Data Augmentation for Deep Transfer Learning

Cameron R. Wolfe
UT Austin, Salesforce
wolfe.cameron@utexas.edu

Keld T. Lundgaard
Salesforce
klundgaard@salesforce.com

ABSTRACT

Current approaches to deep learning are beginning to rely heavily on transfer learning as an effective method for reducing overfitting, improving model performance, and quickly learning new tasks. Similarly, such pre-trained models are often used to create embedding representations for various types of data, such as text and images, which can then be fed as input into separate, downstream models. However, in cases where such transfer learning models perform poorly (i.e., for data outside of the training distribution), one must resort to fine-tuning such models, or even retraining them completely. Currently, no form of data augmentation has been proposed that can be applied directly to embedding inputs to improve downstream model performance. In this work, we introduce four new types of data augmentation that are generally applicable to embedding inputs, thus making them useful in both Natural Language Processing (NLP) and Computer Vision (CV) applications. For models trained on downstream tasks with such embedding inputs, these augmentation methods are shown to improve the AUC score of the models from a score of 0.9582 to 0.9812 and significantly increase the model's ability to identify classes of data that are not seen during training.

1 INTRODUCTION

In many deep learning applications, the ability to generalize from one distribution of data to another is quite difficult. Due to the lack of alignment between training and testing distributions, many deep learning models struggle with overfitting and fail to perform well in real world environments, despite high training accuracy. Much of this performance discrepancy is caused by the sampling bias that is created when randomly selecting a training set, as the distribution of this training set will not perfectly match the distribution of data that is seen in the real world. To combat issues with overfitting, many methods, such as transfer learning and data augmentation, have been employed. However, the generalization performance of deep learning models is still in need of further improvement.

Transfer learning has become an extremely popular method of reducing overfitting in deep learning. In CV, large, pre-trained convolutional neural networks (CNNs) have been shown to perform well when fine-tuned to accomplish other, downstream tasks after pre-training [4, 13]. In a similar vein, deep learning models in NLP have began to heavily utilize self-supervised learning, in which deep learning models are pre-trained on large corpora of unlabeled text to yield pre-trained language models that can be fine-tuned to accomplish downstream NLP tasks [5, 20]. In addition to fine-tuning pre-trained models to accomplish downstream tasks, one also can use a pre-trained model to create vector embeddings of data that can be used as input to a separate model, which is then trained to accomplish the downstream task [2, 9]. Such an approach avoids the cost of fine-tuning large pre-trained models, focusing instead on training the smaller, downstream model.

Generalization performance can be improved with the use of data augmentation. In CV, such augmentations typically take the form of either geometric or color augmentations on input images, which have been extremely effective at reducing overfitting in CNNs [3, 18]. More recently, Mixup was proposed, which applies data augmentation by taking a weighted average of two images and trying to predict the weights of each class in the output layer [22]. Mixup has been shown to yield several benefits, such as reducing overfitting and better calibrating the confidence of deep learning models [11, 19]. Data augmentation has also recently been expanded to NLP tasks [6, 10]. However, augmenting textual data has proven to be quite difficult due to the fact that replacing or changing words within a corpus of text could easily destroy the semantic meaning of a sentence or phrase. In this work, the proposed forms of data augmentation draw on ideas from all such forms of data augmentation, but are applied to embedding inputs (i.e., either textual or image vectors), instead of raw data inputs. To our knowledge, such methodology of augmenting embedding inputs for deep learning has not yet been explored.

Recent research has begun to address the tendency of one-hot labels to foster overfitting and overconfidence in supervised learning domains. Such overconfidence, characterized by a model outputting extremely high logits for a single class relative to other classes in the output distribution, can be detrimental to model performance because it diminishes the model's generalization accuracy and leads to poor performance in situations that rely on sampling from the output distribution of the model, such as beam search [12], because the output distribution is too peaked. Label smoothing was proposed as a way to better calibrate model confidence and avoid such issues created by one-hot target vectors by computing the target vector as a weighted average between the one-hot target and a uniform distribution [16]. Similarly, Mixup was also shown to regularize model confidence [11], due to the fact that it produces soft labels for use during training. In this work, we further explore the use of label softening in combination with the proposed forms of embedding augmentation and expand the use of label softening to situations in which a softmax output transformation is not used.

