A Novel Embedding Augmentation Method for Pretrained Transformer-Based Language Models

1st Rouzbeh Meshkinnejad
Department of Computer Science
Western University
London, Canada
rmeshkin@uwo.ca

2nd Sepehr Asgarian
Department of Computer Science
Western University
London, Canada
sasgaria@uwo.ca

***Abstract*—Data augmentation has proven to be effective in a variety of scenarios, including Supervised, Semi-Supervised, and Unsupervised Machine Learning. While data augmentation is fairly straightforward in Computer Vision, due to the complexities and discreet nature of language, it remains challenging to manipulate sentences without changing their meanings. While there have been methods for data augmentation in Natural Language Processing, their inherent limitations deny applying some Semi-Supervised techniques such as Consistency Regularization. Moreover, with the emergence of large pretrained transformer-based language models, the performance gain of word-level data augmentation methods is in doubt. In this project a novel data augmentation technique is introduced on the embeddings of pretrained transformer-base language models that enables flexible and controllable generation of new data samples. Empirical experiments show that our algorithm outperforms other augmentation methods in a baseline sentiment classification task.**

***Index Terms*—Data Augmentation, Natural Language Processing, Semi-Supervised Learning**

I. Introduction

A. Problem Definition

In use cases regarding classification, detection, or segmentation of images, deep neural networks or more specifically, convolutional neural networks (CNN) [1], are among the highest performing models. This performance is partially due to the scalability of these models, i.e. the more data there is to train these models on, the better the performance [2] [3] [4]. Consequently, as the size of these models increases, to get superior performance a larger labelled dataset is needed. If the labelled dataset is small in size, these models may suffer from overfitting. One way to mitigate this issue, is to simply find and label more data, while the other solution is to employ data augmentation techniques to increase the dataset size [5]. Other times, while the labelled dataset is small, there exists a large unlabelled dataset that can be made use of by Semi-Supervised Learning methods. Obtaining a large labelled dataset is usually costly or time-consuming, but a large unlabelled dataset is often available at low cost. For this reason, any improvement achieved by Semi-Supervised models will result in a better performance on a variety of tasks while inducing a low cost. Some of the techniques used in the Semi-Supervised machine learning make use of unlabelled data by introducing various loss functions on augmented data samples [6], making data augmentation a requirement for exploiting unlabelled data. For example, in a technique called Consistency Regularization [7] [8] [9], the model is trained to make consistent predictions regarding different augmented versions of a data sample. Due to this significant use of data augmentation either in the Supervised or Semi-Supervised domain, there has been an ongoing research into different ways a data sample can be manipulated while keeping the label unchanged [10] [11] [12]. While a variety of data augmentation methods have been introduced and employed in Computer Vision, data augmentation in Natural Language Processing (NLP) remains a challenging task, mainly due to the complexity and ambiguity associated with language. It is often the case that even small manipulations to a sentence can cause the meaning to change, making augmentations that keep the label intact hard to find. Moreover, similar to Computer Vision, any data augmentation method proposed for NLP should be flexible and fast enough to be efficiently incorporated in the relevant pipelines. Current augmentation techniques for NLP rely mostly on synonym replacement (based on a thesaurus [13], word vectors [14], or contextual embeddings [15]), random insertion of words, random swap of words, random deletion [16], or back-translation i.e. translating a sentence to another language and back [17]. Each of these techniques are limited in terms of the number of augmented samples they can generate without changing the meaning. Moreover, methods like back-translation are slow and computationally expensive. Intractability is another problem, making it difficult to control the strength of augmentation and how much change is introduced to the sentence. For these reasons, a flexible, fast, and controllable augmentation technique is needed. Some have attributed the limitation of the mentioned augmentations is due to the discreet nature of language [18]. Concurrently, with the rise of pretrained transformer [19] based language models, researchers have been looking to utilize the rich embeddings produced by these models [11]. These embeddings

provide a continuous representation of words and sentences in a high dimensional space, inviting researchers to come up with methods to augment embeddings instead of words and sentences themselves. Moreover, there is evidence showing word-level data augmentation does not improve performance when pretrained transformer-base language models are being used [20]. To address the issues above, in this work we present a novel embedding-level data augmentation that efficiently and flexibly generates new data samples. In the experiments that are described later, we show that our model provides performance gains compared to the baseline, opening up doors to future research by enabling applications of Semi-Supervised methods typically designed for Computer Vision on NLP.

