Adversarial Mixing Policy for Relaxing Locally Linear Constraints in *Mixup*

Guang Liu, Yuzhao Mao,
Hailong Huang, Weiguo Gao, and **Xuan Li**
PingAn Life Insurance of China
https://github.com/PAI-SmallIsAllYourNeed/Mixup-AMP

Abstract

Mixup is a recent regularizer for current deep classification networks. Through training a neural network on convex combinations of pairs of examples and their labels, it imposes locally linear constraints on the model's input space. However, such strict linear constraints often lead to under-fitting which degrades the effects of regularization. Noticeably, this issue is getting more serious when the resource is extremely limited. To address these issues, we propose the Adversarial Mixing Policy (AMP), organized in a "min-max-rand" formulation, to relax the Locally Linear Constraints in Mixup. Specifically, AMP adds a small adversarial perturbation to the mixing coefficients rather than the examples. Thus, slight non-linearity is injected in-between the synthetic examples and synthetic labels. By training on these data, the deep networks are further regularized, and thus achieve a lower predictive error rate. Experiments on five text classification benchmarks and five backbone models have empirically shown that our methods reduce the error rate over Mixup variants in a significant margin (up to 31.3%), especially in low-resource conditions (up to 17.5%).

1 Introduction

Deep classification models have achieved impressive results in both images (He et al., 2016; Dosovitskiy et al., 2020) and language processing (Devlin et al., 2019; Kim, 2014; Wang et al., 2016). One of the most significant challenges to train a deep model is the great efforts and costs to collect large-scale labels. Without sufficient labels, the deep networks tend to generalize poorly, leading to unsatisfactory performance. Thus, the regularization techniques under augmentation schema, which generate labeled data to regularize models (Hernández-García and König, 2018), are widely explored (Wei and Zou, 2019; Liu et al., 2021).

Mixup (Zhang et al., 2018) is an effective regularizer under the augmentation schema. In recent years, topics related to *Mixup* have warranted serious attention (Lee et al., 2020; Xu et al., 2020; Verma et al., 2019; Archambault et al., 2019; Berthelot et al., 2019b,a; Beckham et al., 2019; Mao et al., 2019; Zhu et al., 2020). The core idea of *Mixup* is to generate synthetic training data via a mixing policy, which convex combines a pair of examples and its labels. Through training on these data, the classification networks will be regularized to reach higher performance. Unlike conventional regularizers (Srivastava et al., 2014; Hanson and Pratt, 1988; Ioffe and Szegedy, 2015), *Mixup* imposes a kind of locally linear constraint (Zhang et al., 2018; Guo et al., 2019b) on the model's input space.

However, vanilla *Mixup* often suffers from under-fitting due to the ambiguous data (Guo et al., 2019b; Guo, 2020; Mai et al., 2021) generated under the strict locally linear constraints. To alleviate the under-fitting, (Guo, 2020) uses extra parameters to project the inputs and labels into a high dimensional space to properly separate the data. (Guo et al., 2019b; Mai et al., 2021) use auxiliary networks to learn the mixing policy in a data-driven way to avoid the generation of ambiguous data. Although existing works effectively reduce the under-fitting, they have limitations to properly regularization networks. Current networks are prone to be over-fitting when adding the extra parameters. Eventually, these methods degrade the effects of regularization. The conflicts between over-fitting and under-fitting get more serious when the labeled resources are rare or hard to obtain. Besides, the methods with auxiliary networks usually have difficulties in integrating with other *Mixup* variants. More importantly, *Mixup* works well in most cases (Guo et al., 2019b). Adding too much non-linearity into *Mixup* will sacrifice the majority of synthetic data that can regularize the networks under locally linear constraints. So, the locally linear constraints in *Mixup* only need to be slightly

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2998–3008
November 7–11, 2021.

relaxed.

In this paper, we propose the Adversarial Mixing Policy (***AMP***) to overcome these limitations. We modify the adversarial training (Goodfellow et al., 2015), which relaxes the linear nature of the network without any extra parameters or auxiliary networks, to relax the Locally Linear Constraints in *Mixup*. Inspired by the "min-max" formulation of adversarial training, we formulate our method as a form of "min-max-rand" regularization. Specifically, the "rand" operation randomly samples a mixing coefficient as in vanilla *Mixup* to generate synthetic example and label. Then, the "max" operation calculates the perturbation of the mixing coefficient and applies it. Note that the updated mixing coefficient is only used to re-synthetic example, keeping the synthetic label unchanged. Thus, slight non-linearity is injected in-between the synthetic example and label. Finally, the "min" operation minimizes the training loss over the non-linearly generated example-label pairs. In summary, we highlight the following contributions:

- We propose an Adversarial Mixing Policy (AMP) to relax the Locally Linear Constraints (LLC) in *Mixup* without any auxiliary networks. It can be seamlessly integrated into other *Mixup* variants for its simplicity.
- To the best of our knowledge, this is the first exploration of the application of adversarial perturbation to the mixing coefficient in *Mixup*.
- We analyze our proposed method with extensive experiments and show that our AMP improves the performance of two *Mixup* variants on various settings and outperforms the non-linear *Mixup* in terms of error rate.

