Dynamic Nonlinear Mixup with Distance-based Sample Selection

Shaokang Zhang[1,2], **Lei Jiang**[1,2] [*], **Jianlong Tan**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
zhangshaokang@iie.ac.cn; jianglei@iie.ac.cn; tanjianlong@iie.ac.cn

Abstract

Data augmentation with mixup has shown to be effective on the NLP tasks. Although its great success, the mixup still has shortcomings. First, vanilla mixup randomly selects one sample to generate the mixup sample for a given sample. It remains unclear how to best choose the input samples for the mixup. Second, linear interpolation limits the space of synthetic data and its regularization effect. In this paper, we propose the dynamic nonlinear mixup with distance-based sample selection, which not only generates multiple sample pairs based on the distance between each sample but also enlarges the space of synthetic samples. Specifically, we compute the distance between each input data by cosine similarity and select multiple samples for a given sample. Then we use the dynamic nonlinear mixup to fuse sample pairs. It does not use a linear, scalar mixing strategy, but a nonlinear interpolation strategy, where the mixing strategy is adaptively updated for the input and label pairs. Experiments on the multiple public datasets demonstrate that dynamic nonlinear mixup outperforms state-of-the-art methods.

1 Introduction

Deep neural networks have achieved great success in NLP tasks, such as text classification (Zhang et al., 2015), machine translation (Sutskever et al., 2014), and dialogue tasks (Serban et al., 2016). These models usually require a large amount of labeled data, but labeling data is time-consuming and expensive. Data augmentation is a common technology to solve the data scarcity problem. Some of them are based on rules (Wei and Zou, 2019) and models (Edunov et al., 2018) to generate similar text. Augmented data are trained by advanced training methods (Park et al., 2021). On the other hand, mixup (Zhang et al., 2017b) trains the classifier on the synthetic data which are generated by the linear interpolation of the input and label pairs.

Recently, many variants of mixup are proposed and successfully applied to NLP tasks. It mainly includes two categories: input-level mixup (Yoon et al., 2021) and hidden-level mixup (Sun et al., 2020; Guo, 2020). Hidden-level mixup is more popular because of the discrete nature of text data and variable sequence lengths. It usually fuses the hidden vectors like embeddings or intermediate representations. While the hidden-level mixup is effective, it still has some issues. First, mixup produces the sample pairs by randomly choosing a sample for a given sample. How to select the optimal input and label pairs is unclear. Second, the space of synthetic data is limited due to its linear nature. Although nonlinear mixup (Guo, 2020) is proposed and utilizes the matrix mixing policy for the sample pairs, the matrix is fixed during the training process and it is difficult to get the appropriate parameters.

In this paper, we propose a dynamic nonlinear mixup with distance-based sample selection method to address the above problems. First, we compute the distance between each input sample by cosine similarity. Then we pick the Top-K samples with the largest distance and the Top-K samples with the smallest distance for a given sample. Second, we obtain mixed inputs by performing a dynamic mixing strategy on these sample pairs. Unlike nonlinear mixup (Guo, 2020), where the fixed matrix fusion method is applied to the sample pairs, our method uses the vector mixing policy and dynamically updates its parameters. The feature space of the generated data is further enlarged by this way. Since the dimensions of the one-hot label and vector mixing policy are different, we learn the label embedding instead of the original label. The mixing policy of labels is adaptively learned based on the mixed input. In summary, the main contributions of our work are summarized as

[*]Corresponding author

follows:

- We propose the dynamic nonlinear mixup, which expands the space of the generated data by the dynamic vector mixing policy. Besides, we introduce distance-based sample selection to generate more mixup samples, which improves the generalization ability of the model.
- We conducted comparative experiments on four datasets. The experimental results demonstrate that our method outperforms other state-of-the-art methods, especially in low-resource scenarios.

2 Related work

Traditional data augmentation methods are categorized into adding noise and back-translation. Easy Data Augmentation (EDA) (Wei and Zou, 2019) proposes four methods to add noise: synonym replacement, random insertion, random deletion, and random swap. It can create a large number of augmented training samples but may lose semantic information due to random deletion. Back-translation (BT) (Zhang and Zong, 2016; Edunov et al., 2018) trains target-to-source system to generate the source language. BT keeps the semantics of the original text, but the augmentation data depends on the quality of the translation model.

