The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Nonlinear Mixup: Out-Of-Manifold Data Augmentation for Text Classification

Hongyu Guo
National Research Council Canada
1200 Montreal Road, Ottawa, ON., K1A 0R6
hongyu.guo@nrc-cnrc.gc.ca

Abstract

Data augmentation with Mixup (Zhang et al. 2018) has shown to be an effective model regularizer for current art deep classification networks. It generates out-of-manifold samples through linearly interpolating inputs and their corresponding labels of random sample pairs. Despite its great successes, Mixup requires convex combination of the inputs as well as the modeling targets of a sample pair, thus significantly limits the space of its synthetic samples and consequently its regularization effect. To cope with this limitation, we propose "nonlinear Mixup". Unlike Mixup where the input and label pairs share the same, linear, scalar mixing policy, our approach embraces nonlinear interpolation policy for both the input and label pairs, where the mixing policy for the labels is adaptively learned based on the mixed input. Experiments on benchmark sentence classification datasets indicate that our approach significantly improves upon Mixup. Our empirical studies also show that the out-of-manifold samples generated by our strategy encourage training samples in each class to form a tight representation cluster that is far from others.

Introduction

Despite their profound successes in many challenging real-world applications such as image classification (Krizhevsky, Sutskever, and Hinton 2012), speech recognition (Graves, Mohamed, and Hinton 2013), and machine translation (Sutskever, Vinyals, and Le 2014), deep learning models typically embrace high modeling freedom with a very large number of parameters. Such extremely high modeling capability models thus require effective regularization techniques to power the models to avoid overfitting and to generalize well. To this end, many regularizers for deep models have been introduced, including weight decay (Hanson and Pratt 1988), dropout (Srivastava et al. 2014), stochastic depth (Huang et al. 2016), batch normalization (Ioffe and Szegedy 2015), and data augmentation schemes (Simard et al. 1998; Lecun et al. 1998), amongst many others.

A recently proposed such technique, Mixup (Zhang et al. 2018), is a simple and yet very effective, data-augmentation based regularizer for enhancing the performance of deep classification models. Unlike conventional regularizers, Mixup constrains the model on the input space beyond the data manifold. Through linearly interpolating random data sample pairs and their training targets in one-hot representation, Mixup generates a synthetic set of examples with soft-labels as the training targets and use these out-of-manifold examples to regularize the deep models. Research has shown that Mixup and its variants (Zhang et al. 2018; Guo, Mao, and Zhang 2019b) can dramatically improve the predictive accuracy of the current art of deep neural networks. Despite its demonstrated effectiveness, the power of Mixup is still limited by its linear nature. That is, Mixup requires a convex combination of the inputs as well as the modeling targets of a sample pair, and thus significantly limits the space of its synthetic samples, and consequently its regularization effect on the training of the deep models.

In this paper, we propose nonlinear Mixup to address the aforementioned limitation in Mixup. Unlike existing Mixup methods (Zhang et al. 2018; Guo, Mao, and Zhang 2019b; Verma et al. 2018) where the input pairs and label pairs share the same linear, scalar mixing policy, our approach enables nonlinear interpolation policy in the form of a matrix to interpolate both the input pairs and label pairs, where the mixing policy for the former differs from the latter. The nonlinear mixing here significantly expands the synthetic sample space for Mixup. Consider a pair of random samples in a 2D training set. Mixup always creates synthetic samples along the straight line between the pair. On the other hand, the nonlinear Mixup gives each dimension of the input pair an independent mixing policy, thus allowing mixed samples spread into a wide region between the two sample points (illustrated in Figure 1). In addition, our nonlinear Mixup enables the mixing policy for the labels to be adaptively learned based on the resulting mixed input. This allows the mixed samples to be relocated during training in order to alleviate the manifold intrusion issue. This problem occurs when a mixed example collides with a real example in the data manifold, but is given a soft label that is different from the label of the real example, resulting in under-fitting and degradation of the model performance (Guo, Mao, and Zhang 2019b).

Experiments on five benchmark sentence classification datasets indicate that our nonlinear Mixup strategy significantly outperforms Mixup and its variants in terms of pre-

dictive accuracy. Our empirical studies also show that the out-of-manifold samples created keep tuning the networks long after the training error on the original training set is minimal, encouraging the learning to generate, for each class of the training samples, very tight representation cluster that is far from other clusters.

Nonlinear Mixup for Text Classification

In this section, we first briefly describe the original Mixup and then introduce the proposed nonlinear Mixup in details.