One of the largest bottlenecks in deploying useful deep learning models is the lack of sufficient labeled data for supervised learning, which worsens sampling bias and makes models more susceptible to overfitting. Therefore, finding ways to improve generalization performance of deep learning models, especially with limited training data available, is an important topic of current deep learning research. In this work, we focus on ways to improve the generalization performance of deep learning models by leveraging embeddings generated by large, pre-trained networks. Specifically, our work shows that by deriving new forms of data augmentation that are generally applicable to such embedding representations, the generalization performance of downstream models that utilize these embeddings as input can be improved. Additionally, these

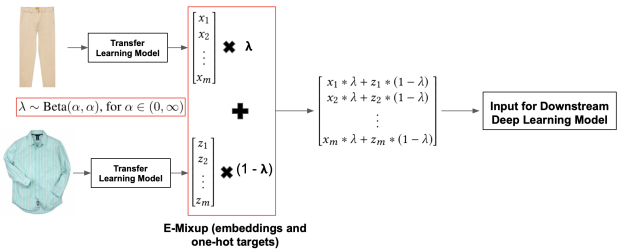

Figure 1: Outlines how embedding representations are augmented using E-Mixup. Both the embedding inputs and their associated target vectors (i.e., one-hot prediction targets) are augmented using the same process outlined above.

novel data augmentations, especially when combined with label softening, can be shown to regulate overconfidence in downstream models and allow such models to accurately identify unseen classes of data, thus pinpointing data that must be labeled and included in the training set to further improve the model's performance.

The organization of the paper is as follows: First, the methodology of the paper will be proposed. This methodology includes several novel forms of data augmentation that are generally applicable to embedding representations used as input to deep learning models. Following the methodology, the experiment details and the results of each of the experiments will be outlined and compared to the control experiment in order to highlight the benefits of the proposed augmentation methods. Next, analysis of these methods will be presented, followed by possible ideas for future research in the area. Lastly, the major conclusions of the work will be summarized.

2 METHODOLOGY

The novel contributions of this work are as follows:

(1) Four new forms of data augmentation are proposed that are generally applicable to embedding representations of data.
(2) These data augmentations are shown to be effective in increasing validation performance and regulating model overconfidence.
(3) The benefits and drawbacks for each of the data augmentation methods are outlined to determine the situations in which they are most useful.

The novel forms of data augmentation proposed in this work avoid any fine-tuning or modification of pre-trained models and ensure that the downstream network will never see the same input twice, thus reducing overfitting. Additionally, because embedding inputs can be pre-calculated for all data such that inference is only run once on the pre-trained model for each element of data (i.e., to produce the associated embedding of the data), these augmentation methods have minimal added cost, only creating an extra constant factor of complexity within the processing of each mini-batch. These new contributions form a basis for optimizing the performance of downstream deep learning models that utilize any type of embedding inputs, especially when minimal training data is available.

2.1 Creating Text and Image Embeddings

For both textual and image data, it is possible to use pre-trained deep learning models to produce embedding representations of data, which contain quantitative characterizations of the associated data and can be used effectively as input to a deep learning model. Such embeddings can be produced by a variety of different pre-trained models. For textual data, recently proposed transformer architectures, such as BERT [2, 5] and XLNet [21], can be used to create textual embeddings of phrases or sentences. Similarly for images, modern deep learning models such as Residual Networks [7] can be used to produce image embeddings, as well as some older CNN architectures such as VGG [15], which has been shown to be surprisingly effective at producing useful image embeddings. Such embeddings are created by passing the image or textual data as input to a pre-trained model and using the activation values within the last layers of the pre-trained network to produce embedding vectors. Because the final layers of such models generally contain informative, semantic information about the input data within their activations, such embedding vectors tend to be quite descriptive and form good inputs for deep learning models. In this work, all data is transformed into an associated embedding before being passed as input to the downstream model.

2.2 E-Mixup

The first form of data augmentation proposed in this work is referred to as E-Mixup, which stands for "Embedding Mixup". This form of data augmentation draws inspiration from the original idea of Mixup [22], a form of data augmentation commonly used in computer vision applications. In E-Mixup, two input embeddings are combined by first sampling a random value, lambda, from a Beta distribution having a parameter alpha. It should be noted that, in this work, alpha is used to refer to this distribution parameter, as opposed to the learning rate, and generally lies in the range of 0 to 0.5. Once the value of lambda has been sampled, E-Mixup takes a weighted average over the embedding inputs of two unique training examples, where lambda is the weight of the average. A different lambda value is sampled each time two embeddings are combined, thus randomly perturbing each sample that is passed as input to the model and ensuring the model will never see the same input twice. E-Mixup, as well as all other proposed embedding augmentation methods, cause the size of each mini-batch to be halved because each mini-batch is constructed by combining pairs of embeddings into a single input. The process of combining two input embeddings using E-Mixup can be visualized in Fig. 1.

