better performance than Baseline(DA+Long). MMEL-H and MMEL-S
Table 3: Development and test sets results on the BERT\textsubscript{BASE} model. The training time is measured on a single NVIDIA V100 GPU. The results of Baseline, Baseline(DA+UNI), MMEL-H and MMEL-S are obtained by five independent runs with “mean(±std)” reported.

<table>
<thead>
<tr>
<th>Method</th>
<th>SST-2</th>
<th>MRPC</th>
<th>MRPC</th>
<th>CoLA</th>
<th>MNLI-m/mm</th>
<th>QQP</th>
<th>QNLI</th>
<th>RTE</th>
<th>Test</th>
<th>Dev</th>
<th>Avg</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline(DA+Long)</td>
<td>61.5</td>
<td>89.1</td>
<td>87.7</td>
<td>59.7</td>
<td>84.0</td>
<td>80.4</td>
<td>71.2</td>
<td>92.0</td>
<td>32.6</td>
<td>32.6</td>
<td>40.1</td>
<td>105.5</td>
</tr>
<tr>
<td>Baseline(DA+UNI)</td>
<td>61.3(±0.75)</td>
<td>93.0(±0.37)</td>
<td>87.6(±0.63)</td>
<td>90.8(±0.10)</td>
<td>91.1(±0.01)</td>
<td>84.8(±0.26)</td>
<td>91.6(±0.16)</td>
<td>75.8(±1.02)</td>
<td>74.5(±0.74)</td>
<td>91.1(±0.48)</td>
<td>31.4k</td>
<td></td>
</tr>
<tr>
<td>SMART</td>
<td>91.1</td>
<td>91.9</td>
<td>87.7</td>
<td>80.4</td>
<td>85.0</td>
<td>91.3</td>
<td>71.2</td>
<td>92.0</td>
<td>32.6</td>
<td>32.6</td>
<td>40.1</td>
<td>105.5</td>
</tr>
<tr>
<td>MMEL-H</td>
<td>62.0(±0.55)</td>
<td>93.1(±0.34)</td>
<td>87.7(±0.20)</td>
<td>90.4(±0.10)</td>
<td>91.8(±0.07)</td>
<td>85.3(±0.19)</td>
<td>91.6(±0.16)</td>
<td>75.8(±1.02)</td>
<td>74.5(±0.74)</td>
<td>91.1(±0.48)</td>
<td>31.4k</td>
<td></td>
</tr>
<tr>
<td>MMEL-S</td>
<td>52.5(±0.63)</td>
<td>93.5(±1.06)</td>
<td>88.3(±0.13)</td>
<td>86.6(±0.07)</td>
<td>72.1(±0.10)</td>
<td>80.6(±0.23)</td>
<td>84.2(±0.15)</td>
<td>91.4(±0.31)</td>
<td>89.5(±0.54)</td>
<td>80.3(±10.0)</td>
<td>32.4k</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Development set results on BERT\textsubscript{BASE} model with different loss functions.

also outperform Baseline(DA+UNI), indicating the superiority of using the proposed reweighting strategy. In addition, our proposed method also beats SMART in both accuracy and efficiency because they use PGD-k (Madry et al., 2018) to construct adversarial augmented samples which requires nearly $k$ times more training cost. Figure 2 shows the development set accuracy across over the training procedure. As can be seen, training with MMEL-H or MMEL-S converges faster and has better accuracy except SST-2 and RTE where the performance is similar.

Effect of Predicted Labels. For the augmented samples from same origin, we use a fine-tuned strategy, we propose to minimize the supremum of the loss under all reweighting strategies, and give a closed-form solution of the optimal weights. Our method can be applied on the top of any

Table 4: Effect of using the predicted label. Development set results are reported.