II. Related Work

A. Easy Data Augmentation (EDA)

EDA is a powerful data augmentation method in NLP which is presented by Jason Wei and Kai Zou [16]. This method includes four different operations for data augmentation; these operations are synonym replacement (SR), random insertion (RI), random swap (RS), and random deletion (RD). In this approach, on each sentence in a training set, one of the discussed operations will be applied. Each of operations are discussed in detail in the following paragraphs.

SR operation: In the first step, from each sentence, n words are randomly chosen, which are not stop words. Next, each of these n words is substituted with one of their synonyms, which has been chosen randomly among all of their synonyms.

RI operation: Insert the random synonym of a random word that is not a stop word in a random location in the sentence. This will be done n times.

RD operation: Deleting words from the sentence randomly.

RS operation: Choose two words randomly from the sentence and switch their locations.

The proposed EDA [16] study showed that their model outperforms other benchmark models based on the speed and the performance. However, one of the things which was not examined in this research is the influence of this approach on pretraining models. In the recent work, [20], the influence of this method and Back-Translation [13] on pretrained model was examined. Moreover, this research has investigated the impacts of these models across five classification tasks on six various datasets and three modern pretrained variations, including BERT [21], XLNET [22], and ROBERTA [23]. In addition, [20] showed that approaches that previously demonstrated significant increases for non-pretrained models do not consistently enhance performance for pretrained models, even when training data is limited. Moreover, based on their results, data augmentation can be effective for pretrained models only when it is used based on the purpose of the problem (classification, question answering, etc).

B. MIXUP

Mixup is a powerful data augmentation approach which is proposed by Zhang et al. [24] for computer vision applications. In their model, they have combined pairs of examples and their labels. Moreover, this model is a data augmentation approach that behaves linearly in between training instances. When forecasting outside of the training instances, this article claims that this linear tendency decreases the number of unwanted oscillations. This approach generates data points between two points, which aids in domain generalisation. Through experimentation on the ImageNet-2012 [25], CIFAR-10, CIFAR-100 [26], Google commands, and UCI datasets [27], they demonstrate that mixing enhances the generalisation of state-of-the-art neural network designs. Additionally, their model improves the resilience of generative adversarial networks to adversarial cases and stabilises the training process of generative adversarial networks. As a result, Mixup broadens the training distribution by including the previously learned information that linear interpolations of feature vectors result in linear interpolations of the targets. The equation 1 is the formula of Mixup, in which (xi, yi) and (xj, yj) are two randomly selected samples from training data, respectively that xi is the feature vector, and y is a label. Furthermore, λ is a probability in the Beta distribution that is between 0 to 1.

$$\begin{aligned} \hat{x} &= \lambda * x_i + (1-\lambda) * x_j \\ \hat{y} &= \lambda * y_i + (1-\lambda) * y_j \end{aligned} \tag{1}$$

Since Mixup is one of the most potent data augmentation approaches, two various approaches, E-Stitchup and E-Mixup [28], which both are based on Mixup, are presented by Wolfe et al. However, since the type of text data is different from images, these two approaches augment the training data in the embedding space, not on word-level. These two approaches will be discussed in detail in the following sections.

1) E-Mixup: E-Mixup [28], which refers to Embedding Mixup as previously explained, is a Mixup approach in texts instead of images. In this method, embedding space is first created by pretrained models. Next, Mixup algorithm is performed on the embeddings of the training dataset. Below, E-Mixup equation is presented, e_1 stands for embedding of feature vector x_1 in (x_1, y_1) and e_2 stands for embedding of feature vector x_2 in (x_2, y_2). Both (x_1, y_1) and (x_2, y_2) has been chosen randomly from the training samples.

$$\begin{aligned} \hat{e} &= \lambda * e_1 + (1-\lambda) * e_2 \\ \hat{y} &= \lambda * y_1 + (1-\lambda) * y_2 \end{aligned} \tag{2}$$

2) E-Stitchup: This approach [28] is inspired by a recent experiment presented by Lian et al. [29]. This approach is similar to E-Mixup. However, it combines two embeddings by random selection. In addition, each of the combined embedding's indexes is the same as one of the two randomly selected embeddings. In the equation below, λ is the chance of selecting an element from either embedding vector, calculated from the Beta distribution.

$$e_i = \begin{cases} e_{1i} & Prob(\lambda) \\ e_{2i} & Prob(1-\lambda) \end{cases} \quad i \in [0, \text{embeddingsize}) \tag{3}$$

$$\hat{y} = \lambda * y_1 + (1-\lambda) * y_2 \tag{4}$$

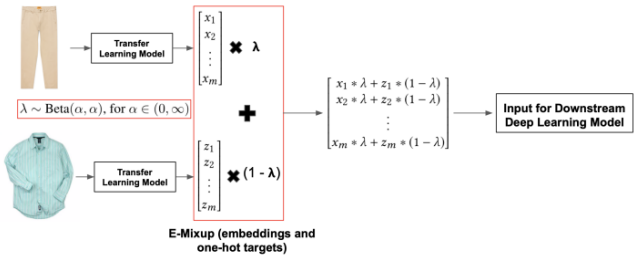

Fig. 1. Data Augmentation process in E-Mixup [28]

