TEXT IS AN IMAGE: AUGMENTATION VIA EMBEDDING MIXING

Kevin Zhao
Wayland High School, MIT PRIMES Program
Wayland, MA, 01778, USA
zhaokevinusa@gmail.com

Vladislav Lialin & Anna Rumshisky
Text Machine Lab
University of Massachusetts Lowell
Lowell, MA, 01854, USA
{vlialin,arum}@cs.uml.edu

ABSTRACT

Data augmentation techniques are essential for computer vision, yielding significant accuracy improvements with little engineering costs. However, data augmentation for text has always been tricky. Synonym replacement techniques require a good thesaurus and domain-specific rules for synonym selection from the synset, while backtranslation techniques are computationally expensive and require a good translation model for the language in interest.

In this paper, we present simple text augmentation techniques on the embeddings level, inspired by mixing-based image augmentations. These techniques are language-agnostic and require little to no hyperparameter tuning. We evaluate the augmentation techniques on IMDB and GLUE tasks, and the results show that the augmentations significantly improve the score of the RoBERTa model.

1 INTRODUCTION

Data augmentation has been widely and successfully applied in computer vision (CV). Starting with simple augmentations such as crop, reflection and translation (Krizhevsky et al., 2012), it rapidly became a standard way to improve generalization of the computer vision models (Simonyan & Zisserman, 2015; He et al., 2016) .

On the other hand, augmentations in natural language processing (NLP) have always been tricky. A simple idea of synonym replacement (Zhang et al., 2015) requires an external data source (e.g., thesaurus) and has additional engineering costs for synset selection algorithm. Other methods require a pretrained language model (Kobayashi, 2018) or a translation system (Wieting et al., 2017; Edunov et al., 2018) to rephrase the training texts. These approaches not only limit the set of languages that can benefit from the data augmentation and make the low-resource language setups even more challenging compared to English, but they also require significant additional computation.

Several NLP researchers have proposed to reapply computer vision methods such as Mixup (Guo et al., 2019; Jindal et al., 2020; Guo, 2020) in a NLP setting. In this study, we show that a wider range of computer vision augmentations are applicable to NLP and demonstrate their efficacy on the IMDB dataset (Maas et al., 2011) as well as a popular set of NLU tasks – GLUE benchmark (Wang et al., 2018).

We summarize our contributions as follows:

- We apply multiple computer vision inspired augmentations to NLP tasks;
- We show that a combined Mixup+Cutout augmentation can significantly improve RoBERTa results on the GLUE benchmark;
- We propose a version of Nonlinear Mixup augmentation that substantially improves RoBERTa scores on the IMDB dataset, but is ineffective on harder tasks;

2 Related Work

Augmentations in CV Krizhevsky et al. (2012) and He et al. (2016) have used simple image augmentations such as image translations, horizontal reflections, and minor color modifications in their fundamental works. These works demonstrated the importance of augmentations for image recognition using deep neural networks. Konno & Iwazume (2018) use feature-map level augmentations to improve generalization in a heavy class imbalance setting. Rusak et al. (2020) in their work show that Gaussian or Speckle noise serves as a strong baseline against adversarial attacks.

A very interesting approach was proposed by Zhang et al. (2017). Their idea was that instead of modifying a single image, it is possible to combine several images to produce a "soft" label that describes this combination. Thulasidasan et al. (2019) demonstrated that such approach improves neural network calibration. Yun et al. (2019) developed a more complex approach that combines two images in a mosaic-way and Kim et al. (2020) moved that idea even further. Additionally, many researchers such as Zhan et al. (2017) and Frid-Adar et al. (2018) have shown that Generative Adversarial Networks (Goodfellow et al., 2014) can be used for augmentation in a wide variety of applications ranging from remote sensing to medical imaging.

Augmentations in NLP The NLP community has not used augmentations as frequently as the CV community. The fundamental papers such as Vaswani et al. (2017); Peters et al. (2018) do not use any kind of augmentation.

An exception for this is the machine translation community which is extensively using methods such as backtranslation to significantly improve the model quality (Sennrich et al., 2016; Edunov et al., 2018), although this method is not directly an augmentation method as it requires additional, although, unlabeled texts. Yu et al. (2018) used a method of data augmentation via backtranslation for a question answering (QA) task, which uses an existing translation model instead of training one.

Zhang et al. (2015) use synonyms to improve classification quality and Wieting et al. (2017) replace parts of the text using a language model.