2 Background

2.1 Linear nature of the networks

Let $(x; y)$ be a sample in the training data, where x denotes the input and y the corresponding label. Deep networks learns a mapping function from x to y, which is:

$$f(x) = y' \rightarrow y\,. \tag{1}$$

Here, y' is the output of the networks, $\rightarrow$ represents the learning process. The linear nature of networks can be interpreted as that a small change in the input will lead to a change of model output:

$$f(x + \nabla x) = y' + \nabla y\,. \tag{2}$$

Here, ∇x is a small perturbation of x, and ∇y is the changing of output caused by the injection of ∇x. This linearity causes the networks vulnerable to adversarial attacks (Goodfellow et al., 2015).

2.2 Relax the linear nature

To relax the linear nature of the networks, adversarial training (Goodfellow et al., 2015) forces the networks to learn the following mapping function,

$$f(x + \nabla x) = y' \rightarrow y\,, \tag{3}$$

where ∇x is an small adversarial perturbation. Such kind of training can effectively relax the linearity of networks and improve the robustness of deep networks. However, there exists a trade-off between model robustness(Equation. 3) and generalization(Equation. 1)(Tsipras et al., 2019).

2.3 Locally linear constraints in *Mixup*

Mixup can be formulated as follows,

$$f(m_x(\lambda)) = y' \rightarrow m_y(\lambda)\,, \tag{4}$$
$$m_x(\lambda) = x_1 \cdot \lambda + x_2 \cdot (1 - \lambda)\,, \tag{5}$$
$$m_y(\lambda) = y_1 \cdot \lambda + y_2 \cdot (1 - \lambda)\,, \tag{6}$$

where $\lambda \in [0, 1]$ is the mixing coefficient. m is the mixing policy. $(x_1; y_1)$ and $(x_2; y_2)$ are a pair of examples from the original training data. By training on synthetic data, $m_x(\lambda)$ and $m_y(\lambda)$, *Mixup* (Zhang et al., 2018; Verma et al., 2019) imposes the Locally Linear Constraints on the input space of networks. Different from Eq. 2, this linearity can be formulated as follow,

$$f(m_x(\lambda+\nabla\lambda)) = y'+\nabla y \rightarrow m_y(\lambda+\nabla\lambda)\,. \tag{7}$$

Here, the $\nabla\lambda$ is a small change in λ. We can observe that the output of the networks is changed accordingly. That is similar to the form of the linear nature of networks. Under these settings, the small change in λ often leads to an undesirable change of output. Eventually, these strict linear constraints lead to under-fitting that degrades the regularization effects (Guo et al., 2019b; Guo, 2020).

2.4 Why relaxing locally linear constraints

Relaxing the strict linear constraints in *Mixup* can alleviate the under-fitting and therefore improve

the regularization effects (Guo, 2020). The under-fitting happens when the synthetic data is corrupted or ambiguous for the network. So, if we can make the networks compatible with such data, like the soft margin (Suykens and Vandewalle, 1999), the under-fitting will be eased. Furthermore, such a technique is best realized the relaxing without extra parameters. Inspired by the adversarial training (Eq. 3), we hypothesize that injecting slight non-linearity into *Mixup* can relax its constraints without extra parameters as follow,

$$f(m_x(\lambda + \nabla\lambda)) = y' \rightarrow m_y(\lambda)\,, \tag{8}$$

where $\nabla\lambda$ is an adversarial perturbation injected to the original mixing coefficient λ.

3 Methodology

As shown in Figure 1, Adversarial Mixing Policy (**AMP**) consists of three operations: **Rand**, **Max** and **Min**. Rand Operation (RandOp) generates the synthetic data by interpolating pairs of training examples and their labels with a random mixing coefficient λ. Max Operation (MaxOp) injects a small adversarial perturbation into the λ to re-synthesize the example and keeps the synthetic label unchanged. This operation injects slight non-linearity into the synthetic data. Min Operation (MinOp) minimizes the losses of these data. Additionally, we use a simple comparison to eliminate the influence caused by the scaling of gradients.

3.1 Method formulation

Given a training set $D = \{x_i, y_i\}$ of texts, in which each sample includes a sequence of words x_i and a label y_i. A classification model encodes the text into a hidden state and predicts the category of text. *Mixup*'s objective is to generate interpolated sample $\hat{g_k}$ and label $\hat{y}$ by randomly linear interpolation with ratio λ applied on a data pair$(x_i; y_i)$ and $(x_j; y_j)$. Our method aims to project a perturbation $\nabla\lambda$ into λ to maximize the loss on interpolated data. Then, it minimizes the maximized loss. Inspired by adversarial training, we formulate this problem as a ***min-max-rand*** optimization problem,

$$\min_{\theta} \mathbb{E}_{\hat{D}} \max_{|\nabla\lambda| \leq \varepsilon} \ell_{mix}(\underset{\lambda \sim Beta(\alpha,\alpha)}{f_{rand}}(\lambda + \nabla\lambda, i, j, k); \theta)\,. \tag{9}$$

Here, $\hat{D} = \{\hat{g_{k_i}}, \hat{y_i}\}$ is the synthetic data set generated by $f_{rand}(\lambda, i, j)$, $\nabla\lambda$ is the adversarial perturbation of λ, ε is the maximum step size, $\ell_{mix}(*)$ is the *Mixup* loss function, $f_{rand}(*)$ represent the random interpolation of data and labels, λ is the random mixing coefficient sampled from a $Beta$ distribution with α parameters, i and j are the randomly sampled data indexes in D, k is the mixed layer.