Mixup and wordMixup

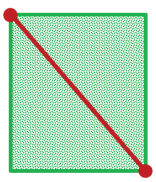
Mixup was first introduced by Zhang et al. (Zhang et al. 2018) for image classification. In a nutshell, Mixup enhances the training of deep classification models by generating synthetic samples through linearly interpolating a pair of training samples as well as their modeling targets. Given a pair of samples $(x^i; y^i)$ and $(x^j; y^j)$ from the original training set, where x denotes the input and y the one-hot encoding of the corresponding class of the sample. Mixup generates a synthetic sample as follows.

$$\widetilde{x}^{ij} = \lambda x^i + (1 - \lambda)x^j \tag{1}$$

$$\widetilde{y}^{ij} = \lambda y^i + (1 - \lambda)y^j \tag{2}$$

where λ is the mixing policy for mixing both the inputs and modeling targets of the sample pair, and it is a *scalar*, sampled from a Beta(α, α) distribution with a hyper-parameter α. The generated synthetic data are then fed into the model for training to minimize the loss function. It is worth noting that unlike the one-hot *hard label* from the original data set, Mixup creates samples with *soft-label* to indicate the possibilities of belonging to multiple classes. Those *soft-labelled* synthetic samples are clearly outside of the manifold of the original *hard-labelled* training data. In other words, Mixup regularizes the deep models by imposing constraints on the model by making use of the regions in the input space of the model that are outside of the data manifold. These synthetic samples are termed out-of-manifold samples in (Guo, Mao, and Zhang 2019b), which differ from conventional synthetic samples associated with one-hot representation labels.

Figure 1: Illustration of sample space for nonlinear Mixup and Mixup for a pair of samples in a 2-dimensional space. For Mixup, the synthetic samples are created along the red line, and the nonlinear Mixup can create samples anywhere within the green rectangle.



Inspired by its great successes in the image domain, the linear Mixup was recently adapted for sentence classification in (Guo, Mao, and Zhang 2019a). The method is termed wordMixup. Unlike image which is consist of pixels, sentence is composed of a sequence of words and typically a sentence representation is constructed to aggregate information from those words. As an example, in the widely used CNN (Kim 2014) model, a sentence is first represented by a sequence of word embeddings, and then convolutional operations are operated on those embeddings to generate the sentence representation. Next, the resulting sentence embeddings are then passed through a soft-max layer to generate the predictive distribution over the provided c different target categories or classes.

In wordMixup, all sentences are first zero padded to the same length and then interpolation is conducted on the word embeddings level. In specific, given a piece of text, such as a sentence with N words. It can be represented as a matrix $B \in R^{N \times d}$. Each row of the matrix corresponds to one word, which is represented by a d-dimensional vector as provided either by a learned word embedding table or being randomly generated. In the wordMixup setting, a pair of samples $(B^i; y^i)$ and $(B^j; y^j)$ are given, where B^i and B^j denote the embedding vectors of the input sentence pairs x^i and x^j, and y^i and y^j denote the corresponding class labels of the samples using one-hot representation. With such a pair of sentences, wordMixup performs the linear interpolation process as follows:

$$\widetilde{B}^{ij} = \lambda B^i + (1 - \lambda)B^j \tag{3}$$

$$\widetilde{y}^{ij} = \lambda y^i + (1 - \lambda)y^j \tag{4}$$

Where the λ is the scalar mixing policy for mixing both the inputs and modeling targets of the sample pair. The same as that in the Mixup, the λ here is a *scalar*, sampled from a Beta(α, α) distribution. The resulting new sample $(\widetilde{B}^{ij}; \widetilde{y}^{ij})$ is then used for training, such as passing through a CNN model f_φ (parameterized with φ) to generate the m-dimensional sentence embedding $S^{ij} \in R^m$:

$$S^{ij} = f_\varphi(\widetilde{B}^{ij}) \tag{5}$$

Finally, a linear fullyconnected layer $W \in R^{c \times m}$ produces the distribution over the c classification classes:

$$\ddot{y}^{ij} = softmax(WS^{ij}) \tag{6}$$

For training, cross entropy loss E is used by wordMixup:

$$E = \widetilde{y}^{ij} log \ddot{y}^{ij} \tag{7}$$

Nonlinear Mixup

Similar to wordMixup, the nonlinear Mixup also mixes sample pairs on the word embedding level. But unlike wordMixup, where all the words share the same *scalar* mixing policy λ, nonlinear Mixup deploys a separate mixing policy for each of the dimensions of each of the words in a given sentence. That is, for a sentence represented by $B \in R^{N \times d}$, the mixing policy for the nonlinear Mixup is a *matrix* $\Lambda \in R^{N \times d}$, where each element of Λ is independently sampled from a Beta(α, α) distribution. The mixing of inputs for the nonlinear Mixup is computed as follows.

$$\widetilde{B}^{ij} = \Lambda \circ B^i + (1 - \Lambda) \circ B^j \tag{8}$$

where $\circ$ denotes the Hadamard product.

