

# DOUBLEMIX: Simple Interpolation-Based Data Augmentation for Text Classification

Hui Chen<sup>\*</sup> Wei Han<sup>\*</sup> Diyi Yang<sup>◇</sup> Soujanya Poria<sup>\*</sup>

<sup>\*</sup> Singapore University of Technology and Design

<sup>◇</sup> Georgia Institute of Technology

{hui\_chen, wei\_han}@mymail.sutd.edu.sg

diyi.yang@cc.gatech.edu

sporia@sutd.edu.sg

## Abstract

This paper proposes a simple yet effective interpolation-based data augmentation approach termed DOUBLEMIX, to improve the robustness of models in text classification. DOUBLEMIX first leverages a couple of simple augmentation operations to generate several perturbed samples for each training data, and then uses the perturbed data and original data to carry out a two-step interpolation in the hidden space of neural models. Concretely, it first mixes up the perturbed data to a synthetic sample and then mixes up the original data and the synthetic perturbed data. DOUBLEMIX enhances models' robustness by learning the "shifted" features in hidden space. On six text classification benchmark datasets, our approach outperforms several popular text augmentation methods including token-level, sentence-level, and hidden-level data augmentation techniques. Also, experiments in low-resource settings show our approach consistently improves models' performance when the training data is scarce. Extensive ablation studies and case studies confirm that each component of our approach contributes to the final performance and show that our approach exhibits superior performance on challenging counterexamples. Additionally, visual analysis shows that text features generated by our approach are highly interpretable. Our code for this paper can be found at <https://github.com/declare-lab/DoubleMix.git>.

## 1 Introduction

Deep neural networks have enabled breakthroughs in most supervised settings in natural language processing (NLP) tasks. However, labeled data in NLP is often scarce, as linguistic annotation usually costs large amounts of time, money, and expertise. With limited training data, neural models will be vulnerable to overfitting and can only capture shallow heuristics that succeed in limited scenarios, which will lead to severe performance degradation when applied to challenging situations.

In order to improve the robustness of models, various data augmentation methods have been proposed. Generally, there are three types of augmentation techniques: token-, sentence-, and hidden-level transformation. Wei and Zou (2019) summarized several common token-level transformations, including word insertion, deletion, replacement, and swap. Sentence-level transformation is to paraphrase a sentence through specific grammatical or syntactic rules. Back-translation (Sennrich et al., 2016; Edunov et al., 2018) is a typical sentence-level augmentation method where a sentence is translated to an intermediate language and then translated back to obtain augmented samples. Additionally, for natural language inference (NLI) tasks that identify whether a premise entails, contradicts, or is neutral with a hypothesis, Min et al. (2020) studied syntactic rules of sentences in inference tasks and proposed several syntactic transformation techniques such as *Inversion* and *Passivization* to construct syntactically informative examples. However, these methods often have high requirements for sentence structures. It is hard to obtain a large number of augmented samples by this method.

In recent years, several hidden-level augmentation methods are proposed and they have exhibited superior performance in a number of popular text classification tasks. TMix (Chen et al., 2020) is a typical approach where a linear interpolation is performed in the hidden space of transformer models such as BERT (Devlin et al., 2019). The main idea of TMix (Verma et al., 2019) comes from Mixup, a method that is based on the principle of Vicinal Risk Minimization (VRM) (Chapelle et al., 2001) and has achieved substantial improvements in computer vision tasks (Verma et al., 2019; Hendrycks et al., 2020; Kim et al., 2020; Ramé et al., 2021) and natural language tasks (Guo et al., 2019; Chen et al., 2020; Kim et al., 2021; Park and Caragea, 2022). Recently, SSMix (Yoon et al., 2021) which interpolates text based on the saliency

of tokens (Simonyan et al., 2014) in hidden space has been introduced. These methods make models learn a mapping from a mixed text representation to an intermediate label which is generated by linearly combining two different source labels. However, the intermediate soft label cannot always accurately describe the true probability of classes that the mixed text representations belong to, which limit the effectiveness of augmentation.

To overcome these limitations, this work proposes a simple yet effective interpolation-based data augmentation method termed DOUBLEMIX, which performs interpolation in the hidden space and does not require label mixing. Firstly, we leverage a collection of simple augmentation operations to generate several perturbed samples from the raw data and then mix up these perturbed samples. Secondly, we mix up the original data with the synthesized perturbed data. We constrain the mixing weight of the original to be larger than the synthesized perturbed data to balance the trade-off between proper perturbations and the potential injected noise. To stabilize the training process, we add a Jensen-Shannon divergence regularization term to our training objective to minimize the distance between the predicted distributions of the original data and the perturbed variants.

To demonstrate the effectiveness of our approach, we conduct extensive experiments by comparing our DOUBLEMIX with previous state-of-the-art data augmentation methods on six popular text classification benchmark datasets. Additionally, we reduce and vary the amount of training data, to observe if DOUBLEMIX can consistently improve over the baselines. We further conduct ablation studies and case studies to investigate the impact of different training strategies on DOUBLEMIX’s effectiveness and whether our method works on challenging counterexamples. Moreover, we visualize the features generated by DOUBLEMIX to interpret why our method works. Experimental results and analyses confirm the efficacy of our proposed approach and every component in DOUBLEMIX contributes to the performance. To sum up, our contributions are:

- We propose a simple interpolation-based data augmentation approach DOUBLEMIX to improve the robustness of neural models in text classification by mixing up the original text with its perturbed variants in hidden space.
- We demonstrate the effectiveness of DOU-

BLEMIX through extensive experiments and analyses on six text classification benchmarks as well as three low-resource datasets.

- We qualitatively analyze why our method works by visualizing its data manifold and quantitatively analyze how our method works by conducting several ablation studies and case studies.

