Mixup-Transformer: Dynamic Data Augmentation for NLP Tasks

Lichao Sun[1]*, **Congying Xia**[2]*, **Wenpeng Yin**[3], **Tingting Liang**[4], **Philip S. Yu**[2], **Lifang He**[1]

[1] Lehigh University; [2] University of Illinois at Chicago;
[3] Salesforce Research; [4] Hangzhou Dianzi University
`james.lichao.sun@gmail.com`; `{cxia8, psyu}@uic.edu`;
`wyin@salesforce.com`; `liangtt@hdu.edu.cn`;
`lih319@lehigh.edu`

Abstract

Mixup (Zhang et al., 2017) is a latest data augmentation technique that linearly interpolates input examples and the corresponding labels. It has shown strong effectiveness in image classification by interpolating images at the pixel level. Inspired by this line of research, in this paper, we explore: i) how to apply *mixup* to natural language processing tasks since text data can hardly be mixed in the raw format; ii) if *mixup* is still effective in transformer-based learning models, *e.g.*, BERT. To achieve the goal, we incorporate *mixup* to transformer-based pre-trained architecture, named "*mixup*-transformer", for a wide range of NLP tasks while keeping the whole end-to-end training system. We evaluate the proposed framework by running extensive experiments on the GLUE benchmark. Furthermore, we also examine the performance of *mixup*-transformer in low-resource scenarios by reducing the training data with a certain ratio. Our studies show that *mixup* is a domain-independent data augmentation technique to pre-trained language models, resulting in significant performance improvement for transformer-based models.

1 Introduction

Deep learning has shown outstanding performance in the field of natural language processing (NLP). Recently, transformer-based methods (Devlin et al., 2018; Yang et al., 2019) have achieved state-of-the-art performance across a wide variety of NLP tasks[1]. However, these models highly rely on the availability of large amounts of annotated data, which is expensive and labor-intensive. To solve the data scarcity problem, data augmentation is commonly used in NLP tasks. For example, Wei and Zou (2019) investigated language transformations like insertion, deletion and swap. Several works (Malandrakis et al., 2019; Yoo et al., 2019; Xia et al., 2020b; Xia et al., 2020a) utilized variational autoencoders (VAEs) (Kingma and Welling, 2013) to generate more raw inputs. Nevertheless, these methods often rely on some extra knowledge to guarantee the quality of new inputs, and they have to be working in a pipeline.

Zhang et al. (2017) proposed *mixup*, a domain-independent data augmentation technique that linearly interpolates image inputs on the pixel-based feature space. Guo et al. (2019) tried *mixup* in CNN (LeCun et al., 1998) and LSTM (Hochreiter and Schmidhuber, 1997) for text applications. Despite effectiveness, they conducted *mixup* only on the fixed word embedding level like Zhang et al. (2017) did in image classification. Two questions arise, therefore: (i) how to apply *mixup* to NLP tasks if text data cannot be mixed in the raw format? Apart from the embedding feature space, what other representation spaces can be constituted and used? ii) whether or not *mixup* can boost the state-of-the-art further in transformer-based learning models, such as BERT (Devlin et al., 2018).

To answer these questions, we stack a *mixup* layer over the final hidden layer of the pre-trained transformer-based model. The resulting system can be applied to a broad of NLP tasks; in particular, it is still end-to-end trainable. We evaluate our proposed *mixup*-transformer on the GLUE benchmark,

---

*Indicates Equal Contribution

[1]Please see the General Language Understanding Evaluation (GLUE) (https://gluebenchmark.com/) for tracking performance.

which shows that *mixup* can consistently improve the performance of each task. Our contributions are summarized as follows:

- We propose the *mixup*-transformer that applies *mixup* into transformer-based pre-trained models. To our best knowledge, this is the first work that explores the effectiveness of *mixup* in Transformer.

- In experiments, we demonstrate that *mixup*-transformer can consistently promote the performance across a wide range of NLP benchmarks, and it is particularly helpful in low-resource scenarios where we reduce the training data from 10% to 90%.