3.2 Rand operation

Rand Operation (RandOp) is identical to *Mixup* (Zhang et al., 2018). It aims to generate random interpolated data between two categories. Specifically, it generates synthetic labeled data by linearly interpolating pairs of training examples as well as their corresponding labels. For a data pair $(x_i; y_i)$ and $(x_j; y_j)$, x denotes the examples and y the one-hot encoding of the corresponding labels. Consider a model $f(x) = f_k(g_k(x))$, g_k denotes the part of the model mapping the input data to the hidden state at layer k, and f_k denotes the part mapping such hidden state to the output of $f(x)$. The synthetic data is generated as follows,

$$\lambda \sim Beta(\alpha, \alpha)\,, \tag{10}$$

$$\hat{g_k} = g_k(x_i) \cdot \lambda + g_k(x_j) \cdot (1 - \lambda)\,, \tag{11}$$

$$\hat{y} = y_i \cdot \lambda + y_j \cdot (1 - \lambda)\,, \tag{12}$$

where λ is the mixing coefficient for the data pair, α indicates the hyper-parameter of $Beta$ distribution, $\hat{g_k}$ is the synthetic hidden state. For efficient computation, the mixing happens by randomly picking one sample and then pairs it up with another sample drawn from the same mini-batch (Zhang et al., 2018). Here, the sample is obtained randomly. To simplify, we reformulate the random interpolation $f_{rand}(*)$ as follow,

$$(f_k(\hat{g_k}), \hat{y}) := \underset{\lambda \sim Beta(\alpha,\alpha)}{f_{rand}}(\lambda, i, j, k)\,. \tag{13}$$

Here, $f_{rand}(*)$ takes the results of Equation 10- 12 as input, outputs the model predictions $f_k(\hat{g_k})$ and the label $\hat{y}$. The model trained on the generated data tends to reduce the volatility of prediction on these data. Then, the model will generalize better on unseen data.

3.3 Max operation

Max operation (MaxOp) injects a small adversarial perturbation to inject slight non-linearity between the synthetic example and synthetic label. It means that the generated synthetic data will not strictly follow the Locally Linear Constraints in *Mixup*. To achieve this, we propose an algorithm,

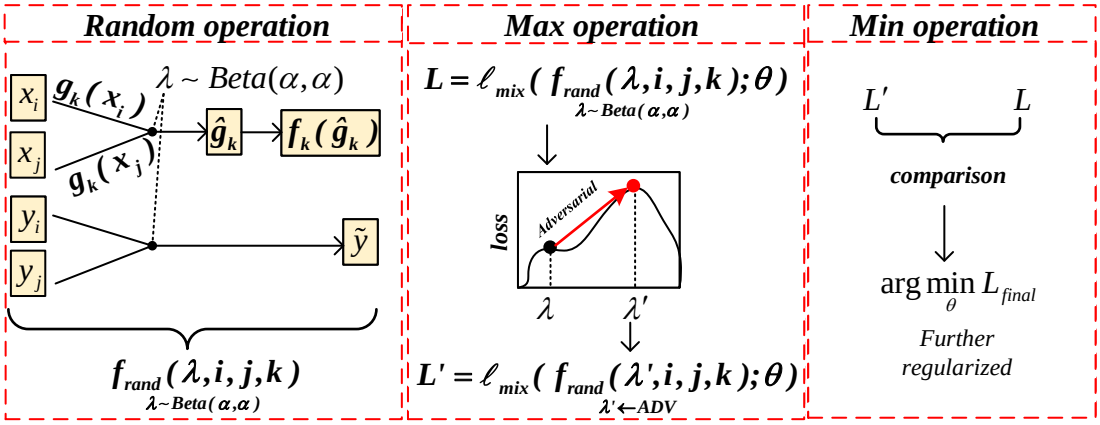


Figure 1: The major operations of Adversarial Mixing Policy (*AMP*).

which is similar to the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), to inject an adversarial perturbation to the λ. It calculates the gradient of λ in the gradient ascend direction,

$$\max_{|\nabla\lambda|\leq\varepsilon} \ell_{mix}(f_{rand}(\lambda+\nabla\lambda,i,j,k);\theta)\,, \quad (14)$$
$$\lambda\sim Beta(\alpha,\alpha)$$

where the $\nabla\lambda$ is the gradients of λ on gradient ascent direction, ε is the step size. Different from the FGSM (Goodfellow et al., 2015), we add a small perturbation on λ instead of the input. Besides, the λ is a scalar, we can get the adversarial direction and strength directly. So, there is no need to perform the normalization on $\nabla\lambda$.

$$\lambda' = \lambda + \varepsilon\cdot\nabla\lambda\,, \quad (15)$$

where λ' is the slight hardness version of mix coefficient, ε is the step size, $\nabla\lambda$ is the clipped (≤ 1) gradient of λ. The perturbation is the gradient in the adversarial direction. We calculate the gradient of λ as follow,

$$\nabla\lambda = \frac{\partial\mathcal{L}}{\partial\lambda}\,. \quad (16)$$

Here, the *Mixup* loss $\mathcal{L}$ is calculated by interpolation of losses on pair of labels (Zhang et al., 2018; Verma et al., 2019) as follow,

$$\begin{aligned}\mathcal{L} =& \ell_{mix}(f_{rand}(\lambda,i,j,k);\theta)\\ &\lambda\sim Beta(\alpha,\alpha)\\ =& \ell_{ce}(f_k(\hat{g}_k),y_i;\theta)\cdot\lambda +\\ & \ell_{ce}(f_k(\hat{g}_k),y_j;\theta)\cdot(1-\lambda)\,.\end{aligned} \quad (17)$$

Here, $\mathcal{L}$ represents the loss of synthetic data generated under mixing coefficient λ, θ is the parameters of the model, $\ell_{mix}(*)$ is the *Mixup* loss, $\ell_{ce}(*)$ represents the cross-entropy function. Notable that the step size of gradient ε may lead to undesirable results that minimize the losses. So, we need to eliminate the influence caused by ε.

