SSMix: Saliency-Based Span Mixup for Text Classification

Soyoung Yoon[1,2*†] **Gyuwan Kim**[1*] **Kyumin Park**[2]
[1]Clova AI, Naver Corp. [2]KAIST
{soyoungyoon,pkm9403}@kaist.ac.kr, gyuwan.kim@navercorp.com

Abstract

Data augmentation with mixup has shown to be effective on various computer vision tasks. Despite its great success, there has been a hurdle to apply mixup to NLP tasks since text consists of discrete tokens with variable length. In this work, we propose *SSMix*, a novel mixup method where the operation is performed on input text rather than on hidden vectors like previous approaches. *SSMix* synthesizes a sentence while preserving the locality of two original texts by span-based mixing and keeping more tokens related to the prediction relying on saliency information. With extensive experiments, we empirically validate that our method outperforms hidden-level mixup methods on a wide range of text classification benchmarks, including textual entailment, sentiment classification, and question-type classification. Our code is available at https://github.com/clovaai/ssmix.

1 Introduction

Data augmentation gains popularity in natural language processing (NLP) (Feng et al., 2021) due to the expensive cost of data collection. Some of them are based on simple rules (Wei and Zou, 2019) and models (Edunov et al., 2018; Ng et al., 2020) to generate similar text. Augmented samples are trained jointly with original samples by a standard way or advanced training methods (Zhu et al., 2019; Park et al., 2021). On the other hand, mixup (Zhang et al., 2018) interpolates input texts and labels for the augmentation.

Training with mixup and its variants become a popular regularization method in computer vision to improve the generalization of neural networks. Mixup approaches are categorized into input-level mixup (Yun et al., 2019; Kim et al., 2020; Walawalkar et al., 2020; Uddin et al., 2021) and hidden-level mixup (Verma et al., 2019) depending on the location of the mix operation. Input-level mixup is a more prevalent approach than hidden-level mixup because of its simplicity and the ability to capture locality, leading to better accuracy.

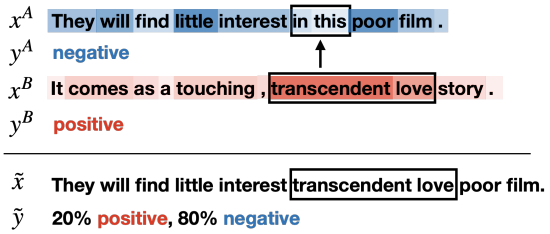


Figure 1: Illustration of *SSMix*. Two data samples x^A and x^B are labeled negative and positive respectively for sentiment classification task. For each token, saliency maps are visualized where darker concentration of colors mean higher contribution to corresponding label. We select the least salient span from x^A and replace it with the most salient span from x^B. The output results in $\tilde{x} = mixup(x^A, x^B)$. We also assign $\tilde{y}$ by the mixup ratio λ. In this example, λ is set to 0.2 as the span length is 2 out of 10.

Applying mixup in NLP is more challenging than in computer vision because of the discrete nature of text data and variable sequence lengths. Therefore, most previous attempts on mixup for texts (Guo et al., 2019; Chen et al., 2020) apply mixup on hidden vectors like embeddings or intermediate representations. However, input-level mixup might have an advantage over hidden-level mixup with a similar intuition from computer vision. This motivation encourages us to examine input-level mixup approaches for text data.

In this work, we propose *SSMix* (Fig 1), a novel input-level spanwise mixup method considering the saliency of spans. First, we conduct a mixup by replacing a span of contiguous tokens with a span in another text, which is inspired from CutMix

*Equal contribution.
†Work done during the internship at Clova AI.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3225–3234

(Yun et al., 2019), to preserves the locality of two source texts in the mixed text. Second, we select a span to be replaced and to replace based on saliency information to make the mixed text contain tokens more related to output prediction, which may be semantically important. Our input-level method is different from hidden level mixup methods in that while current hidden level mixup methods linear interpolate original hidden vectors, our method mix tokens on the input level, resulting in a *nonlinear* output. Also, we utilize saliency values to select span from each sentence and discretely define the length of span and mixup ratio, which is outside the hidden level.

SSMix has empirically proven effective through extensive experiments on a wide range of text classification benchmarks. Especially, we prove that input-level mixup methods generally outperform hidden-level methods. We also show the importance of using saliency information and restricting token selection in span-level when conducting our method via ablation study.

