OoMMix: Out-of-manifold Regularization in Contextual Embedding Space for Text Classification

Seonghyeon Lee[1], Dongha Lee[2] and Hwanjo Yu[1]*
[1]Dept. of Computer Science and Engineering, POSTECH, Republic of Korea
[2]Institute of Artificial Intelligence, POSTECH, Republic of Korea
{sh0416,dongha.lee,hwanjoyu}@postech.ac.kr

Abstract

Recent studies on neural networks with pre-trained weights (i.e., BERT) have mainly focused on a low-dimensional subspace, where the embedding vectors computed from input words (or their contexts) are located. In this work, we propose a new approach, called OoMMix, to finding and regularizing the remainder of the space, referred to as out-of-manifold, which cannot be accessed through the words. Specifically, we synthesize the out-of-manifold embeddings based on two embeddings obtained from actually-observed words, to utilize them for fine-tuning the network. A discriminator is trained to detect whether an input embedding is located inside the manifold or not, and simultaneously, a generator is optimized to produce new embeddings that can be easily identified as out-of-manifold by the discriminator. These two modules successfully collaborate in a unified and end-to-end manner for regularizing the out-of-manifold. Our extensive evaluation on various text classification benchmarks demonstrates the effectiveness of our approach, as well as its good compatibility with existing data augmentation techniques which aim to enhance the manifold.

1 Introduction

Neural networks with a word embedding table have been the most popular approach to a wide range of NLP applications. The great success of transformer-based contextual embeddings as well as masked language models (Devlin et al., 2019; Liu et al., 2019b; Raffel et al., 2020) makes it possible to exploit the pre-trained weights, fully optimized by using large-scale corpora, and it brought a major breakthrough to many problems. For this reason, most recent work on text classification has achieved state-of-the-art performances by fine-tuning the network initialized with the pre-trained weight (Devlin et al., 2019). However, they suffer from extreme over-parameterization due to the large pre-trained weight, which allows them to be easily overfitted to its relatively small training data.

Along with outstanding performances of the pre-trained weight, researchers have tried to reveal the underlying structure encoded in its embedding space (Rogers et al., 2021). One of the important findings is that the contextual embeddings computed from words usually form a low-dimensional manifold (Ethayarajh, 2019). In particular, a quantitative analysis on the space (Cai et al., 2021), which measured the effective dimension size of BERT after applying PCA on its contextual embedding vectors, showed that 33% of dimensions covers 80% of the variance. In other words, only the low-dimensional subspace is utilized for fine-tuning BERT, although a high-dimensional space (i.e., model weights with a high capacity) is provided for training. Based on this finding on contextual embedding space, we aim to regularize the contextual embedding space for addressing the problem of over-parameterization, while focusing on the outside of the manifold (i.e., out-of-manifold) that cannot be accessed through the words.

In this work, we propose a novel approach to discovering and leveraging the out-of-manifold for contextual embedding regularization. The key idea of our out-of-manifold regularization is to produce the embeddings that are located outside the manifold and utilize them to fine-tune the network for a target task. To effectively interact with the contextual embedding of BERT, we adopt two additional modules, named as embedding generator and manifold discriminator. Specifically, 1) the generator synthesizes the out-of-manifold embeddings by linearly interpolating two input embeddings computed from actually-observed words, and 2) the discriminator identifies whether an input embedding comes from the generator (i.e., the synthesized embed-

* Corresponding author

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 590–599
August 1–6, 2021.

ding) or the sequence of words (i.e., the actual embedding). The joint optimization encourages the generator to output the out-of-manifold embeddings that can be easily distinguished from the actual embeddings by the discriminator, and the discriminator to learn the decision boundary between the in-manifold and out-of-manifold embeddings. In the end, the fine-tuning on the synthesized out-of-manifold embeddings tightly regularizes the contextual embedding space of BERT.

The experimental results on several text classification benchmarks validate the effectiveness of our approach. In particular, our approach using a parameterized generator significantly outperforms the state-of-the-art mixup approach whose mixing strategy needs to be manually given by a programmer. Furthermore, our approach shows good compatibility with various data augmentation techniques, since the target space we focus on for regularization (i.e., out-of-manifold) does not overlap with the space the data augmentation techniques have paid attention to (i.e., in-manifold). The in-depth analyses on our modules provide an insight into how the out-of-manifold regularization manipulates the contextual embedding space of BERT.

2 Related Work

In this section, we briefly review two approaches to regularizing over-parameterized network based on auxiliary tasks and auxiliary data.

2.1 Regularization using Auxiliary Tasks

Regularization is an essential tool for good generalization capability of neural networks. One representative regularization approach relies on designing auxiliary tasks. Liu et al. (2019a) firstly showed promising results by unifying a bunch of heterogeneous tasks and training a single unified model for all the tasks. In particular, the synthesized task that encodes desirable features or removes undesirable features turns out to be helpful for network regularization. Devlin et al. (2019) introduced the task which restores masked sentences, termed as masked language model, to encode the distributional semantic in the network; this considerably boosts the overall performance of NLP applications. In addition, Clark et al. (2020) regularized the network by discriminating generated tokens from a language model, and Gong et al. (2018) utilized an additional discriminator to remove the information about word frequency implicitly encoded in the word embeddings.