3.4 Min operation

Min operation (MinOp) minimizes loss of constraints relaxed synthetic data as follow,

$$\arg\min_\theta \mathcal{L}_{final}\,, \quad (18)$$

where $\mathcal{L}_{final}$ is the final loss. In addition, MinOp leans to minimize the larger loss in the previous two steps to eliminate the influence of the step size ε. Besides, this preference will help model learning from the one with larger loss to reduce the risk of under-fitting. We use a mask-based mechanism to realize the operation as follow,

$$\mathcal{L}_{final} = \mathcal{L}\cdot(1-mask) + \mathcal{L}'\cdot mask\,. \quad (19)$$

Here, the $mask$ is used as a selector of losses. The comparison is carried out on losses before and after updated λ in the synthetic example. The latter one $\mathcal{L}'$ is calculated as follow,

$$\mathcal{L}' = \ell_{mix}(f_{rand}(\lambda',i,j,k);\theta)\,. \quad (20)$$
$$\lambda'\leftarrow ADV$$

Here, λ' is the mixing coefficient after injecting perturbation (we only inject the perturbation into mixing coefficient of input, as Eq. 8), $\mathcal{L}'$ is the *Mixup* loss on synthetic example generated under λ'. Note that the λ for the synthetic label is unchanged. $mask$ is calculated as follow,

$$mask = \begin{cases}1 & \delta_{\mathcal{L}} > 0\\ 0 & \delta_{\mathcal{L}} \leq 0\,.\end{cases} \quad (21)$$

Here, the $mask$ is batch size vector, $\delta_{\mathcal{L}}$ is the direct comparison $\mathcal{L}' - \mathcal{L}$. By doing this, the proposed method achieves steady improvement under different settings of step size.

4 Experiments

4.1 Data

We evaluate the proposed *AMP* on five sentence classification benchmark datasets as used in (Guo et al., 2019a). TREC is a question dataset which aims to categorize a question into six types (Li and Roth, 2002). MR is a movie review dataset aiming at classifying positive/negative reviews (Pang and Lee, 2005). SST-1 is the Stanford Sentiment Treebank dataset with five sentiment categories: very positive, positive, neutral, negative, and very negative (Socher et al., 2013). SST-2 is a binary label version of SST-1. SUBJ is a dataset aiming to judge a sentence to be subjective or objective (Pang and Lee, 2004). Table 1 summarizes the statistical characteristics of the five datasets after prepossessing.

Table 1: The statistics of datasets. c is the category number. l is the average length. V is the vocabulary size. N is the size of the training set. T is the size of the testing set. CV denotes the 10-fold cross-validation.

Data	c	l	V	N	T
TREC	6	10	9592	5952	500
SST-1	5	18	17836	11855	2210
SST-2	2	19	16185	9613	1821
SUBJ	2	23	21323	10000	CV
MR	2	20	18765	10662	CV

4.2 Baselines and Settings

Our *AMP* is evaluated by integrating to two recent proposed *Mixup* variants. We choose five popular sentence classification models as the backbone to test the performance of all *Mixups* on the five benchmark datasets.

Classification backbone. We test *Mixups* on five classification backbones. $LSTM_{rand}$ and $LSTM_{glove}$ (Wang et al., 2016) are two versions of bi-directional Long Short Term Memory(LSTM) with attention, where the former uses randomly initiated word embeddings and the latter uses GloVe (Pennington et al., 2014) initiated word embeddings. CNN_{rand} and CNN_{glove} (Kim, 2014) are two versions of convolutional neural networks. They are fed with randomly and GloVe initiated word embeddings, repectively. The above four methods are popular sentence classification models without pre-training techniques. We employ $BERT_{base}$ (Devlin et al., 2019) as the pre-training classification backbone.

Mixup. We choose three popular *Mixup* variants for sentence classification as baselines. ***Word-Mixup*** (Guo et al., 2019a) is the straightforward application of *Mixup* on NLP tasks where linear interpolation applying on the word embedding level (first layer). ***SentMixup*** (Verma et al., 2019; Sun et al., 2020) is the *Mixup* applying to NLP tasks where linear interpolation is conducted in the last layer of hidden states. ***Non-linear Mixup*** is the non-linear version of *SentMixup*.

AMP. ***WordAMP*** is applied on the word embedding level, the same as *WordMixup*. ***SentAMP*** is applied on the last layer of hidden states, the same as *SentMixup*.