2 *SSMix*

We propose *SSMix* to synthesize a new text $\tilde{x}$ by replacing a span x_S^A from one text x^A into another span x_S^B from another text x^B based on saliency information. Also, we have to set a new label $\tilde{y}$ for $\tilde{x}$ using y^A and y^B which are one-hot labels corresponding to x^A and x^B, respectively. Consequently, we can additionally use this generated virtual sample $(\tilde{x}, \tilde{y})$ for training.

Saliency Saliency measures how each portion of data (in this case, tokens) affects the final prediction. Gradient-based methods (Simonyan et al., 2013; Li et al., 2016) are widely used for the saliency computation. We compute the gradient of classification loss $\mathcal{L}$ with respect to input embedding e, and use its magnitude as the saliency: i.e., $s = \|\partial\mathcal{L}/\partial e\|_2$. We apply the L2 norm to obtain the magnitude of a gradient vector, which becomes a saliency of each token similar to PuzzleMix (Kim et al., 2020).

Mixing text Text data x^A and x^B are discrete token sequences. Using saliency scores as explained earlier, we can find the least salient span in x^A with a length l_A as x_S^A and the most salient span in x^A with a length l_B as x_S^B. We set $l_A = l_B = max(min([\lambda_0|x^A|], |x^B|), 1)$ given a prior mixup ratio λ_0. Then, final $\tilde{x}$ becomes the concatenation of $(x_L^A; x_S^B; x_R^A)$ where x_L^A and x_R^A are tokens located to the left and the right side of x_S^A respectively in the original text x^A.

Algorithm 1 Mixup loss calculation

procedure SSMIX_LOSS($x^A, x^B, y^A, y^B, \lambda$)
  $\tilde{x} \leftarrow SSMix(x^A, x^B)$
  $logit \leftarrow model(\tilde{x})$
  $loss^A \leftarrow CrossEntropy(logit, y^A)$
  $loss^B \leftarrow CrossEntropy(logit, y^B)$
  $total\_loss \leftarrow loss^A * \lambda + loss^B * (1 - \lambda)$
return $total\_loss$
end procedure

Same span length We set the length of the original (l_A) and replaced (l_B) span to be the same, since allowing different length of spans would result in redundant and ambiguous mixup variations. Also, calculating the mixup ratio between different span length would be too complex. This same-size replacement strategy is also adopted in (Yun et al., 2019) and (Uddin et al., 2021). In situations where span length is the same, our method maximizes the effect of saliency. Since *SSMix* doesn't restrict the position of tokens, we can pick the *most* salient span and replace it with *least* salient span on the other text.

Mixing label We set mixup ratio λ for label as $\lambda = |x_S^B|/|\tilde{x}|$. Since λ is recalculated by counting the number of tokens in the span, it may differ from λ_0. We set the label of $\tilde{x}$ to $\tilde{y} = (1 - \lambda)y^A + \lambda y^B$. Algorithm 1 shows how we utilize the original sample pairs to compute the mixup loss for augmented samples. We calculate the cross-entropy loss of the augmented output logit with respect to the original target label of each sample and combine them by weighted sum, which is similar to the original implementation of (Zhang et al., 2018).[1] Therefore, applying *SSMix* is independent of the total number of labels of the classification dataset. On any dataset, output label ratio is calculated by linear combination of *two* original labels.

Paired sentence tasks For tasks requiring a pair of texts as an input such as textual entailment and similarity classification, we conduct mixup in a pairwise manner and calculate the mixup ratio by aggregating token counts in each mixup result. Denoting $x^A = (p^A, q^A)$, $x^B = (p^B, q^B)$, and

[1] https://github.com/hongyi-zhang/mixup/blob/master/cifar/utils.pyL34

Dataset	Task	# Label	Size
SST-2	Sentiment	2	67k / 1.8k
QQP	Paraphrase	2	364k / 391k
MNLI	NLI	3	393k / 20k
QNLI	QA/NLI	3	105k / 5.4k
RTE	NLI	2	2.5k / 3k
MRPC	Paraphrase	2	3.7k / 1.7k
TREC-coarse	Classification	6	5.5k / 500
TREC-fine	Classification	47	5.5k / 500
ANLI	NLI	3	162.8k / 3.2k / 3.2k

Table 1: Dataset name, task, number of total labels, and dataset size of datasets we used as benchmark. Task column describes the objective of each dataset. ANLI dataset shows aggregated dataset statistics among different rounds. GLUE tasks report the size as (train / validation) format, TREC reports (train / test) and ANLI reports (train / validation / test).