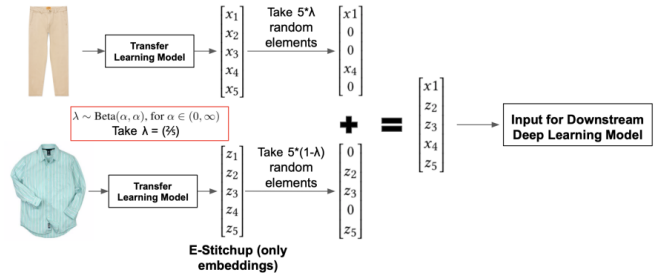

Fig. 2. Data Augmentation process in E-Stitchup [28]

Both E-mixup and E-Stitchup [28] have been tested on the fashion dataset [30], which includes both images and texts. Both methods outperformed benchmark models in confidence-accuracy correlation. In the figure 1 and 2 the overview of both approaches is presented. As described in figure 1, E-Mixup takes all the elements in the word vectors of training data. In contrast, in figure 2, E-Stitchup chooses $5 * \lambda$ elements randomly.

III. PRELIMINARY

A. Language Representation Learning

As described in previous sections, a good representation should describe information that doesn't belong to a specific field but can be helpful in model learning. A good representation should contain language rules and common sense, such as literary meaning, grammatical structure, and semantic roles. The main idea is to represent and explain the texts by vectors in a relatively low dimensions. Overall, there are two methods to embed each word. In the first method, contextual embeddings assign each word a representation based on its context; however, in the second method, the non-contextual word embedding model generates only one vector, ignoring the location of words in the sentence and the various meanings they may have.

B. Neural Contextual Encoders

Sequence and non-sequence models are two different groups of Neural Contextual Encoders. Sequence models can be grouped into Convolutions [31] and Recurrent [11] models. Non-sequence models used in this project also learn the underlying meaning of words by defining a tree or graph structure between words. One of the powerful methods in this regard is self-attention [32]. Self-attention is an attention mechanism that connects distinct points of the same sequence in order to compute a representation of it [32].

C. Pretraining

Due to the development of the deep neural networks in recent years, models in this field are getting more complex with increasing number of parameters. However, many parameters can make the machine learning models overfit, especially when only small datasets are available. In addition, creating large datasets with labels is difficult since it is time-consuming and it is costly to make labels. However, acquiring large datasets without labels is relatively easy. In the first step, these datasets can be used for different representation learning methods, and in the next step, with the use of the learned representation, they can be used in various machine learning tasks. Advantage of using pretrained models can be described as follows:

1) Pretraining on a large dataset can be used to learn the representation of words and sentences, and in the next step, it can be used to solve different tasks.

2) Pretraining can help initialize variables in the machine learning model.

3) Pretraining can be thought as a kind of regularization that helps prevent models (classifiers, generators, etc.) from overfitting on small datasets.

D. Application of data augmentation in Semi-supervised algorithms

Semi-supervised learning is a machine learning technique that involves training using a small amount of labeled data and a large amount of unlabeled data. Data augmentation is one of the useful techniques in machine learning that increase the amount of data for different purposes without collecting new data. In recent years many semi-supervised methods used this approach for the lack of data and to improve the accuracy of their model. One of the most recent methods in this area is Fixmatch [6]. This model creates pseudo-labels from weakly augmented unlabeled images utilizing the model's predictions. Moreover, in the next step, the pseudo-label is only kept if the model makes a high-confidence prediction on it. After that, the model will receive the strong augmentation of the same image, and it will be trained to predict the pseudo-label of the same image. In the figure 3 Fixmatch architecture can be seen. The main idea of FixMatch is to generate an artificial label for each unlabeled image and use it in a cross-entropy loss. As described, data augmentation is an effective technique used in many machine learning methods like Fixmatch [6]. There are different types of data augmentation specially in computer vision not applicable to NLP. However, data augmentation in NLP is not easy to make since any introduced modification can change the label of the data.

