BatchMixup: Improving Training by Interpolating Hidden States of the Entire Mini-batch

Wenpeng Yin, Huan Wang, Jin Qu and Caiming Xiong
Salesforce Research
wyin@salesforce.com

Abstract

Usually, we train a neural system on a sequence of mini-batches of labeled instances. Each mini-batch is composed of k samples, and each sample will learn a representation vector. Mixup implicitly generates synthetic samples through linearly interpolating inputs and their corresponding labels of random sample pairs in the same mini-batch. This means that Mixup only generates new points on the edges connecting every two original points in the representation space. We observed that the new points by the standard Mixup cover pretty limited regions in the entire space of the mini-batch. In this work, we propose BatchMixup—improving the model learning by interpolating hidden states of the entire mini-batch. BatchMixup can generate new points scattered throughout the space corresponding to the mini-batch. In experiments, BatchMixup shows superior performance than competitive baselines in improving the performance of NLP tasks while using different ratios of training data.

1 Introduction

The study of data augmentation techniques has a long history in the NLP community. Typical data augmentations include synonym replacement (Kobayashi, 2018), back-translation (Fadaee et al., 2017), adding data noise (Xie et al., 2017), etc. Mostly, these techniques are combined with the augmentation-free models in pipeline. Mixup (Zhang et al., 2018) is able to augment the data by linearly combining each two examples by their hidden representations, keeping the whole system trained in end-to-end.

Mixup has shown effectiveness in a range of NLP tasks (Sun et al., 2020). Nevertheless, it has two drawbacks. First, Mixup generates new points merely and exactly on the connecting edges of random point pairs; these new points cover pretty limited region in the representation space of the mini-batch. Second, the training of a system equipped with Mixup is considerably inefficient—generally, Mixup slows down the training by n times if it generates n new points for each original point pair. In this work, we propose BatchMixup, an improved mixup paradigm that generates new points scattered uniformly throughout the whole representation region of the mini-batch. Specifically, within a mini-batch, each example and its label will first learn a representation vector respectively, BatchMixup then generates n new points (including a new input representations and a new label representation) simultaneously by non-linearly interpolating all the examples in the same mini-batch. The new n points are expected to better identify the space represented by the mini-batch. Finally, the n mixed points will act as one batch to update the model.

Our model BatchMixup, as a batch-wise non-linear Mixup, shows advantages in two aspects. (i) Compared with the standard Mixup, BatchMixup further improves the representation learning in solving downstream NLP tasks, yielding better performance. (ii) BatchMixup works much more efficient than the conventional Mixup and other pair-wise mixup variants.

2 Related Work

Mixup was originally proposed in the computer vision community. The standard Mixup (Zhang et al., 2018) interpolates the raw pixels of each two images in a mini-batch. Verma et al. (2019) conducted interpolation in the hidden states of images. Guo et al. (2019b) discovered a limitation of Mixup, called "manifold intrusion", which is the conflict between the synthetic label of the mixed-up points and the labels of the original examples. They came up with "AdaMixup", an adaptive Mixup, where the mixing policies are automatically learned from the data using an additional network and objective function designed to avoid manifold intrusion. Other work tried to explain the work mechanisms of MIXUP from different threads, such as "MIXUP as directional adversarial training" (Archambault et al., 2019), "MIXUP training as the complexity reduction" (Kimura, 2020)

To date, only a couple of previous studies explored the effectiveness of the standard MIXUP in NLP. Guo et al. (2019a) tried two strategies: interpolating word embeddings or sentence embeddings generated by convolutional/recurrent neural networks. Sun et al. (2020) incorporated MIXUP into BERT (Devlin et al., 2019), the state of the art architecture in NLP. To improve the standard MIXUP, Guo (2020) added non-linearity to the MIXUP for text classification tasks. However, that non-linear MIXUP works on word embedding level, which is less applicable to Transformer-style (Vaswani et al., 2017) systems. All the work above are pair-wise mixup, this work is the first work that interpolates all the examples in the same mini-batch to cover the representation space better.

3 The Base Model: MIXUP

Given a pair of samples (x_i, y_i) and (x_j, y_j) from the original mini-batch (x: input, y: the one-hot label), the standard MIXUP (Zhang et al., 2018) generates a synthetic sample as follows.

$$\hat{x}_{ij} = \beta x_i + (1-\beta)x_j \tag{1}$$

$$\hat{y}_{ij} = \beta y_i + (1-\beta)y_j \tag{2}$$

where β is a mixing scalar, sampled from a Beta(α, α) distribution with a hyper-parameter α, for mixing both the inputs and the corresponding targets. The generated synthetic data are then fed into the model for training to minimize the loss function.