We obtained the source codes of backbone models from the public available implementations[1]. In our experiments, we follow the exact implementation and settings in (Kim, 2014; Wang et al., 2016; Devlin et al., 2019; Guo et al., 2019a; Verma et al., 2019). Specifically, we use filter sizes of 3, 4, and 5, each with 100 feature maps; dropout rate of 0.5 and L2 regularization of 1e-8 for the *CNN* baselines. We use hidden size of 1024 of single-layer; dropout rate of 0.5 and L2 regularization of 1e-8 for the *LSTM* baselines. For datasets without a standard development set, we randomly select 10% of training data as a development set. Training is done through Adam (Kingma and Ba, 2015) over mini-batches of size 50 (*CNN*, *LSTM*) and 24 ($BERT_{base}$) respectively. The learning rate is 2e-4 for *CNN* and *LSTM*, and 1e-5 for $BERT_{base}$. The word embeddings are 300 dimensions for *CNN* and *LSTM*. The step size $\varepsilon = 0.002$ for all experiments. The α for all *Mixup* is set to one. For each dataset, we train each model 10 times with different random seeds each with 8k steps and compute their mean error rates and standard deviations.

4.3 Main results

To evaluate the predictive performance of *AMP*, we conduct five sets of experiments. For each setting, we compare the performance of without *Mixup* (*w/o*), *WordMixup* (*Word*), *SentMixup* (*Sent*) and *non-linear Mixup*(*non-linear*[2]. As presented in Table 2, *AMP* outperform *Mixup* comparison baselines. For example, compared with the *Sent* base-

[1]LSTM: https://github.com/songyouwei/ABSA-PyTorch
CNN: https://github.com/harvardnlp/sent-conv-torch
BERT: https://github.com/huggingface/transformers
GloVe: https://nlp.stanford.edu/projects/glove/

[2]Only on CNN_{glove} our baseline results close to the baseline results reported in (Guo, 2020)). For a fair comparison, we only cite the results of CNN_{glove} of non-linear *Mixup*.

Table 2: The results of our *AMP* method compared with two recent *Mixup* methods on five different datasets under five different classification models. For a fair comparison, we re-implement the *Mixup* baselines based on backbone models. The results may not the same as the results in (Guo et al., 2019a; Sun et al., 2020). RP indicates the relative improvement.† indicates the results are cited from (Guo, 2020).

Model	*Mixup*	TREC(%)	SST-1(%)	SST-2(%)	SUBJ(%)	MR(%)
RNN_{rand}	*w/o*	11.3±1.48	63.7±3.00	18.0±0.85	10.7±0.57	24.9±1.11
	Sent	10.5±1.16	55.8±0.75	16.6±0.38	10.3±0.55	24.2±0.72
	*Sent(**our**)*	9.8±0.73	**55.0±0.37**	15.9±0.43	10.0±0.78	23.6±0.65
	RP(%)	$6.7_{\uparrow}$	$1.4_{\uparrow}$	$4.2_{\uparrow}$	$2.9_{\uparrow}$	$2.5_{\uparrow}$
	Word	9.8±0.86	55.9±0.62	16.1±0.62	9.4±0.77	23.6±0.75
	*Word(**our**)*	**9.5±0.84**	55.6±0.67	**15.3±0.43**	**8.8±0.48**	**22.7±0.96**
	RP(%)	$3.1_{\uparrow}$	$0.5_{\uparrow}$	$5.0_{\uparrow}$	$6.4_{\uparrow}$	$3.8_{\uparrow}$
RNN_{glove}	*w/o*	8.3±0.47	56.6±0.30	13.0±0.51	6.1±0.76	18.5±0.97
	Sent	6.9±0.55	48.1±0.37	12.1±0.61	6.0±0.69	18.1±0.95
	*Sent(**our**)*	6.7±0.27	**48.0±0.45**	11.5±0.31	5.8±0.79	17.8±0.98
	RP(%)	$2.9_{\uparrow}$	$0.2_{\uparrow}$	$5.0_{\uparrow}$	$3.3_{\uparrow}$	$1.7_{\uparrow}$
	Word	**6.5±0.45**	48.6±0.33	11.8±0.34	5.5±0.73	17.8±0.87
	*Word(**our**)*	6.6±0.52	**48.0±0.66**	**11.1±0.42**	**5.2±0.72**	**17.5±0.91**
	RP(%)	$1.5_{\downarrow}$	$1.2_{\uparrow}$	$5.9_{\uparrow}$	$5.5_{\uparrow}$	$1.7_{\uparrow}$
CNN_{rand}	*w/o*	8.8±0.86	63.2±0.54	**17.6±0.52**	9.5±0.64	24.2±1.39
	Sent	8.3±0.63	58.1±0.48	19.9±0.32	9.5±0.52	25.1±0.91
	*Sent(**our**)*	**8.1±0.71**	57.9±0.51	19.9±0.51	9.4±0.45	25.1±0.93
	RP(%)	$2.4_{\uparrow}$	$0.5_{\uparrow}$	$\rightarrow$	$1.1_{\uparrow}$	$\rightarrow$
	Word	8.3±0.71	58.0±0.55	19.4±0.22	9.7±0.57	24.6±0.78
	*Word(**our**)*	8.4±0.92	**57.5±0.50**	19.2±0.53	**9.2±0.68**	**24.1±0.98**
	RP(%)	$1.2_{\downarrow}$	$1.0_{\uparrow}$	$1.0_{\uparrow}$	$5.2_{\uparrow}$	$2.0_{\uparrow}$
CNN_{glove}	*w/o*	7.9±0.12	57.5±0.50	13.1±0.49	5.6±0.36	20.2±0.60
	Non-linear	5.3±0.29†	50.7±0.42†	**11.4±0.29**†	6.1±0.19†	**16.6±0.36**†
	Sent	6.7±0.23	51.4±0.23	12.8±0.35	5.1±0.34	19.4±0.56
	*Sent(**our**)*	**4.6±0.33**	50.6±0.40	11.7±0.25	**5.1±0.62**	17.4±0.69
	RP(%)	$31.3_{\uparrow}$	$1.6_{\uparrow}$	$8.6_{\uparrow}$	$\rightarrow$	$10.3_{\uparrow}$
	Word	6.3±0.80	51.8±0.91	12.9±0.26	5.3±0.45	18.7±0.28
	*Word(**our**)*	4.8±0.26	**50.4±0.60**	11.7±0.24	**5.1±0.58**	17.4±0.66
	RP(%)	$23.8_{\uparrow}$	$2.7_{\uparrow}$	$9.3_{\uparrow}$	$3.8_{\uparrow}$	$7.0_{\uparrow}$
$BERT_{base}$	*w/o*	2.6±0.18	47.3±0.47	6.9±0.21	2.4±0.47	11.5±1.19
	Sent	2.2±0.24	44.5±0.37	6.3±0.29	2.4±0.56	11.3±1.44
	*Sent(**our**)*	2.1±0.20	**44.3±0.54**	**5.9±0.30**	2.3±0.49	11.2±1.31
	RP(%)	$4.5_{\uparrow}$	$0.4_{\uparrow}$	$9.5_{\uparrow}$	$4.2_{\uparrow}$	$0.9_{\uparrow}$
	Word	2.1±0.20	45.6±0.37	6.5±0.25	2.3±0.54	11.1±1.44
	*Word(**our**)*	**1.9±0.13**	45.5±0.37	6.4±0.23	**2.2±0.56**	**10.8±1.29**
	RP(%)	$9.5_{\uparrow}$	$0.2_{\uparrow}$	$1.5_{\uparrow}$	$4.3_{\uparrow}$	$2.7_{\uparrow}$