<table>
<thead>
<tr>
<th>Method</th>
<th>Label</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI-m/mm</th>
<th>QNLI</th>
<th>RTE</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMEL-H Original</td>
<td></td>
<td>48.8</td>
<td>91.5</td>
<td>79.2</td>
<td>80.3</td>
<td>88.6</td>
<td>80.4/79.8</td>
<td>88.8</td>
<td>65.3</td>
<td>78.1</td>
</tr>
<tr>
<td>MMEL-H Predicted</td>
<td></td>
<td>62.8</td>
<td>93.3</td>
<td>87.5</td>
<td>90.4</td>
<td>91.6</td>
<td>85.4/85.6</td>
<td>92.1</td>
<td>72.2</td>
<td>84.5</td>
</tr>
<tr>
<td>MMEL-S Original</td>
<td></td>
<td>56.6</td>
<td>91.7</td>
<td>85.8</td>
<td>81.6</td>
<td>90.0</td>
<td>81.9/81.3</td>
<td>89.9</td>
<td>61.0</td>
<td>80.0</td>
</tr>
<tr>
<td>MMEL-S Predicted</td>
<td></td>
<td>61.7</td>
<td>93.8</td>
<td>89.2</td>
<td>90.2</td>
<td>91.6</td>
<td>85.0/85.5</td>
<td>92.5</td>
<td>73.3</td>
<td>84.7</td>
</tr>
</tbody>
</table>

5 Conclusion

In this work, we propose to minimize a reweighted loss over the augmented samples to that directly considers the implicit impacts on the loss. Since we can not access the optimal reweighting strategy, we propose to minimize the supremum of the loss under all reweighting strategies, and give a closed-form solution of the optimal weights. Our method can be applied on the top of any
data augmentation methods. Experiments on both image classification tasks and natural language understanding tasks show that the proposed method improves the generalization performance of the model, while being efficient in training.

REFERENCES


A PROOF OF THEOREM[1]

Proof. For any given \( \mathbf{x}_i \) and \( B(\mathbf{x}_i) \), we aim to find \( \mathbb{P}_{\theta}(\cdot | \mathbf{x}_i) \) on \( B(\mathbf{x}_i) \) such that

\[
\max_{\mathbb{P}_{\theta}(\cdot | \mathbf{x}_i)} \sum_{\mathbf{z} \in B(\mathbf{x}_i)} \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) \ell(f_{\theta}(\mathbf{z}), y_{\mathbf{x}_i}) - \lambda_P \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) \log ((B(\mathbf{x}_i)) \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i))
\]

s.t. \( \sum_{\mathbf{z} \in B(\mathbf{x}_i)} \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) = 1. \)

(10)

Since the objective is convex, by Lagrange multiplier method, let

\[
\mathcal{L}(\mathbb{P}_{\theta}, \lambda) = \sum_{\mathbf{z} \in B(\mathbf{x}_i)} \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) \ell(f_{\theta}(\mathbf{z}), y_{\mathbf{x}_i}) - \lambda_P \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) \log ((B(\mathbf{x}_i)) \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i))
\]

\[+\lambda \left( \sum_{\mathbf{z} \in B(\mathbf{x}_i)} \mathbb{P}_{\theta}(\mathbf{z} | \mathbf{x}_i) - 1 \right).\]

(11)

From \( \nabla_{\mathbb{P}_{\theta}} \mathcal{L}(\mathbb{P}_{\theta}, \lambda) = \nabla_{\lambda} \mathcal{L}(\mathbb{P}_{\theta}, \lambda) = 0 \), for any pairs of \( \mathbf{z}_u, \mathbf{z}_v \in B(\mathbf{x}_i) \), we have

\[
\ell(f_{\theta}(\mathbf{z}_u), y_{\mathbf{x}_u}) - \lambda_P (\log |B(\mathbf{x}_i)| + 1 + \log \mathbb{P}_{\theta}(\mathbf{z}_u | \mathbf{x}_i))
\]

\[= \ell(f_{\theta}(\mathbf{z}_v), y_{\mathbf{x}_v}) - \lambda_P (\log |B(\mathbf{x}_i)| + 1 + \log \mathbb{P}_{\theta}(\mathbf{z}_v | \mathbf{x}_i)).\]