With the mixing policy as a *matrix* instead of a *scalar*, the synthetic sample space for the nonlinear Mixup and Mixup for a pair of samples in a 2-dimensional space can be illustrated in Figure 1. For the linear Mixup, the synthetic samples are created along the *red* diagonal line, and the nonlinear Mixup can create samples anywhere within the *green* rectangle. As a result, the input space of the synthetic samples created by the nonlinear Mixup is much larger than that of the linear Mixup, thus enforcing further constraints and potentially providing better regularization for the training.

Label Embedding One more problem needs to be resolved for the nonlinear Mixup. That is, the mixing policy now is a matrix instead of a scalar, which cannot be directly applied to the one-hot label pairs due to the dimension difference between the input and the target. To cope with this challenge, we adopt the idea of label embedding (Bengio, Weston, and Grangier 2010). In specific, we use a vector $z \in R^k$ to encode the one-hot representation of the modeling target y. That is, a matrix $M \in R^{c \times k}$ is used to represent the modeling targets where each of the c categories is encoded as a k-dimensional vector z. To prevent the set of c k-dimensional label vectors from being too close to each other, a Gram-Schmidt process (Pussell and Trimble 1991) is performed to obtain an orthogonal set of label vectors in M.

Policy Mapping Function With the above label embedding strategy, the mixing policy for the modeling targets of a sample pair in the nonlinear Mixup is created by a Policy Mapping Function $\digamma$ as follows.

$$\Phi = \digamma(\widetilde{B}^{ij}) \tag{9}$$

where the input for $\digamma$ is the mixed input $\widetilde{B}^{ij}$ created using Equation 8, and Φ has the same dimension as the label embedding z. In this way, the mixing policy for the modeling targets is based on the mixed input, which allows the model to adaptively assign, based on the mixed inputs, the proper modeling targets for the synthetic samples, namely relocating them in the input space. Such relocation capability greatly benefits the nonlinear Mixup model as will be discussed in the experimental section. In practice, the $\digamma$ can be simply implemented as a Sigmoid function σ as

$$\digamma_\theta(\widetilde{B}^{ij}) = \sigma(\theta \,\mathrm{vec}(\widetilde{B}^{ij})) \tag{10}$$

here vec denotes the matrix vectorization operation and $\theta \in R^{k \times t}$ ($t = N \times d$) is the parameters that will be learned as part of the model. In other words, by tuning the θ, the nonlinear Mixup is able to assign the proper modeling target labels to its synthetic samples.

With the mixing policy for the label pairs in Equation 9, the mixed label for the newly created sample is computed as

$$\widetilde{z}^{ij} = \Phi z^i + (1 - \Phi) z^j \tag{11}$$

Equations 8 and 11 give us the resulting new sample ($\widetilde{B}^{ij}$; $\widetilde{z}^{ij}$), which is then passed through the sentence embedding encoder f_φ to generate the m-dimensional sentence embedding $S^{ij} \in R^m$:

$$S^{ij} = f_\varphi(\widetilde{B}^{ij}) \tag{12}$$

Parameter Optimization At the last step of the nonlinear Mixup, a linear fullyconnected layer $W \in R^{k \times m}$ produces the predicted k-dimensional class vector over the c classification classes:

$$\ddot{z}^{ij} = W S^{ij} \tag{13}$$

For training, we minimize the negative-cosine loss between the prediction vector $\ddot{z}^{ij}$ and its true label vector $\widetilde{z}^{ij}$, through optimizing the set of parameters of the model $(\theta, \varphi, W, M, B)$ with gradient descent:

$$E = -\ddot{z}^{ijT} \widetilde{z}^{ij} \tag{14}$$

It is worth noting that, although there are many degrees of freedom (i.e., θ, φ, W, M, and B) for optimizing the model to drive down the cosine loss between $\ddot{z}^{ij}$ and $\widetilde{z}^{ij}$. However, due to the need to reduce the losses for both the real samples and synthetic instances, the cosine loss in Equation 14 will not be trivially reduced to zero. Also note that, in the current implementation of nonlinear Mixup we give an equal weight to losses for both the real samples and synthetic instances. In the future, it would be beneficial to investigate different weighting schemes.

In testing time, the cosine distance of the predicted vector $\ddot{z}^{ij}$ and the set of c gold label vectors in M are computed, and the closest one is chosen as the predicted label.

In summary, the nonlinear Mixup attains the regularization effect by forcing the model to fit an additional set of synthetic data, with the aim of further constraining the hypothesis search space. The synthetic samples are created with two conditions: between pairs of real samples and with learnable synthetic labels. The former ensures the synthetic samples are not too far from the given training data; the latter makes sure those synthetic samples not to collide with the original data set. Along with the nonlinear mixing policy, which significantly expands the input space of the synthetic examples over the linear version, such constrains power the nonlinear Mixup to achieve models with less prediction biases.

