DropMix: A Textual Data Augmentation Combining Dropout with Mixup

Fanshuang Kong[1], **Richong Zhang**[1,2]*, **Xiaohui Guo**[3], **Samuel Mensah**[4], **Yongyi Mao**[5]
[1]SKLSDE, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]Hangzhou Innovation Institute, Beihang University, Hangzhou, China
[4]Department of Computer Science, University of Sheffield, UK
[5]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
{kongfs,zhangrc,guoxh}@act.buaa.edu.cn
s.mensah@sheffield.ac.uk, ymao@uottawa.ca

Abstract

Overfitting is a common problem when there is insufficient data to train deep neural networks in machine learning tasks. Data augmentation regularization methods such as Dropout, Mixup, and their enhanced variants, are effective and prevalent, and achieve promising performance to overcome overfitting. However, in text learning, most of the existing regularization approaches merely adopt ideas from computer vision without considering the importance of dimensionality in natural language processing. In this paper, we argue that the property is essential to overcome overfitting in text learning. Accordingly, we present a saliency map informed textual data augmentation and regularization framework, which combines Dropout and Mixup, namely DropMix, to mitigate the overfitting problem in text learning. In addition, we design a procedure that drops and patches fine grained shapes of the saliency map under the DropMix framework to enhance regularization. Empirical studies confirm the effectiveness of the proposed approach on 12 text classification tasks.

1 Introduction

Deep neural networks have shown its effectiveness, achieving state-of-the-art result in many natural language processing (NLP) tasks. However, these models are likely to overfit training sets with few samples due to the large number of parameters they contain. Regularization techniques, such as weighted decay (Krogh and Hertz, 1992), dropout (Srivastava et al., 2014), and data augmentation (Goodfellow et al., 2015; Zhang et al., 2017) are powerful to overcome overfitting by making slight modifications on the neural network or data space to improve generlization.

Data augmentation methods have shown to be useful for overcoming overfitting by increasing the training dataset size via adding augmented versions of already existing training data. For example, Goodfellow et al. (2015) adds the worst case adversarial sample into the training dataset; Mixup (Zhang et al., 2017) introduces new samples by the linear interpolation of samples and their corresponding labels. The majority of these methods are however designed for computer vision tasks, requiring adaptation for text learning. However, due to the discreteness and high-dimensional nature of text data spaces, adapting these methods is non-trivial.

Among the proposed data augmentation methods, Mixup's linearity inductive bias is succinct and effective since it reduces the discreteness in-between samples or the space in which new samples are generated. Thus, motivating researchers to adapt this method for text learning (Verma et al., 2019; Chen et al., 2020; Zhang et al., 2020; Yoon et al., 2021). The current state-of-the-art SSMix (Yoon et al., 2021) which builds upon Mixup, generates new samples by performing a saliency-based span mixup on the input text rather than its embedding vectors as like previous approaches (Verma et al., 2019; Guo et al., 2019). As shown in a text sentiment classification task (Table 1), SSMix replaces the least salient span in X_A with the

X_A	Food in this restaurant is delicious	P
X_B	The environment is bad	N
X_C	Food in this restaurant bad delicious	0.83P

Table 1: Illustration of an example X_C generated by SSMix. The darker the red (or blue) shade is, the more salient the words in the positive (or negative) sentiment decision. SSMix uses a Saliency Map (Simonyan et al., 2013) to measure the saliency of words. SSMix substitute the least salient word "is" in X_A with the most salient "bad" in X_B. The mixup ratio $\lambda = 0.17$ as the length of the salient words (i.e., "bad") is one out of six words in X_C. Thus, the generated label is 17% negative or 83% positive.

*Corresponding author

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 890–899
December 7-11, 2022

most salient span in X_B to generate the new sample X_C and uses a mixup ratio λ to determine the label X_C.

Unfortunately, three problems concurrently occur in SSMix which limits its ability to generate expressive examples. Firstly, although SSMix has a strong flexibility by exploring a larger synthetic space to generate new samples, this space is in fact high-dimensional as the input data consists of words rather than embeddings. Besides, words within this space is disordered and therefore may lead to unpredictable sampling results. This leads to the second and third problems, where there is a potential of generating *"dirty"* examples or causing a *label drift* (i.e., newly added sample is categorized under a wrong class), as it does not consider the position of words or the sentence syntax structure in the mixup. For instance, X_C can be said to be dirty. Besides being wrong grammatically, the generated example contains contrasting opinion words (i.e., "bad","delicious") offering confusing clues to determine its label.