Another class of methods is based on the technique of interpolation. A data augmentation method mixup (Zhang et al., 2017b) is proposed and has shown good performance in the image classification. Mixup trains the classifier on the synthetic data which are generated by the linear interpolation of the input and label pairs. In NLP tasks, it usually uses hidden-level mixup. WordMixup and senMixup (Guo et al., 2019) are two linear interpolation methods that are implemented on the word embeddings and sentence embeddings. Sun et al. (2020) incorporate mixup to transformer-based pre-trained architecture to boost the performance of NLU tasks. Park and Caragea (2022) propose a novel mixup strategy for pre-trained language models by both the Area Under the Margin (AUM) statistic and the saliency map of each sample. Chen et al. (2020) propose a data augmentation method called Tmix for semi-supervised learning, which interpolates labeled and unlabeled samples in the hidden space to improve the performance of text classification. BatchMixup (Yin et al., 2021) utilizes interpolation strategy in a mini-batch to improve model performance on the pre-trained language model RoBERTa (Liu et al., 2019). Yoon et al. (2021) propose the Saliency-Based Span Mixup (SSMix), which generates a sentence while reserving the locality of two original texts by span-based mixing and keeping more tokens related to the prediction relying on saliency information.

The above methods just randomly select one sample for a given sample. The number of generated samples is insufficient. We choose Top-K largest distance samples and Top-K smallest distance samples according to the distance between each sample. It can generate more sample pairs for mixup. Besides, mixup uses linear interpolation which limits the space of generated data. We utilize dynamic nonlinear mixup which expands the space of the generated data. Similar to us, the nonlinear mixup (Guo, 2020) uses the fixed matrix mixing strategy. Finding an appropriate hyperparameter is difficult. In contrast, our method utilizes a vector mixing strategy that can be dynamically updated throughout the training process.

3 Approach

In this section, we first present the mixup, followed by distance-based sample selection, and the details of the dynamic nonlinear mixup. Finally, the training strategy is introduced. Figure 1 gives an overview of our method.

3.1 Mixup

Mixup is first introduced for image classification, which uses the linear interpolation to generate the synthetic data based on the input and label pairs. Specifically, a sample pair (x_i, y_i) and (x_j, y_j), where x and y denote the input samples and corresponding labels, respectively. The synthetic sample is generated as follows:

$$\begin{aligned} x_{ij} &= \lambda x_i + (1-\lambda)x_j \\ y_{ij} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{1}$$

where λ is the mixing-ratio and sampled from a Beta(α,α) distribution with the hyper-parameter $\alpha \in (0, \infty)$. Inspired by its great success in the image domain, this method is introduced into text classification. Unlike the image data, the text is composed of discrete and variable-length tokens. Generally, we extract the sentence embedding of the sample pairs by CNN, LSTM, or Transformer. Then, the sample pairs are linearly interpolated.

3.2 Distance-based Sample Selection

For sample (x_i, y_i), mixup randomly selects another sample (x_j, y_j) to generate the new sample