## 2 Related Work

Data augmentation techniques are widely employed in NLP tasks to improve the robustness of models (Sun et al., 2020; Xie et al., 2020; Cheng et al., 2020; Guo et al., 2020; Kwon and Lee, 2022). One way to enrich the original training set is to perturb the tokens in each sentence. For example, Wei and Zou (2019) introduced a set of simple data augmentation operations such as synonym replacement, random insertion, swap, and deletion. However, token-level perturbation sometimes does not guarantee that the augmented sentences are grammatically correct.

Thus, sentence-level augmentation methods are introduced, where people paraphrase the sentence by some specific rules. Minervini and Riedel (2018); McCoy et al. (2019) leveraged syntactic rules to generate adversarial examples in inference tasks. Moreover, Andreas (2020) investigated the compositional inductive bias in sequence models and augmented data by compositional rules. However, these methods require careful design, and they are often customized for a specific task, which makes them hard to generalize to different datasets.

Recently, a couple of hidden-level augmentation techniques which perform interpolation in hidden space have been studied (Guo et al., 2019; Verma et al., 2019; Hendrycks et al., 2020). Inspired by PuzzleMix (Kim et al., 2020) and SaliencyMix (Uddin et al., 2020) which is popular in computer vision, Yoon et al. (2021) proposed SSMix which utilizes the saliency information of spans (Simonyan et al., 2014) in each sentence to interpolate in hidden space to create informative examples. Yin et al. (2021) interpolate hidden states of the entire minibatch to obtain better representations. Inspired by the prior work, our DOUBLEMIX aims at improving models’ robustness by mixing up text features and their perturbed samples in hidden space.

### 3 Proposed Method: DOUBLEMIX

To regularize NLP models in a more efficient manner, we introduce a simple yet effective data augmentation approach, DOUBLEMIX, that enhances the representation of each training data by learning the features sampled from a region constructed by the original sample itself and its perturbed samples. The perturbed samples are generated by simple token- or sentence-level augmentation operations. DOUBLEMIX is a hidden-level regularization technique and our base model is a pre-trained transformer network, as they have achieved great performance in various NLP tasks. Algorithm 1 shows the training process of our approach.

#### 3.1 Robust Interpolation in Hidden Space

For an input sequence  $x = \{w_0, w_1, \dots, w_S\}$  with  $S$  tokens associated with a label  $y$ , our goal is to predict a label of this sequence. At the beginning of this approach, we prepare a perturbation operation set containing simple token- and sentence-level data augmentation techniques such as back-translation (sentence-level), synonym replacement (token-level), and Gaussian noise perturbation (sentence-level). Thereafter, we randomly sample the operations  $N$  times and use the selected augmentation operations to generate  $N$  perturbed samples of each training instance. Note that each type of operation can be selected multiple times. We generated different perturbed samples by adjusting the hyper-parameters. For example, if we select synonym replacement, we can produce different perturbations by adjusting the proportion of tokens to be substituted. For back-translation, we can use different intermediate languages.

Our approach is performed in hidden space, so as to encourage the model to fully utilize the hidden information within the multi-layer networks. We employ a pre-trained model  $f(\cdot; \theta)$  containing  $L$  layers to encode the text to hidden representations. Then we select a layer  $i$  which ranges in  $[0, L]$  to interpolate. At the  $i$ -th layer of  $f(\cdot; \theta)$ , a two-step interpolation is performed where the first step is to mix up all the perturbed samples by a group of weights sampled from Dirichlet distribution, and the second step is to mix up the synthesized perturbed sample and the original sample by some weights  $\in [0, 1]$  sampled from Beta distribution. We follow Zhang et al. (2018) to use Beta distribution for weight sampling, and Dirichlet distribution is a multi-variate Beta distribution. Note that when

we mix up the original data and the synthesized perturbed data, we constrain the mixing weight of the original data to be larger, so as to make the final perturbed representation to be close to the original one. This balances the trade-off between proper perturbation and potential injected noise. After the two-step interpolation, the synthesized hidden representation is fed to the remaining layers  $f_{[i:L]}(\cdot; \theta)$  and a classifier.

#### 3.2 Training Objectives

During the training period, we do not directly minimize the Cross-Entropy loss of the probability distribution of the synthesized sample, as it may introduce too much noise. We employ a consistency regularization term, Jensen-Shannon Divergence (JSD) loss (Bachman et al., 2014; Zheng et al., 2016; Hendrycks et al., 2020), to minimize the difference between the prediction distribution of synthetic data and original data, and meanwhile, we minimize the Cross-Entropy loss of the model output of original data and the gold label. The training objective can be written as:

$$\mathcal{L}(y|x, x_a, \dots, x_N) = \mathcal{L}_{CE}(y|x) + \gamma \mathcal{L}_{JSD}(y|x, x_a, \dots, x_N) \quad (1)$$

where  $\gamma$  is a hyper-parameter.

In the consistency regularization, we do not employ Kullback-Leibler divergence (KL) because it is not symmetric, i.e.,  $\text{KL}(P||Q) \neq \text{KL}(Q||P)$  when  $P \neq Q$ . It is not a promising choice to measure the similarity of  $p_{mix}$  and  $p_{orig}$  using KL, as neither  $p_{mix}$  nor  $p_{orig}$  are true predictions and we deem that they share equal status. JSD provides a smoothed and normalized version of KL divergence, with scores between 0 (identical) and 1 (maximally different). We believe using such a symmetric metric can make the training more stable.