To tackle the aforementioned issues, we propose a patch-aware DropMix framework, a data augmentation method which takes advantage of Dropout and Mixup, to generate new examples based on the dimension and saliency of words in texts. In contrast to SSMix, DropMix maps the high-dimensional input data (i.e., words) into a low-dimensional space to encode the semantics of the text input and mitigate the unpredictability of generated examples. Despite the problems encountered by SSMix, empirical results indicate that sampling from a large synthetic space may have some benefits. This may be intuitively desirable, as the model is not constrained to generate examples from only the input text representations. To compensate for this, DropMix applies Dropout on the embeddings of X_A (and X_B) and replaces the dropped out units with noise from X_B (and X_A) to reduce the over-dependence on the input features. However, since the saliency of words in texts impacts the class decision, the activation neurons in the features to be dropped depends on the saliency with a Beta prior (or drop ratio). In other words, we present a general patch-aware DropMix framework that drops a patch surrounding the peak salient region of one text input embedding and mixes it with the corresponding patch of another text input embedding to generate new examples. Under this framework, we develop other variants of DropMix that mixes text input embeddings using different fine-grained patches based on the saliency and dimension of words in text.

In a nutshell, we make the following contributions in this paper:

- We propose a unified framework DropMix, which leverages dropout and mixup for data augmentation and regularization in natural language processing tasks.
- Under this framework, we propose an efficient fine grained patch selection for the mixup, with three other variants for the text augmentation.
- Empirical studies on 12 commonly used datasets in text classification confirms the effectiveness of our model.

2 Related Work

A variety of regularization techniques have been proposed to solve overfitting in deep neural networks (Zhang et al., 2017; Yun et al., 2019; Verma et al., 2019). Some of the notable and recent methods include, Vanilla Mixup (Zhang et al., 2017) which linearly interpolates two input samples and their associated labels to generate augmented versions for regularization. CutMix (Yun et al., 2019) cuts and pastes image patches to construct new examples, and mixes the labels in proportion to the cut patch sizes. Manifold Mixup (Verma et al., 2019) on the other hand interpolates the middle hidden layers' feature maps. PatchUp (Faramarzi et al., 2020) proposes a method of patches swap or interpolation on the hidden layer feature maps. SaliencyMix (Uddin et al., 2021) mixes a target image with a saliency map (Simonyan et al., 2013) of the source image that highlights important objects. Although the majority of these approaches were designed for vision tasks, they have been widely adopted for NLP (Guo et al., 2019; Verma et al., 2019; Yoon et al., 2021).

In the context of text classification, Mixup regularization variants including, wordMixup and senMixup (Guo et al., 2019) perform interpolation on the word and sentence embeddings. Guo et al. (2019) shows that Mixup can improve the accuracy upon both CNN and LSTM sentence classification models. MixText (Chen et al., 2020) extends the hidden layer feature interpolation method to semi-supervised text classification task. To overcome the

mixing difficulty in the discrete text input spaces, SeqMix (Zhang et al., 2020) performs token-level interpolation in the embedding space and selects a token closest to the interpolated embedding. SSMix (Yoon et al., 2021) performs mixup operation on the input text rather than hidden vectors as like previous methods (Guo et al., 2019; Chen et al., 2020; Zhang et al., 2020). Specifically, SSMix synthesizes a sentence while preserving the locality of two original texts by relying on high saliency span-based mixing. This work is closely related to our methods, but we argue that the synthesis sanity of its input word level mixup rather generates "dirty" samples.

3 Preliminaries

Our proposed model, DropMix, builds upon notable works (Srivastava et al., 2014; Guo et al., 2019) relating to data augmentation for text classification. Specifically, given a text $\boldsymbol{x} = [x_1, x_2, ..., x_N]$ with N words, $\boldsymbol{x}$ is typically represented as low-dimensional features $\mathbf{X} \in \mathbb{R}^{N \times D}$, generated through an encoder such as Bert (Devlin et al., 2018). A classifier takes $\mathbf{X}$ as input and maps it to one or more class labels Y. To improve the classification problem, conventional data augmentation methods such as Dropout (Srivastava et al., 2014) and Mixup (Guo et al., 2019) have been utilized to encourage generalization. Here, we describe the basics of these data augmentation methods as they are fundamental to our approach.

3.1 Dropout

Dropout (Srivastava et al., 2014) is a classical regularization scheme which works by randomly dropping inputs of a layer during training, which may be the neurons of a neural network or the features (or units) of the data sample. For text classification, randomly dropping features of the data sample refers to applying a dropout on the word embedding layer to generate an augmented version of $\mathbf{X}$, denoted as $\widetilde{\mathbf{X}}$ and formulated as follows:

$$\widetilde{\mathbf{X}} = \mathbf{M} \odot \mathbf{X}, \quad \widetilde{Y} = Y$$

where $\mathbf{M} \in \mathbb{R}^{N \times D}$ is a mask matrix with entry $\mathbf{M}_{ij} \sim Bernoulli(p)$. p is a dropout ratio. The operator $\odot$ denotes an element-wise product. The new label $\widetilde{Y}$ of $\widetilde{\mathbf{X}}$ remains unchanged. With Dropout, the over-dependence on the input features is reduced to encourage the learning model to generalize.