It should be noted that E-Mixup is performed both on a pair of input embeddings and on the target, or label, vectors associated with these input embeddings. The resulting target vector for a combined embedding input after E-Mixup has been applied is a weighted average of the two original, one-hot target vectors, thus creating two classes with positive probability in the target output distribution. This has the effect of softening the target labels that are being predicted by the model. This softening of the target vector has a regularizing effect on the downstream classification model, as will be further outlined in following sections (see Sec. 2.4).

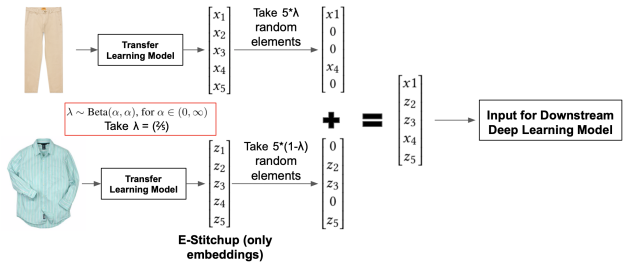

Figure 2: Outlines how embedding representations are augmented in E-Stitchup. The label vectors associated with these embedding inputs are not handled in the same way, but are instead handled by taking a weighted average of the two label vectors, as in E-Mixup.

2.3 E-Stitchup

The second proposed form of data augmentation is referred to as E-Stitchup, which stands for "Embedding Stitchup". Similarly to E-Mixup, E-Stitchup creates a combination of two unique training examples. However, instead of taking a weighted average of a pair of embedding inputs, this method randomly samples elements from each of the two vectors to create a combination of the two (e.g., 1/4 of the elements may come from one vector, while 3/4 of the elements will come from the other). This sampling of embedding elements is performed by randomly choosing an element from one of the two vectors to populate each index of the resulting vector. Similarly to E-Mixup, the probability of choosing an element from either vector in E-Stitchup (i.e., the ratio of elements to take from each vector) is determined by sampling a value, lambda, from a random Beta distribution with parameter alpha. The resulting, mixed embedding is the same size as the original and contains elements from either of the original embeddings at each of its indices. The process of combining two input embeddings using E-Stitchup can be seen in Fig. 2.

Although E-Stitchup augments the input embeddings differently than E-Mixup, the associated label vectors are handled identically as in E-Mixup, by taking a weighted average of the two label vectors. The weight used to take this average is the same randomly sampled weight, lambda, that determines the ratio of elements to sample from each embedding vector when E-Stitchup is performed. Therefore, similar to E-Mixup, E-Stitchup yields softened target vectors where multiple classes have nonzero probability.

2.4 Soft E-Mixup and Soft E-Stitchup

The third and fourth forms of data augmentation incorporate a change that can be added to both E-Mixup and E-Stitchup. The process of creating the mixed input embeddings is identical to E-Mixup and E-Stitchup, respectively. However, the way in which the associated label vector is created is slightly modified. Again, a weighted average of the two one-hot target vectors is taken, thus resulting in a soft target vector to be predicted by the model. Once this target vector is created, its values are randomly perturbed by subtracting a small value from the two positive classes and adding a small value to the rest of the negative classes. The values added to the negative classes are normalized such that the total probability distributed among all negative classes is equal to a value of one, which is possible because a binomial output transformation is used in the downstream classification model instead of softmax (See Sec. 3.3). By default, a value of 0.05 was subtracted from each of the positive classes and the subtraction was clamped such that it will never result in a negative class probability (i.e., if subtracting 0.05 results in a negative value, the value is set equal to 0). This process, in effect, further softens the target vector and introduces noise into the target that the model is trying to predict, thus creating a regularizing effect on the network. This method of softening the resulting label vector is applied to both E-Mixup and E-Stitchup, which, when used with softened labels, are referred to as Soft E-Mixup and Soft E-Stitchup, respectively.

As an example of how soft labels would be created during training, consider two training elements that are classified as class one and two, respectively. Now, assume that the lambda value, or the weight, sampled during E-Mixup is equal to 0.2. Then, the target vectors, when mixed together, would yield a single target vector with a value of 0.2 for class one and 0.8 for class two, representing the mixed probability of the two original classes. In order to soften this label vector, a value of 0.05 is subtracted from each of the positive classes, resulting in probabilities of 0.15 and 0.75 for classes one and two, respectively. It should be noted that if this subtraction resulted in any negative value, the negative value would be set equal to zero, as there cannot exist any negative probabilities in the target vector. Then, a small value must be added to each of the negative classes, such that a total value of one is distributed across all negative classes. Assume that there exist a total of 12 possible output classes, leaving 10 negative classes to which a small value must be added. In this case, a value of 0.1 would be added to each of the negative classes, resulting in a final target vector having probabilities of 0.15 and 0.75 for the first two classes and a probability of 0.1 for all other classes. This softened vector would then be used as the target for the model if soft labels are being utilized during training, such as in Soft E-Mixup or Soft E-Stitchup.