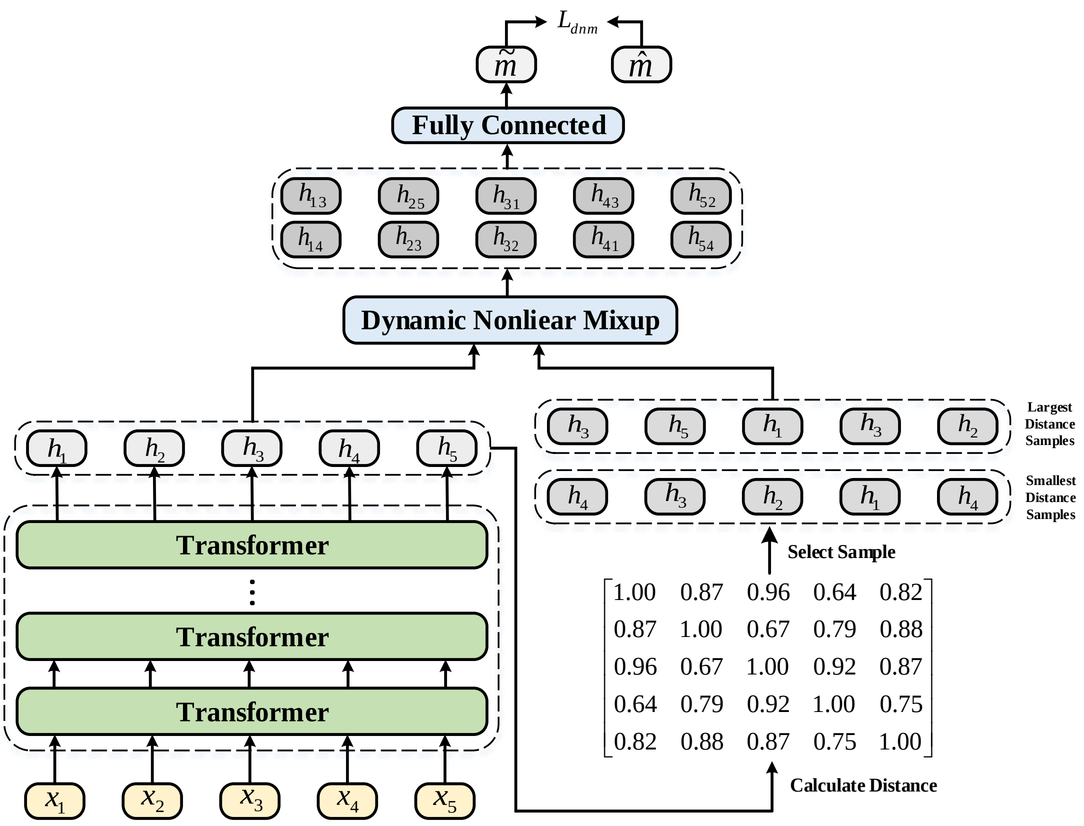


Figure 1: The architecture of the dynamic nonlinear mixup with distance-based sample selection, where $\tilde{m}$ is the predicted value and $\hat{m}$ is the ground truth. L_{dnm} is loss. For easy explanation, we select Top-1 largest distance samples and Top-1 smallest distance samples.

(x_{ij}, y_{ij}). The number of generated samples is limited because only one sample is picked. Intuitively, if the distance between a given sample and another sample is larger, the potential generation space of generated sample is larger. If only the samples with the largest distance are selected, the distribution of synthetic samples may not cover the middle area, which will reduce the performance of the model. In this paper, We propose the distance-based sample selection method which combines Top-K largest distance samples and Top-K smallest distance samples. We assume that there is training data $\mathbf{X} = \{x_i, y_i\}_{i=1}^{N}$, where N is the number of training data. The text representation $\mathbf{H} = \{h_i\}_{i=1}^{N}$ is obtained through the feature encoder. We compute the distance $L \in R^{N\times N}$ between each sample embedding representations by cosine similarity as the basis for selecting samples. For sample (x_i, y_i), we find Top-K largest distance samples $\mathbf{X}^{L_i} = \{x_j^{L_i}, y_j^{L_i}\}_{j=1}^{K}$ and Top-K smallest distance samples $\mathbf{X}^{S_i} = \{x_j^{S_i}, y_j^{S_i}\}_{j=1}^{K}$. We generate 2K mixed samples for a given sample (x_i, y_i) by interpolating the Top-K largest distance samples $\mathbf{X}^{L_i}$ and Top-K smallest distance samples $\mathbf{X}^{S_i}$.

3.3 Dynamic Nonlinear Mixup

To expand the space of generated samples, we propose the dynamic nonlinear mixup method. First, we use dynamic vector mixing policy to obtain the mixed input, where the parameters of the vector are learnable. Second, The mixing policy is a vector instead of a scalar, which cannot be applied to the label pairs because of dimension difference. We use the label embeddings to encode the one-hot labels. The label weights are updated based on the mixed input samples. In the following, we describe the details of this method successively.

3.3.1 Mixed Input

Although nonlinear mixup (Guo, 2020) expands the space of generated samples, the parameters of matrix mixing policy are constant. In this paper, we propose the dynamic nonlinear mixup which updates the mixing policy during the training process. We assume that $v_1, v_2 \in R^{1\times d}$ denote the weight vectors, where d is the dimension. In order to make the probability between 0 and 1, we normalize the vectors v_1 and v_2 in each dimension to get $\hat{v_1}$ and $\hat{v_2}$ by softmax. The input of dynamic nonlinear mixup is computed as follows:

$$h_{ij} = \hat{v_1} \circ h_i + \hat{v_2} \circ h_j \tag{2}$$

where h_{ij} is the input representation of the synthetic sample. $\circ$ denotes the element-wise product. The parameters of the vector are updated throughout the training process. Compared with the matrix mixing method, the space of the synthetic samples is wider and the model has better generalization ability. Note that our method mixes samples at the [CLS] hidden state representations level of the top layer of the pre-trained language model.

3.3.2 Mixed Label

Since our mixing strategy is a vector instead of a scalar, it cannot be applied to one-hot label because the vector and label have different dimensions. To overcome this problem, we use label embedding (Zhang et al., 2017a) which adopts different vectors to encode each category. $M \in R^{c\times d}$ is the label embedding matrix, where c is the number of categories. The mixed label $\hat{m}_{ij}$ is computed as follows:

$$\begin{aligned} \Phi_i &= (\hat{v_1} \circ h_i)W \\ \Phi_j &= (\hat{v_2} \circ h_j)W \\ \hat{m}_{ij} &= \hat{\Phi}_i \circ m_i + \hat{\Phi}_j \circ m_j \end{aligned} \tag{3}$$

where $W \in R^{d\times d}$ is the weight matrix. Similarly, we also normalize the weight vector Φ_i, Φ_j in each dimension to get $\hat{\Phi}_i, \hat{\Phi}_j$. m_i is the label embedding of the i-th sample. In this way, the target weight vector is generated based on the mixed input sample, which makes the model automatically obtain the appropriate synthetic sample target. Finally, the input and label of the generated samples are $(h_{ij}, \hat{m}_{ij})$.