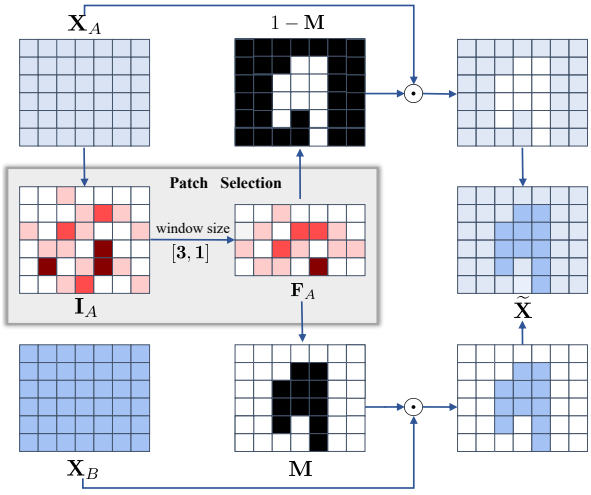

Figure 1: Overall process of DropMix. The darker the red shade, the more salient the units in $\mathbf{I}_A$ and $\mathbf{F}_A$. With the mask matrix $\mathbf{M}$, black units represent 1 and white units represent 0.

3.2 Mixup

Mixup is a data augmentation method that generates new training data $(\widetilde{\mathbf{X}}, \widetilde{Y})$ by linearly interpolating a pair of random inputs, $\mathbf{X}_A$ and $\mathbf{X}_B$, and their corresponding labels $\mathbf{Y}_A$ and $\mathbf{Y}_B$.

$$\begin{aligned} \widetilde{\mathbf{X}} &= (1-\lambda)\mathbf{X}_A + \lambda \mathbf{X}_B \\ \widetilde{Y} &= (1-\lambda)Y_A + \lambda Y_B \end{aligned} \tag{1}$$

where λ is beta-distributed with parameter α, denoted by $\lambda \sim Beta(\alpha, \alpha)$. By generating such samples, Mixup encourages a model to behave linearly in-between training samples to improve generalization. We refer the reader to the standard source (Guo et al., 2019) for a detailed review on Mixup.

4 DropMix for Text Augmentation

In this paper we combine the advantages of Dropout and Mixup to develop a new approach for data augmentation, namely, DropMix (Fig. 1). Specifically, DropMix encourages a model to reduce the over-dependence on input units while behaving non-linearly in-between training samples. The idea is to dropout units in $\mathbf{X}_A$ and replace their positions with corresponding units in $\mathbf{X}_B$ based on the saliency of the units. We define the augmentation approach as follows:

$$\begin{aligned} \widetilde{\mathbf{X}} &= (1-\mathbf{M}) \odot \mathbf{X}_A + \mathbf{M} \odot \mathbf{X}_B \\ \widetilde{Y} &= (1-\lambda)Y_A + \lambda Y_B \end{aligned} \tag{2}$$

where $\lambda \sim Beta(\alpha, \alpha)$. $\mathbf{M} \in \mathbb{R}^{N \times D}$ is constrained on λ, thus, λND units of $\mathbf{X}_A$ is replaced by the units in $\mathbf{X}_B$.

Typically, the mask matrix $\mathbf{M}$ is generated randomly by setting units to 0. This works for Dropout because it significantly reduces the overdependence on input units without focusing on any particular unit. However, our approach is more concerned with the model's dependency on the saliency on both inputs $\mathbf{X}_A$ and $\mathbf{X}_B$. Hence, rather than random sampling, we develop algorithms that identify the salient units of $\mathbf{X}_A$ to construct $\mathbf{M}$. That is, the resulting features $\widetilde{\mathbf{X}}$ becomes a mixture of $\mathbf{X}_A$ and $\mathbf{X}_B$ which considers the saliency of both inputs into account.

4.1 Patch Selection and Mixing

In order to determine the salient units in $\mathbf{X}_A$, we use a saliency map (Simonyan et al., 2013), denoted as $\mathbf{I}_A \in \mathbb{R}^{N \times D}$. Specifically, each unit in $\mathbf{I}_A$ is the magnitude of the derivative of classification loss w.r.t its corresponding feature in $\mathbf{X}_A$, found through back-propagation. The saliency map (or saliency matrix) highlights the important units that affect the text classification decision.

We now aim to extract patches of the saliency map for the mixup. To this end, we first use sliding filter windows with size $n \times d$, where $n \in \{1, 2, ..., N\}$ and $d \in \{1, 2, ..., D\}$ and perform a convolution over the saliency map. Under the assumption that we use a sliding stride of 1, we obtain $(N - n + 1)(D - d + 1)$ windows. An L2 norm is then used as an activation function and applied on each window to highlight the saliency of features. A saliency activation map $\mathbf{F}_A \in \mathbb{R}^{(N-n+1)\times(D-d+1)}$ is created by concatenating all the extracted windows.