2.2 Regularization using Auxiliary Data

Another approach to network regularization is to take advantage of auxiliary data, mainly obtained by data augmentation, which eventually supplements the input data space. Inspired by (Bengio et al., 2011) that additionally trained the network with noised (i.e., augmented) images in computer vision, Wei and Zou (2019) simply augmented sentences by adding a small perturbation to the original sentences, such as adding, deleting, and swapping words within the sentences. Recent work tried to further exploit the knowledge from a pre-trained model for augmenting the sentences: sentence back translation by using a pre-trained translation model (Xie et al., 2019), and masked sentence reconstruction by using a pre-trained masked language model (Ng et al., 2020).

Mixup (Zhang et al., 2018) is also a kind of data augmentation but differs in that it performs linear interpolation on multiple input sentences and their corresponding labels. Verma et al. (2019) validated that mixup in the hidden space (instead of the input space) is also effective for regularization, and Guo et al. (2019b) found that mixup of images can regularize the out-of-manifold in image representations. In the case of NLP domain, Guo et al. (2019a) and Guo (2020) firstly adopted mixup to text data for text classification, using the traditional networks such as CNN and LSTM; they sample their mixing coefficients from the beta distribution at the sentence-level and at the word-level, respectively. To fully utilize the contextual embedding of transformer-based networks, Chen et al. (2020) applied mixup in the word-level contextual embedding space using a pre-trained language model (i.e., BERT), whereas Sun et al. (2020) focused on mixup in the sentence-level embedding space specifically for improving GLUE score.

3 Method

In this section, we propose a novel mixup approach, termed as OoMMix, to regularize the out-of-manifold in contextual embedding space for text classification. We first briefly remind the architecture of BERT, then introduce two modules used for out-of-manifold regularization, which are embedding generator and manifold discriminator.

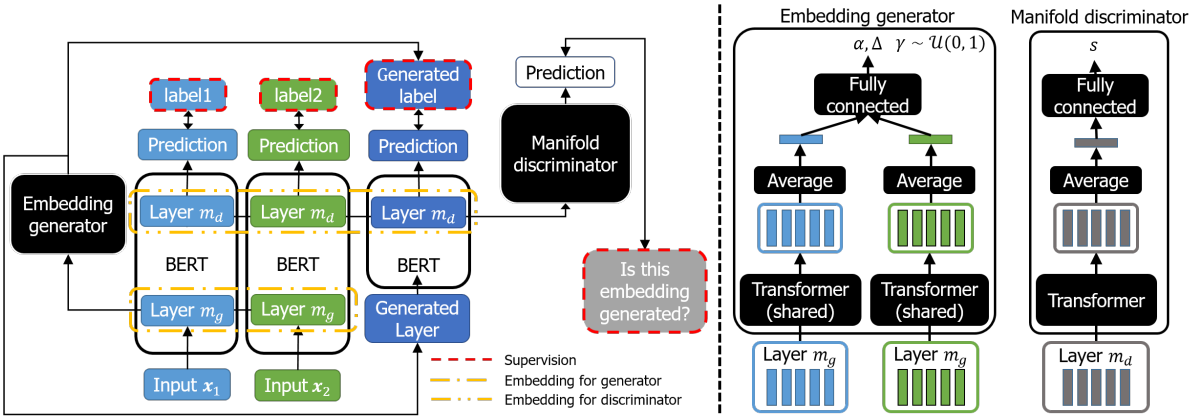

Figure 1: The overview of OoMMix for fine-tuning BERT (Left) and the structure of our embedding generator and manifold discriminator (Right).

3.1 Preliminary

BERT is a stack of M transformer encoders pre-trained on the objective of the masked language model (Devlin et al., 2019). First, a raw sentence is split into the sequence of tokens $\mathbf{x} \in \{0, ..., |V|\}^L$ using a tokenizer with the vocabulary V, where L is the sequence length. Each token is mapped into a D-dimensional vector based on the embedding table. The sequence of embedding vectors $\mathbf{h}^{(0)} \in \mathbb{R}^{L \times D}$ is transformed into the m-th contextual embedding $\mathbf{h}^{(m)} \in \mathbb{R}^{L \times D}$ by m transformer layers (Vaswani et al., 2017).