$\tilde{x} = (\tilde{p}, \tilde{q})$, we define mixup of paired sentence data as $\tilde{x} = (mixup(p^A, p^B), mixup(q^A, q^B))$. Here, we set the mixup ratio on paired sentence tasks as $\lambda = (|p_S| + |q_S|)/(|\tilde{p}| + |\tilde{q}|)$, where p_S and q_S are replacing spans of independent mixup operations. Illustation is available in Appendix B.3.

3 Experimental Setup

3.1 Dataset

As listed in table 1, to evaluate the effectiveness of *SSMix*, we perform experiments on various text classfication benchmarks: six datasets in GLUE benchmark (Wang et al., 2018), TREC (Li and Roth, 2002; Hovy et al., 2001), and ANLI (Nie et al., 2020). Two of them are single sentence classification tasks, and six of them are sentence pair classification tasks. All datasets are extracted from HuggingFace datasets library.[2]

For GLUE, we use SST-2 (Socher et al., 2013), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009), MRPC (Dolan and Brockett, 2005), and QQP[3]. Among GLUE, we leave out datasets that were not evaluated by accuracy, along with WNLI, because the size is too small to show any general trend of effectiveness.

TREC is a commonly used dataset to evaluate mixup methods in sentence classification (Guo et al., 2019; Thulasidasan et al., 2019). We use two different versions of TREC (coarse, fine) that have different levels of label number to test the dependency of mixup effectiveness on the number of class labels. In addition, we use ANLI to see how mixup can help to improve model robustness. For training ANLI, we concatenate all training data from different rounds and use them to train the model.

3.2 Baseline

We compare *SSMix* with three baselines: (1) standard training without mixup, (2) EmbedMix, and (3) TMix. EmbedMix apply mixup on the embedding layer, which is similar to the wordMixup in Guo et al. (2019) except their experiments are performed with LSTM or CNN architecture. TMix, borrowed from Chen et al. (2020), interpolates hidden states of two different inputs at a particular encoder layer and forward the combined hidden states to the remaining layers. For EmbedMix and TMix, we follow the best settings stated in the original papers: mixup ratio is set by $\lambda' \sim Beta(\alpha, \alpha)$, $\lambda = max(\lambda', 1 - \lambda')$ with $\alpha = 0.2$. During the training with TMix, we randomly sample the mixup layer from $[7, 9, 12]$.

3.3 Ablation study

To investigate how much (1) considering saliency and (2) restricting mixup operation on the span-level individually benefit our proposed method, we conduct an ablation study. We implement *SSMix* without considering saliency information (*SSMix* - saliency) where the spans are randomly selected, and additionally without the span-level restriction (*SSMix* - saliency - span). For *SSMix* - saliency - span, we randomly sample tokens from x^B, which need not be a contiguous span and are conducted on a per-token basis. Then, we replace tokens accordingly with the position of the token be preserved, meaning that the second token from x^A is replaced with the second token from x^B, and so on. For all ablation studies, the lambda values were set to 0.1 to compare methods with the same setting as *SSMix*. Detailed implementation and illustration of ablation methods and comparison with simple word dropout methods are described in Appendix B.

3.4 Training Details

Among the entire experiment, we use sequence classification task with the pre-trained BERT-base model having 110M parameters from Hugging-

[2] https://github.com/huggingface/datasets

[3] https://www.quora.com/First-Quora-Dataset-Release-Question-Pairs

Face Transformers library.[4] We perform all experiments with five different seeds (0 to 4) on a single NVIDIA P40 GPU and report the average score. We set a maximum sequence length of 128, batch size of 32, with AdamW optimizer with eps of 1e-8 and weight decay of 1e-4. We use a linear scheduler with a warmup for 10% of the total training step. We update the best checkpoint by measuring validation accuracy on every 500 steps. For datasets that have less than 500 steps per epoch, we update and validate every epoch.