Recently, Wei & Zou (2019) proposed a method of combining synonym replacement with random insertion/deletion/swap for CNN and RNN networks which achieves the same accuracy as a normal model while only using half of the training data.

On the other hand, simpler techniques similar to Zhang et al. (2017) have been overlooked by the community. Guo et al. (2019) proposed WordMixup and SenMixup techniques that mix either word embeddings or sentence embeddings and demonstrated the efficacy of the approach using CNN and LSTM networks. Guo (2020) went on to propose a new variation of Mixup called Nonlinear Mixup which performs better than his previous two methods. Jindal et al. (2020) demonstrated that it is possible to apply Mixup to the hidden representations of a BERT classifier.

In this work, we demonstrate that it is possible to apply a wider range of computer vision techniques for NLP tasks despite the fact that texts are discrete in nature and that the word sequences are supposed to follow a specific, grammatical structure.

3 Text-Mixing Augmentations

We adapt three data augmentation techniques used in computer vision – Mixup, Cutout, and CutMix – for NLP. To apply mixing augmentations, the word embeddings of the input sentence are stacked together to form a matrix $A \in \mathbb{R}^{N \times d}$ where N is the sequence length and d is the embedding dimension. In other words, the i-th row of the matrix corresponds to the word embedding for the i-th token in the sentence. Then, Mixup, Cutout, CutMix, or a combination of Mixup with Cutout is applied by treating each matrix as a 1+1D image.

To draw an analogy, an RGB image is a 2+1D tensor (2 spatial dimensions, 1 feature dimension – color), and text can be represented as a 1+1D embeddings matrix (1 spatial dimension, 1 feature dimension). The main difference here is the variability of the spatial dimension size. While for the images it is usually fixed to a certain value (e.g., 128), for the text, it is variable and usually all sequences are padded to the longest sequence in the batch with zero vectors.

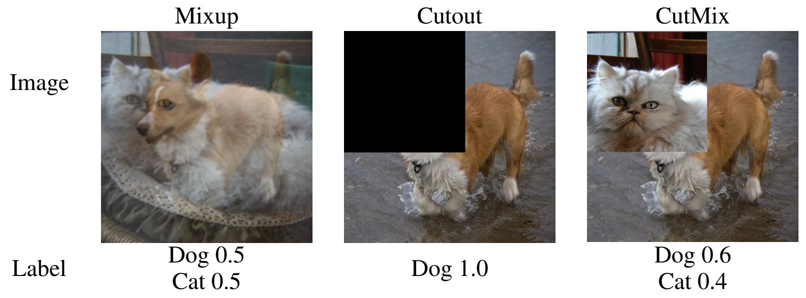

Figure 1: A visual example of different mixing strategies. Source: Yun et al. (2019).

3.1 MIXUP

Zhang et al. (2017) proposed an image augmentation technique that combines two images and their labels in a weighted way:

$$\begin{aligned} X_{mixed} &= \alpha X_1 + (1-\alpha)X_2 \\ y_{mixed} &= \alpha y_1 + (1-\alpha)y_2 \end{aligned} \tag{1}$$

Where the weight α is chosen from the beta distribution.

Our experiments (Section 5) suggest that a constant mixing value or a gaussian distribution can perform significantly better on NLP tasks.

3.2 CUTOUT

DeVries & Taylor (2017) augment images by randomly cutting out a fixed shape from each image. This is similar to dropout (Srivastava et al., 2014), because it forces the model to look at the entire input to make the prediction, instead of relying on a small section. However, unlike dropout we only apply cutout to half of the input images, while leaving the other half of images unchanged, following the recommendations from DeVries & Taylor (2017). We remove a rectangular section with a width of half the text length and height of half the embedding size. The position to be cut out is selected uniformly from anywhere in the embedded text matrix.

3.3 CUTMIX

Following the success of Mixup and Cutout, Yun et al. (2019) proposed a similar method that combines both ideas. An intuitive illustration of the method can be found in Figure 1. In order to combine two input tensors, a random section is selected similar to Cutout. After that, instead of zeroing out the corresponding features in the first example, they are replaced with the corresponding features from the second example.

The resulting label is selected according to the CutMixed section size. This section size is not fixed like it was in Cutout; instead, it is selected uniformly from the range of 0.3 to 0.7 of the total size of the image, which ensures that over 1/4 of both images are always visible. The shape of the CutMixed section is always proportional to the shape of the image (i.e. the ratio of the length to the width is constant).