#### 3.3 Why does DOUBLEMIX work?

To further discuss why our method works, we visualize the original data and the sample space of synthesized data in Mixup and DOUBLEMIX in Fig. 1. For brevity, we assume the number of perturbed samples in Step I is two. As shown in Fig. 1, blue dots indicate training data, and orange dots are perturbed data generated by our selected operations. Synthesized data in Mixup (Zhang et al., 2018) can only be created along a line, such as the blue full line connecting the two points  $X$  and  $X'$  since it

---

**Algorithm 1: DOUBLEMIX**


---

**Input:** Model  $f(\cdot; \theta)$  containing  $L$  layers, the  $l$ -th layer of the model  $f_l(\cdot; \theta)$ , classifier  $\hat{p}(\cdot; \phi)$ , training set  $\mathcal{X} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , perturbation operation set  $O = \{\text{back-translation (BT), synonym replacement (SR), \dots, Gaussian noise (GN)}\}$ , interpolation layer set  $I = \{i_1, \dots, i_k\}$ , number of augmented samples  $N$ , number of training epochs  $K$ , the global learning rate  $\eta$ , Beta distribution hyper-parameter  $\alpha$ , Dirichlet distribution hyper-parameter  $\tau$ , loss hyper-parameter  $\gamma$

**Output:** Updated network weights  $\theta$ ,  $\phi$

```

1  $\{o_1, \dots, o_N\} \leftarrow O$   $\triangleright$  Select  $N$  operations. Each type
   of operation can be selected multiple times.
2 for  $k \leftarrow 0$  to  $K$  do
3   for  $(x, y) \in \mathcal{X}$  do
4      $\{x_a, \dots, x_N\} \leftarrow \{o_1, \dots, o_N\}(x)$   $\triangleright$  Apply
       the selected operations to generate  $N$ 
       different augmented samples of  $x$ .
5      $i \leftarrow I = \{i_1, \dots, i_k\}$   $\triangleright$  Randomly select an
       interpolation layer from Set  $I$ 
6      $h_{orig}^i \leftarrow f_{[0:i]}(x; \theta)$ 
7      $\{h_a^i, \dots, h_N^i\} \leftarrow f_{[0:i]}(\{x_a, \dots, x_N\}; \theta)$   $\triangleright$ 
       Encode  $\{x, x_a, \dots, x_N\}$ , and interpolate at
       the  $i$ -th layer.
8     Sample  $(w_1, \dots, w_n) \sim \text{Dirichlet}(\tau, \dots, \tau)$ 
9      $h_{aug}^i \leftarrow w_1 \cdot h_a^i + \dots + w_n \cdot h_N^i$   $\triangleright$  First
       mixup (Step I).
10    Sample  $\lambda \sim \text{Beta}(\alpha, \alpha)$ 
11     $\lambda \leftarrow \max(\lambda, 1 - \lambda)$   $\triangleright$  Constrain the
       synthetic data to a region closer to the
       original example.
12     $h_{mix}^i \leftarrow \lambda \cdot h_{orig}^i + (1 - \lambda) \cdot h_{aug}^i$   $\triangleright$ 
       Second mixup (Step II).
13     $h_{mix} \leftarrow f_{[i+1:L]}(h_{mix}^i; \theta)$ 
14     $p_{mix} \leftarrow \hat{p}(y|h_{mix}; \phi)$ 
15     $p_{orig} \leftarrow \hat{p}(y|f(x; \theta); \phi)$ 
16     $\bar{p} \leftarrow \frac{1}{2}(p_{mix} + p_{orig})$ 
17     $\mathcal{L}_{JSD}(y|x, x_a, \dots, x_N) \leftarrow$ 
        $\frac{1}{2}(\text{KL}(p_{mix}||\bar{p}) + \text{KL}(p_{orig}||\bar{p}))$ 
18     $\mathcal{L} \leftarrow \mathcal{L}_{CE}(y|x) + \gamma \mathcal{L}_{JSD}(y|x, x_a, \dots, x_N)$ 
19     $\theta \leftarrow \theta - \eta \nabla_{\theta, \phi} \mathcal{L}; \phi \leftarrow \phi - \eta \nabla_{\theta, \phi} \mathcal{L}$   $\triangleright$ 
       Update the network weight  $\theta$  of the base
       model  $f$  and  $\phi$  of the classifier  $\phi$ .
20   end
21 end

```

---

is a simple linear combination. Hence, Mixup enforces the regularization to behave linearly among the training data. In contrast to Mixup, the sample space of synthesized data of DOUBLEMIX is a polygon. In this example,  $\triangle XAB$  is the sample space of the synthesized data in DOUBLEMIX. Firstly, Step I samples a point  $P$  on the line connecting

$X_a$  and  $X_b$ . Secondly, Step II finds a point  $Q$  on the line connecting  $X$  and  $P$ . Note that  $Q$  should be closer to  $X$  or at the middle of Line  $XP$ , as in Step II, we constrain the mixing weight of original data to be larger than that of synthesized perturbed data. Taken together, our approach enforces the model to learn nearby features for each training data so that it is robust to representation shifts.

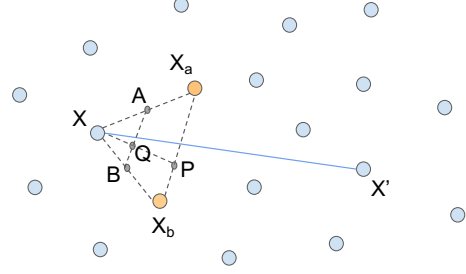


Figure 1: Visualization of the sample space of synthesized data in Mixup and DOUBLEMIX. Blue dots  $X$  and  $X'$  indicate two data points in the training set. Orange dots  $X_a$  and  $X_b$  are perturbed data of  $X$ . Grey dots  $A$ ,  $B$ ,  $P$  and  $Q$  are sampled points.

## 4 Experimental Setup

### 4.1 Datasets

We compare our approach with several data augmentation baselines on six text classification datasets, covering sentiment polarity classification, question type classification, humor detection, and natural language inference: IMDB (Maas et al., 2011) and SST-2 (Socher et al., 2013) which predict the sentiment of movie reviews to be positive or negative, 6-class open-domain question classification TREC (Li and Roth, 2002), Pun of the day (Puns) (Yang et al., 2015) which detects humor in a single sentence, and two inference datasets SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) which identify whether a premise entails, contradicts or is neutral with a hypothesis. Statistics of the six text classification datasets can be found in Appendix A. As the test set of MNLI is not publicly available, we used the matched development set as our development set and the mismatched development set as our test set in our experiments.