(12)

Hence we have

\[
\mathbb{P}_{\theta}(\mathbf{z}_v | \mathbf{x}_i) = \mathbb{P}_{\theta}(\mathbf{z}_u | \mathbf{x}_i) \exp \left( \frac{\ell(f_{\theta}(\mathbf{z}_u), y_{\mathbf{x}_u}) - \ell(f_{\theta}(\mathbf{z}_v), y_{\mathbf{x}_v})}{\lambda_P} \right).
\]

(13)

Summing over \( \mathbf{z}_v \in B(\mathbf{x}_i) \), we have

\[
\mathbb{P}_{\theta}(\mathbf{z}_u | \mathbf{x}_i) \sum_{\mathbf{z}_v \in B(\mathbf{x}_i)} \exp \left( \frac{\ell(f_{\theta}(\mathbf{z}_v), y_{\mathbf{x}_u}) - \ell(f_{\theta}(\mathbf{z}_u), y_{\mathbf{x}_u})}{\lambda_P} \right) = 1.
\]

(14)

The proof completes. \( \square \)

B GENERATING AUGMENTED SAMPLES FOR TEXTUAL SAMPLES

In this section, we elaborate the procedure of generating augmented sentences using greedy-based and beam-based method for a sequence. For each original input sentence, we randomly mask \( k \) tokens (which is obtained by rounding the product of masking ratio and length of the sequence to the nearest number) and then we do a forward propagation of the BERT to predict the tokens in those masked positions using greedy search. The detailed procedure is shown in Algorithm 2. We also use beam search (Yang et al., 2018) to generate augmented data. The details of beam search can be referred to (Yang et al., 2018). For sentence-pair tasks, we treat the two sentences separately and generate augmented samples for each of them.

**Algorithm 2** Augmented Sample Generation by Greedy Search

**Input:** Pre-trained language model \( \text{BertModel} \), original sentence \( \mathbf{x} \), number of augmented samples \( |B(\mathbf{x})| - 1 \), number of masked tokens \( k \).

**Output:** Augmented samples \( B(\mathbf{x}) = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_{|B(\mathbf{x})|-1}\} \).

1. Randomly sample \( k \) positions \( \{p_1, \cdots, p_k\} \) and get \( \mathbf{x}_{\text{mask}} \).
2. for \( i = 1, 2, \cdots, |B(\mathbf{x})| - 1 \) do
   3. \( \mathbf{z}_i \leftarrow \mathbf{x}_{\text{mask}} \).
   4. \( \mathbf{z}_i[p_j] \leftarrow \) the \( j \)th most likely word predicted by \( \text{BertModel}(\mathbf{z}_i[p_j]|\mathbf{z}_i) \).
5. for \( j \) in \( \{2, 3, \cdots, k\} \) do
6. \( \mathbf{z}_i[p_j] \leftarrow \) the \( j \)th most likely word predicted by \( \text{BertModel}(\mathbf{z}_i[p_j]|\mathbf{z}_i) \).
7. end for
8. end for

In the following, we vary the factors that may affect the quality of the generated augmented samples. These factors include

---

Under review as a conference paper at ICLR 2021
1. The number of masked tokens, which equals the replacement proportion multiplied with the sentence length. This affects the diversity of augmented samples, i.e., replacing a large proportion of tokens makes the augmented sample less similar to the original one.