3.4 NONLINEAR MIXUP

Guo (2020) proposed the method of Nonlinear Mixup, where instead of mixing examples using a single parameter α, a matrix $\Lambda \in \mathbb{R}^{N \times d}$ of the same size as the input is used. The values of the matrix are randomly sampled from a gaussian distribution and then the inputs are mixed in the following way:

$$X_{mixed} = \Lambda \circ X_1 + (1-\Lambda) \circ X_2 \tag{2}$$

where $\circ$ denotes the element-wise product.

Then, Guo (2020) flatten Λ and pass it through a linear layer which outputs a vector of size c, where c is the number of classes. Finally, the sigmoid function is applied to each value of the output vector, resulting in c numbers which corresponds to the degree that each class is weighted. Additionally, Guo (2020) uses label embedding to represent the labels as a matrix $M \in \mathbb{R}^{c \times k}$, where each of the c categories is encoded as a k-dimensional vector.

Building on Guo (2020), we use multiple methods to learn the mixing weights in Λ instead of randomly sampling them. The most straightforward method is to train a second, smaller Transformer network (Vaswani et al., 2017) that will use the input embeddings to predict the mixing weights. A benefit of this method is that it can account for variable-length inputs.

Thus, the mixing of the embeddings and labels are computed in the following way:

$$\begin{aligned}
\Lambda_i &= \text{Transformer}(X_i),\ i \in 1, 2 \\
\Lambda &= \sigma([\Lambda_1; \Lambda_2] \times \Theta) \\
\hat{\Lambda} &= \frac{\Lambda}{\text{mean}(\Lambda)} \\
X_{mixed} &= \alpha \hat{\Lambda} X_1 + (1 - \alpha)(1 - \hat{\Lambda}) X_2 \\
y_{mixed} &= \alpha y_1 + (1 - \alpha) y_2
\end{aligned} \tag{3}$$

where $\Theta \in \mathbb{R}^{2d \times d}$ and d is the embedding dimension.

Even a small transformer is complex and has many weights, so a simper alternative is to use a linear layer or a fully connected network. With this simplification, other steps to calculate Λ can also be changed.

$$\begin{aligned}
\Lambda_i &= \text{NN}(X_i),\ i \in 1, 2 \\
\Lambda_i &= \frac{\Lambda_i - \text{mean}(\Lambda_i)}{\text{std}(\Lambda_i)},\ i \in 1, 2 \\
\hat{\Lambda} &= \sigma(\Lambda_1 - \Lambda_2) \\
X_{mixed} &= \alpha \hat{\Lambda} X_1 + (1 - \alpha)(1 - \hat{\Lambda}) X_2 \\
y_{mixed} &= \alpha y_1 + (1 - \alpha) y_2
\end{aligned} \tag{4}$$

The above version of Nonlinear Mixup is the most straightforward and the smaller network is trained together with the main model. Both networks share the common goal of minimizing the regression or classification loss. Another idea is an adversarial approach, where the smaller network acts as an adversary that tries to maximize the loss of the main classification model, allowing the main model to become more robust and generalize better to unseen examples.

4 DATASETS

We evaluate our augmentation methods using GLUE, The General Language Understanding Evaluation benchmark (Wang et al., 2018). GLUE is a collection of datasets that measure general natural language understanding capabilities. It is made up of a diverse set of tasks which uses either a single or pair of sentences, and includes tasks that evaluate natural language inference, sentiment analysis, and semantic similarity. It is a common text classification benchmark and was used in previous work such as Devlin et al. (2018) and Liu et al. (2019).

In addition to GLUE, we also use the IMDB dataset (Maas et al., 2011), which is a sentiment analysis task with a collection of 50,000 movie reviews. Unlike the GLUE benchmark, IMDB is only a single task, so it is used to quickly evaluate the performance of our augmentation methods. Because IMDB is just one task, it is less comprehensive than GLUE and does not reflect the effectiveness of our augmentation methods on other, harder tasks such as natural language inference.

5 Experiments

Our finetuning procedure follows RoBERTa with minor changes. The only hyperparameters we modify are the sequence length and number of training epochs, as well as the learning rate for select tasks.

The original RoBERTa paper (Liu et al., 2019) padded all sequences to 512 tokens, but we find that this is unnecessary for the vast majority of examples and instead pad all sequences to either 64 tokens (single sentence task) or 128 tokens (sentence-pair task).