We fine-tune the pre-trained weight to classify input texts into C classes. A classifier produces the classification probability vector $o \in \mathbb{R}^C$ using the last contextual embedding $\mathbf{h}^{(M)}$. Then, the optimization problem is defined based on a labeled dataset $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$.

$$\underset{w_f}{\text{minimize}} \underset{(\mathbf{x},y)\in D}{\mathbb{E}} \left[\mathcal{L}_C \left(\mathbf{x}, y \right) \right]$$
$$\mathcal{L}_C \left(\mathbf{x}, y \right) := \mathcal{L}_{kl} \left(f \left(\mathbf{x} \right), \mathbf{e}_y \right)$$

where $\mathcal{L}_{kl}$ is the Kullback-Leibler divergence and $\mathbf{e}_y \in \mathbb{R}^C$ is a one-hot vector representing the label y. The function f is the whole process from $\mathbf{h}^{(0)}$ to o, called a target model, and w_f is the trainable parameters for the function f, including the pre-trained weight of BERT and the parameters in the classifier. For notation, f can be split into several sub-processes $f(\mathbf{x}) = (f_{m'} \circ h_m^{m'} \circ h_0^m)(\mathbf{x})$ where $h_m^{m'}(\mathbf{x})$ maps the m-th contextual embedding into the m'-th contextual embedding through the layers.

3.2 Embedding Generator

The goal of our generator network G is to synthesize an artificial contextual embedding by taking two contextual embeddings (obtained from layer m_g) as its input. We use linear interpolation so that the new embedding belongs to the line segment defined by the two input embeddings. Since we limit the search space, the generator produces a single scalar value $\lambda \in [0, 1]$, called a mixing coefficient.

$$G\left(\mathbf{h}_1^{(m_g)}, \mathbf{h}_2^{(m_g)}\right) = \lambda \cdot \mathbf{h}_1^{(m_g)} + (1 - \lambda) \cdot \mathbf{h}_2^{(m_g)}$$
$$\lambda = g\left(\mathbf{h}_1^{(m_g)}, \mathbf{h}_2^{(m_g)}\right)$$

We introduce the distribution of the mixing coefficient to model its uncertainty. To this end, our generator network produces the lower bound α and the interval Δ by using $\mathbf{h}_1^{(m_g)}$ and $\mathbf{h}_2^{(m_g)}$, so as to sample the mixing coefficient from the uniform distribution $\mathcal{U}(\alpha, \alpha + \Delta)$.

To avoid massive computational overhead incurred by the concatenation of two input sequences (Reimers and Gurevych, 2019), we adopt the Siamese architecture that uses the shared weights on two different inputs. The generator first transforms each sequence of contextual embedding vectors by using a single transformer layer, then obtains the sentence-level embedding by averaging all the embedding vectors in the sequence. From the two sentence-level embeddings $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{R}^D$, the generator obtains the concatenated embedding $\mathbf{s} = \mathbf{s}_1 \oplus \mathbf{s}_2 \in \mathbb{R}^{2D}$ and calculates α and Δ by using a two-layer fully-connected network with the softmax normalization. Specifically, the last fully-connected layer outputs a normalized 3-dimensional vector, whose first and second values become α and Δ, thereby the range of sampling distribution $(\alpha, \alpha + \Delta)$ lies in $[0, 1]$. In this work, we consider the structure of the generator to efficiently process the sequential input,

but any other structures focusing on different aspects (e.g. the network that enlarges the search space) can be used as well. For effective optimization of λ sampled from $\mathcal{U}(\alpha, \alpha+\Delta)$, we apply the re-parameterization trick which decouples the sampling process from the computational graph (Kingma and Welling, 2014). That is, we compute the mixing coefficient by using $\gamma \sim \mathcal{U}(0,1)$.

$$\lambda = \alpha + \gamma \times \Delta$$

The optimization problem for text classification can be extended to the new embeddings and their labels, provided by the generator network.

$$\underset{w_{f_{m_g}}, w_g}{\text{minimize}} \underset{(\mathbf{x}_1, y_1) \in D}{\mathbb{E}} \Big[\mathcal{L}_G(\mathbf{x}_1, y_1) \Big] \quad (1)$$

$$\mathcal{L}_G(\mathbf{x}_1, y_1) := \underset{(\mathbf{x}_2, y_2) \in \mathcal{D}}{\mathbb{E}} \Big[\mathcal{L}_{kl}(f_{m_g}(\tilde{\mathbf{h}}), \tilde{\mathbf{y}}) \Big]$$

$$\lambda \sim g\left(h_0^{m_g}(\mathbf{x}_1), h_0^{m_g}(\mathbf{x}_2)\right)$$
$$\tilde{\mathbf{h}} := \lambda \cdot h_0^{m_g}(\mathbf{x}_1) + (1-\lambda) \cdot h_0^{m_g}(\mathbf{x}_2)$$
$$\tilde{\mathbf{y}} := \lambda \cdot \mathbf{e}_{y_1} + (1-\lambda) \cdot \mathbf{e}_{y_2}$$

where $w_{f_{m_g}}$ is the trainable parameters of the function f_{m_g} (i.e., the process from $\mathbf{h}^{(m_g)}$ to o), and w_G is the ones for the generator. Similar to other mixup techniques, we impose the mixed label on the generated embedding.