Experiments

Datasets

We evaluate the nonlinear Mixup method with five benchmark sentence classification tasks as used for evaluating the wordMixup and senMixup for text augmentation in (Guo, Mao, and Zhang 2019a). They are as follows. **TREC** is a question dataset with the aim of categorizing a question into six question types (Li and Roth 2002). **MR** is a movie review dataset aiming to detect positive/negative reviews (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** dataset is the same as SST-1 but with neutral reviews removed and binary labels. **Subj** is a data set with the aim of classifying a sentence as being subjective or objective (Pang and Lee 2004). Table 1 summarizes the statistical characteristics of the five data sets after being tokenized.

Baselines and Settings

We evaluate our nonlinear Mixup model using the popular CNN (Kim 2014) for sentence classification model. We

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

Table 1: Summary for the datasets after tokenization. c: number of target labels. l: average sentence length. N: number of samples. V: vocabulary size. Test: test set size (CV means no standard train/test split was provided and thus 10-fold CV was used).

compare with four baselines: the original CNN (Kim 2014) (denoted as CNNsen), and three recent text augmentation methods including EDA (Wei and Zou 2019), wordMixup and senMixup (Guo, Mao, and Zhang 2019a). These augmentation strategies are less relied on additional text resources or domain knowledge.

- **CNNsen** is a convolutional neural networks model, and has been widely used for sentence classification baseline.
- **EDA** is a recent data augmentation method containing a set of 4 text augmentation techniques, including synonym replacement, random insertion, random swap, and random deletion.
- **wordMixup** is the straightforward application of Mixup on NLP tasks where linear interpolation applying on the word embedding level.
- **senMixup** is the Mixup applying to NLP tasks where linear interpolation is conducted in the hidden representation space, namely the layer before the Softmax layer.

We obtained the source codes for the comparison models from the authors in (Kim 2014) and in (Wei and Zou 2019). In our experiments, we follow the exact implementation and settings in (Kim 2014), (Guo, Mao, and Zhang 2019a), and (Wei and Zou 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 0.2 for the baseline CNN. For datasets without a standard development set we randomly select 10% of training data as development set. Training is done through Adam (Kingma and Ba 2014) over mini-batches of size 50. The pre-trained word embeddings are 300 dimensional GloVe (Pennington, Socher, and Manning 2014). For the nonlinear Mixup the mixing policy α is set to the default value of one. The dimension of the label embedding in the nonlinear Mixup is 100. For each dataset, we train each model 10 times each with 80k steps, and compute their mean test errors and standard deviations.

Predictive Accuracy

To evaluate the predictive performance of the nonlinear Mixup, we conduct four sets of experiments with various word embeddings settings: 1) learnable, randomly initialized word embeddings (denoted as RandomTune); 2) fixed, randomly initialized word embeddings (RandomFix); 3) learnable, pretrained word embeddings (PretrainTune), and fixed, pretrained word embeddings (PretrainFix).

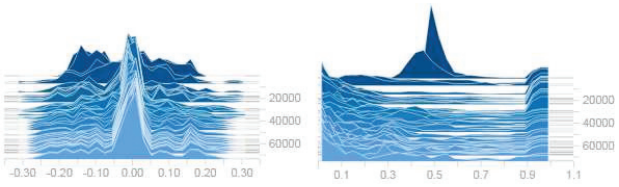
The results on the learnable, randomly initialized word embeddings are presented in Table 2. The results in Table 2 show that the nonlinear Mixup outperformed all the four comparison baselines, ie., CNNsen, EDA, wordMixup and senMixup, on all the five datasets. For example, when compared with the CNNsen baseline, nonlinear Mixup improved over CNNsen with a large margin on all the five datasets. For instance, against the SST-2 and MR datasets, the improvements were 3.9% and 4.9%, respectively. When compared with wordMixup, the nonlinear Mixup increased the accuracy by at least 2% for four of the five testing datasets. When compare with the senMixup, nonlinear Mixup also improved over it by respectively 3.2%, 2.2%, and 2.7% on the SST-1, SST-2, and MR datasets, while only slight improvement was obtained on the Trec and Subj datasets. When consider the comparison with the EDA method, the nonlinear Mixup improved the predictive accuracy by at least 2% for all the five datasets. Interestingly, one can see that the EDA obtained similar predictive accuracy as the wordMixup on almost all the five datasets.

When the randomly initialized word embeddings are fixed during training, the results in Table 3 show that, again, the nonlinear Mixup outperformed all the other four comparison baselines on all the five datasets. Also, it is interesting to see that EDA outperformed both wordMixup and senMixup on the SST-2 and MR datasets with at lease 2.5%. For these two data sets, the nonlinear Mixup outperformed the wordMixup and senMixup by at least 3.8%.

