Augmenting Data with Mixup for Sentence Classification: An Empirical Study

Hongyu Guo
National Research Council Canada
1200 Montreal Road, Ottawa
hongyu.guo@nrc-cnrc.gc.ca

Yongyi Mao
Electrical Engineering & Computer Science
University of Ottawa, Ottawa, Ontario
yymao@eecs.uottawa.ca

Richong Zhang
BDBC, School of Computer Science and Engineering
Beihang University, Beijing, China
zhangrc@act.buaa.edu.cn

Abstract

Mixup (Zhang et al., 2017), a recent proposed data augmentation method through linearly interpolating inputs and modeling targets of random samples, has demonstrated its capability of significantly improving the predictive accuracy of the state-of-the-art networks for image classification. However, how this technique can be applied to and what is its effectiveness on natural language processing (NLP) tasks have not been investigated. In this paper, we propose two strategies for the adaption of Mixup on sentence classification: one performs interpolation on word embeddings and another on sentence embeddings. We conduct experiments to evaluate our methods using several benchmark datasets. Our studies show that such interpolation strategies serve as an effective, domain independent data augmentation approach for sentence classification, and can result in significant accuracy improvement for both CNN (Kim, 2014b) and LSTM (Hochreiter and Schmidhuber, 1997) models.

1 Introduction

Deep learning models have achieved state-of-the-art performance on many NLP applications, including parsing (Socher et al., 2011), text classification (Kim, 2014b; Tai et al., 2015), and machine translation (Sutskever et al., 2014). These models typically have millions of parameters, thus require large amounts of data for training in order for over-fit avoidance and better model generalization. However, collecting a large annotated data samples is time-consuming and expensive.

One technique aiming to address such a data hungry problem is automatic data augmentation. That is, synthetic data samples are generated as additional training data for regularizing the learning models. Data augmentation has been actively and successfully used in computer vision (Simard et al., 1998; Krizhevsky et al., 2017; Zhang et al., 2017) and speech recognition (Jaitly and Hinton, 2015; Ko et al., 2015). Most of these methods, however, rely on human knowledge for label-invariant data transformation, such as image scaling, flipping and rotation. Unlike in image, there is, however, no simple rule for label-invariant transformation in natural languages. Often, slight change of a word in a sentence can dramatically alter the meaning of the sentence. To this end, popular data augmentation approaches in NLP aim to transform the text with word replacements with either synonyms from handcrafted ontology (e.g., WordNet (Zhang et al., 2015)) or word similarity (Wang and Yang, 2015; Kobayashi, 2018). Such synonym-based transformation, however, can be applied to only a portion of the vocabulary due to the fact that words having exactly or nearly the same meanings are rare. Some other NLP data augmentation methods are often devised for specific domains thus makes them difficult to be applied to other domains (Sahin and Steedman, 2018).

Recently, a simple yet extremely effective augmentation method Mixup (Zhang et al., 2017) has been proposed and shown superior performance on enhancing the accuracy of image classification models. Through linearly interpolating pixels of random image pairs and their training targets, Mixup generates synthetic examples for training. Such training has been shown to act as an effective model regularization strategy for image classification networks.

How Mixup can be applied to and what is its effectiveness on NLP tasks? We here aim to answer these questions in this paper. In specific, we propose two strategies for the application of Mixup on sentence classification: one performs interpolation on word embedding and another on

sentence embedding. We empirically show that such interpolation strategies serve as a simple, yet effective data augmentation method for sentence classification, and can result in significant accuracy improvement for both CNN (Kim, 2014b) and LSTM (Hochreiter and Schmidhuber, 1997) models. Promisingly, unlike traditional data augmentation in NLP, these interpolation based augmentation strategies are domain independent, exclusive of human knowledge for data transformation, and of low additional computational cost.

2 Data Augmentation through Sentence Interpolation

2.1 MixUp for Image Classification

Zhang et al. (Zhang et al., 2017) proposed the Mixup method for image classification. The idea is to generate a synthetic sample by linearly interpolating a pair of training samples as well as their modeling targets. In detail, consider a pair of samples $(x^i; y^i)$ and $(x^j; y^j)$, where x denotes the input and y the one-hot encoding of the corresponding class of the sample. The synthetic sample is generated as follows.