3.4 Training strategy

We obtain the predicted d-dimensional class vector $\tilde{m}_{ij}$ by feeding the mixing input into fully connected layer:

$$\tilde{m}_{ij} = fc(h_{ij}) \tag{4}$$

We use cosine similarity to measure the correlation between the predicted vector $\tilde{m}_{ij}$ and true vector $\hat{m}_{ij}$. Since the cosine value is closer to 1, the similarity of the two vectors is higher. We use the mean squared error (MSE) to compute the loss:

$$L_{ij} = MSE(Cos(\tilde{m}_{ij}, \hat{m}_{ij}), 1) \tag{5}$$

We add up the losses of all synthetic samples to get L_{dnm}. Finally, the loss is as follows:

$$L_{dnm} = \frac{1}{N \times 2K} \sum_{i=1}^{N} \sum_{j=1}^{2K} L_{ij} \tag{6}$$

In testing phase, the cosine value is calculated between the predicted class vector $\tilde{m}_{ij}$ and the set of label embedding matrix M, and the class corresponding to the maximum value is regarded as the predicted label. Algorithm 1 describes the training procedure of our method.

Algorithm 1 Training procedure of our method

Input: Labeled data D, label embedding matrix M, weight vector v_1, v_2, weight matrix W, F is feature encoder, N is the batch size, $iters$ is the total number of iterations.

```
1:  Let $t = 0$.
2:  while t <$iters$ do
3:      t = t+1
4:      $Total\_Loss = 0$
5:      $Loss = 0$
6:      $T_b$ = RandomSelect(D,N)
7:      Obtain the hidden layer representation $H_{T_b} = F(T_b)$
8:      Compute the distance $L$ according to $H_{T_b}$
9:      for $i = 1$ to $N$ do
10:         Select Top-K largest distance samples $\mathbf{X}^{L_i}$
11:         Select Top-K smallest distance samples $\mathbf{X}^{S_i}$
12:         Generate 2K mixed inputs using Eq. (2)
13:         Generate 2K mixed labels using Eq. (3)
14:         Compute the predicted value of each mixed input using Eq. (4)
15:         Compute the dynamic nonlinear mixup loss of each mixed sample using Eq. (5)
16:         $Loss = Loss + Loss^{L_i} + Loss^{S_i}$
17:     end for
18:     $Total\_Loss = Total\_Loss + Loss$
19:     Update the model parameters by minimizing $Total\_Loss$
20: end while
```

4 EXPERIMENT

4.1 Dataset Preparation

We use four datasets to evaluate our method. They are single-sentence classification tasks. **Subj** is a subjective dataset and the label is subjective or objective (Pang and Lee, 2004). **MR** is a movie review dataset with positive or negative (Pang and Lee, 2005). **SST-1** is the Stanford Sentiment Treebank with five categories of very positive, positive, neural, negative, and very negative (Socher et al., 2013). **SST-2** removes the neural label of SST-1 and is a binary classification task. Table 1 summarizes the all datasets.

Data	c	l	Train	Test	Val
Subj	2	23	8000	1000	1000
MR	2	20	8528	1068	1066
SST-2	2	19	6920	1821	872
SST-1	5	18	8544	2210	1101

Table 1: Statistics of the experimental datasets. c: number of labels. l: average sentence length.

4.2 Implementation Details

We utilize the WordPiece to split the sentences into tokens. In our experiment, we adopt the BERT$_{base}$(uncased) to obtain the sentence embedding. The maximum sequence length, batch size, step, and dropout is 128, 20, 4000, and 0.1, respectively. The learning rate is 2e-5. The dimension of the label embedding is 768. For each dataset, the average and standard error of the accuracy are calculated over 5 runs with different random seeds. Besides, we explore the effectiveness of our method under low-resource scenarios, and we select 50, 200, 500, 1000, and 2000 training data. For the hyper-parameter K, we select the optimal parameters on the three datasets (Table6 and Table7). Finally, We set K=5 in all our experiments. Additionally, all parameters are optimized by the adaptive momentum algorithm.

4.3 Baselines

We consider the following approaches for comparisons:

WordMixup: it applies linear interpolation at word embedding level (Guo et al., 2019).

SenMixup: it applies linear interpolation at sentence embedding level, namely the layer before the softmax layer (Guo et al., 2019).

NonlinearMixup: it uses the matrix mixing method to obtain the synthetic data (Guo, 2020).