2.5 The "None" Category

Because the ability to identify data that belongs to an unseen class is a useful skill for deep learning models in production environments, the proposed augmentation methods are not only evaluated on their ability to improve validation performance and reduce overfitting, but also on their ability to correctly identify data that belongs to a class that is not included in the training distribution (i.e., assign low probability to all classes in the output distribution). In order to identify data that belongs to no class, a confidence threshold is created. To predict data into a given class, the model's probability assigned to this class in the output distribution must be greater than the confidence threshold, otherwise the prediction is discarded. If multiple classes are given probabilities greater than the confidence threshold, the class with the greatest probability is selected (i.e., multiple classes cannot be predicted for a single input). Additionally, if no classes are assigned probabilities greater than the confidence threshold, the data is considered to not be a part of any class, which is referred to in this work as the "none" category. This concept of the "none" category is used throughout the analysis of the proposed

augmentation methods, as the ability to identify products accurately as part of the "none" category is one of the major goals of the proposed augmentation methods. Such an ability to identify data that belongs to an unseen class is important for deep learning models that are exposed to continuously expanding datasets (i.e., any deep learning model in production), as it can be used to identify data that needs to be labeled and data on which the model is not performing well.

3 EXPERIMENTAL DETAILS

All experiments were performed using the Fashion Product Images Dataset. Separate experiments were performed to analyze the effect of each proposed data augmentation method on the performance of downstream classification models. Each of the experiments that are presented were repeated for several trials with different training and validation splits to ensure the consistency of the results. For each of the augmentation methods, various settings of the alpha parameter, or the parameter to the distribution from which the augmentation weight is sampled, are explored using a grid search. However, only a single alpha value is generally presented for each experiment, which was determined through grid search to be the empirically optimal value. The control experiment, which is presented alongside results for all proposed augmentation methods, corresponds to an experiment in which no augmentation is used. All parameters and settings for the control experiment, besides the use of embedding augmentation, are kept identical to the other experiments.

3.1 Fashion Product Images Dataset

For the experiments performed in this work, the Fashion Product Images dataset is used, which is available at [1]. This dataset contains data for 44K apparel products, each of which has an associated image, product title, and product description. These products are classified into 171 unique categories of products. The textual data associated with each product (i.e., the product title and product description) is converted into an embedding vector following the procedure outlined in 3.2. Additionally, all images are converted into an associated embedding representation following the same procedure. These embeddings, including two text vectors and one image vector, are concatenated together before being fed as input into a fully connected classification model, and the model is trained to predict a product's associated class given these input embeddings.

3.2 Embedding Models

In this work, the BERT transformer model [5] is utilized to create phrase and sentence embeddings. Our implementation utilized the BERT Base model (i.e., HuggingFace PyTorch implementation) for the creation of all textual embeddings. This model was never fine tuned or modified in any way. To create these textual embeddings, input phrases are first tokenized using a WordPiece tokenizer [14]. The resulting tokens are then converted into token embeddings and fed as input into the BERT Base model. Once the forward pass of BERT is complete, the sentence embedding is created by averaging the output activation vectors corresponding to each input token in each layer of the transformer, thus yielding a single average activation vector for each layer, and concatenating the average output activation vectors of the final two layers. This process creates an embedding vector with 1536 elements to represent a textual phrase. If there are multiple phrases associated with a single data element (e.g., a product on an e-commerce site may have both a product title and description), embeddings are created separately for each of these phrases and then concatenated together.

All embeddings for image data were created with the EfficientNet B4 model [17]. This pre-trained CNN model was never fine-tuned or modified in any way. To create the image embeddings, the original image is passed as input into EfficientNet to retrieve the activation maps at each layer of the CNN. From these activation maps, the resulting image embedding is created by performing a global average pooling on the final convolutional layer of the network, thus yielding a single value for every channel of the feature map at this layer. This process creates an embedding vector with 1792 elements to represent each image. In cases where both image and textual data are available, all image and textual embeddings are created separately and concatenated together before being passed as input to the downstream model.