$$\widetilde{x}^{ij} = \lambda x^i + (1-\lambda)x^j \quad (1)$$

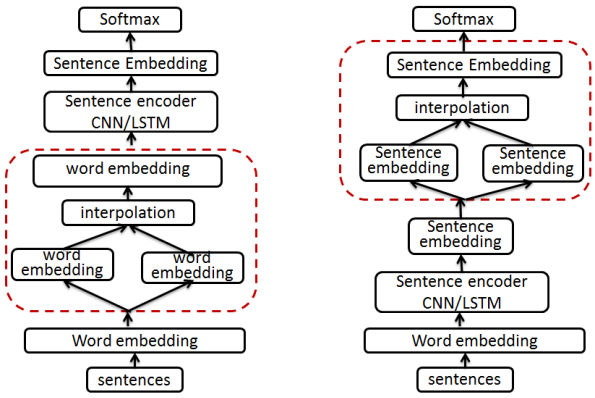
$$\widetilde{y}^{ij} = \lambda y^i + (1-\lambda)y^j \quad (2)$$

where λ is the mixing policy or mixing-ratio for the sample pair. λ is sampled from a Beta(α, α) distribution with a hyper-parameter α. It is worth noting that, when α equals to one, then the Beta distribution is equivalent to an uniform distribution. The generated synthetic data are then fed into the model for training to minimize the loss function such as the cross-entropy function for the supervised classification. For an efficient computation, the mixing happens by randomly pick one sample and then pairs it up with another sample drawn from the same mini-batch.

2.2 Adaptation of Mixup for Sentence Classification

Unlike image which is consist of pixels, sentence is composed of a sequence of words. Therefore, a sentence representation is often constructed to aggregate information from a sequence of words. Specifically, in a standard CNN or LSTM model, a sentence is first represented by a sequence of word embeddings, and then fed into a sentence encoder. The most popular such encoders are CNN and LSTM. The sentence embedding generated by CNN or LSTM are then passed through a soft-max layer to generate the predictive distribution over the possible target classes for predictions.

Figure 1: Illustration of wordMixup (left) and senMixup (right), where the added part to the standard sentence classification model is in red rectangle.



To this end, we propose two variants of Mixup for sentence classification. The first one conducts sample interpolation in the word embedding space (denoted as wordMixup), and the second on the final hidden layer of the network before it is passed to a standard soft-max layer to generate the predictive distribution over classes (denoted as senMixup). The two models are illustrated in Figure 1, where the standard CNN (Kim, 2014b) or LSTM (Hochreiter and Schmidhuber, 1997) model for sentence classification corresponds to the one without the red rectangle.

In specific, in the wordMixup, all sentences are zero padded to the same length and then interpolation is conducted for each dimension of each of the words in a sentence. Given a piece of text, such as a sentence with N words, it can be represented as a matrix $B \in R^{N \times d}$. Each row t of the matrix corresponds to one word (denoted B_t), which is represented by a d-dimensional vector as provided either by a learned word embedding table or being randomly generated. Formally, consider a pair of samples $(B^i; y^i)$ and $(B^j; y^j)$, where B^i and B^j denotes the embedding vectors of the input sentence pairs and y^i and y^j denote the corresponding class labels of the samples using one-hot representation. Then for the t^{th} word in the sentence, linear interpolation process can be formulated as:

$$\widetilde{B}_t^{ij} = \lambda B_t^i + (1-\lambda)B_t^j \quad (3)$$

$$\widetilde{y}^{ij} = \lambda y^i + (1-\lambda)y^j \quad (4)$$

The resulting new sample $(\widetilde{B}^{ij}; \widetilde{y}^{ij})$ is then used for training.

In senMixup, the hidden embeddings (with the same dimension) for the two sentences are first generated by an encoder such as CNN or LSTM. Next, the pair of sentence embeddings are interpolated linearly. In specific, let f denote the sentence encoder, then a pair of sentences B^i and B^j will be first encoded into a pair of sentence embeddings $f(B^i)$ and $f(B^j)$, respectively. In this case, the mixing is conducted for each k^{th} dimension of the sentence embedding, as follows.

$$\widetilde{B}^{ij}_{\{k\}} = f(B^i)_{\{k\}} + (1-\lambda) f(B^j)_{\{k\}} \quad (5)$$

$$\widetilde{y}^{ij} = \lambda y^i + (1-\lambda) y^j \quad (6)$$

Finally, the embedding vector $\widetilde{B}^{ij}$ will be passed to a softmax layer to produce a distribution over the possible target classes. For training, we use multi-class cross entropy loss.

3 Experiment

We evaluate the proposed methods with five benchmark tasks for sentence classifications.