2. Treating the two sentences separately in sentence-pair tasks when generating augmented examples, or concatenate them as a single sentence;

3. Different generation methods like greedy search (Algorithm 2) and beam search.

The results are shown in Table 5. As can be seen, compared with Baseline without data augmentation, MMEL-H and MMEL-S under all hyperparameter configurations have higher accuracy, showing the efficacy of data augmentation and the proposed reweighting strategy. There is no significant difference in using greedy search or beam search to generate the augmented samples. In this natural understanding task, training with augmented samples generated with proper larger replacement proportion (i.e., larger diversity) has slightly better performance. For sentence-pair tasks, treating the two sentences separately and generate augmented samples for each of them has slightly better performance. In the experiments in Section 4.2 we use Greedy search, masking proportion 0.4, and generate augmented sentence for each sentence in sentence-pair tasks.

Table 5: Ablation study on generating augmented samples on the GLUE benchmark. Development set results are reported.

<table>
<thead>
<tr>
<th>Method</th>
<th>Separate sentence-pair</th>
<th>Replacement proportion</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI-mm</th>
<th>QNLI</th>
<th>RTE</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>True</td>
<td>0.2</td>
<td>59.0</td>
<td>93.3</td>
<td>87.5</td>
<td>89.8</td>
<td>91.3</td>
<td>85.0/85.8</td>
<td>91.6</td>
<td>71.1</td>
<td>83.7</td>
</tr>
<tr>
<td>Greedy</td>
<td>True</td>
<td>0.4</td>
<td>62.8</td>
<td>93.3</td>
<td>87.5</td>
<td>90.4</td>
<td>91.6</td>
<td>85.4/85.6</td>
<td>91.6</td>
<td>72.2</td>
<td>84.5</td>
</tr>
<tr>
<td>MMEL-H</td>
<td>False</td>
<td>0.2</td>
<td>60.2</td>
<td>93.6</td>
<td>87.0</td>
<td>89.8</td>
<td>91.5</td>
<td>85.3/85.7</td>
<td>92.5</td>
<td>69.3</td>
<td>84.0</td>
</tr>
<tr>
<td>Greedy</td>
<td>False</td>
<td>0.4</td>
<td>61.8</td>
<td>92.7</td>
<td>88.0</td>
<td>90.0</td>
<td>91.5</td>
<td>85.3/85.5</td>
<td>92.2</td>
<td>72.2</td>
<td>84.4</td>
</tr>
<tr>
<td>Beam</td>
<td>True</td>
<td>0.2</td>
<td>60.0</td>
<td>93.2</td>
<td>87.0</td>
<td>90.0</td>
<td>91.4</td>
<td>85.5/85.5</td>
<td>92.4</td>
<td>71.1</td>
<td>84.0</td>
</tr>
<tr>
<td>Beam</td>
<td>False</td>
<td>0.2</td>
<td>60.0</td>
<td>93.5</td>
<td>86.7</td>
<td>89.8</td>
<td>91.5</td>
<td>85.6/85.7</td>
<td>92.3</td>
<td>70.3</td>
<td>84.5</td>
</tr>
</tbody>
</table>

C HYPERPARAMETERS FOR THE EXPERIMENT ON THE GLUE BENCHMARK.

The optimizer we used is AdamW (Loshchilov & Hutter 2018). The hyperparameters of BERT_BASE model are listed in Table 6.

Table 6: Hyperparameters of the BERT_BASE model.

<table>
<thead>
<tr>
<th>Hyperparam</th>
<th>MMEL-H</th>
<th>MMEL-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>3e-5</td>
<td>3e-5</td>
</tr>
<tr>
<td>Batch Size</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hidden Layer Dropout Rate</td>
<td>0, 0.1</td>
<td>0, 0.1</td>
</tr>
<tr>
<td>Attention Probability Dropout Rate</td>
<td>0, 0.1</td>
<td>0, 0.1</td>
</tr>
<tr>
<td>Max Epochs (MNLI, QQP)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Max Epochs (Others)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Learning Rate Decay</td>
<td>Linearly</td>
<td>Linearly</td>
</tr>
<tr>
<td>Warmup Ratio</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\lambda_T$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$\lambda_P$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Candidates ($</td>
<td>B(x)</td>
<td>$)</td>
</tr>
</tbody>
</table>