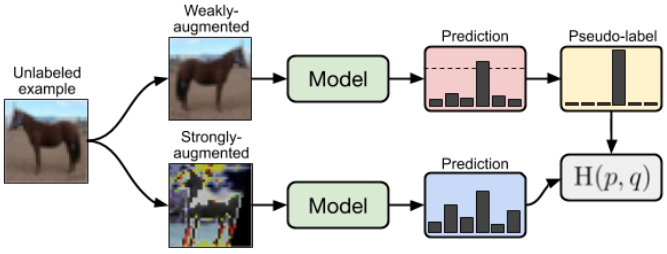

Fig. 3. FixMatch's architecture [6]

E. Bidirectional Encoder Representations from Transformers (BERT)

BERT [21], is a language representation approach introduced by Devlin et al. in 2018. Compared to previous language models, BERT learns representations from both left and proper contexts at each layer. BERT is simultaneously trained on the Masked Language Model (MLM) and Next Sentence Prediction tasks (NSP). In the MLM task, 15% of input tokens are randomly masked, and the model is trained to predict masked tokens. Since BERT's pretraining does not include any [MASK] tokens for downstream tasks, each token chose to be masked only 80% of the time. Moreover, they remain unmodified 10% of the time. In the remaining 10% of the time, they get replaced with a random token. In the NSP task, the model is given two phrases, A and B, and must determine if A follows B. In 50% of the time, B is the actual sentence that follows A, whereas, in the other 50%, B is selected randomly from the sentence corpus. Before any other token, BERT is additionally supplied a special [CLS] token whose output indicates whether B is the next sentence following A or it is not.

IV. Proposed Method

bej As pointed before, a data augmentation method that is simple, flexible, and controllable is needed. To address this requirement, a new method is proposed to first augment data samples in the word-level using EDA, and then use a transformation in the embedding space of a transformer-based language model to generate several augmented sentence embeddings. Given x is a data sample (a sequence of words), and y is its corresponding label, EDA is used to generate m new augmented sentences which with the original sentence itself will constitute $m+1$ samples. Next, BERT is employed to output an embedding for each of these sentences, resulting in $m+1$ sentence embeddings. A $(m+1) \times \text{Embeddingdimension}$ matrix is created by stacking these embeddings on top of each other. Embedding dimension is 768 since the bert-base-uncased model was used. The goal of the proposed method is to generate new augmented embeddings using the embeddings in the said matrix. Normally, a normal distribution could be fitted to these embeddings using a maximum-likelihood approach, but since the number of embeddings will be much smaller than the dimensionality of the embedding (latent) space, a dimensionality reduction algorithm (SVD) is first used to make the matrix smaller ($(m+1) \times (m+1)$) and then a normal distribution is fitted to the data. The fitting is made under the simplifying assumption that the columns of the new reduced embeddings are independent from each other. As a result, only the mean and standard deviation (std) across each column (dimension) is computed and new data will be generated using the calculated mean and std from a normal distribution. The new generated embeddings will be $m+1$ dimensional. Hence, the inverse of SVD operation is applied to the new embeddings to create embeddings in the original high dimensional embedding space. The details of this algorithm with the choice of hyperparameters are described below:

A. Algorithm Details

- (x, y) data sample is inputted, in which x is a sequence of words and y is the corresponding label.
- EDA with parameters $\alpha = 0.2$ and $n_{\text{aug}} = m = 7$ is used while limiting operations to synonym replacement, random insertion, and random deletion to prevent the sentence meaning from changing. As a result, 7 new augmented sentences will be generated.
- The original data sample is passed to BERT along with the 7 new generated augmented samples and the 768 dimensional embedding is received.
- Embeddings are stacked in a 8×768 matrix. We will call this matrix A.
- Mean of the rows of A is computed and is assigned to a variable called (mean_augmented_embeddings). A is then modified to have mean zero across each column:

$$A = A - \text{mean\_augmented\_embeddings} \tag{5}$$

- SVD is employed to reduce the dimensionality of A:

$$A = U \times S \times V^T \tag{6}$$

  We will call the reduced matrix B which will be set to $U \times S$.
- Mean and std of B is computed and assigned to mean_latent_embeddings (or μ for short) and std_latent_embeddings (or σ for short) respectively.
- New data in the 8-dimensional latent embedding space will be generated as follows:

$$\hat{x_{latent,i}} = \mathcal{N}(\mu_i, \sigma_i) \qquad i \in [0, 8) \tag{7}$$

- Data will be transformed to the original embedding space as follows:

$$\hat{x} = \hat{x_{latent}} \times V^T + \text{mean\_augmented}_{\text{e}}\text{mbeddings} \tag{8}$$

B. Algorithm Intuition and Justification

BERT embeddings have been used successfully in a variety of tasks, indicating the generalizability of these embeddings. Intuitively, the embedding space can be thought of as a meaning space. In the algorithm described above, we first try to find the borders of a meaning subspace by generating sentences in the word level that are assumed to have similar meaning and