- **TREC** is a question dataset to categorize a question into six question types (Li and Roth, 2002).
- **MR** is a movie review dataset for detecting positive/negative reviews (Pang and Lee, 2005).
- **SST-1** is the Stanford Sentiment Treebank with five categories labels (Socher et al., 2013).
- **SST-2** dataset is the same as SST-1 but with neutral reviews removed and binary labels.
- **Subj** is a subjectivity detection dataset for classifying a sentence as being subjective or objective (Pang and Lee, 2004).

The summary of the data sets is presented in Table 1. Note that, for comparison purposes on the SST-1 and SST-2 datasets, following (Kim, 2014b; Tai et al., 2015), we trained the models using both phrases and sentences, but only evaluated sentences at test time.

We evaluate our wordMixup and senMixup using both CNN and LSTM for sentence classification. We implement the CNN model exactly as reported in (Kim, 2014b,a). For LSTM, we just simply replace the convolution/pooling components in CNN with standard LSTM units as implemented in (Abadi et al., 2016). The final feature map of CNN and the final state of LSTM are passed to a logistic regression classifier for label prediction.

To evaluate our models in terms of their regularization effects on the training, we present four word embedding settings: random and trainable word embedding (denoted **RandomTune**), random and fix word embedding (denoted **RandomFix**), pre-trained and tunable word embedding (denoted **PretrainTune**), and pre-trained fix word embedding (denoted **PretrainFix**).

Data	c	l	N	V	Test
TREC	6	10	5952	9592	500
SST-1	5	18	11855	17836	2210
SST-2	2	19	9613	16185	1821
Subj	2	23	10000	21323	CV
MR	2	20	10662	18765	CV

Table 1: Summary for the datasets after tokenization. c: number of target labels. l: average sentence length. N: number of samples. V: vocabulary size. Test: test set size (CV means no standard train/test split was provided and thus 10-fold CV was used).

In our experiments, following the exact implementation and settings in (Kim, 2014a) we use filter sizes of 3, 4, and 5, each with 100 feature maps; dropout rate of 0.5 and L2 regularization of 0.2 for the baseline CNN and LSTM. For datasets without a standard dev set we randomly select 10% of training data as dev set. Training is done through Adam (Kingma and Ba, 2014) over mini-batches of size 50. The pre-trained word embeddings are 300 dimensional GloVe (Pennington et al., 2014). The hidden state dimension for LSTM is 100.

For senMixup and wordMixup, the mixing policy α is set to the default value of one. Also, following the original Mixup (Zhang et al., 2017), we did not use dropout or L2 constraint for the wordMixup and senMixup models.

We train each model 10 times each with 20000 steps, and compute their mean test errors and standard deviations.

3.1 Main Results

RandomTune has the largest number of parameters, compared to RandomFix, PretrainTune, and PretrainFix, and thus requires a strong regularization method to avoid over-fit the training data. We, therefore, focus our experiments on the Random-

RandomTune	Trec	SST-1	SST-2	Subj	MR
CNN- KIM Impl. (Kim, 2014b)	91.2	45.0	82.7	89.6	76.1
CNN- HarvardNLP Impl. [1]	88.2	42.2	83.5	89.2	75.9
CNN - Our Impl.	90.2±0.20	43.6±0.19	82.3±0.47	90.6±0.45	75.5±0.36
CNN+wordMixup	90.9±0.42	45.2±0.90	82.8±0.45	**92.9±0.41**	**78.0±0.39**
CNN+senMixup	**92.1±0.31**	**45.2±0.22**	**83.0±0.35**	92.7±0.38	77.9±0.76

Table 2: Accuracy (%) of the testing methods using CNN (with randomly initialized, trainable embeddings). We report mean scores over 10 runs with standard deviations (denoted ±). Best results highlighted in Bold.

RandomTune	Trec	SST-1	SST-2	Subj	MR
LSTM-StanfordNLP Impl. (Tai et al., 2015)	N/A	46.4	84.9	N/A	N/A
LSTM-AgrLearn Impl. (Guo et al., 2018a)	N/A	N/A	N/A	90.2	76.2
LSTM - Our Impl.	86.5±0.61	45.9±0.58	84.4±0.35	90.9±0.42	77.2±0.75
LSTM + wordMixup	**90.5±0.50**	48.2±0.18	86.3±0.35	**93.1±0.49**	**78.0±0.33**
LSTM + senMixup	89.4±0.40	**48.3±0.77**	**86.7±0.33**	91.9±0.34	77.9±0.33

Table 3: Accuracy (%) obtained by the testing methods using LSTM (with randomly initialized, trainable embeddings). We report mean scores over 10 runs with standard deviations (denoted ±). Best results highlighted in Bold.

Tune setting. The results on the RandomTune setting are presented in Table 2.