In addition, the original RoBERTa paper finetuned for 10 epochs on every dataset. For our baseline we train for the same amount on smaller datasets (CoLA, MRPC, STS-B, RTE) but decrease the number of epochs for larger datasets (while making sure the model still converges). When training with Mixup, we finetune for the same number of epochs as our baseline on larger datasets (SST-2, QNLI, QQP, MNLI), while we finetune for twice as many epochs on the smaller datasets.

For two sentence classification tasks, we pad the first sequence of each example within a batch to the same length before adding the SEP token and the second sequence (i.e. the SEP token is at the same place for each example within a batch). This ensures that the first sequence of one example is only mixed with the first sequence of a different example, and the second sequence of one example is only mixed with the second sequence of another example.

Additionally, for Nonlinear Mixup we found that a smaller transformer or fully connected network was too complex, so we use a simple linear layer in all experiments. Because the linear layer has to be trained from scratch, the learning rate for the linear layer is increased by a factor of 10 and the weight decay is set to 0.

Task-specific Modifications In order to achieve similar performance to the reported results, we had to change some additional hyperparameters for some tasks. For SST-2 and QNLI, we switched from the triangular scheduler to the Noam scheduler (Vaswani et al., 2017) in order for the baseline scores to reach a similar score to the reported results. For other tasks, we found that changing the learning rate improved the baseline score to be comparable with the reported results. We doubled the learning rate for QQP to 2e-5 and halved the learning rate for RTE to 1e-5.

The modifications above were applied to both the baselines and the augmentation methods, but we also made changes to two tasks just for Mixup. While most of the GLUE tasks involve two classes, MNLI has three and STS-B has one (regression task). For MNLI, we mixed the one hot encoded labels, while for STS-B we mixed the regression target.

Mixing Weight Selection One of the most important parts of Mixup and CutMix is how the weight is chosen. In both Mixup (Zhang et al., 2017) and CutMix (Yun et al., 2019), the weight varies for every sample and is drawn at random from a beta distribution. We experiment with other methods of determining the weight, namely setting it to a constant and choosing it from a Gaussian distribution.

We find that using a Gaussian distribution centered at 0.1 with a standard deviation of 0.015 works well for normal Mixup. For Nonlinear Mixup, we keep the mixing weight α constant at 0.1 instead of sampling it from a distribution.

6 Results

Regularizers Interaction As data augmentation has a regularizing effect, we also experiment with turning off different forms of regularization (dropout, weight decay) used in the initial RoBERTa fine-tuning process. We find that Mixup + Cutout works well with either dropout or weight decay, but not both together (Table 2). Both types of regularization achieve improvements over the baseline, but finetuning with weight decay obtains a higher score compared to finetuning with dropout. Based on these results, we turn off dropout but continue to use weight decay for all of our Mixup methods.

GLUE benchmark results GLUE results for models trained with different mixing strategies are presented in Table 1. The combination of Mixup+Cutout shows the most promising results and usually improves upon our RoBERTa implementation. These two augmentation techniques have a

$\mathrm{RoBERTa}_{\mathrm{large}}$	CoLA	SST-2	MRPC	STS-B	QNLI	RTE	QQP	MNLI
Reported Results	68.0	96.4	90.9/-	92.4/-	**94.7**	**86.6**	**92.2/-**	90.2
Our implementation	68.7	**96.7**	90.2/93.1	92.5/92.2	94.6	**86.6**	91.9/89.2	**90.7**
Mixup	71.2	**96.7**	90.0/93.0	92.3/92.2		79.8		
CutMix	70.7		89.7/92.7	91.8/91.5				
Cutout	67.8		89.0/92.2			81.2		
Mixup + Cutout	**72.8**	**96.7**	**92.4/94.5**	**92.6/92.5**	94.5	**86.6**	92.1/89.3	90.6

Table 1: Score of different methods on GLUE tasks, with **best** and second best scores marked. Mixup+Cutout consistently performs better or on par with the baseline (no augmentation) with the exception of QNLI (see Table 3) and MNLI, while other augmentation methods tend to perform worse.

Mixup + Cutout	CoLA	MRPC	RTE
+ Dropout + Weight Decay	61.57	88.97	76.90
+ Dropout - Weight Decay	60.35	88.48	75.81
- Dropout + Weight Decay	64.51	**89.46**	**82.31**
- Dropout - Weight Decay	**64.71**	**89.46**	81.95

Table 2: The effect of regularization techniques on smaller GLUE tasks. Notice that using dropout with Mixup+Cutout usually hurts the final performance while the weight decay may be beneficial.

significant improvement on CoLA and MRPC, a smaller improvement on STS-B and QQP, is on par with the baseline for SST-2 and RTE, and is slightly worse (by 0.1) on QNLI and MNLI.