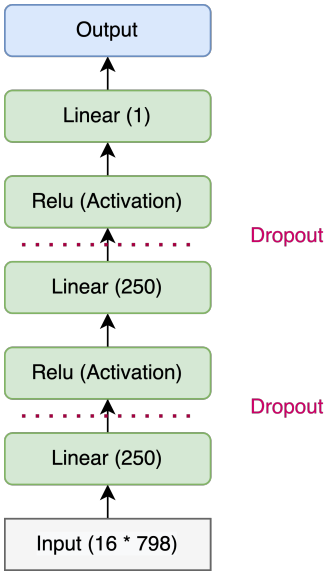

Fig. 4. Architecture of the classifier

computing their *meaning space* representation using BERT. Then this subspace is simplified to a lower-dimensional one, providing a way to flexibly generate as many samples as needed while the strength of augmentation is controllable by specifying the std of the fitted distribution. Moreover, computing variables such as mean_augmented_embeddings, mean_latent_embeddings and std_latent_embeddings can be done as a preprocessing step causing very small overhead compared to methods such as Back-Translation.

V. Experiments and Results

A. IMDb Dataset

This dataset includes a large set of highly polar movie reviews for binary sentiment classification. There are 25,000 samples for training, and 25,000 for testing, while additional unlabelled data is available.

B. Experimental Setup

1) Preprocessing: For simplicity and consistency across experiments, all augmentation operations are performed as preprocessing. Moreover, it should be noted that the goal of these experiments is to compare the new different augmentation methods and show empirically that the proposed algorithm (will be called DoubleA onwards) is superior in performance. Experiments will involve computing embeddings using BERT with and without EDA augmentation and performing E-Mixup and E-Stitchup afterwards. Then the resulting embedding are fed into a simple feed-forward classifier.

2) Classifier: There are two hidden layers of size 250 followed by Dropout ($p = 0.2$) and ReLU [33] activation function in the classifier, outputting a single value to predict the binary label. Architecture of the said classifier can be seen in

C. Results Summary

Summary of experimental results is described in Table V-C and Table V-C. The details of each experiment are represented in the following sections. Considering different combinations of and tuning the hyperparameters, there were 3 trials for each experiment and the reported results are the mean accuracy on test data across trials. Learning rate is 10^{-3} for Table V-C and 10^{-4} for Table V-C. As seen in the Table V-C and Table V-C, DoubleA is outperforming other augmentation methods. While the difference between reported accuracies of DoubleA and no augmentation is not large, the improvement is 3 percent or more compared to the rest of the methods. This is considering that the hyperparameter tuning was limited and further tuning could provide additional improvements. All in all, results show the simple yet effective DoubleA method could be used to enhance generalizability of classifier models.

Algorithm	Accuracy on Test Data
No Augmentation	77.884
DoubleA ($p = 1$ strength $= 1$)	77.886
EDA Only	72.9466
E-Mixup ($p = 0.2$, $\alpha = 0.1$)	76.696
E-Mixup ($p = 1$, $\alpha = 0.1$)	76.696
E-Stitchup ($p = 0.2$, $\alpha = 0.1$)	76.696
E-Stitchup ($p = 1$, $\alpha = 0.4$)	76.696

TABLE I
SUMMARY OF RESULTS WITH LEARNING RATE SET TO 10^{-3}

Algorithm	Accuracy on Test Data
No Augmentation	76.6746
DoubleA ($p = 1$ strength $= 1$)	76.9026
EDA Only	72.9466
E-Mixup ($p = 0.2$, $\alpha = 0.1$)	76.668
E-Mixup ($p = 1$, $\alpha = 0.1$)	60.8506
E-Stitchup ($p = 0.2$, $\alpha = 0.1$)	72.7253
E-Stitchup ($p = 1$, $\alpha = 1$)	61.452

TABLE II
SUMMARY OF RESULTS WITH LEARNING RATE SET TO 10^{-4}

1) No Augmentaion: When no data augmentation was applied, the only hyperparameter affecting performance was learning rate. Training loss and test accuracy curves are depicted in FigureV-C1 and Figure V-C1 respectively.

2) EDA Only: In the experiments where the only augmentation performed was EDA, the best performance was achieved when $\text{augment}_\text{p}\text{rob} = 1$, i.e. all data given to the model are augmented. Consistent with (REF Apple), classifier performance declined by 4%. One reason may be that the $\alpha = 0.2$ was too high and a lot of change was introduced to the sentences, resulting in changes in meaning. If this has been the case, it shows that DoubleA is robust to changes in meaning introduced by word-level augmentation. Training loss and test accuracy curves can be seen in Figure V-C2 and Figure V-C2 respectively.