The results obtained by the five testing methods when applying the GloVe pretrained word embeddings are presented in Tables 4 and 5. Results in the tables indicate that the nonlinear Mixup improved over all the comparison baselines on four of the five testing datasets (with a slight degradation on the Subj dataset). For example, when compared to the CNNsen, the nonlinear Mixup increased the predictive accuracy of the baseline by at least 2.5%. When compared with wordMixup and senMixup, the improvements were less than that with randomly initialized word embeddings as presented in Table 2. The largest improvement come from the MR data set, where the nonlinear Mixup outperformed CNNsen, wordMixup, and senMixup by 3.6%, 2.1%, and 2.8%, respectively. Similar patterns can be seen in Table 5 when the pretrained embeddings are fixed during training.

In short, these results indicate that the nonlinear Mixup significantly outperformed the linear Mixup, the CNN baseline, and two recent introduced text augmentation methods, in terms of predictive accuracy, on the five benchmarking text classification tasks.

Figure 2: Evolving gold label embeddings (left) and mixing policies for the label pairs (right) during training.



RandomTune	Trec	SST-1	SST-2	Subj	MR
CNNsen†	90.2±0.20	43.6±0.19	82.3±0.47	90.6±0.45	75.5±0.36
EDA‡	89.2±0.65	45.1±0.54	83.1±0.49	90.8± 0.44	78.0±0.79
wordMixup†	90.9±0.42	45.2±0.90	82.8±0.45	92.9±0.41	78.0±0.39
senMixup†	92.1±0.31	45.2±0.22	83.0±0.35	92.7±0.38	77.9±0.76
nonlinear Mixup	**92.9±0.80**	**47.4±0.41**	**86.2±0.56**	**93.0±0.31**	**80.6±0.52**

Table 2: Accuracy (%) of the testing methods with randomly initialized, trainable word embeddings. We report mean scores over 10 runs with standard deviations (denoted ±). Best results highlighted in **Bold**. † indicates results extracted from the paper in (Guo, Mao, and Zhang 2019a); ‡ indicates results obtained by running the source codes provided by the authors of (Wei and Zou 2019).

RandomFix	Trec	SST-1	SST-2	Subj	MR
CNNsen†	88.4±0.52	40.3±0.77	80.4±0.17	88.2±0.50	72.9±0.74
EDA‡	90.6±0.45	40.9±0.52	80.5±0.60	89.2±0.41	76.7±0.89
wordMixup†	90.9±0.58	40.5±1.17	77.5±0.33	89.3±0.47	74.2±1.15
senMixup†	88.8±1.10	41.0±0.64	77.6±0.76	90.5±0.36	72.6±0.67
nonlinear Mixup	**91.2±0.55**	**42.6±0.85**	**81.5±0.79**	**90.7±0.49**	**78.0±0.80**

Table 3: Accuracy (%) obtained by the testing methods with randomly initialized and fixed word embeddings. Best results highlighted in **Bold**. † indicates results extracted from the paper in (Guo, Mao, and Zhang 2019a); ‡ indicates results obtained by running the source codes provided by the authors of (Wei and Zou 2019).

PretrainTune	Trec	SST-1	SST-2	Subj	MR
CNNsen†	92.1±0.12	46.3±0.35	86.9±0.49	94.4±0.36	79.8±0.60
EDA‡	92.4±0.51	46.9±0.27	85.9±0.22	92.9±0.31	81.2±0.09
wordMixup†	93.7±0.80	48.2±0.91	87.1±0.26	94.7±0.45	81.3±0.28
senMixup†	93.3±0.23	48.6±0.23	87.2±0.35	**94.9±0.34**	80.6±0.56
nonlinear Mixup	**94.7±0.29**	**49.3±0.42**	**88.6±0.29**	93.9±0.19	**83.4±0.36**

Table 4: Accuracy (%) of the testing methods with pre-trained GloVe and trainable word embeddings. Best results highlighted in **Bold**. † indicates results extracted from the paper in (Guo, Mao, and Zhang 2019a); ‡ indicates results obtained by running the source codes provided by the authors of (Wei and Zou 2019).

PretrainFix	Trec	SST-1	SST-2	Subj	MR
CNNsen†	92.0±0.20	44.6±0.56	85.7±0.33	94.5±0.36	79.7±0.68
EDA‡	93.9±0.36	46.4±0.64	86.6±0.26	93.6±0.40	82.8±0.37
wordMixup†	94.2±0.52	46.6±0.85	84.5±0.54	94.3±0.23	79.7±0.52
senMixup†	94.8±0.35	46.5±0.23	84.7±0.48	**95.0±0.22**	80.3±0.57
nonlinear Mixup	**95.1±0.28**	**46.8±0.52**	**86.8±0.34**	94.1±0.13	**83.3±0.56**

Table 5: Accuracy (%) obtained by the testing methods with pretrained GloVe and fixed word embeddings. Best results highlighted in **Bold**. † indicates results extracted from the paper in (Guo, Mao, and Zhang 2019a); ‡ indicates results obtained by running the source codes provided by the authors of (Wei and Zou 2019).