### 4.2 Baselines

We compare our approach with several widely-used baseline models, including token-, sentence-, and

Method	SST-2		TREC		Puns	
	Acc.	F1.	Acc.	F1.	Acc.	F1.
BERT (Devlin et al., 2019)	91.08 <sub>0.1</sub>	91.09 <sub>0.1</sub>	96.90 <sub>0.4</sub>	96.21 <sub>0.7</sub>	94.20 <sub>0.2</sub>	94.38 <sub>0.2</sub>
+ Easy Data Augmentation (Wei and Zou, 2019)	91.66 <sub>0.2</sub>	91.75 <sub>0.1</sub>	97.10 <sub>0.3</sub>	96.74 <sub>0.2</sub>	93.71 <sub>0.6</sub>	93.83 <sub>0.5</sub>
+ Back Translation (Edunov et al., 2018)	91.33 <sub>0.2</sub>	91.29 <sub>0.1</sub>	96.90 <sub>0.4</sub>	96.72 <sub>0.1</sub>	93.27 <sub>0.4</sub>	93.34 <sub>0.4</sub>
+ Manifold Mixup (Verma et al., 2019)	91.33 <sub>0.4</sub>	91.44 <sub>0.3</sub>	97.00 <sub>0.3</sub>	96.29 <sub>0.5</sub>	94.37 <sub>0.9</sub>	94.44 <sub>1.0</sub>
+ TMix (Chen et al., 2020)	91.13 <sub>0.5</sub>	91.29 <sub>0.3</sub>	96.90 <sub>0.1</sub>	96.32 <sub>0.3</sub>	94.21 <sub>0.7</sub>	94.28 <sub>0.8</sub>
+ SSMix (Yoon et al., 2021)	91.45 <sub>0.5</sub>	91.50 <sub>0.2</sub>	97.00 <sub>0.2</sub>	96.44 <sub>0.2</sub>	94.33 <sub>0.7</sub>	94.46 <sub>0.6</sub>
+ DOUBLEMIX (Ours)	<b>92.21</b> <sub>0.1</sub>	<b>92.11</b> <sub>0.1</sub>	<b>97.40</b> <sub>0.1</sub>	<b>97.32</b> <sub>0.6</sub>	<b>94.59</b> <sub>0.1</sub>	<b>94.66</b> <sub>0.1</sub>

Method	IMDB		SNLI		MNLI	
	Acc.	F1.	Acc.	F1.	Acc.	F1.
BERT (Devlin et al., 2019)	83.52 <sub>0.1</sub>	83.65 <sub>0.4</sub>	90.31 <sub>0.5</sub>	90.28 <sub>0.5</sub>	84.16 <sub>0.2</sub>	84.05 <sub>0.2</sub>
+ Easy Data Augmentation (Wei and Zou, 2019)	83.43 <sub>0.1</sub>	83.85 <sub>0.1</sub>	90.27 <sub>0.4</sub>	90.28 <sub>0.4</sub>	84.04 <sub>0.1</sub>	83.97 <sub>0.2</sub>
+ Back Translation (Edunov et al., 2018)	83.69 <sub>0.1</sub>	84.11 <sub>0.2</sub>	90.21 <sub>0.5</sub>	90.23 <sub>0.4</sub>	84.50 <sub>0.3</sub>	84.44 <sub>0.3</sub>
+ Syntactic Data Augmentation (Min et al., 2020)	-	-	90.35 <sub>0.3</sub>	90.31 <sub>0.4</sub>	84.19 <sub>0.3</sub>	84.08 <sub>0.2</sub>
+ Manifold Mixup (Zhang et al., 2018)	83.63 <sub>0.1</sub>	83.81 <sub>0.2</sub>	90.04 <sub>0.2</sub>	90.02 <sub>0.2</sub>	83.37 <sub>0.2</sub>	83.31 <sub>0.1</sub>
+ TMix (Chen et al., 2020)	83.47 <sub>0.3</sub>	83.91 <sub>0.2</sub>	90.12 <sub>0.1</sub>	90.09 <sub>0.1</sub>	83.43 <sub>0.1</sub>	83.38 <sub>0.1</sub>
+ SSMix (Yoon et al., 2021)	83.55 <sub>0.3</sub>	83.88 <sub>0.2</sub>	90.21 <sub>0.2</sub>	90.14 <sub>0.3</sub>	83.66 <sub>0.2</sub>	83.54 <sub>0.2</sub>
+ DOUBLEMIX (Ours)	<b>84.14</b> <sub>0.5</sub>	<b>84.39</b> <sub>0.2</sub>	<b>91.03</b> <sub>0.1</sub>	<b>91.02</b> <sub>0.1</sub>	<b>84.72</b> <sub>0.2</sub>	<b>84.64</b> <sub>0.1</sub>

Table 1: Test accuracy (%) and F1 scores (%) for BERT when comparing our proposed DOUBLEMIX with baseline methods on six text classification datasets. We randomly select two augmented samples to mix up in Step I. Best scores are marked in bold. As syntactic transformations (Min et al., 2020) are rule-based data augmentation techniques customized for inference tasks, we only show their performances on SNLI and MNLI. We report the mean accuracy and F1 scores across five different runs with the standard deviation shown in subscript (e.g., 91.08<sub>0.1</sub> indicates 91.08 ± 0.1).

hidden-level augmentation techniques. Token-level baselines contain the operations in Easy Data Augmentation (EDA) (Wei and Zou, 2019) where they randomly insert, swap, and delete tokens in each sentence. Sentence-level baselines include paraphrasing sentences such as back-translation (Senrich et al., 2016) and applying some syntactic rules to create augmented sentences such as syntactic transformation (Min et al., 2020). Hidden-level baselines include Manifold Mixup (Verma et al., 2019), TMix (Chen et al., 2020), and SSMix (Yoon et al., 2021), where they mix up two different training samples in hidden space and learn a mapping from the intermediate representation to an intermediate label. Details of the implementation of the baselines can be found in Appendix B.