3.3 Classification Model

Once the input embeddings are augmented using the chosen form of data augmentation (e.g., E-Mixup, E-Stitchup, Soft E-Mixup, or Soft E-Stitchup), the augmented examples are fed into a downstream classification model. This model is a deep, fully-connected network, which accepts a fixed size input and outputs a probability distribution over all possible classes. The model used in this work is comprised of two hidden layers of size 250, although the model size may need to be increased or decreased depending on the situation. This downstream model is significantly smaller than most pre-trained models used for transfer learning and can be retrained at a low computational cost. Each hidden layers is followed by a Dropout layer with probability of 0.3, as well as a Rectified Linear Unit activation. The last layer of the model, however, is not followed by a Rectified Linear Unit activation.

A binomial output transformation, or an element-wise sigmoid activation, is applied to the model's output layer before the predicted class is determined. This binomial output transformation is used instead of Softmax so that the model has the ability to assign low probability to all classes, thus enabling the model to handle data that does not belong to a class that was seen during training by assigning low probability to all classes. The ability to identify such unknown classes is a common problem in deep learning. Models trained using softmax output layers tend to be overly confident when running inference on such unknown data due to the fact that they are forced to assign non-zero probability to some class. In cases where there is very limited training data, such an ability to identify data within an unseen class is useful, as it can prevent inaccurate predictions on such data and identify portions of a dataset that are in need of labeling. Furthermore, in a setting with a continuously expanding dataset (i.e., any deep learning model running in production), the proposed methodology is quite useful, as data that belongs to an unknown class can be identified and labeled before retraining to maximize the performance of the model.

3.4 Training Parameters

Each experiment for the proposed forms of data augmentation utilize identical training parameters. For every experiment, a linear learning rate cycle is utilized that fluctuates from a learning rate of 0.0003 to 0.003 with a step size of 12 epochs and weight decay was set to 0.0001. Training is continued for 576 epochs for most cases, including both control experiments (i.e., those that utilize no data augmentation) and augmentation experiments, to ensure convergence.

For all experiments, only 10 percent of the available data is used for training (i.e., about 4,400 of the available 44,000 total products). The rest of the data is used for validation, and all results presented in this work are measured using this validation set. Such a small training set is used in order to simulate a scenario when very limited training data is available, which is when such augmentation is most useful. Such a limited amount of training data is common for deep learning models in production, in which one would want to label the least amount of data possible before having a high-performing model. In such cases, reaching high accuracy with only 10 percent of known data, or even less, would be ideal. Additionally, including fewer products in the training set allows some classes of products to be excluded from the training set, which, in turn, allows the augmentation methods to be evaluated by their ability to identify products that belong to the "none" category, as described in 2.5. For each trial of every experiment, a different training and validation split is created with 10 percent training data and 90 percent validation data, thus allowing the consistency of the results across different validation splits to be ensured.

3.5 Accuracy Metric

There exist multiple manners in which the accuracy metric can be defined for the purposes of this work, which raises the need to clearly define the accuracy metric as it is used in this context. For all results presented, accuracy is considered to be the top-one accuracy of class probabilities (i.e., the class that is assigned the highest probability in the model's output layer is considered the predicted class). However, if the confidence of the prediction (i.e., the probability of the output element after sigmoid is applied) is less than a certain confidence threshold, then the product is considered to be in the "none" category. Similarly, all product classes that are not present in the training set, but are present in the validation set, are considered to be a part of the "none" category. A product is considered to be classified correctly if the top-one class, or the class with the highest probability, is equal to the correct class and the top-one prediction has a probability that is higher than the confidence threshold. A correct classification occurs when either the top-one prediction, with probability above the confidence threshold, is equal to the actual labeled class, or if no classes are assigned a probability higher than the confidence threshold when the product is part of the "none" category.

The confidence threshold is a hyperparameter that can be used to tune the performance of the different augmentation methods. By default, this value is set to 0.6. However, the confidence threshold can be slightly changed to yield different performance, as described in Sec. 5.1.

4 RESULTS

In this section, the results of the proposed augmentation methods, as well as a control model trained without any augmentation, are provided. These models were evaluated in terms of their ability to both improve generalization accuracy and identify products in the "none" category. The alpha values used for each of the augmentation methods were chosen using a grid search over possible alpha values, which yields the optimal alpha value that is used for each of the respective augmentation methods.

Aug. Method	Weighted AUC
Control	0.9582 ± 0.0015
E-Mixup	0.9763 ± 0.0002
E-Stitchup	0.9768 ± 0.0002
Soft E-Mixup	0.9795 ± 0.0001
Soft E-Stitchup	**0.9812 ± 0.0004**

Table 1: Displays weighted average AUC scores across all product categories, including the "none" category, for models trained with each of the proposed augmentation methods. These values represent the average of AUC scores recorded across multiple trials of the experiment. The maximum deviation from each of the average values is listed next to each value.