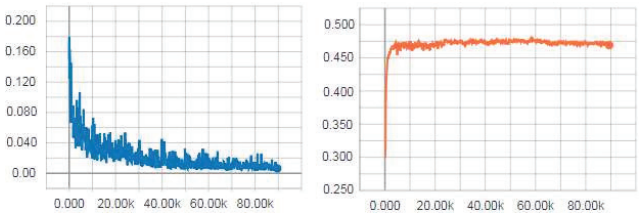
Ablation Studies

To better understand the working mechanism of the nonlinear Mixup method, we conduct ablation studies using the SST-1 data set with randomly initialized, tunable word embeddings. This data set was chosen since it is the most difficult data set (with error rate lower than 50%) in our experimental studies.

Label Assignment for Synthetic Samples In Figure 2, we visualize how the label embeddings for the gold labels (left subfigure) and the learned mixing policies for the modeling targets (right subfigure) evolve during the training on the SST-1 data set. Note that, in our experiments, the mixing policy for the inputs is a uniform distribution due to the value of one is used in the Beta distribution.

Results in the left subfigure of Figure 2 indicate that the embeddings for the gold labels, namely matrix M, keep refining during training, but with more fluctuation at the beginning of the training and gradually stabilize at the end of the training. Results in the right subfigure of Figure 2 show that the mixing policies for the label pairs of the nonlinear Mixup tend to sit in the middle of [0, 1] at the beginning of

Figure 3: Training loss (left) and validation accuracy (right) obtained by the nonlinear Mixup.



Label Dimension	Accuracy (%)	Standard Deviation
5	46.52	±0.47
10	46.92	±0.72
50	46.82	±0.40
100	47.42	±0.41
300	47.50	±0.46
500	46.14	±0.10

Table 6: Accuracy (%) obtained when varying the dimension of the gold label embedding in nonlinear Mixup.

nonlinear Mixup	Tuned Mappings	Fixed Mappings
Trec	92.9±0.80	91.2±0.63
SST-1	47.4±0.41	45.1±0.12
SST-2	86.2±0.56	84.9±0.60
Subj	93.0±0.31	90.9±0.49
MR	80.6±0.52	79.5±0.61

Table 7: Accuracy (%) of the testing methods using randomly initialized, trainable embeddings with fixed Policy Mapping Function in the nonlinear Mixup.

the training, and gradually move to put more weights to the two ends of the policy region, namely zero and one. At the end of the training, most of the mass sit at between [0, 0.5], and many having the value of zero or one, indicating that at the end of the training the learning tends to assign one of the sample pairs' label as the modeling target of the mixed sample.

Regularization Effect We also plot the training loss and testing accuracy across the 80K training steps on the SST-1 dataset in Figure 3. Figure 3 shows that the training loss curve of nonlinear Mixup (left subfigure) maintains a relatively high level, when compared to the CNNsen baseline where the training loss reduced to zero in less than 2k training steps. The relative high loss here allows the model to keep tuning. Such high loss is due to the much larger space of the synthetic samples, thus preventing the model from being over-fitted by limited number of examples. As a result, as shown on the right subfigure, even training for a long time, the nonlinear Mixup model is not overfitting.

Sensitivity to Label Embedding Dimension We also evaluate the sensitivity of the dimension for the label embedding in the nonlinear Mixup. We vary the dimension for the label embedding with 2, 10, 50, 100, 300, and 500, respectively. Results are presented in Table 6.

Results in Table 6 indicate that the nonlinear Mixup is insensitive to the dimensions used. As can be seen in the table, the accuracy obtained when varying the dimension of the label embeddings is similar.

With Fixed Policy Mapping Function We also conduct experiments with fixing the Policy Mapping Function in the nonlinear Mixup as depicted in Equation 9, aiming to understand the impact of the label assignment process in the nonlinear Mixup. Results are presented in Table 7.

Results in Table 7 indicate that when the function for label assignment is not allowed to modified after initialization, the predictive performance of the nonlinear Mixup degrades on all the five testing datasets. For example, against the SST-1, Trec, and Subj datasets, the accuracy dropped by 2.3%, 1.7%, and 2.1%, respectively.

Our hypothesis here is that allowing the Policy Mapping Function to be tuned enabled the networks to effectively relocate the mixed samples to avoid creating mixed samples that are collided with samples from the original dataset, which is an inherent issue in Mixup as discussed in (Guo, Mao, and Zhang 2019b). More discussions regarding this issue will be presented in the next section.

Effect of Out-Of-Manifold Samples

In this section, we aim to visualize the impact of the out-of-manifold samples in the nonlinear Mixup model. We present the results on the Subjectivity dataset with a 2D bottleneck hidden representation in the middle of the network and with label embedding dimension of 2D as well. Doing so, we can visualize the evolving of the learned embeddings for both the sentences and labels. The Subjectivity data set was chosen because it is a relative easy data set for classification and perform well with the 2D bottleneck and label embeddings.