## 5 Results and Analysis

We evaluate our baselines and proposed approach on six text classification benchmark datasets. We also show the performance of our approach in low-resource settings to confirm DOUBLEMIX is efficient and robust when the training samples are scarce. This section will discuss the performance of models in detail and quantitatively analyze how and why DOUBLEMIX works.

### 5.1 Main Results

Table 1 shows the performance of DOUBLEMIX and the relevant baselines on six text classification datasets. Our base model is the BERT-base-uncased model. We observe that DOUBLEMIX achieves the best average results compared to previous state-of-the-art baselines across six datasets, where DOUBLEMIX shows the greatest improvements over BERT on SST-2, increasing the test accuracy and binary F1 score by 1.13% and 1.02%.

In addition, we find token-level Easy Data Augmentation and sentence-level Back Translation are not able to improve the BERT baseline on Puns and SNLI. Especially on Puns, Back Translation’s test accuracy and F1 score are about 1% lower than BERT. This might be that labels in humor detection and inference tasks are closely related to the presence of some important words, and Easy Data Augmentation and Back Translation may perturb these words, making the true label flip, but the label learned by the model does not change accordingly, which leads to an inefficient learning process. Moreover, hidden-level augmentations such as Manifold Mixup, TMix, and SSMix fail to improve the base model on SNLI and MNLI. As subtle changes in the sentences in inference tasks

Method	SNLI				MNLI				Avg. (Acc./F1.)
	1K (Acc./F1.)	2.5K (Acc./F1.)	5K (Acc./F1.)	10K (Acc./F1.)	1K (Acc./F1.)	2.5K (Acc./F1.)	5K (Acc./F1.)	10K (Acc./F1.)	
BERT	69.77/69.59	76.10/75.92	79.28/79.25	82.36/82.28	55.81/54.61	65.63/65.15	71.24/71.01	74.24/74.14	71.80/71.49
+ BT	70.23/69.96	76.51/76.54	79.57/79.57	82.68/82.65	57.28/55.53	66.97/66.95	72.28/72.01	74.49/74.44	72.50/72.21
+ M-Mix	71.45/71.34	76.48/76.42	79.91/79.83	82.14/82.13	57.01/56.70	67.08/66.96	71.76/71.68	74.68/74.54	72.56/72.45
+ TMix	71.04/71.12	76.38/76.10	79.85/79.81	82.09/82.09	<b>57.31</b> /56.99	67.10/67.01	71.66/71.59	74.86/74.70	72.54/72.43
+ SSMix	71.32/71.21	76.87/76.72	80.02/79.93	82.41/82.20	57.25/ <b>57.10</b>	67.13/67.05	71.70/71.63	74.77/74.62	72.68/72.56
+ Ours	<b>71.82/71.72</b>	<b>77.43/77.42</b>	<b>80.75/80.72</b>	<b>83.18/83.26</b>	56.15/55.91	<b>67.57/67.33</b>	<b>72.35/72.15</b>	<b>75.07/74.97</b>	<b>73.04/72.94</b>
RoBERTa	78.11/77.95	82.47/82.30	83.17/83.36	85.58/85.50	70.23/70.45	75.50/75.53	79.01/79.04	81.00/80.98	79.38/79.39
+ BT	78.32/78.25	82.47/82.61	84.23/84.08	86.07/86.05	70.67/70.52	<b>77.58/77.39</b>	79.17/79.07	81.26/81.08	79.97/79.88
+ M-Mix	79.32/78.73	82.71/82.63	84.60/84.63	86.00/85.95	71.78/71.04	76.00/75.96	79.43/79.34	81.40/81.28	80.16/79.95
+ TMix	79.17/79.11	82.84/82.91	85.09/85.13	86.16/86.13	<b>72.05/72.18</b>	76.57/76.44	79.92/79.80	81.30/81.17	80.39/80.36
+ SSMix	79.43/79.35	82.91/82.88	85.33/85.36	86.32/86.28	71.96/71.88	76.44/76.38	79.86/79.77	81.25/81.22	80.44/80.39
+ Ours	<b>80.41/80.31</b>	<b>83.96/83.92</b>	<b>85.42/85.42</b>	<b>86.91/86.88</b>	71.20/71.15	77.12/76.96	<b>80.43/80.28</b>	<b>82.46/82.24</b>	<b>80.99/80.90</b>

Table 2: Test accuracy (%) and F1 score (%) comparison on the SNLI and MNLI datasets training with varying amounts of training data (1000, 2500, 5000, and 10000). Best scores are marked in bold in yellow background. BT and M-Mix represent Back Translation and Manifold Mixup. We only use BT operations in DOUBLEMIX in this experiment.

will flip the true label, mixing up two different samples and learning an intermediate representation in hidden space cannot ensure the learned soft label is the true label of the intermediate representation.

In contrast, our model shows consistent improvements over BERT on these datasets. The consistent improvements indicate that, by strategically mixing up samples with similar meanings in the hidden space, DOUBLEMIX not only helps pre-trained models to become insensitive to feature perturbations in an effective way but also injects less potential noise during the augmentation process compared to other baselines.

## 5.2 Performance in Low-Resource Settings

To investigate the performance of our approach in low-resource settings, we randomly sample 1000, 2500, 5000, and 10000 examples from the original training data of SNLI and MNLI to construct our training sets for low-data setting evaluations, while the size of the development and test sets is unchanged. Apart from the BERT-base-uncased model (Devlin et al., 2019), we also conduct experiments on the RoBERTa-base model (Liu et al., 2019).

Table 2 presents the results in low-data settings. Compared with the BERT baseline in Table 1, we can observe that although pre-trained language models are powerful across text classification tasks, the test accuracy and F1 scores might decrease a lot when the training data is very scarce. DOUBLEMIX consistently improves the base model with no data augmentation on both SNLI and MNLI and outperforms all the baselines on the SNLI dataset. On MNLI, we observe that our method always

achieves the top performance except in the 1K training samples. As the training set grows larger, our model gradually outperforms the baselines and the leading gap keeps expanding—when the number of training samples reaches 10K, our model can achieve at least 1% higher accuracy and F1 score than RoBERTa on both SNLI and MNLI.