2 Mixup-Transformer

In this section, we first introduce the *mixup* used in previous works. Then, we show how to incorporate the *mixup* into transformer-based methods and how to do the fine-turning on different text classification tasks. Last, we will discuss the difference between the previous works and our new approach.
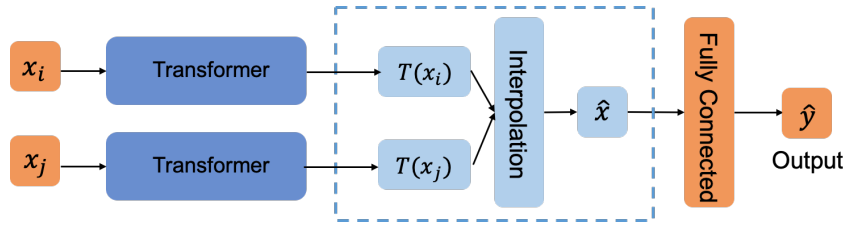

Figure 1: The overall framework of *mixup*-transformer. x_i, x_j are two separate sentences, fed to the same Transformer T. $T(x_i)$ is the representation of the input x_i, generated by T. $\hat{x}$ and $\hat{y}$ are the interpolated representation and label, respectively.

2.1 Mixup

Mixup is first proposed for image classification (Zhang et al., 2017), which incorporates the prior knowledge that linear interpolations of feature representations should lead to the same interpolations of the associated targets. In *mixup*, virtual training examples are constructed by two examples drawn at random from the training data:

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j, \text{where } x_i, x_j \text{ are raw inputs;}$$
$$\hat{y} = \lambda y_i + (1 - \lambda)y_j, \text{where } y_i, y_j \text{ are corresponding labels,}$$

where λ could be either fixed value in $[0, 1]$ or $\lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. In previous works, *mixup* is a static data augmentation approach that improves and robusts the performance in image classification.

2.2 Mixup for Text Classification

Text classification is the most fundamental problem in the NLP field. Unlike image data, text input consists of discrete units (words) without an inherent ordering or algebraic operations - it could be one sentence, two sentences, a paragraph or a whole document.

The first step of text classification is to use the word embedding to convert each word of the text into a vector representation. In the traditional approaches, the word embedding method can be bag-of-words, or a fixed word to vector mapping dictionary built by CNN or LSTM. In our approach, instead of using the traditional encoding methods, we use transformer-based pre-trained language models to learn the representations for text data. For downstream tasks, we fine-tune transformer-based models with the *mixup* data augmentation method. Formally, *mixup*-transformer constructs virtual hidden representations dynamically durning the training process as follows:

$$\hat{x} = \lambda \cdot T(x_i) + (1 - \lambda) \cdot T(x_j), \text{where } T(x_i), T(x_j) \text{ are output features from transformers;}$$
$$\hat{y} = \lambda y_i + (1 - \lambda)y_j, \text{where } y_i, y_j \text{ are corresponding labels,}$$

where $T(\cdot)$ represents outputs of the transformer layers as shown in Figure 1. Note that, the mixup process is trained together with the fine-tuning process in an end-to-end fashion, and the hidden *mixup* representations are dynamic during the training process.

2.3 Discussion

In this section, we highlight two main differences between our approach and previous methods using *mixup* techniques (Zhang et al., 2017; Guo et al., 2019) for comparison.

• Dynamic *mixup* representation. For each input pair, x_i and x_j, the previous approaches produce a fixed *mixup* representation given a fixed λ. However, the *mixup* hidden representations in our approach are dynamic since they are trained together with the fine-tuning process.

• Dynamic *mixup* activation. Since a pre-trained network needs to be fine-tuned for a specific task, we can dynamically activate the *mixup* during the training. For example, if the training epoch is 3, we can choose to use *mixup* in any epoch or all epochs. In our experiments, we fine-tune the model without *mixup* in the first half of epochs for good representations and add *mixup* in the last half of the epochs.