line over CNN_{glove}, *Sent(our)* achieves a significant improvement on all five datasets. For instance, *Sent(our)* outperform *Sent* on the TREC, SST2 and MR datasets over CNN_{glove}, the relative improvements are 31.3%, 8.6% and 10.3%, respectively[3]. Compared with *Word* over RNN_{glove}, *Word(our)* reduces the error rate over 1.2% (up to 5.9%) on all five testing datasets. Interestingly, one can see that the *Word(our)* outperform *Non-linear Mixup* on three out of five datasets. That shows the slightly relaxing of LLC achieves similar sometimes even better results than changing the LLC into a non-linear version.

We use different initial embeddings to evaluate the effectiveness of augmentation as (Guo et al., 2019a). From the embedding perspective, we have three kinds of embeddings: the randomly initiated embeddings (RNN_{rand} and CNN_{rand}), the pre-trained fixed embeddings (RNN_{glove} and CNN_{glove}) and the pre-trained context-aware embeddings ($BERT_{base}$). For each kind of embeddings, *AMP* outperforms the *Mixup* baselines. For instance, when compared with *Sent* under randomly initiated embeddings, the proposed method *Sent(our)* obtains lower predictive error rate on eight out of ten experiments. While *Word(our)* outperforms *Word* on nine out of ten experiments. Similar results can be observed on the pre-trained embeddings settings. Even under the context-aware embeddings setting ($BERT_{base}$), our *AMP* can fur-

[3]Our methods are tuned on CNN_{glove} may cause the significant higher level of improvements.

ther improve the performance against the *Mixup* with advanced backbone models. Significantly, on SST1, our method help $BERT_{base}$ outperforms the SOTA model ($BERT_{large}$, 44.5) (Munikar et al., 2019), which is as two times large as $BERT_{base}$. The results show the effectiveness of our method.

Table 3: The results of $BERT_{base}$ with *SentAMP* on low-resource settings. The experiments are run ten times on each scaled TREC datasets. The average error rate and standard deviation are reported.

%	labels	*Sent*	*Sent(**our**)*	*RP(%)*
3	160	51.0±7.34	42.1±7.34	**+17.5**
4	215	29.8±4.05	25.6±4.01	+14.1
5	270	10.2±1.00	9.2±0.80	+9.8
10	543	5.1±0.64	4.6±0.37	+9.8
15	815	4.1±0.64	4.0±0.67	+2.4
20	1089	3.6±0.62	3.5±0.48	+2.8
40	2179	2.9±0.35	2.7±0.38	+6.7
80	4359	2.2±0.17	2.1±0.10	+4.5
100	5452	2.2±0.24	2.1±0.20	+4.5

4.4 Low-resource conditions

With low resources, the under-fitting caused by the strict LLC has a serious impact on the model generalization. To evaluate our *AMP* performance with different amounts of data, particularly in the case of low-resource settings. We scale the size of the dataset by a certain ratio of data for each category. If the scaled category is less than 0, we retain at least one sample. We randomly generate ten different datasets for each scale ratio and then run the experiment on each dataset. The mean error rate and standard deviation are reported. As shown in Table 3, we can see that our method reduces the mean error rate against *Mixup* with a significant margin. For instance, *Sent(our)* reduces the error rate over *Sent* with 17.5% and 14.1% on 3% and 4% training data, separately. AMP works well as we expected in low resource conditions for its effectiveness in relaxing LLC in *Mixup*.