We trained a nonlinear Mixup with tailored architecture. In detail, following the 512 filters generated by the CNNsen (namely the layer before the Softmax layer), we use a fully-connected Tanh layer of 2 units as a bottleneck layer, followed by two more fully-connected Tanh layers with 100 and 512 units respectively. For the label embedding dimension, we also use 2 instead of 100 for visualization purpose.

Embedding Clusters We visualize the 2D bottleneck representation for the original input samples as well as the label embeddings at 2k, 10k, and 20k steps. For these steps, the training errors are 80%, 100%, and 100% respectively. Results are presented in Figure 4, with positive inputs in red, negative inputs in blue, predicted labels for positive inputs in purple, predicted labels for negative inputs in green, and learned label embedding for the true positive label in yellow and true negative labels in brown.

Results in Figure 4 suggest that, as the learning progresses, the embeddings for both the inputs and true labels are tuning: after 2k steps (left subfigure), the input embeddings are gradually towards separating the two classes and the model achieves 100% accuracy at the $10k^{th}$ step (middle subfigure). At the same time, the networks also learned to generate predicted labels that are close to the gold label embeddings as shown in the middle of Figure 4.

Figure 4: 2D data embeddings, their corresponding predicted labels, and gold label embeddings when training on the Subjectivity data set with 2k, 10k, 20k steps, where the training accuracy for these steps were 80%, 100%, and 100%, respectively.

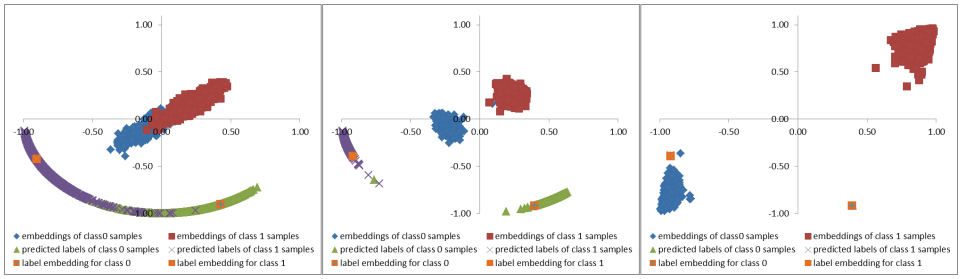
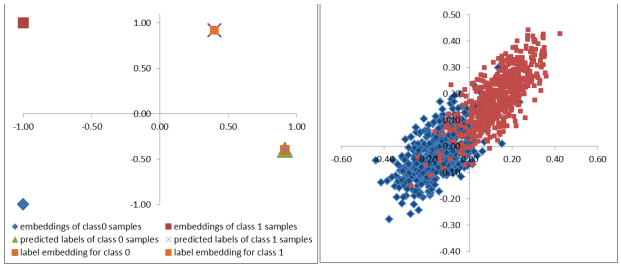


Figure 5: At the 80kth step of training, all training data samples from the same class are projected to the same point by the nonlinear Mixup (left subfigure). On the right, the projected embeddings of the validation data set are depicted.



On the right of Figure 4, where the training accuracy has reached 100%, the network is still tuning using the newly created mixed samples. As can been seen, both the input and label embeddings are continuously tuning after the 10k step. At the end of the 20k training steps (right subfigure), the input embeddings are separated into two tight clusters with distance far away than that of the 10kth step (middle subfigure). Also, as a result, the predicted labels are also highly overlapped with the embeddings of the gold labels.

These results suggest that the out-of-manifold samples help create tight representation cluster for training samples in each class and widen the gap between these clusters. These turnings happened long after the training accuracy on the original training set was not able to further increase.

Manifold Intrusion Avoidance Recall that, nonlinear Mixup expands the synthetic sample space from line to region for a sample pair (as illustrated in Figure 1), thus increasing the possibility of collision with the original samples. Nevertheless, we here argue that with the learnable Policy Mapping Function that is based on the input pairs, the networks is able to relocate the mixed sample to better avoid the manifold intrusion issue. In other words, through adaptively assigning soft-labels for the synthetic samples, the networks are in a position to re-locate the mixed samples to avoid collision with the original data samples.

In our experiments, keep training the nonlinear Mixup allows the networks to be tuned with new out-of-manifold samples that have not been seen before. Consequently, at the end of the 80kth training step, as depicted in the left subfigure of Figure 5, all original samples belonging to the same class are projected to the same point. This observation suggests that at the end of the training, the model may have difficulty to find a valid space to avoid the intrusion issue. By collapsing the real samples of each class to a single point, the model can effectively alleviate the manifold intrusion issue because any mixing of the original data point pairs will not collide with samples from the original data points.