BERT: it fine-tunes vanilla BERT on training data.

BERT+WordMixup: it fuses the the BERT-based word embeddings by the mixup.

BERT+SenMixup: it uses mixup to fuse the BERT-based sentence embeddings.

BERT+NonlinearMixup: it utilizes the BERT-based word embeddings and NonlinearMixup to train model.

We compare our method with other state-of-the-art methods on the four datasets and the experimental results are shown in Table 2, Table 3, Table 4, and Table 5. As can be seen, our method has achieved the best performance on most tasks. Besides, the results reveal several interesting observations. (1) The first is that the training data is sufficient. WordMixup, SenMixup, and NonlinearMixup use CNN or LSTM to obtain the embedding representation and achieve good performance. The classification accuracy of vanilla BERT all surpasses WordMixup, SenMixup, and NonlinearMixup. It shows that BERT model can produce better word embedding vectors. Comparing with BERT, BERT+WordMixup outperforms 0.7% on Subj dataset, 0.4% on MR dataset, 0.3% on SST-2 dataset and 0.3% on SST-1 dataset, respectively. BERT+SenMixup also outperforms BERT. Mixup is beneficial to improve the performance of large pre-trained language models. Comparing with BERT+NonlinearMixup, our method exceeds 0.7% on Subj dataset, 0.4% on MR dataset, 0.9% on SST-2 dataset and 1.0% on SST-1 dataset, respectively. The distance-based sample selection method can generate multiple sample pairs, and the dynamic nonlinear mixup expands the space of generated samples. Therefore, the generalization ability of the model is improved. (2) The second is that the training data is scarce. The classification accuracy of WordMixup, SenMixup, and NonlinearMixup dropped sharply. BERT can achieve better performance due to the pre-training on the large-scale unlabeled corpus. The classification accuracy of BERT+WordMixup and BERT+SenMixup also outperforms BERT. Compared with BERT+NonlinearMixup, our method exceeds 1.0% on Subj dataset, 2.3% on MR dataset, 1.8% on SST-2 dataset, and 2.4% on SST-1 dataset for 50 training data, respectively. The performance improvement with less data is higher than using full training data. It shows that our method is more effective in low-resource settings.

Training Data	**50**	**200**	**500**	**1000**	**2000**	**all**
WordMixup	72.4±0.6	80.6±0.7	84.1±0.6	87.4±0.5	91.6±0.4	94.3±0.6
SenMixup	73.1±0.7	81.4±0.5	84.8±0.4	88.1±0.3	92.4±0.2	95.0±0.3
NonlinearMixup	74.5±0.2	82.2±0.3	86.1±0.2	89.7±0.3	93.2±0.2	94.1±0.1
BERT	89.5±0.4	93.1±0.6	93.5±0.4	94.3±0.5	94.9±0.2	96.0±0.3
BERT+WordMixup	91.8±0.3	93.9±0.2	94.7±0.4	94.9±0.2	96.0±0.5	96.7±0.3
BERT+SenMixup	92.1±0.4	93.8±0.3	94.2±0.3	94.8±0.4	95.6±0.4	96.6±0.5
BERT+NonlinearMixup	92.4±0.5	94.3±0.4	94.9±0.4	94.6±0.2	95.9±0.3	97.3±0.4
Ours	**93.4±0.2**	**95.2±0.5**	**95.6±0.4**	**95.8±0.3**	**96.9±0.4**	**98.0±0.3**

Table 2: Classification accuracy (%) on the Subj dataset.

Training Data	**50**	**200**	**500**	**1000**	**2000**	**all**
WordMixup	65.7±0.6	68.7±0.6	70.5±0.4	72.2±0.7	76.5±0.3	79.7±0.5
SenMixup	66.3±0.4	67.9±0.5	69.4±0.3	73.1±0.5	77.1±0.5	80.3±0.6
NonlinearMixup	67.4±0.2	69.5±0.3	71.1±0.1	74.6±0.3	78.2±0.5	83.4±0.4
BERT	76.6±0.5	81.0±0.5	82.2±0.6	82.9±0.5	84.7±0.4	87.8±0.3
BERT+WordMixup	77.2±0.5	81.3±0.4	82.3±0.4	83.9±0.3	85.0±0.5	88.2±0.4
BERT+SenMixup	77.0±0.4	81.4±0.3	82.6±0.3	83.7±0.5	85.4±0.5	88.1±0.3
BERT+NonlinearMixup	77.3±0.5	81.7±0.6	83.0±0.4	84.3±0.4	85.6±0.3	88.8±0.5
Ours	**79.6±0.3**	**82.6±0.4**	**83.9±0.4**	**85.2±0.4**	**86.4±0.3**	**89.2±0.4**

Table 3: Classification accuracy (%) on the MR dataset.