3) DoubleA: Two hyperparameters were analyzed when experimenting with DoubleA. Similar to previous experiments, first parameter was augment probability or augment_prob. After various experiments $\text{augment\_prob} = 1$ was chosen

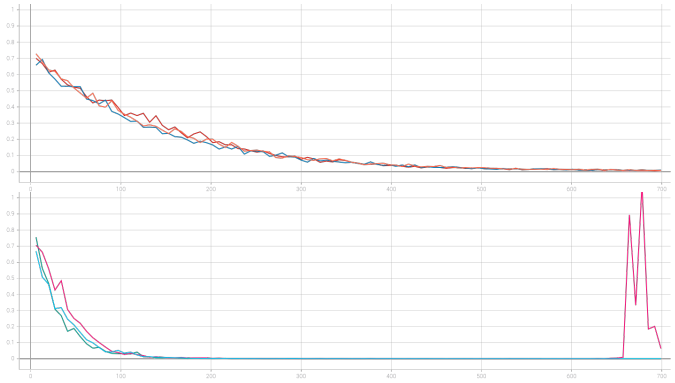

Fig. 5. Training Loss curves when no augmentation was involved for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

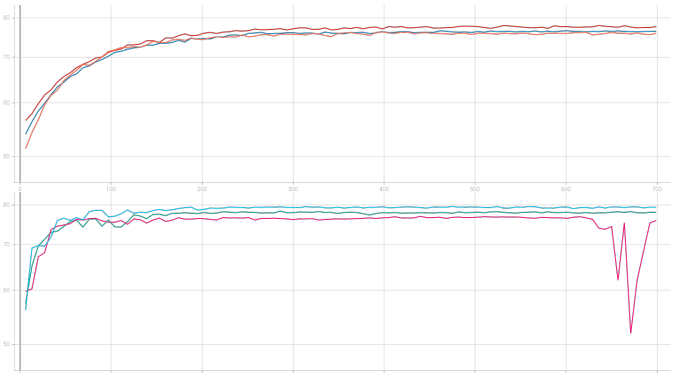

Fig. 6. Accuracy of model using no augmentation on test data during training for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

as the best value. Second parameter was augment_strength, values of which were picked from $\{0.5, 1, 1.5\}$. Experiments showed that augmentation with $\mathrm{augment\_strength} = 1$ and $\mathrm{augment\_strength} = 1.5$ significantly outperformed augmentation with $\mathrm{augment\_strength} = 0.5$. Among 1 and 1.5, 1 was chosen as the best value because of marginally better performance and smaller oscillations in the training and test loss curves. The Training loss and test accuracy curves are depicted in the Figure V-C4 and Figure V-C4 respectively.

4) E-Mixup: Hyperparamter tuning was performed on augment_prob and α. While EDA Only and DoubleA algorithms reached their highest performance when $\mathrm{augment_prob} = 1$, the accuracy for E-Mixup for this hyperparameter values was between 50-60%. The best hyperparameter values for E-Mixup were $\mathrm{augment\_prob} = 0.2$ and $\alpha = 0.1$ with which accuracy of 76% was achieved. As a result, using E-Mixup even when it is applied weakly (low α) does not improve the model's performance. Training loss and test accuracy curves are shown in Figure V-C5 and Figure V-C5 respectively.

5) E-Stitchup: Similar to E-Mixup, performance was significantly low when $\mathrm{augment\_prob} = 1$ while decreasing augment_prob and α would improve classifier's accuracy.

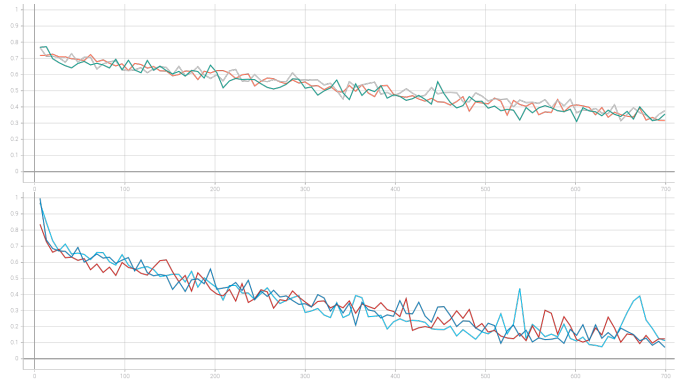

Fig. 7. EDA training Loss curves for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