4.5 Ablation study

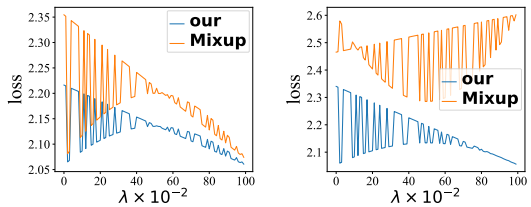
To further understand the Max Operation (MaxOp) and Min Operation (MinOp) effects in *AMP*, we make several variations of our model. The variations are tested under CNN_{glove} and $BERT_{base}$ on TREC. As presented in Table 4, the model trained without augmentation is denoted as $Baseline$. $+RandOp$ is identical to the model trained with *Mixup*, $+MaxOp$ indicates *Mixup* with MaxOp is used for model training, *AMP* is the fully functional method of our proposed method. As the results presented in Table 4, MaxOp contributes the majority cut down of error rate. For instance, the CNN_{glove} under *Sent Mixup* settings, MaxOp reduces the error rate from 6.7 to 4.8. That suggests the effectiveness of adversarial perturbation in relaxing the LLC in *Mixup*. The comparison in MinOp can mostly (three out of four times) further reduce the error rate. Specifically, it brings down the mean error rate from 4.8 to 4.6 on CNN_{glove}. That indicates the effectiveness of MinOp in eliminating the influence of step size.

Table 4: Ablation study.

Method	Model	Operation	TREC
Word	CNN_{glove}	*Baseline*	7.9±0.12
		+RandOp	6.3±0.80
		+MaxOp	4.7±0.35
		AMP	4.8±0.26
	$BERT_{base}$	*Baseline*	2.6±0.18
		+RandOp	2.1±0.24
		+MaxOp	2.0±0.23
		AMP	1.9±0.13
Sent	CNN_{glove}	*Baseline*	7.9±0.12
		+RandOp	6.7±0.23
		+MaxOp	4.8±0.22
		AMP	4.6±0.33
	$BERT_{base}$	*Baseline*	2.6±0.18
		+RandOp	2.2±0.24
		+MaxOp	2.1±0.13
		AMP	2.1±0.15

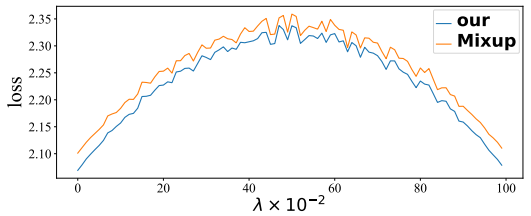
Table 5: The results under different setting of α.

α	Methods	TREC	SST2	MR
0.2	*Word*	1.9±0.13	6.3±0.23	11.0±1.25
	Word(our)	1.8±0.13	6.0±0.20	10.9±1.22
	RP(%)	+5.3	+4.8	+0.9
0.5	*Word*	1.9±0.13	6.7±0.24	11.1±1.25
	Word(our)	1.9±0.16	6.1±0.18	10.8±1.25
	RP(%)	+0.0	**+8.9**	+2.7
1.0	*Word*	2.1±0.20	6.5±0.25	11.1±1.44
	Word(our)	2.0±0.12	6.4±0.23	10.8±1.29
	RP(%)	+4.8	+1.5	+2.7
1.5	*Word*	2.1±0.18	6.8±0.13	11.2±1.44
	Word(our)	2.0±0.12	6.5±0.28	11.0±1.34
	RP(%)	+4.8	+4.4	+1.8



(a) Random pair1

(b) Random pair2



(c) Full-size testing set

Figure 2: The visualization of loss on unseen synthetic data. The results conduct by $BERT_{base}$ on 3% TREC dataset, as listed in Table 3.

4.6 Mix ratio distribution

To analyze the effects of different shapes of mixing coefficient distributions, we compare *Word(out)* with *Word* on $BERT_{base}$ on four α settings (from 0.2 to 1.5) and three datasets: TREC, SST2, and MR. The α is the parameter of the $Beta$ distribution. It controls the shape of how the mixing coefficient λ is distributed. As presented in Table 5, our method can achieve lower mean error rates than *Word* on all α settings. For instance, *Word(our)* achieve 8.9% lower mean error rate than *Word* on SST2 with $\alpha = 0.5$. The improvements come mainly from training the models with the slightly non-linear data generated by AMP.

4.7 Visualization

To intuitively demonstrate the effects of relaxing LLC, we visualize the loss of networks trained by our *AMP* and *Mixup*. The synthetic data is generated strictly follow the LLC based on the testing data. The network trained with relaxed LLC has a smaller loss value shows the effectiveness of our method in alleviate under-fitting. As shown in Figure 2(a), 2(b) and 2(c), we draw the losses on synthetic data generated with mixing coefficient $\in [0, 1]$. Figure 2(a) and 2(b) each uses one random pair of data in the testing set for generating. For two random pair $(x_1, y_1)(x_4, y_4)$ and $(x_2, y_2)(x_3, y_3)$, we calculate the Mixup loss of each pair on different λ to get Figure 2(a) and 2(b). The loss curves on random pairs are not symmetric for the loss of each example of the pairs are different. The loss curves are encouraged (by LLC) to be a line in-between two examples. The line should start with the loss of one example and end with the loss of another example. The Mixup loss (interpolation on cross-entropy loss) and the different examples result in different shapes of the loss curves in Figure 2(a) and 2(b).As illustrated in Figure 2(a) and 2(b), one can observe that *AMP* have a smaller loss than *Mixup*. That indicates the effectiveness of training on the slightly non-linear synthetic data in the micro view.