Training Data	**50**	**200**	**500**	**1000**	**2000**	**all**
WordMixup	66.3±0.6	73.2±0.7	79.4±0.4	78.7±0.5	83.2±0.6	87.1±0.3
SenMixup	67.2±0.7	74.0±0.5	78.7±0.3	77.9±0.6	83.4±0.3	87.2±0.4
NonlinearMixup	68.1±0.3	75.2±0.4	80.2±0.5	79.3±0.5	84.3±0.5	88.6±0.3
BERT	82.6±0.5	85.8±0.3	86.6±0.2	87.9±0.3	89.7±0.4	90.7±0.3
BERT+WordMixup	83.1±0.4	86.4±0.5	87.6±0.3	88.6±0.4	89.6±0.4	91.0±0.3
BERT+SenMixup	83.0±0.5	86.1±0.4	87.8±0.3	88.4±0.2	89.9±0.3	91.1±0.2
BERT+NonlinearMixup	83.3±0.5	86.7±0.6	**88.4±0.4**	88.9±0.5	90.9±0.3	91.6±0.6
Ours	**85.1±0.3**	**87.7±0.4**	88.1±0.3	**89.5±0.2**	**91.4±0.3**	**92.5±0.2**

Table 4: Classification accuracy (%) on the SST-2 dataset.

Training Data	**50**	**200**	**500**	**1000**	**2000**	**all**
WordMixup	26.3±0.6	32.4±0.7	35.4±0.5	38.3±0.6	42.5±0.8	48.2±1.0
SenMixup	27.1±0.7	33.0±0.4	36.2±0.5	37.8±0.5	41.7±0.3	48.6±0.2
NonlinearMixup	27.8±0.3	33.8±0.3	36.7±0.5	38.9±0.4	43.5±0.2	49.3±0.4
BERT	33.3±0.5	39.7±0.4	45.6±0.3	46.1±0.4	48.8±0.2	52.6±0.3
BERT+WordMixup	33.7±0.4	39.3±0.5	45.1±0.3	48.0±0.6	49.4±0.4	52.9±0.2
BERT+SenMixup	33.8±0.4	38.8±0.3	44.8±0.4	47.4±0.2	49.3±0.3	53.0±0.2
BERT+NonlinearMixup	34.9±0.3	40.6±0.4	46.6±0.5	47.5±0.4	50.3±0.4	53.4±0.3
Ours	**37.3±0.3**	**41.8±0.2**	**47.7±0.5**	**48.6±0.2**	**51.4±0.3**	**54.4±0.2**

Table 5: Classification accuracy (%) on the SST-1 dataset.

4.4 Regularization Effect

We plot the training set loss and testing set accuracy using all training data on Subj and SST-1 datasets in Figure 2. Compared with BERT+NonlinearMixup, our method can reduce the training loss faster and have higher classification accuracy. Besides, the classification accuracy has not dropped even through a long training time. It shows that our

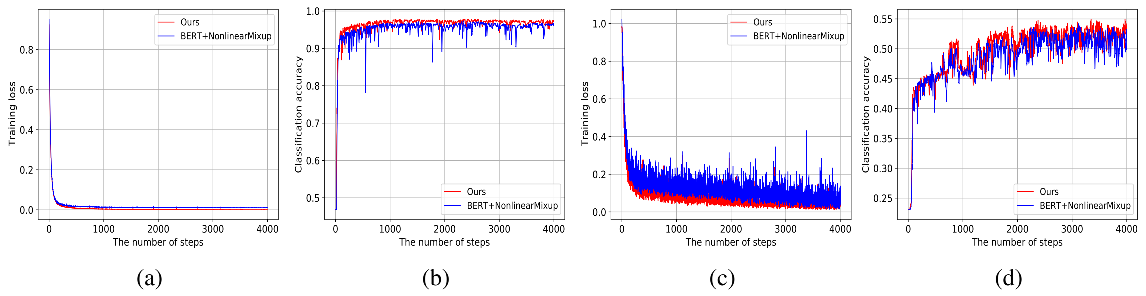


Figure 2: Training set loss and testing set accuracy on Subj dataset (a) (b) and SST-1 dataset (c) (d) when using all training data.

method can effectively prevent overfitting because of more training samples and larger generated sample space.

4.5 Selection of Top-K

We investigate the effects of the K value on Subj, MR, and SST-1 datasets using 50 and all training data in Table 6 and Table 7. We vary the K value with 1, 3, 5, 7, 9, respectively. We find that the classification accuracy increases first and then decreases as K value increases. The best performance is achieved when K = 5 on most tasks. Compared with all training data, the improvement of model performance is more obvious in 50 training data. Moreover, as K value increases, the generated samples cannot improve the performance of the model due to overfitting. It is necessary to find an appropriate parameter K.