Considering our objective of enhancing performance through mixup, we conduct training in two steps. We first train without mixup with a learning rate of 5e-5 for three epochs, and then train with mixup starting from previous training's best checkpoint, with a learning rate of 1e-5 for five epochs. This two-step training, which also utilized by Zhang et al. (2018), speeds up the model convergence. We report the best accuracy among both training with and without mixup. For the ANLI task, we select the best checkpoint for training without mixup separately for each round, then conduct training with mixup and report the best accuracy of each round's evaluation dataset.

For each iteration, we split the batch into two smaller batches with the same size, A and B. Since mixup operation in *SSMix* is not symmetric, we conduct mixup back-and-forth so that mixup performance is evaluated regardless of the data position in batch. To prevent the training data distribution getting too far from the original data distribution, we train with and without mixup together as He et al. (2019). As a result, we forward each step with average loss from A, B, $mixup(A, B)$, and $mixup(B, A)$.

We leave out tokens specific to transformer architecture (e.g., $[CLS]$, $[SEP]$) when conducting a mixup to preserve special signs. As stated by Zhang et al. (2018), giving too high values for mixup ratio may lead to underfitting, while giving λ close to 0 leads to the same effect of giving non-augmented original data. From our experiments, we found out that augmentation with prior ratio $\lambda_0 = 0.1$ is the optimal hyperparameter.

In terms of computation time, *SSMix* takes about twice the training time compared with other mixup methods since we need an additional forward and backward step to compute the saliency of tokens. Among hidden-level mixup methods, TMix takes a slightly longer time to train than EmbedMix.

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

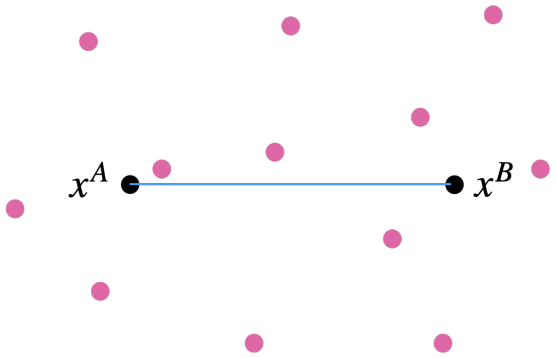


Figure 2: Visualization of original data and synthesized data by hidden-level mixup (EmbedMix or TMix) and *SSMix* in the hidden space. Black dots indicate the original data, x^A and x^B. For hidden-level mixup, synthetic data ($\tilde{x}$) are created only along the line (blue) connecting two points, since it is a linear combination within the hidden space. However, *SSMix* explore larger synthetic sample space for $\tilde{x}$, since it consists of a discrete combination within the *input* space. Synthetic data for *SSMix* are illustrated in pink dots.

4 Results and Discussion

Table 2 illustrates our results. We investigate the effectiveness of *SSMix* compared with hidden layer mixup methods on the aspect of dataset size, number of class labels, and paired sentence tasks.

Dataset size Compared with hidden-level mixup methods, *SSMix* fully demonstrate its effectiveness on datasets having a sufficient amount of data. Since *SSMix* is a discrete combination rather than a linear combination of two data samples, it creates data samples on a synthetic space in a larger range than hidden-level mixup (Fig. 2). We hypothesize that a large amount of data help better representation in synthetic space.

The number of class labels *SSMix* is especially effective for multiple class label datasets (TREC, ANLI, MNLI, QNLI). Accordingly, the accuracy gain of *SSMix* from the training without mixup is much higher on TREC-fine (47 labels) than TREC-coarse (6 labels), with +3.56 and +0.52, respectively. We hypothesize that this result originates from the mixup characteristic that benefits more from cross-label mixup than mixup with the same label, as stated at Zhang et al. (2018).[5] Since datasets with multiple total class labels increase the

[5] Zhang et al. (2018) states that mixing random pairs from all classes (per-batch basis) has the strongest regularization effect compared with mixup by per-class (same class) basis.