Mixup alone also does better than the baseline on CoLA and acheives the same score as the baseline on SST-2, but is generally worse for other tasks. CoLA and SST-2 are the only two single sentence classification tasks in GLUE, so this suggests that Mixup alone works well with one sentence.

Interestingly, CutMix combines Mixup and Cutout in a more natural way but did not perform as well. Furthermore, Cutout without Mixup actually performs worse than the no-augmentation baseline on all tasks.

However, it should be noted that GLUE results are significantly affected by random seeding (Card et al., 2020). All $\mathrm{RoBERTa}_{\mathrm{large}}$ experiments were done with the same random seed, but we also ran multiple $\mathrm{RoBERTa}_{\mathrm{base}}$ experiments with different random seeds and the results are noticeably worse.

	SST-2	QNLI
Reported Results	96.4	94.7
Our implementation	96.0	92.9
Mixup + Cutout	96.4	93.2

Table 3: Results for SST-2 and QNLI without the Noam scheduler. Mixup+Cutout performs better than our baseline.

Using a Noam scheduler for SST-2 and QNLI improved the scores for both the baseline and the augmentation methods, but it had a more noticeable effect on the baseline. With the triangular scheduler used for other tasks, Mixup+Cutout did better than the baseline by 0.4 on SST-2 and 0.3 on QNLI.

However, after switching to the Noam scheduler, Mixup+Cutout was only on par with the baseline for SST-2 and even performed worse by 0.1 for QNLI. This suggests that while the Noam scheduler improved the score of the augmentation methods, it may need to be adapted (e.g. changing number of warmup steps) to work better with Mixup+Cutout.

Nonlinear Mixup results Results of Nonlinear Mixup methods on IMDB and QNLI are presented in Table 4. In Nonlinear Mixup (Random), the mixing weights Λ are randomly sampled with a mean of 0.5, while Λ is the output of a linear layer in both Nonlinear Mixup (Normal) and (Adversarial).

$\mathrm{RoBERTa_{base}}$	IMDB	QNLI
Our Baseline	91.29	92.46
Nonlinear Mixup (Random)	91.43	92.68
Nonlinear Mixup (Normal)	91.50	92.04
Nonlinear Mixup (Adversarial)	91.63	92.35

Table 4: Score of different Nonlinear Mixup methods on IMDB and QNLI.

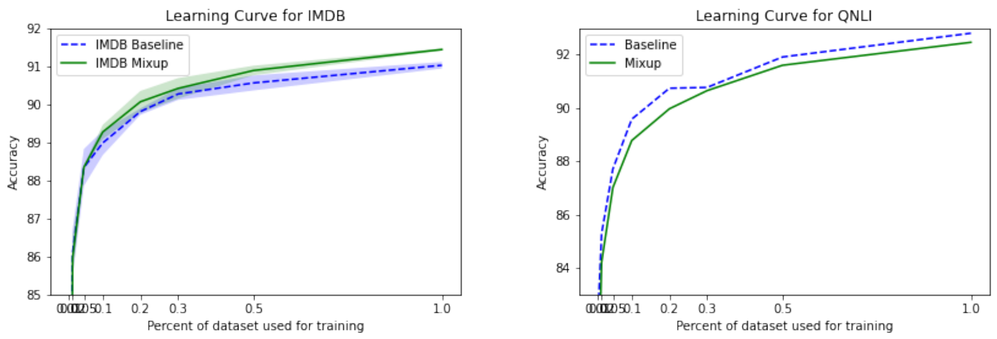

Figure 2: Learning curves of Nonlinear Mixup (Normal) on IMDB (three random seeds) and QNLI (one random seed).

All methods of Nonlinear Mixup perform better than the baseline for IMDB, but for QNLI the baseline outperforms both normal and adversarial Nonlinear Mixup. This suggests that Nonlinear Mixup helps on simple tasks like sentiment analysis but does not work on harder tasks such as natural language inference.