From the same mini-batch, the standard MIXUP will sample the β value n times so that totally n new mixed points for a sampled input pair will be generated sequentially. The model, as a result, will be updated n times more than the mixup-free model.

4 Our Model: BATCHMIXUP

BATCHMIXUP mixes all the samples in the same mini-batch on the level of hidden states generated by RoBERTa (Liu et al., 2019).[1]

[1]Please note that BATCHMIXUP also works for other deep neural encoders.

To start, we first think about how the standard text classifier works: For the labeled input (x_i, y_i), first RoBERTa (optionally with a multilayer perceptron block) generates a representation for x_i ("$v(x_i) \in \mathbb{R}^d$"), then $v(x_i)$ is fed to a logistic regression (LR) layer to classify to y_i. The LR layer has a weight matrix $W \in \mathbb{R}^{c\times d}$ where c is the class size and d is the dimension size of representations. Each row in W, i.e., $w_i \in \mathbb{R}^d$, can be treated as the representation vector of the class y_i. So, LR essentially uses the dot-product to derive the matching score ($s_i \in \mathbb{R}$) between the input x_i and the label y_i: $s_i = (v(x_i))^T \cdot w_i$.

For the same mini-batch of inputs $\{v(x_i)\}$ and labels $\{w_i\}$, BATCHMIXUP deploys the same mixing policy to interpolate the $\{v(x_i)\}$ and $\{w_i\}$.

We denote the whole batch of input representations $\{v(x_i)\}$ as $\mathcal{X} \in \mathbb{R}^{d\times b}$ where b is the batch size and the whole mixing policy for this batch is $\mathcal{M} \in \mathbb{R}^{n\times d\times b}$. To generate a single mixed point $\hat{x}_i \in \mathbb{R}^d$, the BATCHMIXUP uses the following mixing policy $\mathcal{M}[i] \in \mathbb{R}^{d\times b}$ ($i = 1, \cdots, n$) on $\mathcal{X}$, where each element of $\mathcal{M}[i]$ is independently sampled from a Beta(α, α) distribution:

$$\hat{x}_i = \sum_{\text{axis}=1} (\text{softmax}(\mathcal{M}[i]) \circ \mathcal{X}) \tag{3}$$

where $\circ$ is the Hadamard product. Equation 3 can be performed for all i values in [1,n] simultaneously; this means the original batch input $\mathcal{X}$ is transformed into a new batch of mixed input $\hat{\mathcal{X}} \in \mathbb{R}^{n\times d}$.

Similarly, the same mixing policy $\mathcal{M}$ is applied to the batch of label representations, denoted as $\mathcal{Y} \in \mathbb{R}^{d\times b}$ ($\mathcal{Y} = \{w_i\}$):

$$\hat{y}_i = \sum_{\text{axis}=1} (\text{softmax}(\mathcal{M}[i]) \circ \mathcal{Y}) \tag{4}$$

Each ($\hat{x}_i$, $\hat{y}_i$) ($i = 1, \cdots, n$) is a newly mixed point. All $\{(\hat{x}_i, \hat{y}_i)\}$ can be generated in parallel and are scattered throughout the space represented by $\mathcal{X}$.

For training, we minimize the negative-dot-product loss between the mixed input and the mixed label. In testing, an input x_i still compares with all classes $\{y_i\}$ by dot-product between $v(x_i)$ and all $\{w_i\}$ to find the best class.

5 Experiments

In experiments, we check the effectiveness of our approach in NLP tasks with two settings: one is