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	coarse	fine	R1	R2	R3
No mixup	92.96	91.32	84.27	91.28	65.56	86.37	97.08	86.68	56.40 57.16	47.10 47.36	47.62 48.00
EmbedMix	93.03	91.36	84.35	91.43	**67.73**	**86.72**	97.44	90.04	56.78 57.16	47.84 47.42	47.67 48.00
TMix	93.03	91.34	84.33	91.40	66.86	86.42	97.52	90.16	56.68 57.28	47.58 47.90	47.78 **48.42**
SSMix	93.10	**91.43**	**84.54**	**91.54**	67.22	86.57	**97.60**	**90.24**	**57.26** **57.34**	**48.36** **48.06**	47.78 48.00
SSMix - saliency	93.12	91.32	84.48	91.29	67.00	86.42	97.44	89.56	57.04 57.16	48.22 47.94	**47.95** 48.07
SSMix - saliency - span	**93.14**	91.32	84.54	91.45	66.93	86.37	97.40	89.20	56.74 57.20	47.52 47.90	47.77 48.00

Table 2: Experimental results of comparison with baselines and ablation study. All values are average accuracy (%) of five runs with different seeds. MNLI indicates MNLI-mismatched dev set accuracy. We report validation accuracy for GLUE, test accuracy for TREC, and valid (upper) / test (lower) accuracy for ANLI. We report variance on Appendix. A.

possibility of being selected cross-label in a random sampling of mixup sources, we assert mixup performance increases in such datasets.

Paired sentence tasks *SSMix* have a competitive advantage on paired sentence tasks, such as textual entailment or similarity classification. We suspect this accuracy gain originates from consideration of individual tokens. Existing methods (hidden-level mixup) apply mixup on the hidden layer, without consideration of special tokens, i.e., $[SEP], [CLS]$. These methods may lose information about the start of the sentence or appropriate separation of pair of sentences. In contrast, *SSMix* can consider the individual token property when applying mixup. Here, our mixup strategy on paired data (Section 2) preserves the property of $[SEP]$, which is not guaranteed by hidden mixup.

Ablation Study The results of *SSMix* and its variants demonstrate that the performance improves as we add span constraint and saliency information. Adding span constraint in the mixup operation benefit from better localizable ability, and most salient spans have more relationship to corresponding labels while discarding least salient spans have a higher probability that those spans are not semantically important with respect to the original labels. Among those two, introducing saliency information contributes to accuracy relatively more than the span constraint.

5 Conclusion

We present *SSMix*, a novel and simple input-level mixup method for text data that improves regularization ability leading to better performance in text classification. *SSMix* preserves the locality of mixing texts by replacing in span-level and keep most discriminative tokens in the mixed text using saliency score. Throughout the experiment, we show that our method improves performance in various types of text classification tasks. For future work, we plan to apply *SSMix* on a broader range of tasks, including generation or different scenarios like semi-supervised learning.

Acknowledgments

The authors would like to thank Clova AI members for proofreading this manuscript and the anonymous reviewers for their constructive feedback. We use Naver Smart Machine Learning (Sung et al., 2017; Kim et al., 2018) platform for the experiments. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

References

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

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.

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

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

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

Zhuoxun He, Lingxi Xie, Xin Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. 2019. Data augmentation revisited: Rethinking the distribution gap between clean and augmented data.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*.

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

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

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

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. *arXiv preprint arXiv:2009.10195*.

Yixin Nie, Adina Williams, Emily Dinan, Jason Bansal, Mohitand Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jungsoo Park, Gyuwan Kim, and Jaewoo Kang. 2021. Consistency training with virtual adversarial discrete perturbation. *arXiv preprint arXiv:2104.07284*.

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

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

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.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. 2017. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899.

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 *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. 2020. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification.

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.

Jason 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 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.

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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032.

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

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*.

A Accuracy Variance

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	Coarse	Fine	R1	R2	R3
No mixup	0.04	0.04	0.12	0.05	3.89	1.73	0.17	2.21	1.21 0.24	0.16 1.26	0.73 0.84
EmbedMix	0.02	0.03	0.14	0.04	3.89	1.39	0.09	0.31	1.38 0.24	0.46 1.18	0.75 0.84
TMix	0.04	0.04	0.09	0.03	1.85	1.55	0.05	0.63	1.44 0.25	0.33 0.75	0.73 1.28
SSMix	0.03	0.07	0.07	0.03	2.57	1.15	0.03	0.49	1.56 0.25	0.27 0.46	0.73 0.84
SSMix - saliency	0.02	0.04	0.11	0.04	2.06	1.55	0.09	0.69	1.33 0.24	0.18 1.99	0.62 0.80
SSMix - saliency - span	0.00	0.04	0.09	0.03	1.86	1.73	0.08	0.14	2.01 0.28	0.11 0.45	0.68 0.84