At this point, we initialize a mask matrix $\mathbf{M}$, which has the same size and dimension as $\mathbf{X}_A$ (or $\mathbf{X}_B$). The initialized $\mathbf{M}$ is a matrix of ones. To identify the units to be zeroed out, we relate it to the feature activation mat $\mathbf{F}_A$. More specifically, first note that each value in $\mathbf{F}_A$ relates to a window of size $n \times d$ in $\mathbf{I}_A$. Since $\mathbf{X}_A$ (or $\mathbf{X}_B$) also share the same shape with $\mathbf{I}_A$, we can infer that each value in $\mathbf{F}_A$ also relates to a window in $\mathbf{X}_A$ (or $\mathbf{X}_B$). The index mapping function f from $\mathbf{F_A}$ to $\mathbf{I_A}$ can be defined as follows:

$$f(i,j) = \bigcup_{p=i}^{i+n-1} \bigcup_{q=j}^{j+d-1} (p,q) \tag{3}$$

where (i, j) is an index of $\mathbf{F_A}$, (p, q) is an index of $\mathbf{I_A}$. We identify the top-k salient units in $\mathbf{F_A}$, sort them and collect their corresponding k windows. For each window, we set all the nd units of $\mathbf{M}$ corresponding to the window to zero. The patch used in the mixup can now be extracted by applying the mask (or the mask's inverse) on $\mathbf{X}_B$ (or $\mathbf{X}_A$). The new features $\widetilde{\mathbf{X}}$ can now be generated, following Eqn. (2).

It should be noted that a unit in $\mathbf{I}_A$ may be involved with several patches. The number k is set to ensure that the total number of units for replacement is no more than λND. The relation between k and λND is defined as follows:

$$\lambda ND = |\bigcup_{(i,j)\in\mathcal{K}} f(i,j)| \tag{4}$$

where $\mathcal{K}$ is the set of top k windows' indexes in $\mathbf{F_A}$.

To ensure the interpolation of features correspond to the interpolation of associated labels, we make an adjustment to the mixing of labels:

$$\widetilde{Y} = (1 - \frac{S}{ND})Y_A + \frac{S}{ND}Y_B \tag{5}$$

where S is the sum of $\mathbf{M}$ elements, i.e., the number of $\mathbf{M}$ elements containing one.

4.2 Window Size Setting

With regard to the window size setting, we are motivated by the assumption that the continuity of words (i.e., words in the D dimensional space) in text include context information which is significant for text classification. Therefore, n and d should be in the set $n \in \{1, 2, ..., N\}, d \in \{1, D\}$. Our filtering window becomes a general formulation of patch selection, which we refer to as *general*. Thus, we refer to this model as DropMix-general (**DropMix-G**).

We introduce other filter window configurations: *unit*, *channel*, *word* and develop the model variants DropMix-unit (DropMix-U), DropMix-word (DropMix-W) and DropMix-channel (DropMix-C).

DropMix-U ($n = 1, d = 1$): DropMix with a filter window which convolves one unit in $\mathbf{X}_A$.

DropMix-C ($n = N, d = 1$): DropMix with a filter window which convolves one dimension of the embedding for all words from a sentence. That means n features of each dimension are replaced or kept simultaneously.

DropMix-W ($n \in \{1, 2, ..., N\}, d = D$): DropMix with a filter window which convolves the whole embedding of n continuous words. It has a similar connotation to modelling with n-grams.

4.3 Loss Function

We adopt cross entropy to calculate the loss to optimize model parameters. Given a pair of inputs $(\mathbf{X}_{A_i}, Y_{A_i})$ and $(\mathbf{X}_{B_i}, Y_{B_i})$ for instance i, the loss function can be defined as follows:

$$L = -(1-\lambda)\sum_i Y_i \log(Y_{Ai}) - \lambda \sum_i Y_i \log(Y_{Bi}) \quad (6)$$

where Y_i is the label for the generated input $\widetilde{\mathbf{X}}$.

5 Experiment

5.1 Dataset

To evaluate DropMix, we use 12 text classification datasets from two sources: (1) 6 datasets selected from the popular benchmark GLUE (Wang et al., 2018): QQP [1], MNLI (Williams et al., 2018), SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009) and MRPC (Dolan and Brockett, 2005). (2) 6 CNN datasets used in the work of LeCun et al. (1998): CR (Hu and Liu, 2004), MR (Pang and Lee, 2005), MPQA (Esuli et al., 2008), SST-1 (Socher et al., 2013), Subj (Pang and Lee, 2004), TREC (Li and Roth, 2002). Both GLUE [2] and CNN datasets [3] can be downloaded from the Github link. Table 2 shows the statistics of all datasets. We use accuracy as the evaluation metrics following the practice in compared works (Devlin et al., 2018).

Dataset	#Train	#Valid	#AvgLen	#Label
MNLI	392702	9815	39.78	3
QQP	363846	40430	30.57	2
QNLI	104743	5463	49.49	2
SST-2	67349	872	13.47	2
MRPC	4076	1725	53.16	2
RTE	2490	277	66.37	2
MPQA	10606	/	5.46	2
MR	10662	/	27.42	2
SST-1	11855	2210	25.02	5
SUBJ	10000	/	29.98	2
TREC	5952	500	13.21	6
CR	3775	/	23.75	2

Table 2: Dataset statistics of 12 text classification datasets. AvgLen represents the average length of all sentences within one dataset.