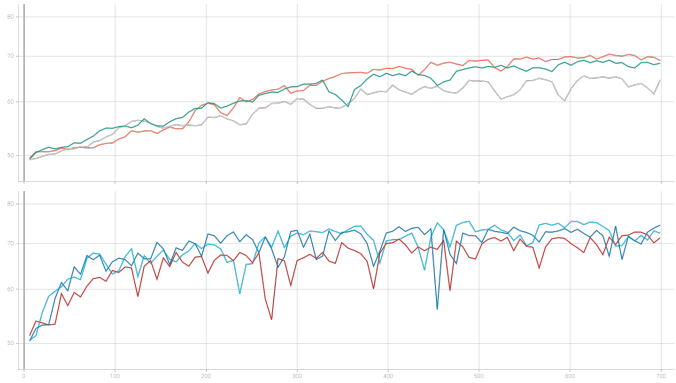

Fig. 8. Accuracy of model using EDA on test data during training for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

Overall, this method did not improve performance compared to when no augmentation was employed. The training and test loss curves are depicted in Figure(REF) and Figure(REF) respectively.

VI. Conclusion and Summary

In this study, we first identified the challenges in data augmentation for textual data and pointed out the possible underlying reasons behind the obstacles for augmentation. Inspired by these reasons and noticing the increasing use of pretrained transformer-based language models, we introduced a novel augmentation method that delivers flexibility on how many samples can be generated and control over the strength of augmentation while inducing a low computational overhead. With empirical experiments we showed that our method can outperform all other augmentation setups while leaving future work open to test and try different setups of our method to examine whether further performance gains are possible. The method provided suggests similar data augmentation capabilities to Computer Vision, enabling many Semi-Supervised models proposed in Computer Vision to be applied in NLP as well, especially in areas where only a limited set of data samples is available.

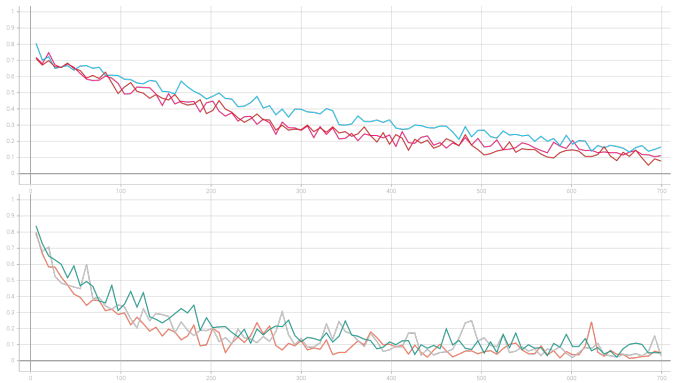

Fig. 9. DoubleA: Training Loss curves for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

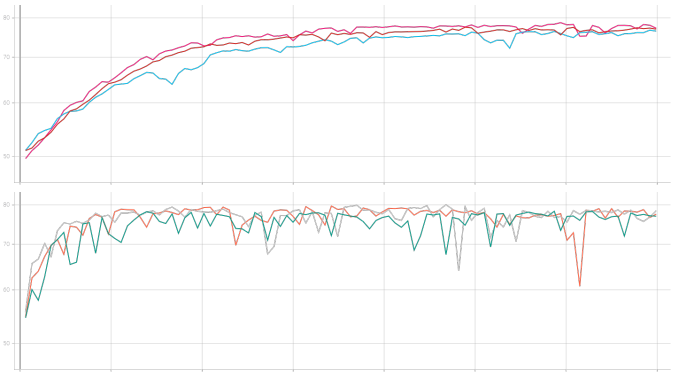

Fig. 10. DoubleA: Accuracy of model on test data during training for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

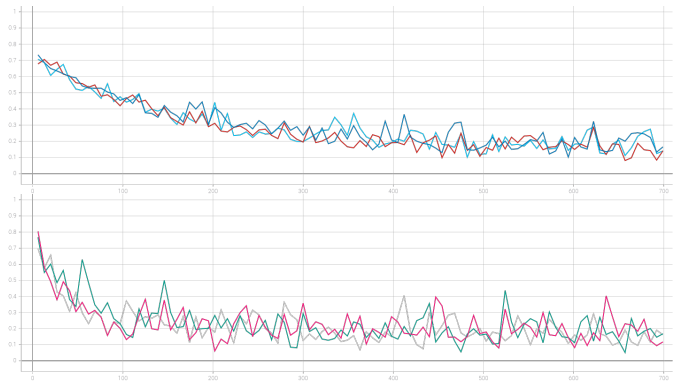

Fig. 11. E-Mixup: Training Loss curves for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