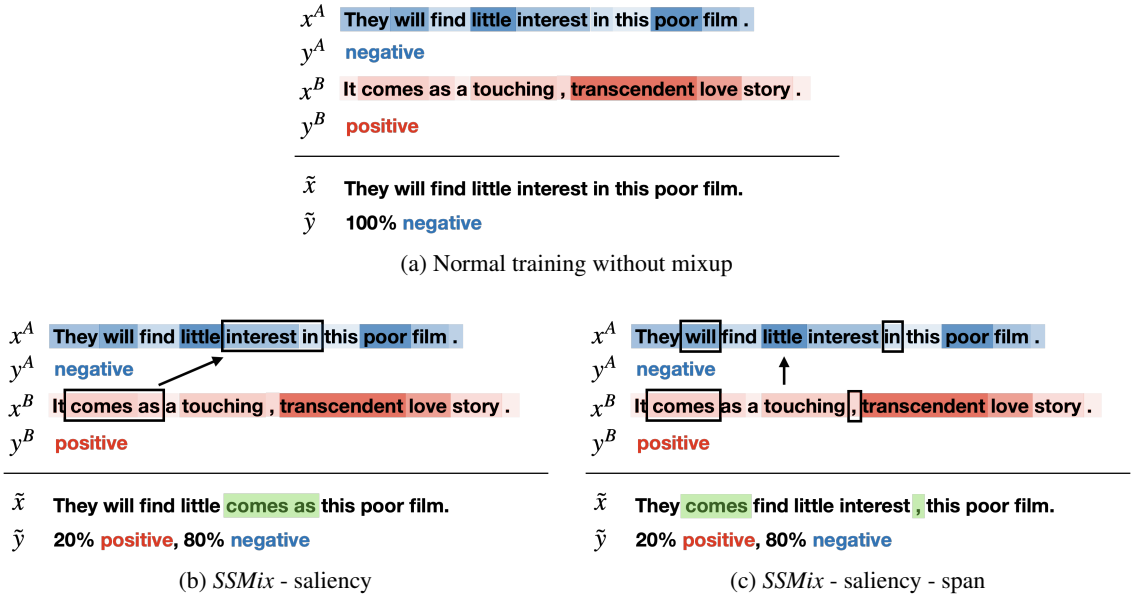
Table A.1: Standard deviation results, corresponding with the average of our experiments. The deviation is conducted by 5 runs with different seeds.

We also report accuracy variance among the five seeds for each experiment (Table. A.1).

B Ablation

Fig. 3 and Fig. 4 shows the illustration of different variants of *SSMix* and random UNK replacement with $\lambda = 0.2$. Fig. 5 shows the illustration of getting the augmented output with lambda calculation by *SSMix* for paired sentence tasks. The saliency maps are visualized where darker concentration of colors mean higher contribution to corresponding label.

B.1 Variants of *SSMix*



(a) Normal training without mixup

(b) *SSMix* - saliency

(c) *SSMix* - saliency - span

Figure 3: Illustration of normal training and variants of *SSMix*

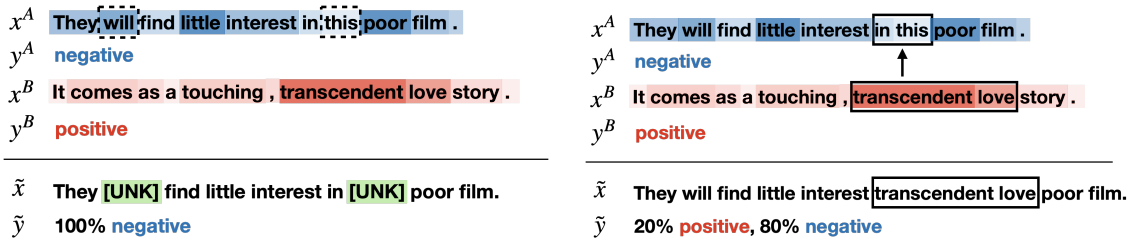
Here, we describe in detail how we implement *SSMix* without saliency (Figure. 3 (b)) and *SSMix* without saliency and span restriction (Figure. 3 (c)).