5.2 Baseline

We implement and compare with a series of baseline models, including Bert, Bert-Dropout, Bert-Mixup, Bert-Dropout+Mixup and SSMix.

Bert in its original form includes a dropout layer. To demonstrate the impact of data augmentation strategies, we ablate the dropout layer in the original Bert and report the performance.

Bert-Dropout is the officially published and extensively utilized version of Bert (Devlin et al., 2018).

Bert-Mixup replaces the Dropout layer in Bert-Dropout with Mixup (Guo et al., 2019).

Bert-Dropout+Mixup applies dropout on the input features and applies a vanilla mixup to generate the new data. This is different from our method DropMix, which drops and mix corresponding features based on the saliency of the input data.

SSMix (Yoon et al., 2021) is an enhanced model of mixup inspired by CutMix and PuzzleMix (Kim et al., 2020).

Unfortunately, most of the image-based Mixup models (Kim et al., 2020; Yun et al., 2019; Uddin et al., 2021) cannot be applied directly to text processing tasks. SSMix on the other hand has been successfully applied in text processing and it is now the current SOTA in text regularization. We therefore exclude the image-based Mixup models but compare with SSMix as well as our baselines.

Hereinafter, we omit 'Bert-' Prefix from model names for brevity.

5.3 Implementation Details

For comparisons, we exploit the pre-trained BERT-base-uncased model from Hugging Face Transformers [4] and the source code from Github [5]. Following the recommended parameter settings: max sentence length N is 128, word embedding dimension D is 768, batch size is 32, epoch is 3 and learning rate is 2e-5, optimizer is AdamW with eps 1e-8. For the specific parameters of regularization

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2] https://github.com/nyu-mll/GLUE-baselines

[3] https://github.com/harvardnlp/sent-conv-torch

[4] https://github.com/huggingface/transformers

[5] https://github.com/huggingface/transformers/tree/master/examples/pytorch/text-classification

Model	MNLI	QQP	QNLI	SST-2	MRPC	RTE
Bert	84.43±0.30	91.11±0.06	91.49±0.09	92.41±0.13	84.00±0.42	64.55±0.95
Dropout	84.38±0.33	90.97±0.19	91.43±0.19	92.61±0.21	84.03±0.82	64.69±0.95
Dropout*	*84.27±0.12*	*91.32±0.04*	*91.28±0.05*	*92.96±0.04*	*86.37± 1.73*	*65.56±3.89*
Mixup	84.38±0.23	91.29±0.06	91.51±0.17	92.66±0.19	**84.48±0.59**	65.13±1.61
Dropout+Mixup	84.52±0.18	90.92±0.08	91.45±0.21	92.91±0.24	82.89±1.27	65.27±0.70
SSMix	84.12±0.22	91.19±0.20	91.48±0.19	92.39±0.41	83.48±0.67	65.85±2.17
SSMix*	*84.54±0.07*	*91.43±0.07*	*91.54±0.03*	*93.10±0.03*	*86.57± 1.15*	*67.22±2.57*
DropMix-U	84.27±0.31	91.35±0.10	91.18±0.12	92.77±0.19	81.75±1.28	66.93±1.28
DropMix-C	84.77±0.16	91.42±0.04	**91.70±0.18**	93.17±0.30	83.48±0.60	67.94±1.06
DropMix-W	83.49±0.13	90.35±0.04	90.80±0.25	91.72±0.38	81.62±1.16	65.99±1.17
DropMix-G	**84.78±0.33**	**91.45±0.05**	91.54±0.23	**93.33±0.34**	83.17±1.26	**68.09±0.98**

Model	TREC	SST-1	Subj	MR	CR	MPQA
Bert	96.64±0.23	54.16±0.66	97.22±0.37	88.34±0.95	91.11±1.65	91.97±0.91
Dropout	96.60±0.36	53.86±0.49	97.10±0.52	88.13±1.05	90.79±0.91	91.91±0.78
Mixup	96.72±0.48	53.92±0.11	97.04±0.29	88.28±1.10	91.06±1.70	92.08±0.75
Dropout+Mixup	96.72±0.30	53.59±0.39	97.14±0.29	88.25±0.57	91.11±1.39	92.03±0.97
SSMix	97.48±0.37	53.77±0.41	97.22±0.62	88.62±1.01	92.01±1.39	91.89±0.64
DropMix-U	97.68±0.16	53.80±0.24	97.38±0.29	88.81±0.88	91.90±1.31	92.29±0.83
DropMix-C	97.76±0.23	53.98±0.23	97.58±0.24	88.66±0.69	92.01±0.77	92.36±0.90
DropMix-W	97.16±0.27	53.14±0.51	97.46±0.37	88.47±0.94	91.48±0.83	91.42±0.40
DropMix-G	**97.92±0.10**	**54.71±0.51**	**97.64±0.27**	**88.88±0.84**	**92.06±1.49**	**92.52±0.47**