Figure 2(c) uses the full-size testing set for generating. Figure 2(c) shows the average loss over all synthetic data generated with the full-size testing set. We freeze the random seeds; thus, we can freeze the data pairs. Let the testing dataset be $X = [(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)]$. The synthetic data is generated by $\lambda X + (1 - \lambda)X$, where $X' = [(x_4, y_4), (x_3, y_3), (x_2, y_2), (x_1, y_1)]$ is shuffled X. So, the loss when $\lambda = 0$ and $\lambda = 1$ are identical. Similarly, we can get a symmetric picture as Figure 2(c).One can observe that our method can achieve a significantly smaller average loss than *Mixup* in the macro view. The visualizations verified our assumption that relaxing LLC can further regularize models.

5 Related work

***Mixup* on text classification**. Text classification has achieved remarkable improvements underlying some effective paradigms, e.g., *CNN* (Kim, 2014), attention-based *LSTM*s (Wang et al., 2016), GloVe (Pennington et al., 2014) and *BERT* (Devlin et al., 2019), etc. The large scale parameter of the model tends to generalize poorly in low-resource conditions. To overcome the limitation, *Mixup* (Zhang et al., 2018) is proposed as a data augmentation based regularizer. Few researches explore the *Mixup* (Guo et al., 2019b; Zhang et al., 2020; Guo, 2020) on NLP tasks. For classification, (Guo et al., 2019a) suggest applying *Mixup* on particular level of networks, i.e., word or sentence level. Although these work make promising progress, the mechanism of Mixup is still need to be explored.

Adversarial Training. The min-max formulation of adversarial training has been theoretically and empirically verified (Beckham et al., 2019; Xu et al., 2020; Pang et al., 2020; Archambault et al., 2019; Lee et al., 2020; Miyato et al., 2015, 2018, 2017). Such training procedure first generates ad-

versarial examples that might maximize the training loss and then minimizes the training loss after adding the adversarial examples into the training set (Madry et al., 2018). The Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) is an efficient one-step method. Inspired by the min-max formulation of adversarial learning, we organize our method into a min-max-rand formulation.

6 Conclusion

For relaxing Locally Linear Constraints (LLC) in *Mixup* to alleviate the under-fitting, this paper proposes an Adversarial Mixing Policy (***AMP***). Inspired by the adversarial training, we organize our method into a ***min-max-rand*** formulation. The proposed method injects slightly non-linearity in-between synthetic examples and synthetic labels without extra parameters. By training on these data, the networks can compatible with some ambiguous data and thus reduce under-fitting. Thus, the network will be further regularized to reach better performance. We evaluate our method on five popular classification models on five publicly available text datasets. Extensive experimental results show that our *AMP* can achieve a significantly lower error rate than vanilla *Mixup* (up to 31.3%), especially in low-resource conditions(up to 17.5%).

7 Acknowledgments

We thank Prof.Xiaojie Wang and Prof.Fangxiang Feng from BUPT for their valuable feedback on an earlier draft of this paper, and Yang Du from XDF for her suggestions of English writing for the final revision. We also thank anonymous reviewers for their helpful comments.

References

Guillaume P Archambault, Yongyi Mao, Hongyu Guo, and Richong Zhang. 2019. Mixup as directional adversarial training. *arXiv preprint arXiv:1906.06875*.

Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R. Devon Hjelm, Yoshua Bengio, and Chris Pal. 2019. On adversarial mixup resynthesis. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4348–4359.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. 2019a. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*.

David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019b. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5050–5060.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4044–4051. AAAI Press.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3714–3722. AAAI Press.

Stephen Hanson and Lorien Pratt. 1988. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1:177–185.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Alex Hernández-García and Peter König. 2018. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. 2020. Adversarial vertex mixup: Toward better adversarially robust generalization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 269–278. IEEE.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Guang Liu, Hailong Huang, Yuzhao Mao, Weiguo Gao, Xuan Li, and Jianping Shen. 2021. A diversity-enhanced and constraints-relaxed augmentation for low-resource classification. In *Database Systems for Advanced Applications - 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11-14, 2021, Proceedings, Part II*, volume 12682 of *Lecture Notes in Computer Science*, pages 262–270. Springer.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zhijun Mai, Guosheng Hu, Dexiong Chen, Fumin Shen, and Heng Tao Shen. 2021. Metamixup: Learning adaptive interpolation policy of mixup with metalearning. *IEEE Transactions on Neural Networks and Learning Systems*.

Xudong Mao, Yun Ma, Zhenguo Yang, Yangbin Chen, and Qing Li. 2019. Virtual mixup training for unsupervised domain adaptation. *arXiv preprint arXiv:1905.04215*.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Tianyu Pang, Kun Xu, and Jun Zhu. 2020. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. 2020. Mixup-transfomer: Dynamic data augmentation for nlp tasks. *arXiv preprint arXiv:2010.02394*.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447. PMLR.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. 2020. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6502–6509.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020. SeqMix: Augmenting active sequence labeling via sequence mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8566–8579, Online. Association for Computational Linguistics.

Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. 2020. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *European Conference on Computer Vision*, pages 633–649. Springer.