#training data		entailment	relation	intent
100%	RoBERTa-large	80.91±1.47	87.24±1.52	93.77±0.44
	w/ MIXUP	82.03±1.69	87.91±0.34	93.41±1.24
	w/ Nonlinear Mixup	82.98±1.17	88.41±0.73	94.29±1.01
	w/ BATCHMIXUP	**83.56±0.90**	**90.04±0.33**	**94.94±0.20**
75%	RoBERTa-large	80.07±1.27	87.49±0.31	90.31±1.11
	w/ MIXUP	80.53±1.31	87.87±0.12	92.44±1.98
	w/ Nonlinear Mixup	80.90±0.39	87.66±0.81	93.00±1.84
	w/ BATCHMIXUP	**82.00±0.91**	**88.41±0.54**	**93.55±1.58**
50%	RoBERTa-large	77.50±0.82	82.83±4.92	85.11±2.88
	w/ MIXUP	75.19±6.07	87.00±0.18	87.09±2.06
	w/ Nonlinear Mixup	77.66±1.90	87.79±0.29	88.13±2.92
	w/ BATCHMIXUP	**79.47±2.07**	**88.29±0.22**	**90.13±2.20**
25%	RoBERTa-large	70.92±4.04	78.95±0.33	80.95±2.22
	w/ MIXUP	71.70±5.66	82.02±0.59	84.95±1.19
	w/ Nonlinear Mixup	72.14±4.27	83.71±0.12	85.44±2.02
	w/ BATCHMIXUP	**74.36±2.82**	**86.66±0.26**	**87.71±1.38**
1%	RoBERTa-large	50.41±0.29	3.41±0.74	42.21±5.54
	w/ MIXUP	**51.74±0.84**	49.93±2.71	50.21±3.21
	w/ Nonlinear Mixup	51.41±0.43	55.29±4.78	52.21±1.49
	w/ BATCHMIXUP	51.46±1.43	**60.29±2.18**	**55.21±1.77**
random or majority baseline		50.16	1.31	1.29

Table 1: Experimental results on three NLP tasks: textual entailment (RTE (Dagan et al., 2005; Wang et al., 2019)), relation classification (FewRel (Han et al., 2018)) and intent classification (BANKING77 (Casanueva et al., 2020)). We decrease the size of training data from 100% to 1% with random sampling. All numbers are averaged over three random seeds.

full-shot setting that trains on the regular full training data; the other is few-shot setting that train with limited training data. Unfortunately, prior work about mixup never evaluated on few-shot scenarios.

Tasks. We evaluate on the following three tasks.

• **Textual Entailment.** Textual entailment is a task that figures out the truth value of a hypothesis sentence given a premise sentence (Dagan et al., 2005). This is a binary classification ("entailment" or "non-entailment") problem where the input is a sentence pair. We use the GLUE RTE (Wang et al., 2019) benchmark which has 2,490/277/2,999 examples in train/dev/test. The smaller size of this dataset (compared MNLI (Williams et al., 2018) for example) makes it a good testbed for data augmentation techniques.

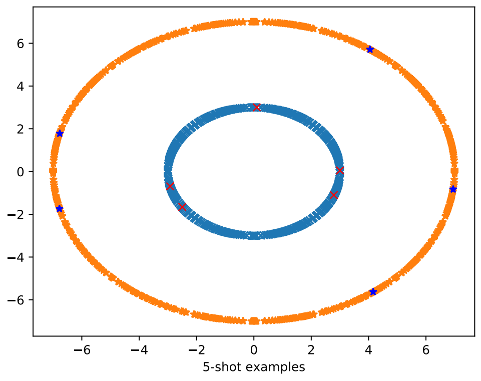
• **Relation Classification.** FewRel (Han et al., 2018) is a large-scale relation classification dataset. It has 100 relation types, each with 700 labeled examples. The original FewRel relation set was split by 64/16/20 for developing meta-learning techniques which only allow a test instance to search for its relation type within the 20 candidates. This is not a practical setting because (i) in relation detection, an input should search for a label in the entire space of defined relations, (ii) we should always define a "None" type in this problem because most span pairs in the input actually do not have a relation. Since the test relations of FewRel is not publicly available, we use the 64+16=80 relations as the entire relation set, in which 5 relations are treated "None" (So, basically this is a regular "75+None" setting).

• **Intent Classification ("intent").** We use the benchmark BANKING77[2] (Casanueva et al., 2020), which is single-domain intent detection dataset comprising 13,083 annotated examples over 77 intents (average: 170 examples per intent). Each intent class is described by a short name, such as "get physical card", "lost or stolen card", etc.