Top-K	Subj	MR	SST-1
1	92.5±0.4	77.2±0.3	35.2±0.3
3	91.7±0.3	78.3±0.2	36.1±0.4
5	**93.4±0.2**	**79.6±0.3**	**37.3±0.3**
7	93.1±0.3	78.8±0.3	36.9±0.5
9	92.9±0.2	79.2±0.4	36.6±0.4

Table 6: Classification accuracy under different K values on Subj, MR, and SST-1 datasets when using 50 training data.

Top-K	Subj	MR	SST-1
1	97.2±0.4	88.7±0.4	53.3±0.3
3	97.6±0.3	88.5±0.5	53.8±0.4
5	**98.0±0.3**	89.2±0.4	**54.4±0.2**
7	97.9±0.5	**89.4±0.2**	54.0±0.3
9	97.4±0.4	89.1±0.5	53.5±0.4

Table 7: Classification accuracy under different K values on Subj, MR, and SST-1 datasets when using all training data.

4.6 Effects of Input Weights and Output Weights

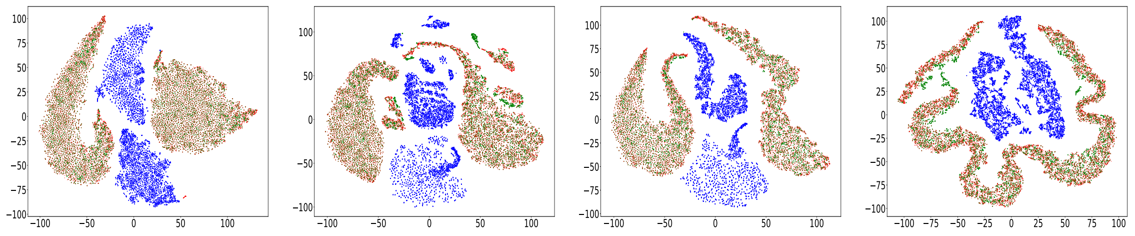
We construct four different ways to observe the effects of weights on Subj, MR, and SST-1 datasets using 50 and all training data in Table 8 and Table 9: tuned input weights (TIW), tuned output weights (TOW), fixed input weights (FIW), and fixed output weights (FOW). The input weights are shown in Eq. (2), and the output weights are shown in Eq. (3). The experimental results show that when the input weights or output weight is fixed, the prediction accuracy of our method is reduced. Compared with FIW+FOW, TIW+TOW exceeds 2.3% on Subj dataset, 2.8% on MR dataset, and 3.1% on SST-1 dataset for 50 training data, respectively. Compared with FIW+FOW, TIW+TOW exceeds 1.5% on Subj dataset, 1.4% on MR dataset, and 2.1% on SST-1 dataset for all training data, respectively. It shows that our method is more suitable for low resource situations.

Method	Subj	MR	SST-1
TIW+TOW	**93.4±0.2**	**79.6±0.3**	**37.3±0.3**
TIW+FOW	91.7±0.3	77.9±0.2	35.1±0.2
FIW+TOW	92.1±0.2	78.2±0.4	34.9±0.3
FIW+FOW	91.1±0.1	76.8±0.3	34.2±0.2

Table 8: Effects of input weights and output weights on Subj, MR, and SST-1 datasets when using 50 training data.

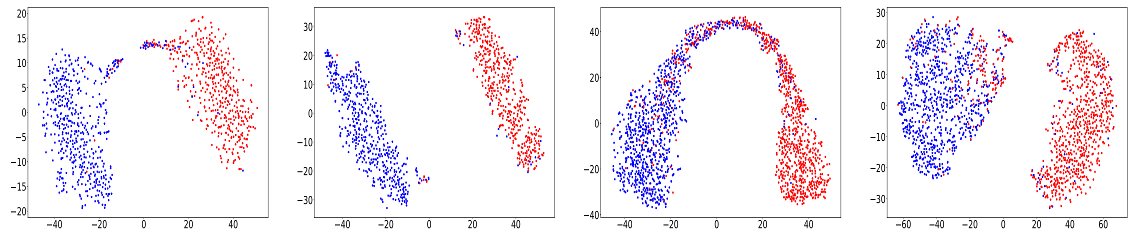
Method	Subj	MR	SST-1
TIW+TOW	**98.0±0.3**	**89.2±0.4**	**54.4±0.2**
TIW+FOW	96.7±0.3	88.4±0.3	53.1±0.3
FIW+TOW	96.5±0.4	88.1±0.2	53.0±0.4
FIW+FOW	96.2±0.2	87.8±0.3	52.3±0.3

Table 9: Effects of input weights and output weights on Subj, MR, and SST-1 datasets when using all training data.