The performance of each of the augmentation methods is measured in various manners. First, accuracies for both the "none" category and other categories are measured separately to determine the performance of each augmentation method in relation to the control (Fig. 3). It should be noted, however, that the accuracy metric is dependent upon the value of the confidence threshold, due to how accuracy is defined (see Sec. 3.5). To avoid this dependence on the confidence threshold, ROC curves and AUC scores, which evaluate model performance in a way that is independent of the value of the confidence threshold, are presented for each of the augmentation methods and the control.

The ROC curve that is provided (see Fig. 4) is a weighted average ROC curve across all product categories within the dataset, including the "none" category. The weight for each product class is determined by the ratio of examples that product class has in the dataset (i.e., a class with lots of examples relative to other classes will be weighted more). Similarly, the associated AUC scores in Table 1 are determined by measuring the AUC of these weighted average ROC curves. Furthermore, each of these ROC curves are recorded across multiple trials of the experiment and the average of all such trials is presented in Fig. 4 and Table 1.

4.1 E-Mixup

As seen in Table 1, E-Mixup has an AUC score of 0.9763, as compared to a score of 0.9582 for the control. From this AUC score, it is known that E-Mixup has better validation performance than the control, both for general product categories and for the "none" category. E-Mixup has the lowest AUC score of all proposed augmentation methods despite its improvements over the control, but its performance is only slightly behind that of E-Stitchup.