## 5.3 Ablation Studies

### 5.3.1 Training Strategies in DOUBLEMIX

We also conduct a series of ablative experiments to examine the contribution of individual components. The results are displayed in Table 3, where our experiments are conducted on the SNLI dataset with only 1000 training samples. We find the performance drops after changing the training strategies, suggesting that the current interpolation method trained with JSD loss in DOUBLEMIX contributes to the final performance.

Concretely, we first remove the JSD loss in our training objective to check if this loss contributes to the performance. We observe that the accuracy and F1 score drop approximately 0.7% and 0.9% after removing the JSD loss, which manifests that JSD loss is capable of stabilizing the training process. Secondly, we merge the two steps and see how the model performs, where we use a Dirichlet distribution to sample  $N + 1$  mixing weights for the original example and other  $N$  augmented examples, and mix up them at a time. In this case, the test accuracy and F1 score drop to 71.11% and 70.96%, respectively. This indicates that the two-step interpolation where we constrain the synthetic data to a region closer to the original sample, as mentioned in Line 11 in Algorithm 1 will inject less

noise. Moreover, we have also tried different mixing samples Step II. We find mixing up with another randomly selected training sample in Line 12 in Algorithm 1 results in a 0.59% accuracy decrease and a 0.70% F1 score decrease. If the selected training sample is restricted to the same category as the original data, the performance degradation will be even larger. This outcome may be caused by the larger semantic difference between the selected example and the original data compared to the augmented examples of the original data.

Method	Acc.	F1.
DOUBLEMIX	71.82	71.72
- w/o JSD loss	71.15	70.87
- merge Step II and Step I	71.11	70.96
- mix with another training sample in Step II	71.23	71.02
- mix with another same-class sample in Step II	70.98	70.74

Table 3: Test accuracy (%) and F1 scores (%) on the SNLI dataset with 1000 training samples after changing different parts of DOUBLEMIX.

Mixup layer set	Acc.	$\Delta_{Acc}$	F1.	$\Delta_{F1}$
$\emptyset$	69.77		69.59	
{0}	71.13	+1.36	71.02	+1.43
{0,1,2}	70.95	+1.18	70.84	+1.25
{3,4}	71.56	+1.79	71.48	+1.89
{3,6,9}	71.24	+1.47	71.16	+1.57
{7,9,12}	71.30	+1.53	71.19	+1.60
{9,10,12}	<b>71.82</b>	<b>+2.05</b>	<b>71.72</b>	<b>+2.13</b>
{3,4,6,9,10,12}	70.90	+1.13	70.90	+1.31
{3,4,6,7,9,12}	71.29	+1.52	71.16	+1.57

Table 4: Test accuracy (%) and F1 scores (%) on SNLI with 1000 training data with different interpolation layer sets.  $\emptyset$  means no interpolation, and {0} is the input space.  $\Delta$  indicates the gap to the baseline with no augmentation.

### 5.3.2 Effect of Interpolation Layers

We believe the hidden layers in pre-trained language models are powerful in representation learning, and interpolation in the hidden space can yield a larger performance improvement than in the input space. In this section, we will investigate which interpolation option in terms of the set of layers in pretrained models can obtain the best performance. Previous work Jawahar et al. (2019) indicates that BERT’s intermediate layers {3, 4} perform best in encoding surface features and layers {6, 7, 9, 12} contain the most syntactic features and semantic features. We refer to Jawahar et al. (2019) to for-

mulate several sets of layers and have conducted a couple of additional experiments on BERT + DOUBLEMIX with different sets of interpolation layers on SNLI with 1000 training samples to see which subsets give the optimal performance. The results are shown in Table 4.

Model	Original and Counterfactual Examples	P.	T.
BERT	<i>P</i> : Students are inside of a lecture hall. <i>H</i> : Students are indoors.	N	E
Ours	<i>P</i> : Students are inside of a lecture hall. <i>H</i> : Students are indoors.	E	E
BERT	<i>P</i> : Students are inside of a lecture hall. <i>H</i> : Students are on the soccer field.	C	C
Ours	<i>P</i> : Students are inside of a lecture hall. <i>H</i> : Students are on the soccer field.	C	C
BERT	<i>P</i> : Man in green jacket with baseball hat on. <i>H</i> : The man is not wearing a hat.	C	C
Ours	<i>P</i> : Man in green jacket with baseball hat on. <i>H</i> : The man is not wearing a hat.	C	C
BERT	<i>P</i> : Man in green jacket with baseball hat on. <i>H</i> : The man is at a baseball game.	E	N
Ours	<i>P</i> : Man in green jacket with baseball hat on. <i>H</i> : The man is at a baseball game.	N	N

Table 5: Predictions of BERT and our method on original examples and their counterfactual examples on the SNLI dataset. The counterfactual examples are extracted from Kaushik et al. (2020) and are constructed by substituting entities or adding details to entities. The examples in the yellow background are counterexamples. **P.** and **T.** represent prediction and true label. E, N and C are *entailment*, *neutral*, and *contradiction*. *P* and *H* are premise and hypothesis. Labels in red are wrong predictions while labels in blue are correct predictions.