3 Experiments

To show the effectiveness of our proposed *mixup*-transformer, we conduct extensive experiments by adding the *mixup* strategy to transformer-based models on seven NLP tasks contained in the GLUE benchmark. Furthermore, we reduce the training data with different ratios (from 10% to 90%) to see how the *mixup* strategy works with insufficient training data. We report the performance on development sets for all the tasks because the test time is limited by the online GLUE benchmark.

Datasets. The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of diverse natural language understanding tasks. Experiments are conducted on eight tasks in GLUE: CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QQP (Z. Chen and Zhao, 2018), MNLI (Williams et al., 2017), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009).

Baselines. Two baselines are conducted in the experiments, including BERT-base and BERT-large (Devlin et al., 2018). We evaluate the performance of our methods by adding the *mixup* strategy to these two baselines.

Implementation details. When fine-tuning BERT with or without the *mixup* strategy for these NLP tasks, we fix the hyper-parameters as follows: the batch size is 8, the learning rate is 2e-5, the max sequence length is 128, and the number of the training epochs is 3. We test different values of λ, (from 0.1 to 0.9) on the default dataset (CoLA) and find *mixup*-transformer is insensitive to this hyper-parameter, so we set a fixed value of $\lambda = 0.5$.

3.1 Results on full data

Dataset	CoLA (Corr)	SST-2 (Acc)	MRPC (Acc)	STS-B (Corr)	QQP (Acc)	MNLI-mm (Acc)	QNLI (Acc)	RTE (Acc)
BERT-base	57.86	92.20	86.76	**89.41**	90.79	84.47	91.61	68.23
BERT-base + *mixup*	**59.58**	**92.78**	**88.48**	88.66	**90.98**	**85.12**	**91.84**	**71.84**
Improved	1.72	0.58	1.72	-0.75	0.21	0.65	0.23	3.61
BERT-large	59.71	91.97	86.27	89.21	90.38	**85.92**	91.96	69.31
BERT-large + *mixup*	**62.39**	**92.78**	**87.99**	**90.10**	**90.80**	85.91	**92.09**	**69.68**
Improved	2.68	0.81	1.72	0.89	0.42	-0.01	0.13	0.37

Table 1: *mixup*-transformer results on eight NLP tasks. Matthew's Correlations are reported for CoLA, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks.

Experimental results for eight different NLP tasks are illustrated in Table 1. By adding the proposed *mixup* technique to BERT-base and BERT-large, the *mixup*-transformer improves the performance consistently on most of these tasks. The average improvement is around 1% for all the settings. The highest performance gain comes from the RTE task by adding the proposed *mixup* technique on BERT-base. In this experiment, the accuracy improves from 68.23% to 71.84%, which is an increase of 3.61%. The

Matthew's correlation for CoLA increases from 59.71% to 62.39% (improved 2.68%) with *mixup* on BERT-large. Some experiments also get performance decrease with *mixup*. For example, adding *mixup* to BERT-base on STS-B decreases the Spearman correlation from 89.41% to 88.66%. Overall, most of the tasks improved (14 out of 16, while 2 got slightly worse) were found by applying *mixup*-transformer.

Training Data	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
BERT-large	74.51	77.45	77.69	82.11	83.58	84.07	84.07	84.56	86.76	85.53
BERT-large + *mixup*	**77.21**	**79.17**	**81.13**	**87.01**	**85.78**	**86.27**	**86.79**	**86.76**	**87.25**	**87.99**
Improved	2.70	1.72	3.44	4.90	2.21	2.21	2.72	2.21	0.49	2.46

Table 2: *mixup*-transformer with reduced training data on MRPC. Accuracy scores are used to evaluate the performance.