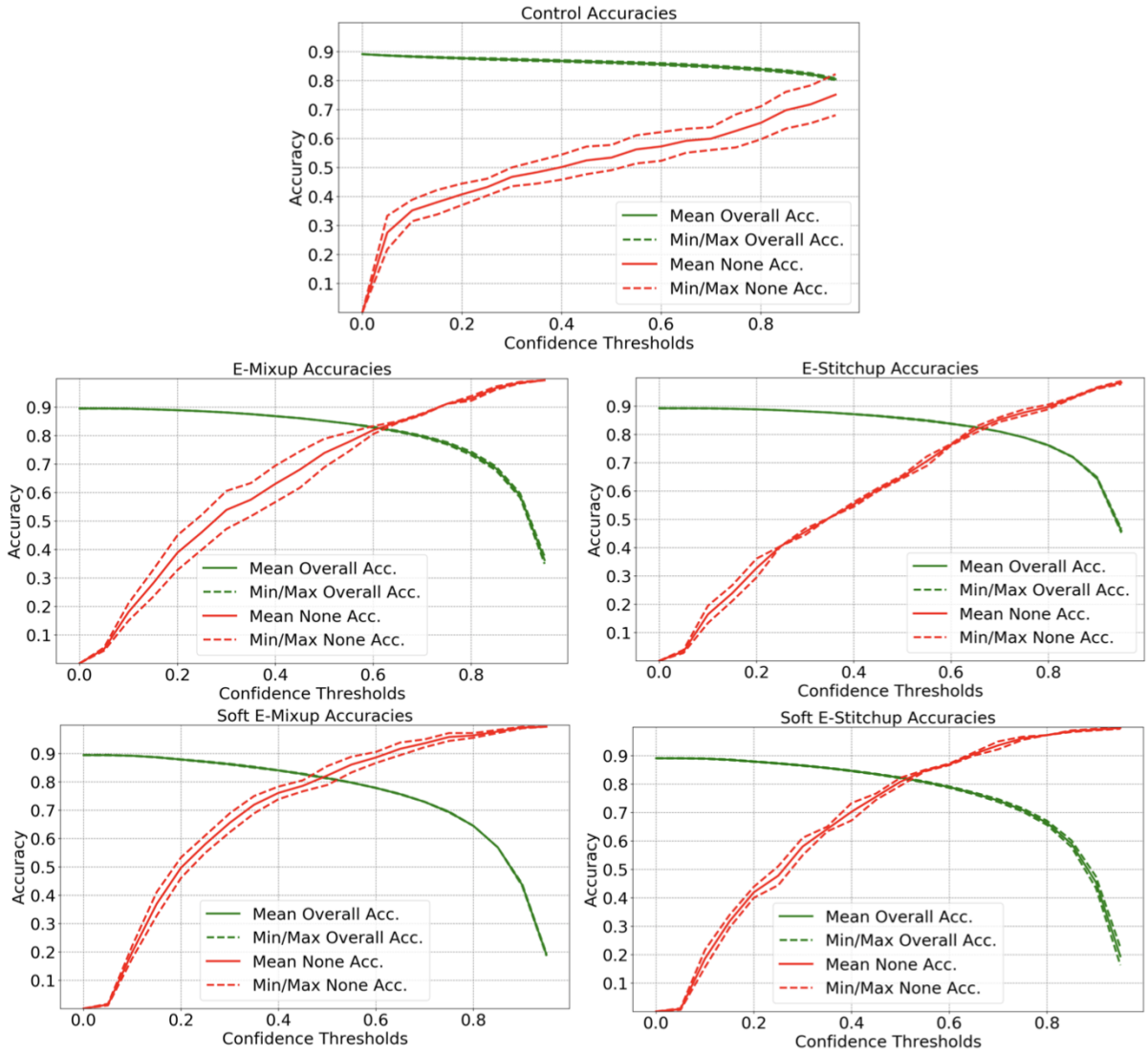

Figure 3: Overall and "none" category accuracies for both the control experiment and all augmentation methods across various confidence thresholds. This figure displays the relationship between overall and "none" category accuracy, as well as the performance capabilities of models trained with each augmentation method. The solid lines represent the average of accuracy values recorded across multiple trials, while the dotted lines represent the minimum and maximum accuracy recorded across multiple trials.

When label softening (see Sec. 2.4) is applied to E-Mixup, the AUC measure is still superior to the control, yielding an AUC score of 0.9795 for Soft E-Mixup. Soft E-Mixup has the 2nd highest AUC score of all proposed augmentation methods, thus highlighting its effectiveness in improving generalization performance. Additionally, Soft E-Mixup performs better than both the control and E-Mixup, as it improves the weighted AUC score by 0.0213 and 0.0032, respectively.

4.2 E-Stitchup

As seen in Fig. 4, E-Stitchup yields an AUC score of 0.9768. E-Stitchup has a better weighted AUC score than the control and results in the 3rd best AUC score of the proposed augmentation methods. E-Stitchup performs better than E-Mixup in terms of AUC score. However, the difference in performance between these two methods is quite minimal (i.e., their AUC scores differ by a value of 0.0005), thus revealing that these augmentation methods yield similar performance in the downstream classification model.

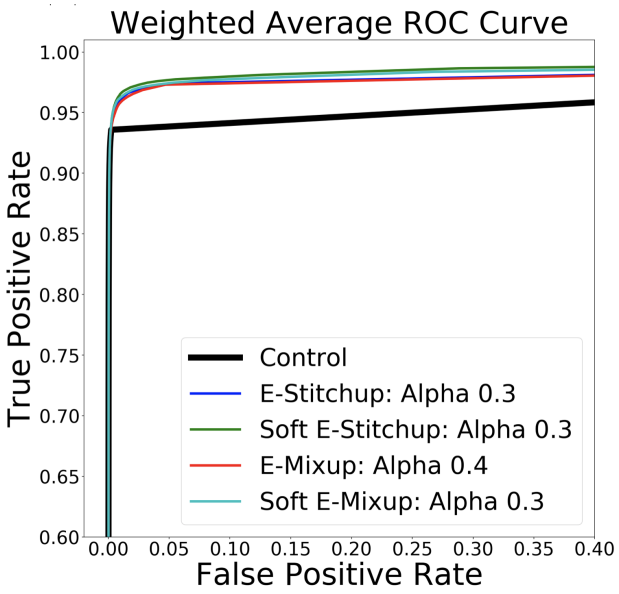

Figure 4: Illustrates the weighted average ROC curve across all product categories, including the "none" category, for models trained with each of the proposed augmentation methods. These curves are the average of ROC curves produced across multiple trials of the experiment. The ROC curve associated with the control experiment, which used no embedding augmentation, is bolded.

Soft E-Stitchup yields an AUC score of 0.9812, which is the best AUC score out of all proposed augmentation methods. Such a high AUC score highlights the effectiveness of Soft E-Stitchup in improving the downstream model's generalization performance. Again, Soft E-Stitchup yields better performance than the control. Additionally, the performance of Soft E-Stitchup and Soft E-Mixup are somewhat similar, as their AUC scores differ only by a value of 0.0017.

5 ANALYSIS

As seen in Table 1 and Fig. 3, each of the different augmentation methods has its own strengths and weaknesses. In this section, the behavior of each of the proposed augmentation methods will be outlined such that the situations in which they would each be most useful can be understood.