The results in Table 2 show that wordMixup and senMixup provide good regularization to CNN, resulting in accuracy improvement on all the five testing datasets. For example, in SST-1 and MR, the relative improvement was over 3.3%. Interestingly, both wordMixup and senMixup failed to significantly improved over the baseline against the SST-2 dataset; with senMixup slightly outperformed the baseline with only 0.7%. Also, results in Table 2 suggest that senMixup and wordMixup were quite competitively, in terms of predicticve performance obtained, against the five testing datasets. For example, on the Trec dataset, senMixup outperformed senMixup with 1.2%, but for the other four datasets, the two methods obtained very similiar predictive accuracy.

Regularization Effect We plot the training and testing cross-entropy loss across the first 12K training steps on the MR dataset in Figure 1. Figure 1 shows that with (top-left subfigure) or without (top-right subfigure) dropout, the training loss of CNN drops to zero quickly and provides no training signal for further tuning the networks. On the otherhand, the training loss of wordMixup (bottom-right subfigure) keeps above zero during the training, continuously provide training signal for the network learning. Also, the training loss curve of senMixup (bottom-left subfigure) maintains a relatively high level, allowing the model to keep tuning. The relatively higher training loss of both wordMixup and senMixup is due to the much larger space of the mixed samples, thus preventing the model from being over-fitted by limited number of individual examples

LSTM Networks as Sentence Encoder We also evaluate the effect of using LSTM as the sentence encoder. Results in Table 3 show that, similar to the case of using CNN as sentence encoder, wordMixup and senMixup with LSTM as encoder also improved the predictive performance of the baseline models. For example, the largest improvements came from the Trec and SST-1 cases (with relative improvement of 4.62% and 5.22%), which have six and five classes, respectively. Results in the table also suggest that, on the Subj dataset, wordMixup outperformed senMixup with 1.2%, but for the other four datasets, the two methods performed comparably well.

One notable fact when compared with the CNN-based models as presented in Table 2 is that, against the SST-2 data sets, both wordMixup and senMixup with LSTM here were able to improve over the baseline with about 2%.

Results for RandomFix, PretrainTune, and PretrainFix Results for the settings of RandomFix, PretrainTune, and PretrainFix are presented in Tables 4, 5, and 6, respectively. Results in these three tables further confirm that data augmentation through wordMixup and senMixup can improve the predictive performance of the base mod-

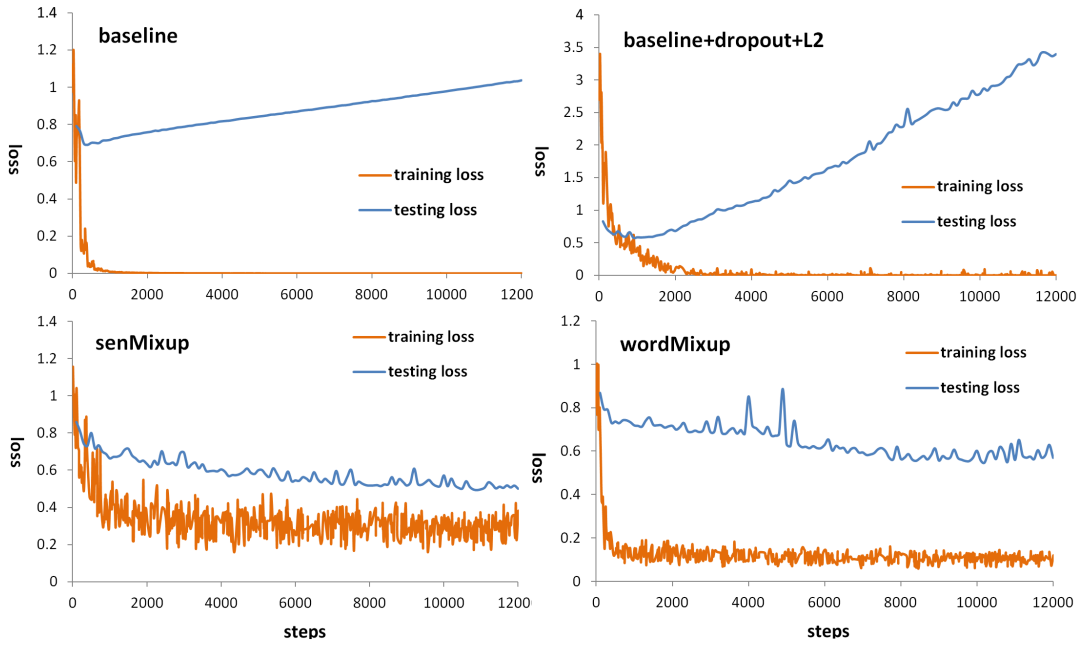


Figure 2: Training and testing entropy loss obtained by the baseline CNN without dropout (top-left), baseline CNN with dropout and L2 (top-right), wordMixup (bottom-right), and senMixup (bottom-left).