3.2 Results on limited data

As *mixup* is a technique for augmenting the feature space, it is interesting to see how it works when the training data is insufficient. Therefore, we reduce the training data with a certain ratio (from 10% to 90% with a step 10%) and test the effectiveness of *mixup*-transformer in low-resource scenarios. As shown in Table 2, BERT-large + *mixup* consistently outperforms BERT-large when we reduce the training data for MRPC, where the highest improvement (4.90%) is achieved when only using 40% of the training data. Using the full training data (100%) gets an increase of 2.46%, which indicates that *mixup*-transformer works even better with reduced annotations. We also report the experiments of reducing training data on other tasks, including STS-B, RTE and CoLA. As shown in Figure 2, the *mixup*-transformer again consistently improves the performance for all the experiments. The performance gains with less training data (like 10% for STS-B, CoLA, and RTE) are higher than using full training data since data augmentation is more effective when the annotations are insufficient. Therefore, the *mixup* strategy is highly helpful in most low-resource scenarios.
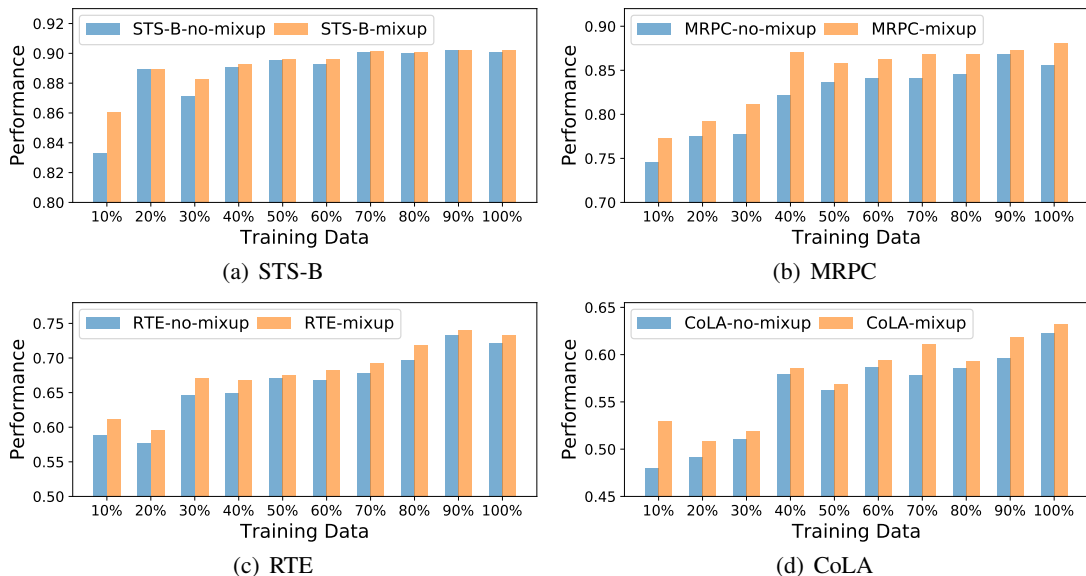

(a) STS-B

(b) MRPC

(c) RTE

(d) CoLA

Figure 2: *mixup*-transformer with BERT-large runs with reduced training data for four tasks: STS-B, MRPC, RTE and CoLA.

4 Conclusion

In this paper, we propose the *mixup*-transformer that incorporates a data augmentation technique called *mixup* into transformer-based models for NLP tasks. Unlike using the static *mixup* in previous works, our approach can dynamically construct new inputs for text classification. Extensive experimental results show that *mixup*-transformer can be dynamically used with a pre-trained model to achieve better performance on GLUE benchmark.

Acknowledgement

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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)*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled text generation for data augmentation in intelligent artificial agents. *arXiv preprint arXiv:1910.03487*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Congying Xia, Caiming Xiong, Philip Yu, and Richard Socher. 2020a. Composed variational natural language generation for few-shot intents. *arXiv preprint arXiv:2009.10056*.

Congying Xia, Chenwei Zhang, Hoang Nguyen, Jiawei Zhang, and Philip Yu. 2020b. Cg-bert: Conditional text generation with bert for generalized few-shot intent detection. *arXiv preprint arXiv:2004.01881*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Kang Min Yoo, Youhyun Shin, and Sang-goo Lee. 2019. Data augmentation for spoken language understanding via joint variational generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7402–7409.

X. Zhang Z. Chen, H. Zhang and L. Zhao. 2018. Quora question pairs.

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