3.3 Manifold Discriminator

We found that the supervision from the objective (1) is not enough to train the generator. The objective optimizes the generator to produce the embeddings that are helpful for the target classification. However, since the over-parameterized network tends to memorize all training data, the target model also simply memorizes the original data to minimize Equation (1). In this situation, the generator is more likely to mimic the embeddings seen in the training set (memorized by the target model) rather than generate novel embeddings. For this reason, we need more useful supervision for the generator, to make it output the out-of-manifold embeddings.

To tackle this challenge, we define an additional task that identifies whether a contextual embedding comes from the generator or actual words. The purpose of this task is to learn the discriminative features between actual embeddings and generated embeddings, in order that we can easily discover the subspace which cannot be accessed through the actually-observed words. For this task, we introduce a discriminator network D that serves as a binary classifier in the contextual embedding space of the m_d-th transformer layer.

The discriminator takes a contextual embedding $\mathbf{h}^{(m_d)}$ and calculates the score $s \in [0,1]$ which indicates the probability that $\mathbf{h}^{(m_d)}$ comes from an actual sentence (i.e., $\mathbf{h}^{(m_d)}$ is located inside the manifold). Its network structure is similar to that of the generator, except that the concatenation is not needed and the output of the two-layer fully connected network produces a single scalar value. As discussed in Section 3.2, any network structures for focusing on different aspects can be employed.

The optimization of the generator and discriminator for this task is described as follows.

$$\underset{w_g, w_d}{\text{minimize}} \underset{(\mathbf{x}_1, y_1) \in D}{\mathbb{E}} \Big[\mathcal{L}_D(\mathbf{x}_1) \Big] \quad (2)$$

$$\mathcal{L}_D(\mathbf{x}_1) := \underset{(\mathbf{x}_2, y_2) \in D}{\mathbb{E}} \Big[\mathcal{L}_{bce}(D(h_{m_g}^{m_d}(\tilde{\mathbf{h}})), 0) + \mathcal{L}_{bce}\left(D\left(h_0^{m_d}(\mathbf{x})\right), 1\right) \Big]$$

where $\mathcal{L}_{bce}$ is the binary cross entropy loss. By minimizing this objective, our generator can produce the out-of-manifold embeddings that are clearly distinguished from the actual (in-manifold) contextual embeddings by the discriminator.

3.4 Training

We jointly optimize the two objectives to train the embedding generator. Equation (1) encourages the generator to produce the embeddings which are helpful for the target task, while Equation (2) makes the generator produce the new embeddings different from the contextual embeddings obtained from the words. The final objective is defined by

$$\underset{(\mathbf{x}, y) \sim D}{\mathbb{E}} \left[\mathcal{L}_C(\mathbf{x}, y) + \mathcal{L}_G(\mathbf{x}, y) + e\mathcal{L}_D(\mathbf{x}) \right]$$

where e regulates the two objectives. The generator and discriminator collaboratively search out informative out-of-manifold embeddings for the target task while being optimized with the target model, thereby the generated embeddings can effectively regularize the out-of-manifold.

4 Experiments

In this section, we present the experimental results supporting the superiority of OoMMix among the recent mixup approaches in text classification. Also, we investigate its compatibility with other data augmentation techniques. Finally, we provide in-depth analyses on our approach to further validate the effect of out-of-manifold regularization.

Dataset	Input sentence	Class	Valid size	Valid length	Test size	Test length
AG News	content	4	7.6K	43.49	7.6K	43.21
Amazon Review	review text	2	8K	95.94	400K	95.62
Yahoo Answer	title, question, answer	10	50K	109.81	60K	110.74
DBpedia	content	14	28K	63.62	70K	63.61

Table 1: Statistics of datasets

Dataset	Train	Original	NonlinearMix	*mixup*-transformer	TMix	OoMMix	TMix†	MixText†
AG News	0.5K	88.22 ± 0.02	88.24 ± 0.05	$\mathbf{88.58 \pm 0.02}$	88.45 ± 0.02	88.41 ± 0.05	-	-
	2.5K	89.92 ± 0.15	88.75 ± 0.36	89.62 ± 0.09	90.07 ± 0.09	$\mathbf{90.25 \pm 0.05}^{*}$	-	-
	10K	91.50 ± 0.05	88.86 ± 0.12	91.37 ± 0.21	91.51 ± 0.08	$\mathbf{91.83 \pm 0.09}^{**}$	91.0	91.5 (+20K)
Amazon Review	0.5K	89.17 ± 0.35	89.02 ± 0.21	89.31 ± 0.14	89.57 ± 0.02	$\mathbf{89.66 \pm 0.01}$	-	-
	2.5K	90.96 ± 0.05	91.04 ± 0.11	90.70 ± 0.05	91.24 ± 0.13	$\mathbf{91.28 \pm 0.12}$	-	-
	10K	92.81 ± 0.05	91.15 ± 0.42	92.12 ± 0.28	92.79 ± 0.07	$\mathbf{92.94 \pm 0.06}$	-	-
Yahoo Answer	0.5K	67.24 ± 0.07	67.56 ± 0.37	67.62 ± 0.06	67.57 ± 0.11	$\mathbf{67.95 \pm 0.16}$	-	-
	2K	70.41 ± 0.04	69.17 ± 0.11	70.29 ± 0.14	70.68 ± 0.15	$\mathbf{71.08 \pm 0.10}^{*}$	69.8	71.3 (+50K)
	25K	73.68 ± 0.03	69.31 ± 0.37	73.52 ± 0.05	73.84 ± 0.00	$\mathbf{74.13 \pm 0.06}^{*}$	73.5	74.1 (+50K)
DBpedia	0.5K	97.86 ± 0.07	97.50 ± 0.25	98.06 ± 0.05	98.15 ± 0.10	$\mathbf{98.26 \pm 0.04}$	-	-
	2.8K	$\mathbf{98.83 \pm 0.03}$	98.74 ± 0.09	98.76 ± 0.01	98.82 ± 0.04	$\mathbf{98.83 \pm 0.05}$	98.7	98.9 (+70K)
	35K	98.96 ± 0.07	98.89 ± 0.01	98.91 ± 0.03	98.97 ± 0.03	$\mathbf{99.03 \pm 0.03}^{*}$	99.0	99.2 (+70K)