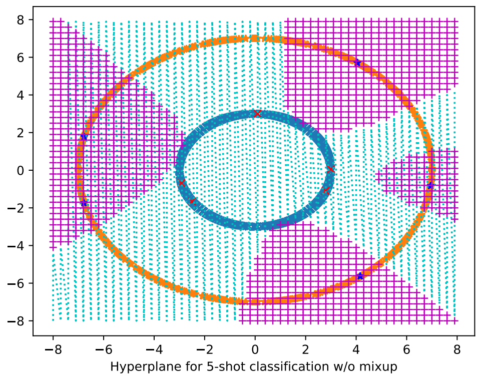
Baselines. The augmentation-free system we use for above tasks consists of a RoBERTa[3] encoder

[2]https://github.com/PolyAI-LDN/task-specific-datasets

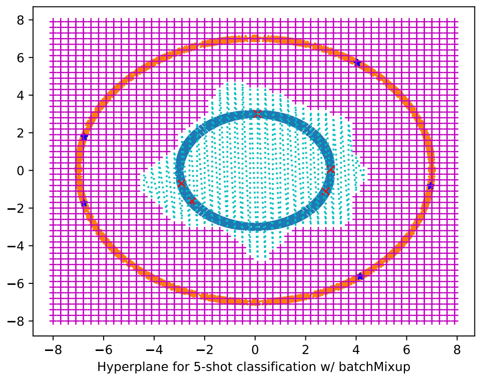
[3]We used the pretrained "RoBERTa-large"



(a) Two classes (orange vs. blue), 5-shot for each



(b) Hyperplane after training w/o mixup; it cannot distinguish the two classes clearly.



(c) Hyperplane after training w/ BATCHMIXUP, it separates the two classes very well

Figure 1: Visualization analysis

and a final logistic regression layer. Based on this RoBERTa system, we compare our system BATCHMIXUP with (i) the standard MIXUP (Zhang et al., 2018; Sun et al., 2020), and (ii) non-linear MIXUP (Guo, 2020). Both baselines conduct data interpolation in the hidden states output by the RoBERTa.

All systems are implemented through the Huggingface's Transformers package.[4]

Results and Analysis. Table 1 lists the main results. We notice that our approach RoBERTa+BATCHMIXUP consistently outperforms the baselines MIXUP and non-linear MIXUP. In "1% entailment", none of systems really worked—all system results are around the majority baseline. This is because that the RTE task is very challenging with over limited annotations. With 2.5K $\times$ 1%=25 labeled examples, the "RoBETTa" cannot learn any useful representations. This is in line with the observations in (Yin et al., 2020) which showed that few-shot RTE (when $k \in \{1, 3, 5, 10\}$) will make RoBERTa fail. So, we conclude that when a system is close to random guess, adding mixup is not helpful. In this situation, maybe using other conventional data augmentation skills makes more sense as the representation learning of synthetic data and that of the original data are decoupled.

To further study how BATCHMIXUP works, we simulate the classification process with a toy experiment: we generate a large amount of 2-dimensional data in Gaussian distributions for two classes (Figure 1(a)), and randomly sample 5 examples for each class to conduct 5-shot classification. We used a MLP as the classifier, trained 100 epochs. Comparing the final hyperplane of training with BATCHMIXUP with that of training without MIXUP, we can observe that BATCHMIXUP can improve the training considerably.

Last but not least, the training of same epochs for "w/ MIXUP", "w/ Nonlinear MIXUP" takes much longer than our system "w/ BATCHMIXUP". For example, when all systems separately run on a GPU Tesla V100, our system BATCHMIXUP and the baseline "RoBERTa-large" both take about 1.5min to finish one epoch on RTE, but "w/ MIXUP" and "w/ Nonlinear MIXUP" will take ~20mins if β is sampled 15 times per point pair.

6 Conclusion

In this work, we proposed a novel MIXUP model, named BATCHMIXUP, to improve the text classifier. Different with prior MIXUP variants, which always interpolate random two points, our system interpolates all the hidden states in the mini-batch. The mixed points by our system are able to better cover the space expressed by the minibatch. The experiments and visualization analysis both show the effectiveness of our model BATCHMIXUP.

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

References

Guillaume P. Archambault, Yongyi Mao, Hongyu Guo, and Richong Zhang. 2019. Mixup as directional adversarial training. *CoRR*, abs/1906.06875.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *ACL*, pages 567–573.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *AAAI*, pages 4044–4051.

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

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *AAAI*, pages 3714–3722.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, pages 4803–4809.

Masanari Kimura. 2020. Mixup training as the complexity reduction. *CoRR*, abs/2006.06231.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT*, pages 452–457.

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

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S. Yu, and Lifang He. 2020. Mixup-transfomer: Dynamic data augmentation for NLP tasks. In *COLING*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

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 *ICML*, pages 6438–6447.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *ICLR*.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *EMNLP*, pages 8229–8239.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *ICLR*.