(a) Subj (b) MR (c) SST-2 (d) SST-1

Figure 3: The t-SNE visualization of original samples and generated samples. The red, green and blue points denote original samples, the generated samples with the Top-k largest distance and the generated samples with the Top-k smallest distance.



(a) BERT (b) Ours (c) BERT (d) Ours

Figure 4: The t-SNE visualization of the Subj task (a) (b) and SST-2 task (c) (d). The red and blue points denote different labels.

4.7 Ablation Studies

To analyze the effect of distance-based sample selection (DSS), largest distance samples (LDS), and smallest distance samples (SDS), and dynamic nonlinear mixup (DNM), we conduct the ablation experiments using 50 and all training data in Table 10 and Table 11. The w/o DSS randomly selects a sample for a given sample. Our method can improve the classification accuracy of the model by generating more sample pairs. Furthermore, we study the contributions of the largest distance samples and smallest distance samples. We can find the samples with the largest distance has a greater impact than samples with the smallest distance. The w/o DNM follows the original mixing strategy to fuse the sentence embeddings. When we interpolate sample pairs by mixup, the classification accuracy of w/o DNM degrades on Subj, MR, and SST-1 datasets. It shows that dynamic nonlinear mixing strategy can expand the space of generated samples.

Method	**Subj**	**MR**	**SST-1**
Ours	**93.4±0.2**	**79.6±0.3**	**37.3±0.3**
w/o DSS	91.3±0.5	78.3±0.3	36.1±0.3
w/o LDS	90.5±0.3	76.4±0.4	32.4±0.3
w/o SDS	91.6±0.4	77.9±0.3	35.2±0.4
w/o DNM	92.4±0.4	78.5±0.2	35.9±0.5

Table 10: Results of ablation on Subj, MR, and SST-1 datasets when using 50 training data.

Method	**Subj**	**MR**	**SST-1**
Ours	**98.0±0.3**	**89.2±0.4**	**54.4±0.2**
w/o DSS	97.4±0.4	88.7±0.2	53.5±0.4
w/o LDS	93.9±0.2	86.5±0.4	50.5±0.5
w/o SDS	97.1±0.3	88.2±0.3	53.1±0.3
w/o DNM	97.3±0.3	88.3±0.2	53.6±0.4

Table 11: Results of ablation on Subj, MR, and SST-1 datasets when using all training data.

4.8 Feature Visualization

To prove the idea in Section 3.2, we visualize the original samples and generated samples by the t-SNE (Van der Maaten and Hinton, 2008) in Figure 3. The distribution of the generated samples with the Top-k largest distance is similar to the original samples, and the generated samples with the Top-k smallest distance are mainly distributed in the middle area of the original samples. Besides, we also select test samples and visualize the feature of the last layer in Figure 4. We perform the visualization on Subj and SST-2 datasets using all training data. The vanilla BERT mixes samples with different labels (Figure 4a and Figure 4c). Our method can better separate samples with different labels (Figure 4b and Figure 4d). The boundary of text classification is clear. It proves that our method can improve the performance of the model.

5 Conclusion

In this paper, we propose the dynamic nonlinear mixup with distance-based sample selection method to enhance the performance of pre-trained language models. First, we introduce distance-based sample selection to choose the Top-K largest distance samples and Top-K smallest distance samples for a given sample. More generated samples improve the generalization ability of the model. Second, we utilize the dynamic nonlinear mixing policy on the input sample pairs to enlarge the space of the synthetic samples. The mixed labels are constructed through the learned label embeddings and mixed input so that the mixed labels are updated adaptively. Experiments on the Subj, MR, SST-2, and SST-1 datasets demonstrate that our method outperforms the state-of-the-art methods. For future work, we plan to further enhance performance by exploring different weighting schemes for origin samples and augmented samples.

References

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4044–4051.

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

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*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Jungsoo Park, Gyuwan Kim, and Jaewoo Kang. 2021. Consistency training with virtual adversarial discrete perturbation. *arXiv preprint arXiv:2104.07284*.

Seo Yeon Park and Cornelia Caragea. 2022. On the calibration of pre-trained language models using mixup guided by area under the margin and saliency. *arXiv preprint arXiv:2203.07559*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Wenpeng Yin, Huan Wang, Jin Qu, and Caiming Xiong. 2021. Batchmixup: Improving training by interpolating hidden states of the entire mini-batch. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4908–4912.

Soyoung Yoon, Gyuwan Kim, and Kyumin Park. 2021. Ssmix: Saliency-based span mixup for text classification. *arXiv preprint arXiv:2106.08062*.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2017a. Multi-task label embedding for text classification. *arXiv preprint arXiv:1710.07210*.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017b. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.