Table 2: Classification accuracy on sentence classification benchmarks. * and ** respectively indicate $p \leq 0.05$ and $p \leq 0.01$ for the paired t-test of OoMMix vs. the best competitor. TMix† and MixText† report the scores presented in (Chen et al., 2020), where the sizes of domain-related unlabeled data are described in the parenthesis.

4.1 Experimental setup

Our experiments consider 4 sentence classification benchmarks (Zhang et al., 2015) of various scales. The statistics of the datasets are summarized in Table 1. We follow the experimental setup used in (Chen et al., 2020) to directly compare the results with ours. Specifically, we split the whole training set into training/validation sets, while leaving out the official test set for evaluation. We choose the classification accuracy as the evaluation metric, considering the datasets are already class-balanced. For the various sizes of training set from 0.5K to 35K, we apply stratified sampling to preserve the balanced class distributions.

In terms of optimization, we use BERT provided by huggingface for the classification tasks.[1] The Adam optimizer is used to fine-tune BERT with the linear warm-up for the first 1000 iterations, and the initial learning rates for the pre-trained weight and the target classifier are set to 2e-5 and 1e-3, respectively. We set the batch size to 12 and the dropout probability to 0.1. We attach the generator and discriminator at the third layer ($m_g = 3$) and the last layer ($m_d = 12$), respectively. The two objectives equally contribute to training the generator, $e = 1$, but we increase the e value if the discriminator fails to discriminate the embeddings. The accuracy is evaluated on validation set every 200 iterations, and stop training when the accuracy does not increase for 10 consecutive evaluations. We report the classification accuracy on the test set at the best validation checkpoint and repeat the experiment three times with different random seeds to report the average with its standard deviation. We implement the code using PyTorch and use NVIDIA Titan Xp for parallel computation. In our environment, the training spends about 30 minutes to 3 hours depending on the dataset.

[1] In our experiments, we use the checkpoint `bert-base-uncased` as the pre-trained weight.

4.2 Comparision with Mixup Approaches

We compare OoMMix with existing mixup techniques. All the existing methods manually set the mixing coefficient, whereas we parameterize the linear interpolation by the embedding generator, optimized to produce out-of-manifold embeddings.

- **NonlinearMix** (Guo, 2020) samples mixing coefficients for each word from the beta distribution, while using neural networks to produce the mixing coefficient for the label. We apply this approach to BERT.
- ***mixup*-transformer** (Sun et al., 2020) linearly interpolates the sentence-level embedding with a fixed mixing coefficient. The mixing coefficient is 0.5 as the paper suggested.