Table 3: Results of baseline models and DropMix on GLUE and other 6 classification datasets using BERT-base-uncased as encoder. Superscript $\star$ indicate the results are taken from (Yoon et al., 2021). The bold result is the best result of a dataset. The underlined result is DropMix-C performs the best or only lower than DropMix-G among four DropMix variants.

models, the dropout ratio p for Dropout is set at 0.1, the beta distribution parameter α is set at 0.1 for Mixup and $\alpha \in \{0.025, 0.05, 0.1, 0.25, 0.5, 1\}$ for SSMix and DropMix. Unlike DropMix-U and DropMix-C which have a determined window size, we set $n \in [2, 32]$ for DropMix-W and DropMix-G. Since regularization methods reduce the speed of convergence, we run multiple epochs (5 to 10 times) for these regularization schemes. The parameter quantity of all regularization models are equal to the original Bert (Devlin et al., 2018). All hyper-parameters are set manually and empirically tuned on the development set during the training process.

All experiments are completed on Tesla V100-SXM2-32GB GPU, costing about 1300s per epoch for the medium scale dataset QNLI.

5.4 Text Classification Performance

We run our experiments five times using different random seeds on all models. The average classification accuracy and its standard deviation are shown in Table 3.

From these results, we find that DropMix outperforms other baseline models on all datasets, except for MRPC. These results indicate that DropMix is an effective scheme for text data augmentation. It is also worth noting that SSMix* starts from the best checkpoint of Dropout* and measures validation accuracy every 500 steps. This proves to be useful in achieving faster optimization. Nonetheless, DropMix still enjoys a better overall performance.

Among the four variants of DropMix, DropMix-G generally achieves the best results, except on QNLI and MRPC datasets. The results indicate that DropMix-G is an efficient variant and choosing fine grained patch on text representation is practical for data augmentation. Besides DropMix-G, DropMix-C achieves comparable results and outperforms DropMix-U and DropMix-W in almost all datasets except MR. This indicates DropMix-C is a satisfying substitute to DropMix-G when there is an efficiency requirement in the training process. Furthermore, DropMix-C performs better than DropMix-W. This demonstrates that *channel* may be more important and appropriate than *word*

Model	TREC	SST-1	Subj	MR	CR	MPQA
Bert-large	97.52±0.29	55.16±0.60	97.29±0.52	89.97±1.00	92.43±1.21	92.27±0.65
Dropout	97.44±0.36	55.38±0.47	97.28±0.37	89.87±0.65	92.27±1.22	92.44±0.48
Mixup	97.72±0.29	54.90±0.71	97.36±0.30	89.78±0.85	92.96±1.19	91.53±1.34
Dropout+Mixup	**97.80±0.17**	54.71±0.47	97.53±0.36	89.93±0.54	92.85±0.96	92.36±0.38
DropMix-C	97.76±0.23	55.22±0.30	97.72±0.23	**90.42±0.71**	**93.70±0.72**	92.57±0.65
DropMix-G	97.76±0.20	**55.67±0.84**	**97.74±0.35**	90.16±0.75	93.60±1.04	**92.66±0.58**
Model	TREC	SST-1	Subj	MR	CR	MPQA
RoBERTa	97.32±0.43	57.68±0.46	97.17±0.41	90.10±0.63	94.17±0.50	91.97±0.59
Dropout	97.36±0.20	57.78±0.69	97.12±0.35	90.25±0.44	94.23±0.20	92.18±0.68
Mixup	97.12±0.10	57.61±0.62	96.90±0.17	89.58±0.66	93.86±0.84	91.80±0.60
Dropout+Mixup	97.56±0.29	57.62±0.60	96.91±0.34	90.17±0.65	94.23±0.35	92.08±0.33
DropMix-C	98.00±0.13	**58.04±0.46**	97.36±0.22	**90.47±0.19**	94.55±0.64	92.10±0.56
DropMix-G	**98.08±0.24**	57.91±0.29	**97.36±0.21**	90.44±0.74	**94.71±0.58**	**92.31±0.53**

Table 4: Results on other encoders.

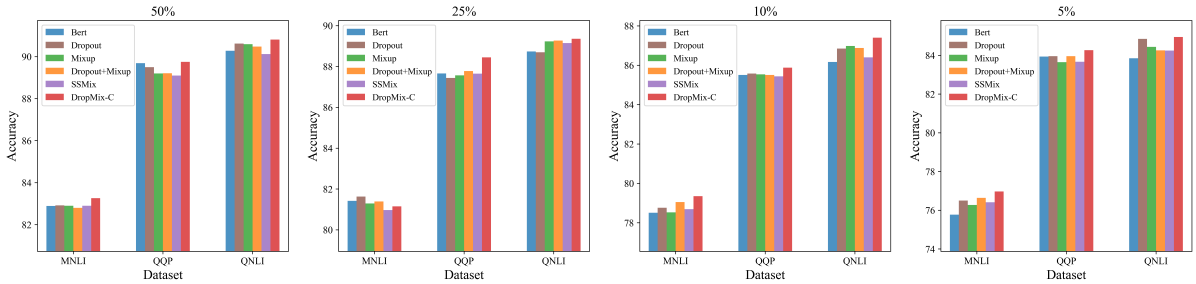

Figure 2: Accuracy on different ratio to sparse data on three large datasets.