When all the interpolation steps are excluded, the test accuracy is 69.77% and the F1 score is 69.59%. When we interpolate in the input space (the 0-th layer), the accuracy and F1 score increase by about 1.3% and 1.4%, showing that interpolation contributes to the performance. When we perform interpolations at layer set {0, 1, 2} with lower layers, the accuracy and F1 increases are smaller than interpolating in input space. However, when we interpolate at some middle layers such as {3, 4} and {3, 6, 9}, the performance improvement is more significant. According to Jawahar et al. (2019), the 9-th layer captures most of the syntactic and semantic information. We have tried several layer sets containing the 9-th layer and find {9, 10, 12} containing upper layers performs best. At the same time, we notice that the number of layers in the interpolation layer set is not the more the bet-

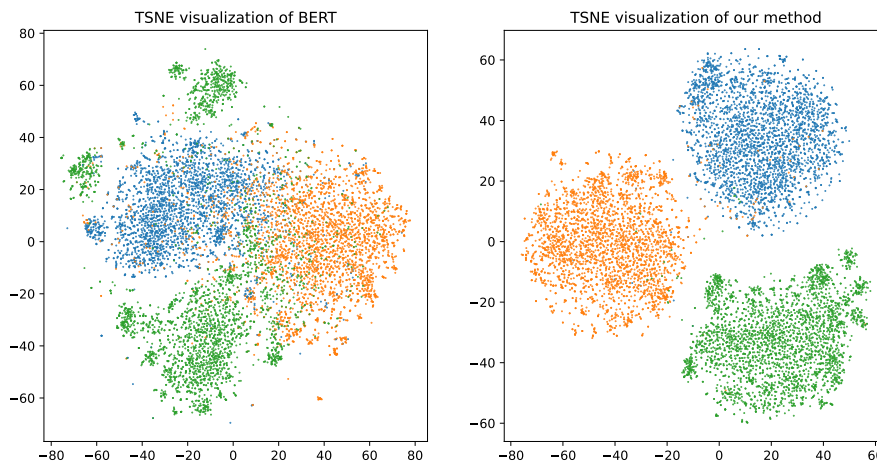


Figure 2: T-SNE projection of the features generated by the 12-layer encoder of BERT baseline (left) and BERT + DOUBLEMIX (right) on SNLI with 10000 training samples. The visualized features of the augmented text are extracted from the last layer of the base model during testing.

ter. The performance of  $\{3,4,6,9,10,12\}$  is only 70.90%, which is lower than any other layer set, and the performance of  $\{3,4,6,7,9,12\}$  is not the best, indicating that too many interpolation layers will reduce the efficiency of representation learning.

#### 5.4 Case Studies

To further understand how DOUBLEMIX works, we randomly pick up some examples from the SNLI dataset and check the discrepancy between the predictions obtained from BERT and our method. Predictions of BERT and DOUBLEMIX are shown in Table 5. We find in those examples with “*contradiction*” label, both BERT and DOUBLEMIX can accurately predict the true label. Additionally, to investigate how our method behaves in challenging scenarios, we also test on the counterfactual version (Kaushik et al., 2020) of our selected samples. Both models excel in detecting negative words “*not*” and location names. However, when the ground truth is “*entailment*” or “*neutral*”, DOUBLEMIX is more likely to make correct predictions. When the premise and hypothesis contain some common words (e.g., *Man in green jacket with **baseball** hat on. The man is at a **baseball** game.*), DOUBLEMIX inclines to make more accurate predictions.

#### 5.5 Manifold Visualization

Finally, we visualize the embedding vectors generated by the 12-layer encoder of the BERT baseline with no data augmentation (BERT) and with DOU-

BLEMIX to qualitatively show the effectiveness of our approach in facilitating the model to learn robust representations in Fig. 2. Our experiments are conducted on SNLI with 10000 training samples. The visualized features are extracted from the output of the last layer of the model. We employ t-SNE (Van der Maaten and Hinton, 2008) which is implemented by the python package scikit-learn<sup>1</sup> to visualize the features. In Fig. 2, there are three clusters with three colors indicating different classes. We observe that the features in DOUBLEMIX are better separated than those in BERT, indicating that our method effectively improves robustness by encouraging the model to learn nearby features of each training sample.

## 6 Conclusion

In this work, we present a simple interpolation-based data augmentation approach DOUBLEMIX to improve models’ robustness on a wide range of text classification datasets. DOUBLEMIX first leverages simple augmentation operations to generate perturbed data of each training sample and then performs a two-step interpolation in the hidden space of models to learn robust representations. Our approach outperforms several popular data augmentation methods on six benchmark datasets and three low-resource datasets. Finally, ablation studies, case studies, and visualization of manifold fur-

<sup>1</sup><https://github.com/scikit-learn/scikit-learn>



ther explain how and why our method works. Our future work includes making the mixing weights learnable as well as extending DOUBLEMIX to natural language generation tasks.

## Acknowledgement

This work is supported by the A\*STAR under its RIE 2020 AME programmatic grant RGAST2003 and project T2MOE2008 awarded by Singapore’s MoE under its Tier-2 grant scheme.

## References

- Jacob Andreas. 2020. [Good-enough compositional data augmentation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.
- Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. [Learning with pseudo-ensembles](#). In *Advances in Neural Information Processing Systems*, volume 27, pages 3365–3373.
- Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.
- Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2001. [Vicinal risk minimization](#). In *Advances in neural information processing systems*, pages 416–422.
- Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. [Mix-Text: Linguistically-informed interpolation of hidden space for semi-supervised text classification](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.
- Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. [AdvAug: Robust adversarial augmentation for neural machine translation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.
- 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.
- Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. [Understanding back-translation at scale](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.
- Demi Guo, Yoon Kim, and Alexander Rush. 2020. [Sequence-level mixed sample data augmentation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics.
- Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. [Mixup as locally linear out-of-manifold regularization](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722.
- Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2020. [Augmix: A simple data processing method to improve robustness and uncertainty](#). In *International Conference on Learning Representations*.
- Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. [What does BERT learn about the structure of language?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.
- Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. [Learning the difference that makes a difference with counterfactually-augmented data](#). In *International Conference on Learning Representations*.
- Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. [Puzzle mix: Exploiting saliency and local statistics for optimal mixup](#). In *International Conference on Machine Learning*, pages 5275–5285. PMLR.
- Yekyung Kim, Seohyeong Jeong, and Kyunghyun Cho. 2021. [Linda: Unsupervised learning to interpolate in natural language processing](#). *arXiv preprint arXiv:2112.13969*.
- Soonki Kwon and Younghoon Lee. 2022. [Explainability-based mix-up approach for text data augmentation](#). *ACM Transactions on Knowledge Discovery from Data (TKDD)*.
- Xin Li and Dan Roth. 2002. [Learning question classifiers](#). In *COLING 2002: The 19th International Conference on Computational Linguistics*.
- 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*.
- Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

- Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. [Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.
- Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. [Syntactic data augmentation increases robustness to inference heuristics](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online. Association for Computational Linguistics.
- Pasquale Minervini and Sebastian Riedel. 2018. [Adversarially regularising neural nli models to integrate logical background knowledge](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 65–74.
- Seo Yeon Park and Cornelia Caragea. 2022. [A data cartography based MixUp for pre-trained language models](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4244–4250.
- Alexandre Ramé, Rémy Sun, and Matthieu Cord. 2021. [Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 823–833.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Improving neural machine translation models with monolingual data](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. [Deep inside convolutional networks: Visualising image classification models and saliency maps](#). In *In Workshop at International Conference on Learning Representations*. Citeseer.
- 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, S Yu Philip, and Lifang He. 2020. [Mixup-transformer: Dynamic data augmentation for nlp tasks](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440.
- AFM Shahab Uddin, Mst Sirazam Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. 2020. [Saliencymix: A saliency guided data augmentation strategy for better regularization](#). In *International Conference on Learning Representations*.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. [Visualizing data using t-sne](#). *Journal of machine learning research*, 9(11).
- 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 *International Conference on Machine Learning*, pages 6438–6447. PMLR.
- 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 6383–6389.
- Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. [A broad-coverage challenge corpus for sentence understanding through inference](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.
- Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. [Unsupervised data augmentation for consistency training](#). *Advances in Neural Information Processing Systems*, 33:6256–6268.
- Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. [Humor recognition and humor anchor extraction](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.
- 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](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3225–3234.
- Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. [mixup: Beyond empirical risk minimization](#). In *International Conference on Learning Representations*.
- Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. [Improving the robustness of deep neural networks via stability training](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4480–4488.

## A Dataset Statistics

Table 6 describes the statistics of the datasets we used. Note that for SST-2, we did not use the one

on the GLUE benchmark, as the test labels are not publicly available. We used the original SST-2 dataset and it can be loaded from huggingface datasets<sup>2</sup>.

Dataset	Task Type	# Label	Size
SST-2	Sentiment	2	6.9k / 872 / 1.8k
TREC	Classification	6	4.9k / 546 / 500
Puns	Humor	2	3.6k / 603 / 604
IMDB	Sentiment	2	22.5k / 2.5k / 25k
SNLI	Inference	3	550k / 9.8k / 9.8k
MNLI	Inference	3	392k / 9.8k / 9.8k

Table 6: Summary statistics of the seven natural language understanding datasets. We report the size of datasets as (train / validation / test) format.

## B Baseline Details

This section introduces the details of our baselines:

- **Easy Data Augmentation** (Wei and Zou, 2019) contains several simple data augmentation techniques in text such as synonym replacement, random insertion, random swap, and random deletion. The experimental setup for these methods is the same as that of back-translation. We use the official code<sup>3</sup> with the default insertion/deletion/swap ratio the author provided.
- **Back-translation** (Sennrich et al., 2016) translates an input text in some source language (e.g. English) to another intermediate language (e.g. German), and then translates it back into the original one. In our experiments, our intermediate languages are German and Russian. And we create two types of augmented text for every training sample. These augmented examples are directly added to the training set. We use the code of fairseq<sup>4</sup> to implement this baseline.
- **Syntactic Transformation** (Min et al., 2020) applies rule-based syntactic transformations such as inversion and passivization to sentences to generate augmentations in inference tasks. We directly add the augmented data to

<sup>2</sup><https://huggingface.co/datasets/gpt3mix/sst2>

<sup>3</sup>[https://github.com/jasonwei20/eda\\_nlp](https://github.com/jasonwei20/eda_nlp)

<sup>4</sup><https://github.com/pytorch/fairseq/blob/main/examples/wmt19/README.md>

the training set. The implementation is based on the official code<sup>5</sup> the author provided.

- **Manifold Mixup** (Verma et al., 2019) performs in hidden space. Similar to Mixup (Zhang et al., 2018), Manifold Mixup samples two training examples and mixes up the hidden representations using a coefficient  $\lambda_0$  randomly sampled from Beta( $\alpha$ ,  $\beta$ ). For the training objective, Manifold Mixup first uses the Cross-Entropy loss to measure the divergence between the predicted distribution and the one-hot vector of gold label, and then mix up the Cross-Entropy losses. Our implementation is based on the official code of Mixup<sup>6</sup>.
- **TMix** (Chen et al., 2020) is similar to Manifold Mixup which performs interpolation in hidden space. We first mix up the gold labels to a sythetic label and secondly minimize the KL divergence between the sythetic label and the predicted distribution. We use the code implemented in the MixText<sup>7</sup> repository.
- **SSMix** (Yoon et al., 2021) is similar to PuzzleMix (Kim et al., 2020) and SaliencyMix (Uddin et al., 2020). It applies Mixup based on the saliency (Simonyan et al., 2014) of tokens. Our implementation is based on the official code of SSMix<sup>8</sup>.

## C Implementation Details

For all the experiments, we set the learning rate of the encoder model as 1e-5, set the learning rate of the two-layer MLP classifier as 1e-3, and tried different batch sizes within 8, 16, and 32 to choose the best performance. For the hyper-parameters in Dirichlet distribution and Beta distribution, we set  $\tau$  as 1.0 and set  $\alpha$  as 0.75. The coefficient  $\gamma$  of JSD loss was set to be 8, and the max number of training epochs was set to be 20. All these hyper-parameters are shared among the models. All the experiments are performed multiple times across different seeds on a single NVIDIA RTX 8000 GPU.

<sup>5</sup><https://github.com/aatlantise/syntactic-augmentation-nli>

<sup>6</sup><https://github.com/hongyi-zhang/mixup>

<sup>7</sup><https://github.com/GT-SALT/MixText>

<sup>8</sup><https://github.com/clovaai/ssmix>