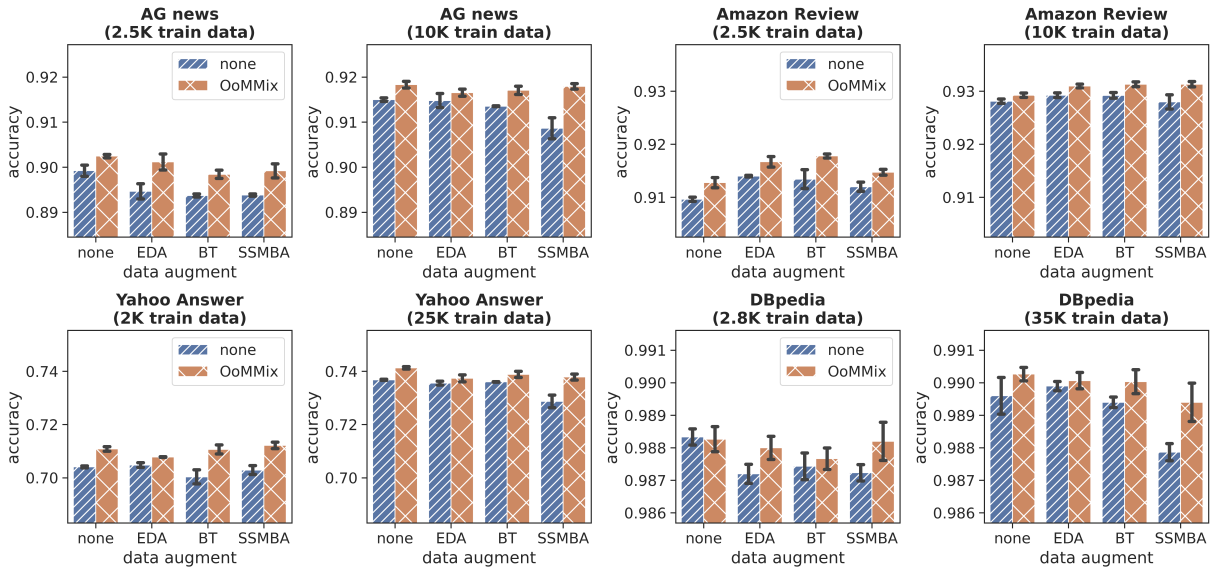

Figure 2: Average classification accuracy and their standard deviation when OoMMix is applied with various data augmentation techniques.

- **TMix** (Chen et al., 2020) performs linear interpolation on the word-level contextual embedding space and samples a mixing coefficient from the beta distribution. We select the best accuracy among different alpha configurations $\{0.05, 0.1\}$ for the beta distribution.
- **MixText** (Chen et al., 2020) additionally utilizes unlabeled data by combining TMix with its pseudo-labeling technique.

Table 2 reports the accuracy on various sentence classification benchmarks. In most cases, OoMMix achieves the best performance among all the competing mixup approaches. In the case of NonlinearMix, it sometimes shows worse performance than the baseline (i.e., fine-tuning only on original data), because its mixup strategy introduces a large degree of freedom in the search space, which loses useful semantic encoded in the pre-trained weight. The state-of-the-art mixup approaches, TMix and *mixup*-transformer, slightly improves the accuracy over the baseline, while showing the effectiveness of the mixup approach. Finally, OoMMix beats all the previous mixup approaches, which strongly indicates that the embeddings mixed by the generator are more effective for regularization, compared to the embeddings manually mixed by the existing approaches. It is worth noting that OoMMix obtains a comparable performance to MixText, even without utilizing additional unlabeled data. In conclusion, discovering the out-of-manifold and applying mixup for such subspace are beneficial in contextual embedding space.

4.3 Compatibility with Data Augmentations

To demonstrate that the regularization effect of OoMMix does not conflict with that of existing data augmentation techniques, we investigate the performance of BERT that adopts both OoMMix and other data augmentations together. Using three popular data augmentation approaches in the NLP community, we replicate the dataset as large as the original one to use them for fine-tuning.

- **EDA** (Wei and Zou, 2019) is a simple augmentation approach that randomly inserts/deletes words or swaps two words in a sentence. We used the official codes[2] with the default insertion/deletion/swap ratio the author provided.
- **BT** (Xie et al., 2019) uses the back-translation for data augmentation. A sentence is translated into another language, then translated back into the original one. We use the code implemented in the MixText repository[3] with the checkpoint fairseq provided.[4]
- **SSMBA** (Ng et al., 2020) makes use of the pre-trained masked language model. They mask the original sentence and reconstruct it by filling in the masked portion. We use the codes provided by the authors[5] with default

[2] https://github.com/jasonwei20/eda_nlp
[3] https://github.com/GT-SALT/MixText
[4] `transformer.wmt19.{en-ru,ru-en}.single_model` are provided through the official torch hub.
[5] https://github.com/nng555/ssmba

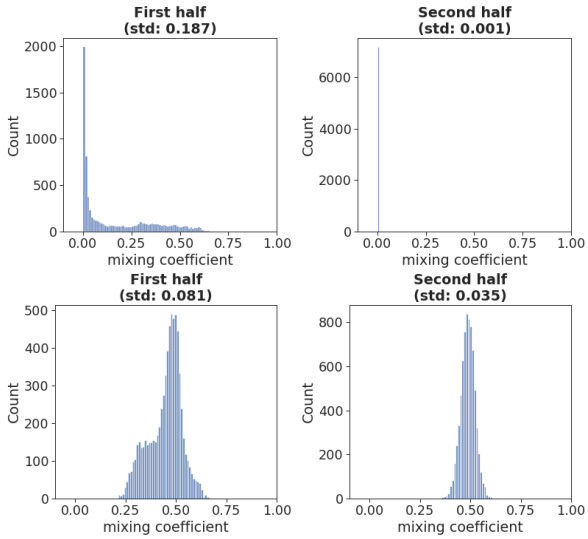

Figure 3: Count of mixing coefficients without the discriminator (Upper) and with the discriminator (Lower).