(DropMix-W) for text augmentation.

We perform additional experiments to investigate the model behaviour with different encoders, particularly, BERT-large-uncased and RoBERTa. Table 4 shows the results of our experiments on six datasets. Similar to the results produced via the BERT-base-uncased encoder (see Table 3), we find that both BERT-large-uncased and RoBERTa leads to a better performance of DropMix variants when compared to Dropout and Mixup. The results indicate the efficiency of DropMix.

5.5 Performance on Sparse Data

To further analyze in sparse settings, we downsample 50%, 25%, 10%, 5% of the training data from the 3 large datasets MNLI, QQP and QNLI. For the sake of efficiency, we use DropMix-C in this experiment. The classification accuracies are shown in Figure 2.

Results indicate that more training samples could improve the performance of all models. On the reduced datasets, we observe that these methods have a certain regularization effect. DropMix-C is better than other regularization models generally, and as the dataset decreases extremely to 10% or 5%, the performance gap between DropMix-C and compared methods is enlarged. These observations demonstrate the effectiveness of the DropMix-C on even sparser datasets.

5.6 Fine Grained Patch Analysis

To discover the patch selection of DropMix-G, we plot the saliency map of a sentence. We employ a fixed λ to determine the ratio of units to retain for convenience. This is to ensure that for a given sentence, the proportion of units to be replaced is consistent in different epochs. We also set $n = 2$ to observe the effect of *general* in DropMix-G. Since the saliency map has an extensive range of values and it cannot represent the result of the final patch, we plot the mask matrices as saliency map to some extent. Figure 3 (a) illustrates the 'saliency map' of the sentence *"like jaglom's films , some is honestly affecting"* taken from SST-2.

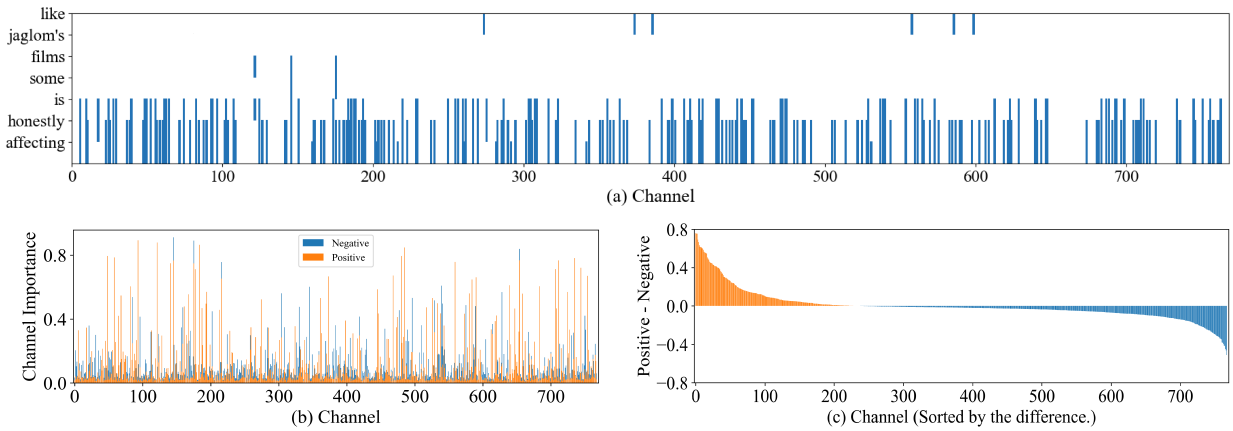

Figure 3: (a) Saliency map of sentence "like jaglom's films , some is honestly affecting". In this heatmap, blue represents a patch with large saliency value that should be dropped. (b) Channel importance of positive and negative samples in training set. (c) The difference between the channel importance on positive and negative samples from (b). To observe clearly, channel in x-axis is sorted by the difference.

In this case example, words that affect the sentiment classification decision include the words, "honestly" and "affecting". These words are easily spotted when we use *word*, indicating its effectiveness. Additionally, not all channels of these two words are highlighted, which implies channel plays a distinct role in the text representation. By comparing the results of *general* as against *word* and *channel* to obtain the fine grained patch, we can conclude that *general* is reasonable and effective for text representation.