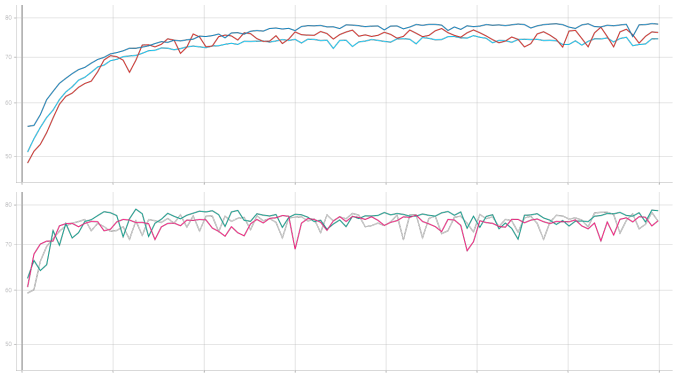

Fig. 12. E-Mixup: Accuracy of model on test data during training for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

REFERENCES

[1] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[2] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[3] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 181–196, 2018.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[5] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," in *2016 international conference on digital image computing: techniques and applications (DICTA)*, pp. 1–6, IEEE, 2016.

[6] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in Neural Information Processing Systems*, vol. 33, pp. 596–608, 2020.

[7] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," *Advances in neural information processing systems*, vol. 27, 2014.

[8] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[9] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[10] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[11] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for nlp," *arXiv preprint arXiv:2105.03075*, 2021.

[12] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," *arXiv preprint arXiv:2003.02245*, 2020.

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[14] W. Y. Wang and D. Yang, "That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 2557–2563, 2015.

[15] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[16] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[17] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6256–6268, 2020.

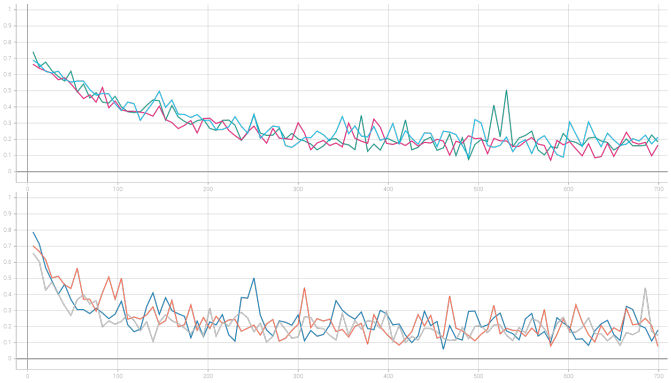

Fig. 13. E-Stitchup: Training Loss curves for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

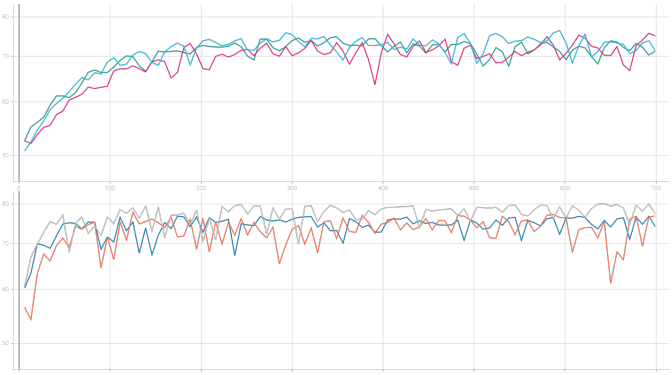

Fig. 14. E-Stitchup: Accuracy of model on test data during training for $\mathrm{lr} = 10^{-4}$ (up) and $\mathrm{lr} = 10^{-4}$ (down)

[18] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *AI Open*, 2022.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] S. Longpre, Y. Wang, and C. DuBois, "How effective is task-agnostic data augmentation for pretrained transformers?," *arXiv preprint arXiv:2010.01764*, 2020.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[24] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[25] "Imagenet large scale visual recognition challenge 2012 (ilsvrc2012)."

[26] "Cifar-10 and cifar-100 datasets https://www.cs.toronto.edu/ kriz/cifar.html."

[27] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[28] C. R. Wolfe and K. T. Lundgaard, "E-stitchup: Data augmentation for pre-trained embeddings," *arXiv preprint arXiv:1912.00772*, 2019.

[29] D. Liang, F. Yang, T. Zhang, and P. Yang, "Understanding mixup training methods," *IEEE Access*, vol. 6, pp. 58774–58783, 2018.

[30] P. Aggarwal, "Fashion product images dataset," Mar 2019.

[31] J. Yang and J. Li, "Application of deep convolution neural network," in *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 229–232, IEEE, 2017.

[32] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[33] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.