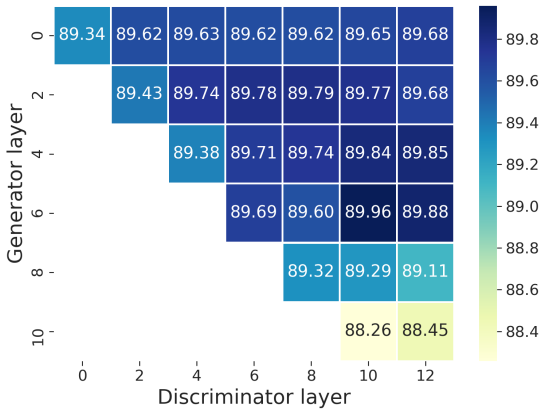

Figure 4: Performance changes with respect to different layers for the generator and discriminator. Dataset: Amazon Review 0.5K, Layer 0: word embedding.

masked proportion and the pre-trained weight.

Figure 2 shows the effectiveness of OoMMix when being used with the data augmentation techniques. For all the cases, OoMMix shows consistent improvement. Especially for the Amazon Review dataset, the data augmentation and our mixup strategy independently bring the improvement of the accuracy, because the subspaces targeted by the data augmentation and OoMMix do not overlap with each other. That is, OoMMix finds out out-of-manifold embedding, which cannot be generated from the actual sentences, whereas the data augmentations (i.e., EDA, BT, and SSMBA) focus on augmenting the sentences whose embeddings are located inside the manifold. Therefore, jointly applying the two techniques allows to tightly regularize the contextual embedding space, including both in-manifold and out-of-manifold.

Moreover, OoMMix has additional advantages over the data augmentations. First, OoMMix is still effective in the case that large training data are available. The data augmentation techniques result in less performance gain as the size of training data becomes larger, because there is less room for enhancing the manifold constructed by enough training data. Second, the class label of the augmented sentences given by the data augmentation techniques (i.e., the same label with the original sentences) can be noisy for sentence classification, compared to the label of out-of-manifold embeddings generated by OoMMix. This is because the assumption that the augmented sentences have the same label with their original sentences is not always valid. On the contrary, there do not exist actual (or ground truth) labels for out-of-manifold embeddings, as they do not correspond to actual sentences; this allows our mixup label to be less noisy for text classification.

4.4 Effect of the Manifold Discriminator

We also investigate how the manifold discriminator affects the training of the embedding generator. Precisely, we compare the distributions of mixing coefficients, obtained from two different generators; they are optimized with/without the manifold discriminator, respectively (Figure 3 Upper/Lower). We partition the training process into two phases (i.e., the first and second half), and plot a histogram of the mixing coefficients in each phase.

The embedding generator without the discriminator gradually moves the distribution of the mixing coefficients toward zero, which means that the generated embedding becomes similar to the actual embedding. Therefore, training the generator without the discriminator fails to produce novel embeddings, which cannot be seen in the original data. In contrast, in the case of the generator with the discriminator, most of the mixing coefficients are located around 0.5, which implies that the generator produces the embeddings which are far from both the two actual embeddings to some extent. We also observe that the average objective value for our discrimination task (Equation (2)) is 0.208 for the last 20 mini-batches; this is much lower than 0.693 at the initial point. It indicates that the generated embeddings are quite clearly distinguished from the ones computed from actual sentences.

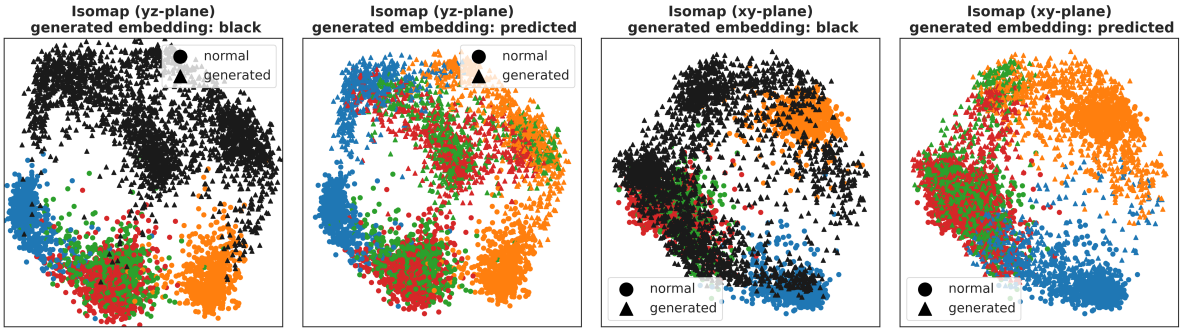

Figure 5: Isomap visualization of the sentence-level embeddings. The embedding vectors are projected into 3-dimensional space and rendered in two different views (xy, and yz-plane). For each view, we colorize the out-of-manifold embeddings with black and their predicted class.