Positive			Negative		
93	183	485	345	96	85
good	good	funny	too	movie	bad
movie	funny	good	bad	bad	too
film	best	fun	movie	film	movie
best	clever	entertaining	film	##iche	no
funny	fascinating	interesting	not	too	dull
interesting	great	fascinating	un	no	not
enjoyable	interesting	clever	no	dull	predictable
smart	fun	best	dull	not	flat
clever	smart	smart	little	##less	worst
wonderful	compelling	compelling	like	un	un
fascinating	movie	love	fails	worst	film
compelling	better	great	worst	flat	less
great	wonderful	enjoyable	nothing	predictable	boring
fun	entertaining	wonderful	story	bland	##less

Table 5: The contributed words of selected channels. In positive samples, the top 3 selected channel are 93, 183 and 485. In negative samples, top 3 channels are 345, 96 and 85. Channels are ordered by their channel importance, and words are ordered by the times that chosen as a contributed word. Positive meaning words are covered with red colorboxes, negative meaning words are covered with blue colorboxes and common negative pattern words are covered with green colorboxes.

5.7 Channel Importance Analysis

For a comprehensive study on the effectiveness of channel, we construct an experiment on channel selection on SST-2. For positive and negative samples, we separately count the number of times that each channel is selected (highlighted in the saliency map). For fair comparison, we divide the count by the number of positive and negative samples. We believe this value represents the importance of a specific channel, call this the channel importance.

We first plot the channel importance of positive and negative samples in the training set as shown in Figure 3 (b). It can be observed that channel importance of positive and negative samples gain peak values on different channels, which indicates that channel plays a disparate role with different sentiment polarity.

We further plot the difference between the channel importance in positive and negative to eliminate the impact of irrelevant channels. As the decreasing ordered difference illustrated in Figure 3 (c), for all 768 channels (from the 768 dimensions in Bert Embeddings), around 100 positive channels (left of the figure) and 100 negative channels (right of the figure) achieves a relatively channel importance. We find that the selected positive channels are more concentrated and have a higher value than negative channels. To further investigate the underlying reason of this phenomenon, we find that negative sentences have richer patterns and are from a more common vocabulary, such as "little", "too" and "to". This means that more negative channels are selected during the training. This observation is also confirmed by the statistics in Table 5.

After obtaining the selected channels of positive and negative, we further analyze which word contributes most in a sentence of the selected channel. Fixing the top 3 selected channels of positive samples, we compute the maximum gradient of each word in this channel. Then we obtain the word importance for each channel. We also conduct the same procedure for negative samples. The top 3 channels for positive samples, negative samples and their contributed words are shown in Table 5. Clearly, the listed contributed words are highly related to their corresponding categories. This experiment demonstrates the effectiveness of our presented model on channel.

5.8 CutMix vs. SSMix vs. DropMix

Our work is most closely related to the line of research that adapt the mixup method (Guo et al., 2019). As such, we briefly discuss how these methods differ in architecture to gain a more in-depth understanding of our proposed mechanism.

CutMix vs. SSMix CutMix cuts a patch of an image and mixes it with a corresponding patch from another image. SSMix is an adaptation of CutMix to text classification. SSMix applies a saliency map to determine the Mixup patch.

SSMix vs. DropMix SSMix disregards the position and sentence structure of words, and simply replaces unimportant word spans in $\mathbf{X}_A$ with salient words in $\mathbf{X}_B$. DropMix on the other hand considers the context of words as well as its structure by mapping words into a low-dimensional space.

CutMix vs. DropMix CutMix cuts rectangular patches on input images with 3 RGB channels. Analogously, for a 2d sentence embedding matrix, DropMix-C uniformly cuts some specific dimension of latent word embedding, i.e., a channel or column vector patch. DropMix advocates and regards channel mixing as suitable and important for text synthesis and augmentation.

6 Conclusion

In this paper, we present a data augmentation framework, DropMix, based on the saliency map which combines Dropout and Mixup to facilitate regularization in text learning. Our best results obtained from our model variant DropMix-G indicates that the features of the word embeddings are important for patch selection and mixing. In general, our empirical results confirm the effectiveness of DropMix.

7 Limitation

DropMix-G has a dominant performance due to the fine grained patch selection using an irregular window size. This makes it computationally expensive. If there is an efficiency requirement for an algorithm, DropMix-C is much more desirable and it also achieves competitive performance with DropMix-G.

8 Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0110700, in part by the Fundamental Research Funds for the Central Universities, in part by the State Key Laboratory of Software Development Environment. SM is supported by a Leverhulme Trust Research Project Grant (No. RPG-2020-148).

References

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09)*.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Andrea Esuli, Fabrizio Sebastiani, and Ilaria Urciuoli. 2008. Annotating expressions of opinion and emotion in the Italian content annotation bank. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Mojtaba Faramarzi, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. 2020. Patchup: A regularization technique for convolutional neural networks.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5275–5285. PMLR.

Anders Krogh and John A Hertz. 1992. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (Workshop Poster)*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

A F M Shahab Uddin, Mst. Sirazam Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. 2021. Saliencymix: A saliency guided data augmentation strategy for better regularization. In *International Conference on Learning Representations*.

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, Long Beach, California, USA. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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, New Orleans, Louisiana. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*.

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

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