Examining the mixing weights (Λ) of Nonlinear Mixup offers an explanation of the difference between tasks. For IMDB, the linear layer is very effective; for example, in one review, the words with the highest corresponding Λ values (ordered from largest to smallest) are ”greatest”, ”loved”, ”unbelievable”, ”hope”, ”sister”, and ”addict[ing]”. All of the words other than ”sister” are very important for a sentiment analysis task and can concisely capture the opinion of the reviewer. The same trend is true in other examples where most words with large Λ values are impactful, but a select few, such as ”sister”, are irrelevant. Even though the linear layer that predicts Λ is unable to understand the context of a word, is effective for this task because only a small set of words are important.

On the other hand, Λ for QNLI is hard to interpret, and the weights corresponding to words appear random. For a natural language inference task, every word is important, because even grammar words like ”although” can drastically alter the meaning of a sentence. Therefore, it is much harder for a linear layer to assign useful weights without understanding the context that a word is in. From the results in Table 4, it appears that random weights continue to outperform the baseline, but both normal and adversarial versions of Nonlinear Mixup do worse. One explanation for the poor performance is that the linear layer has overfit to the training examples and does not actually learn any meaningful information. The random mixing appears to always do better than the baseline, but it was done on only two tasks and needs more experimentation.

The learning curves of Nonlinear Mixup (Normal) on IMDB and QNLI show the differences between the tasks. After a certain amount of training data, Nonlinear Mixup consistently performs better than the baseline on IMDB across three random seeds. It is intriguing that Nonlinear Mixup augmentations are not effective with a small subset of data (less than 10% the total training data). This suggests that Nonlinear Mixup still requires a certain threshold of data to train the linear layer before yielding performance improvements.

The learning curve for QNLI shows that Nonlinear Mixup consistently performs worse than the baseline, regardless of the amount of training data. This reinforces the explanation that the linear layer simply cannot learn useful information for a NLI task without understanding the context of a word.

7 Practical recommendations

Here we provide a short list of recommendations that can be used for real-world applications:

- Use Mixup+Cutout combination to improve test performance of the model;
- A gaussian distribution $\mathcal{N}(\mu = 0.1, \sigma^2 = 0.015)$ usually works better than a beta distribution for the weight selection;
- Nonlinear Mixup may yield substantial improvements depending on the task
- Dropout negatively interacts with the mixing augmentations;

8 Conclusion

In this work, we showed that the computer vision augmentations such as Mixup, Cutout and CutMix are applicable to NLP tasks when used on a word embedding level. We demonstrated that a combination of Mixup and Cutout can considerably improve the results of RoBERTa Large on the GLUE benchmark with a minimal modifications, such as adding the mixing weight distribution and turning off dropout. We proposed a new method of Nonlinear Mixup which achieved promising results on IMDB but requires further experimentation.

This illustrates that a wider range of augmentations is applicable to natural language processing and potentially opens a way to a broader adoption of computer vision techniques in NLP. Furthermore, potentially the more simple (compared to Zhang et al. (2015) and Sennrich et al. (2016)) augmentation techniques studied in this work can simplify the research and ease of application of modern data-hungry algorithms for low-resource languages.

9 Acknowledgements

I would like to thank my mentors Vladislav Lialin and Dr. Anna Rumshisky for their close guidance and insightful ideas. I am also grateful for Dr. Gerovitch and the MIT PRIMES program, under which this research was conducted.

References

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. *arXiv preprint arXiv:2010.06595*, 2020.

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

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

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

Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

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

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

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Amit Jindal, Dwaraknath Gnaneshwar, Ramit Sawhney, and Rajiv Ratn Shah. Leveraging bert with mixup for sentence classification (student abstract). In *AAAI*, pp. 13829–13830, 2020.

Janghyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. 2020.

Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*, 2018.

Tomohiko Konno and M. Iwazume. Cavity filling: Pseudo-feature generation for multi-class imbalanced data problems in deep learning. *arXiv: Learning*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

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

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. doi: 10.18653/v1/n18-1202. URL `http://dx.doi.org/10.18653/v1/N18-1202`.

E. Rusak, Lukas Schott, R. S. Zimmermann, Julian Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel. A simple way to make neural networks robust against diverse image corruptions. 2020.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016. doi: 10.18653/v1/p16-1009. URL `http://dx.doi.org/10.18653/v1/P16-1009`.

K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958, 2014.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

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

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

John Wieting, Jonathan Mallinson, and Kevin Gimpel. Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847*, 2017.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

Ying Zhan, Dan Hu, Yuntao Wang, and Xianchuan Yu. Semisupervised hyperspectral image classification based on generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*, 15(2):212–216, 2017.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.