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	coarse	fine	R1	R2	R3
No mixup	92.96	91.32	84.27	91.28	65.56	86.37	97.08	86.68	56.40 57.16	47.10 47.36	47.62 48.00
Random UNK replacement	93.10	91.33	84.46	91.45	66.86	**86.62**	97.44	89.24	56.98 57.26	47.86 **48.36**	**47.98** **48.32**
SSMix	93.10	**91.43**	**84.54**	**91.54**	**67.22**	86.57	**97.60**	**90.24**	**57.26** **57.34**	**48.36** 48.06	47.78 48.00
SSMix - saliency	93.12	91.32	84.48	91.29	67.00	86.42	97.44	89.56	57.04 57.16	48.22 47.94	47.95 48.07
SSMix - saliency - span	**93.14**	91.32	84.54	91.45	66.93	86.37	97.40	89.20	56.74 57.20	47.52 47.90	47.77 48.00

Table B.1: Accuracy (%) comparison with simple data augmentation method(random UNK replacement) and input mixup methods. The results are average of five runs with different seeds. Results show that our input level mixup methods are generally competitive with simple word dropout methods.

At normal training, only two real data samples (x^A and x^B) are used to train the model. For Figure. 3 (b), we *randomly* select each span from x^A and x^B. Then, we replace x^B to x^A to make a new data $\tilde{x}$. For Figure. 3 (c), input level mixup is conducted on a per-token basis. After calculaton of l given the prior mixup ratio, we randomly sample tokens from x^A. The tokens need not be a contiguous span. Then, we replace tokens accordingly with the position of the token be preserved, meaning that the second token from x^A is replaced with second token from x^B, the sixth token from x^A is replaced with sixth token from x^B (by the illustration example), and so on.

B.2 Comparison with other simple augmentation methods



(a) Random [UNK] replacement (b) *SSMix*

Figure 4: Comparison of our methods with word dropout

We also compare *SSMix* with simple word dropout methods, which may seem similar in the perspective that they create noisy sentences. The difference is whether label mixup is performed. Illustration of the implementation of random [UNK] replacement is available at Fig. 4. Random UNK replacement is similar to word dropout. We don't use x^B when making synthetic samples ($l = 0$). Instead, we randomly sample a set of tokens from x^A and replace each token in that span with [UNK]. The process is similar to Figure. 3 (c), except that the selected tokens at x^A are replaced into [UNK]. Another difference is that the output label ($\tilde{y}$) completely follow the origin (y^A) and no label mixup is performed. The illustration is available at 3.

We evaluate the random [UNK] replacement method on all dataset with *SSMix* and variants of *SSMix* at ablation study. By the experiment results at Table B.1, we show that input level mixup methods generally outperform simple regularization methods. This means that datasets synthesized from *SSMix* and the according target vectors have more gain on the generalization ability than word dropout.

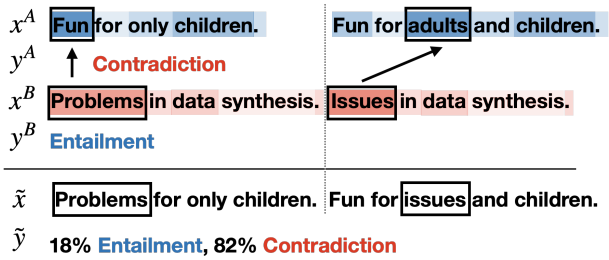


Figure 5: Illustration of applying SSMix to make $\tilde{x}$ for paired sentence, in particular NLI tasks, which classifies whether the relation of sentence pairs is entailment, neutral, or contradiction. Mixup is conducted individually, sentence by sentence.

B.3 Illustration of SSMix on paired sentence tasks

Fig. 5 shows the illustration of example for paired sentence. Here, "Fun for only children." and "Fun for adults and children." correspond to p^A and q^A, "Problems in data synthesis." and "Issues in data synthesis." correspond to p^B and q^B, and "Problems for only children.", "Fun for issues and children." correspond to p and q, respectively. λ is calculated as : $\lambda = (|p_S| + |q_S|)/(|\tilde{p}| + |\tilde{q}|) = (1+1)/(5+6) = 2/11 \approx 0.18$.