RandomFix	Trec	SST-1	SST-2	Subj	MR
CNN	88.4±0.52	40.3±0.77	**80.4±0.17**	88.2±0.50	72.9±0.74
CNN + wordMixup	**90.9±0.58**	40.5±1.17	77.5±0.33	89.3±0.47	**74.2±1.15**
CNN + senMixup	88.8±1.10	**41.0±0.64**	77.6±0.76	**90.5±0.36**	72.6±0.67

Table 4: Accuracy (%) obtained by the testing methods using CNN with randomly initialized and fixed embeddings. Best results highlighted in Bold.

PretrainTune	Trec	SST-1	SST-2	Subj	MR
CNN	92.1±0.12	46.3±0.35	86.9±0.49	94.4±0.36	79.8±0.60
CNN + wordMixup	93.7±0.80	48.2±0.91	87.1±0.26	94.7±0.45	**81.3±0.28**
CNN + senMixup	**93.3±0.23**	**48.6±0.23**	**87.2±0.35**	**94.9±0.34**	80.6±0.56

Table 5: Accuracy (%) of the testing methods using CNN with pre-trained GloVe and trainable embeddings. Best results highlighted in Bold.

PretrainFix	Trec	SST-1	SST-2	Subj	MR
CNN	92.0±0.2	44.6±0.56	**85.7±0.33**	94.5±0.36	79.7±0.68
CNN + wordMixup	94.2±0.52	**46.6±0.85**	84.5±0.54	94.3±0.23	79.7±0.52
CNN + senMixup	**94.8±0.35**	46.5±0.23	84.7±0.48	**95.0±0.22**	**80.3±0.57**

Table 6: Accuracy (%) obtained by the testing methods using CNN with pretrained GloVe and fixed embeddings. Best results highlighted in Bold.

els, except on the SST-2 dataset. On the SST-2 data set, both wordMixup and senMixup degraded the predictive accuracy of the baseline when the word embeddings were not allowed to be tuned during training. With learnable word embeddings, although both wordMixup and senMixup failed to significantly improve over the baseline on this dataset, but they did obtain similar predictive performance as the baseline. In short, our experiments here suggest that when the word embeddings are tuned, both wordMixup and senMixup are able to improve the predictive accuracy of the base models.

4 Conclusion and Future Work

Inspired by the success of Mixup, a simple and effective data augmentation method through sample interpolation for image recognition, we investigated two variants of Mixup for sentence classification. We empirically show that they can improve the accuracy upon both CNN and LSTM sentence classification models. Interestingly, our studies here show that such interpolation strategies can serve as an effective, domain independent regularizer for overfitting avoidance for sentence classification.

We plan to investigate some lately proposed variants of Mixup, such as Manifold Mixup (Verma et al.), where interpolation is performed in a randomly selected layer of the networks, and AdaMixup (Guo et al., 2018b), which addresses the manifold intrusion issues in Mixup. We are also interested in questions such as what the mixed sentences look like and why interpolation works for sentence classification.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2018a. Aggregated learning: A vector quantization approach to learning with neural networks. *CoRR*, abs/1807.10251.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2018b. Mixup as locally linear out-of-manifold regularization. *CoRR*, abs/1809.02499.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, (8):1735–1780.

Navdeep Jaitly and Geoffrey E. Hinton. 2015. Vocal tract length perturbation (vtlp) improves speech recognition.

Kim. 2014a. Url https://github.com/yoonkim/cnn_sentence.

Yoon Kim. 2014b. Convolutional neural networks for sentence classification. In *EMNLP2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3586–3589.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT 2018*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, 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 Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Gözde Gül Sahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *EMNLP 2018*, pages 5004–5009.

Patrice Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1998. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 239–27.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Richard Socher, Alex Perelygin, Jean Y. 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 Conference on Empirical Methods in Natural Language Processing*,

EMNLP '13, Seattle, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Kai Sheng Tai, Richard Socher, and Christopher Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of ACL (to appear)*.

Vikas Verma, Alex Lamb, Christopher Beckham, Aaron C. Courville, Ioannis Mitliagkas, and Yoshua Bengio. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *CoRR*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *EMNLP2015*.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15.