We also depicted the embeddings of the test samples on the right of Figure 5. The right subfigure shows that the embeddings of the test data samples are also squeezed into two clusters, but with much larger overlapped area than that of the training samples in the left subfigure.

Related Work

Data augmentation is still a less touched area in natural language processing tasks due to high complexity of language. To this end, popular text augmentation approaches aim to transform the text with word replacements with either synonyms from handcrafted ontology such as a WordNet (Zhang, Zhao, and LeCun 2015) or word similarity (Wang and Yang 2015; Kobayashi 2018). Some other NLP data augmentation methods are often devised for specific domains or require extra text resources (Sennrich, Haddow, and Birch 2015; Fadaee, Bisazza, and Monz 2017; Sahin and Steedman 2018). Recently, Easy Data Augmentation (EDA) (Wei and Zou 2019) is proposed. This approach contains a set of 4 different text augmentation techniques: synonym replacement, random insertion, random swap, and random deletion. Unlike these methods, our nonlinear Mixup does not rely on domain knowledge or additional text resources and leverages out-of-manifold samples.

Our method closely relates to wordMixup (Guo, Mao, and Zhang 2019a), which is a straightforward adaptation of Mixup in image to sentence classification. Also, our approach is related to senMixup (Guo, Mao, and Zhang 2019a), which is exactly the same as wordMixup except that the interpolation is performed on the last hidden layer. Unlike wordMixup and senMixup which deploy linear interpolation with a scalar policy, our method embraces a nonlinear interpolation policy. Also, the input and label in the wordMixup and senMixup share the same mixing policy. On the contrary, nonlinear Mixup learns the mixing policy for the modeling target pairs based on the mixed input pairs.

Conclusion and Future Work

We proposed "nonlinear Mixup" to relax the constraint of convex combination in Mixup. The nonlinear mixing policy significantly expands the synthetic sample space for Mixup. Also, the newly introduced target-mixing policy function allows the network to relocate the synthetic samples during training to alleviate the manifold intrusion issue.

We conducted experiments on several benchmark sentence classification datasets, showing that our approach significantly outperformed Mixup and its variants in terms of predictive accuracy. We also empirically demonstrated that the out-of-manifold samples created by our method encourage the learning to form tight representation clusters (one for samples in each class) that are far from each other.

Our future research will investigate the time sensitivity property (Golatkar, Achille, and Soatto 2019) and the working mechanism (Archambault et al. 2019) of the nonlinear Mixup in terms of its regularization effect.

References

Archambault, G. P.; Mao, Y.; Guo, H.; and Zhang, R. 2019. Mixup as directional adversarial training. *CoRR* abs/1906.06875.

Bengio, S.; Weston, J.; and Grangier, D. 2010. Label embedding trees for large multi-class tasks. NIPS'10, 163–171.

Fadaee, M.; Bisazza, A.; and Monz, C. 2017. Data augmentation for low-resource neural machine translation. *CoRR* abs/1705.00440.

Golatkar, A.; Achille, A.; and Soatto, S. 2019. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *CoRR* abs/1905.13277.

Graves, A.; Mohamed, A.; and Hinton, G. E. 2013. Speech recognition with deep recurrent neural networks. *CoRR* abs/1303.5778.

Guo, H.; Mao, Y.; and Zhang, R. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *CoRR* abs/1905.08941.

Guo, H.; Mao, Y.; and Zhang, R. 2019b. Mixup as locally linear out-of-manifold regularization. In *AAAI2019*.

Hanson, S. J., and Pratt, L. Y. 1988. Comparing biases for minimal network construction with back-propagation. In *NIPS1988*, 177–185.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. *CoRR* abs/1603.09382.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, 1746–1751.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Kobayashi, S. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT 2018*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105. USA: Curran Associates Inc.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Li, X., and Roth, D. 2002. Learning question classifiers. COLING '02.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL2004*, 271–278.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. ACL '05, 115–124.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Pussell, L., and Trimble, S. Y. 1991. Gram-schmidt orthogonalization by gauss elimination. *Am. Math. Monthly* 98(6).

Sahin, G. G., and Steedman, M. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *EMNLP 2018*, 5004–5009.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Improving neural machine translation models with monolingual data. *CoRR* abs/1511.06709.

Simard, P.; LeCun, Y.; Denker, J. S.; and Victorri, B. 1998. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *NIPS Workshop*, 239–27.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP '13.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–1958.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215.

Verma, V.; Lamb, A.; Beckham, C.; Courville, A.; Mitliagkas, I.; and Bengio, Y. 2018. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *CoRR*.

Wang, W. Y., and Yang, D. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *EMNLP2015*.

Wei, J. W., and Zou, K. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR* abs/1901.11196.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond empirical risk minimization. In *ICLR2018*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. NIPS'15.