4.5 Effect of Different Embedding Layers

We further examine the effect of the location of our generator and discriminator (i.e., m_g and m_d) on the final classification performance. Figure 4 illustrates the changes of the classification accuracy with respect to the target contextual embedding layers the modules are attached to. To sum up, BERT achieves high accuracy when the generator is attached to the contextual embedding lower than the sixth layer while the discriminator works for a higher layer. It makes our out-of-manifold regularization affect more parameters in overall layers, which eventually leads to higher accuracy. On the other hand, in case that we use both the generator and discriminator in the same layer, the gradient of the loss for manifold discrimination cannot guide the generator to output out-of-manifold embeddings, and as a result, the generator is not able to generate useful embeddings.

4.6 Manifold Visualization

Finally, we visualize our contextual embedding space to qualitatively show that OoMMix discovers and leverages the space outside the manifold for regularization. We apply Isomap (Tenenbaum et al., 2000), a neighborhood-based kernel PCA for dimensionality reduction, to both the actual sentence embeddings and generated embeddings. We simply use the Isomap function provided by scikit-learn, and set the number of the neighbors to 15. Figure 5 shows the yz-plane and xy-plane of our embedding space, whose dimensionality is reduced to 3 (i.e., x, y, and z). We use different colors to represent the class of the actual embeddings as well as the predicted class of the generated embeddings.

In the yz-plane, the actual sentence embeddings form multiple clusters, optimized for the text classification task. At the same time, the generated embeddings are located in the different region from the space enclosing most of the actual embeddings. In the second plot, we colorize the generated embeddings with their predicted class. The predicted class of out-of-manifold embeddings are well-aligned with that of the actual embeddings, which means that OoMMix imposes the classification capability on the out-of-manifold region as well. We change the camera view to xy-plane and repeat the same process to show the alignment of class distribution clearly (in the third/fourth plots). By imposing the classification capability on the extended dimension/subspace (i.e., out-of-manifold), OoMMix significantly improves the classification performance for the original dimension/subspace (i.e., in-manifold).

5 Conclusion

This paper proposes OoMMix to regularize out-of-manifold in the contextual embedding space. Our main motivation is that the embeddings computed from the words only utilize a low-dimensional manifold while a high-dimensional space is available for the model capacity. Therefore, OoMMix discovers the embeddings that are useful for the target task but cannot be accessed through the words. With the help of the manifold discriminator, the embedding generator successfully produces out-of-manifold embeddings with their labels. We demonstrate the effectiveness of OoMMix and its compatibility with the existing data augmentation techniques.

Our approach is a bit counter-intuitive in that the embeddings that cannot be accessed through the actual words are helpful for the target model. As the discrete features from texts (i.e., words), embedded into the high-dimensional continuous space where

their contexts are encoded, cannot cover the whole space, the uncovered space also should be carefully considered for any target tasks. In this sense, we need to regularize the out-of-manifold to prevent anomalous behavior in that space, which is especially important for a large pre-trained contextual embedding space.

Acknowledgments

This work was supported by the NRF grant funded by the MSIT (No. 2020R1A2B5B03097210), and the IITP grant funded by the MSIT (No. 2018-0-00584, 2019-0-01906).

References

Yoshua Bengio, Frédéric Bastien, Arnaud Bergeron, Nicolas Boulanger–Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, François Savard, and Guillaume Sicard. 2011. Deep learners benefit more from out-of-distribution examples. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 164–172, Fort Lauderdale, FL, USA. JMLR Workshop and Conference Proceedings.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: 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, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In *Advances in Neural Information Processing Systems*, volume 31, pages 1334–1345. Curran Associates, Inc.

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

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

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, 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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 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 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pages 649–657. Curran Associates, Inc.

A Preprocessing decisions, model parameters and other details

We list the minor implementation details but useful for reproducing our experiments.

- The train/validation split is implemented using `train_test_split` function in scikit-learn with seed 42.
- The `bert-base-uncased` tokenizer provided by huggingface is used to split the sentence.
- We take the first 256 tokens for the sentence which length is longer than 256.
- The embedding table in the pre-trained weight is frozen for all experiments.
- The data cleaning process in EDA deteriorates the performance, so we omit that process.
- Due to the different optimization variables for the two objectives, we perform the backward process twice and update the parameter.

B Hyper-parameter search

Since performing grid search on all datasets is intolerable due to the lack of computational resources, we perform different configurations on one small dataset. The candidate for the embedding layer for the generator (m_g) is $[0, 2, 4, 6, 8, 10]$ and the candidate for the embedding layer for the discriminator (m_d) is $[0, 2, 4, 6, 8, 10, 12]$. For the case the discriminator could not be trained well, e.g. the discriminator loss does not decrease at all, we increase e to give more weight to the discriminator loss. For all the experiments, we fix the m_g and m_d and manually change the e to make the discriminator classify the embedding. The hyper-parameter choices are summarized in Table 3.

Hyper-parameter	Value
Embedding space for the generator (m_g)	3
Embedding space for the discriminator (m_d)	12
Coefficient for two objectives (e)	1 (or 1.5)

Table 3: Hyper-parameter configuration for the experiment