5.1 E-Mixup and E-Stitchup

From the provided results, it is clear that both E-Mixup and E-Stitchup provide better overall and "none" category performance in comparison to the control, as revealed by their superior AUC scores. However, the actual performance, or accuracy measure, of models trained with these methods is dependent upon on the value of the confidence threshold, which can be manipulated to find different balances between overall and "none" category accuracy as seen in Fig. 3. For example, using E-Mixup or E-Stitchup, the confidence threshold can be set to 0.65 to yield an accuracy of about 83 percent both overall and for the "none" category. Similarly, the validation accuracy can be increased to a maximum value of about 90 percent by decreasing the confidence threshold to 0.1, which leads to an associated decrease in "none" category accuracy.

In the provided experiments, the control is not able to achieve a "none" category accuracy that is equal to or greater than the model's overall accuracy. However, both E-Mixup and E-Stitchup are able to achieve such a balance between the two accuracy metrics, as revealed by the points at which the two curves intersect in Fig. 3. Such an ability to achieve a performant balance between overall and "none" category accuracy highlights the ability of E-Mixup and E-Stitchup to simultaneously balance performance in each of these domains, which, as seen in the control experiment, is not easily achievable without embedding augmentation. Additionally, the highest overall accuracy achieved by both of these methods (i.e., at a confidence threshold of 0.1) is slightly higher than that of the control experiment, thus revealing that these methods can also be used to achieve higher generalization performance.

The sweep over possible confidence thresholds, as seen in Fig. 3, was performed to examine the entire scope of performance for each of the possible augmentation methods. In this experiment, it can be seen that the overall accuracy decreases as the confidence threshold increases, while the "none" category accuracy increases as the confidence threshold increases for all experiments. Similarly, as the confidence threshold decreases, the overall accuracy increases while the "none" category accuracy decreases for all experiments. The confidence threshold must be set such that a good balance between these metrics is achieved. Although the control may yield higher overall accuracies at certain confidence thresholds, the control is not able to achieve a balance between overall and "none" category accuracy that is comparable to that of the augmentation experiments and, instead, tends to only perform well in terms of overall accuracy. Therefore, this sweep over possible confidence threshold values both illustrates the relationship between the confidence threshold and accuracy and reveals that E-Stitchup and E-Mixup are able to achieve a more performant balance between overall accuracy and "none" category accuracy when compared to the control.

5.2 Soft E-Mixup and Soft E-Stitchup

Both E-Mixup and E-Stitchup were also evaluated with the use of softened labels (see Sec. 2.4). Both methods of softened embedding augmentation perform better than the control, as revealed by the respective AUC scores of Soft E-Mixup and Soft E-Stitchup (see Table 1). Moreover, models trained with Soft E-Stitchup have the highest AUC scores of models trained with all proposed augmentation methods, thus highlighting the effectiveness of Soft E-Stichup in improving validation performance. Additionally, Soft E-Mixup yields the second highest overall AUC score when compared to the other augmentation methods. From these observations, it is evident that label softening, which results in the two highest performing experiments in terms of AUC score, effectively regularizes the downstream model and improves generalization accuracy more than the other proposed augmentation methods. Therefore, Soft E-Stitchup and Soft E-Stitchup are shown to be best choice of embedding augmentation for improving the generalization performance

of a downstream classification model. As observed in Fig. 3, Soft E-Mixup and Soft E-Stitchup are able to achieve a wide range of different performance balances between overall accuracy and "none" category accuracy. For example, at a confidence threshold of 0.5, both methods achieve overall and "none" category accuracies of about 82 percent. As mentioned in the previous section, the control is never able to achieve such a balance between "none" category and overall accuracy and, instead, tends to only perform well in terms of overall accuracy.

Although Soft E-Mixup and Soft E-Stitchup are both able to achieve more performant balances between overall and "none" accuracy when compared to the control, it is also interesting to observe their scope of performance in comparison to the other proposed augmentation methods. As seen in Fig. 3, the augmentation methods with soft labels achieve equal overall and "none" category accuracies at a confidence threshold of about 0.5, while those without soft labels achieve such a balance at a confidence threshold of about 0.65. By finding this point of equal overall and "none" category accuracy at a lower confidence threshold, Soft E-Mixup and Soft E-Stitcup are able to yield a wider range of overall and "none" category performance. This increased range of performance across different confidence thresholds is revealed by the fact that both Soft E-Mixup and Soft E-Stitchup achieve slightly higher "none" category accuracies at a confidence threshold of 0.9 when compared to E-Mixup and E-Stitchup. However, these slightly higher "none" category accuracies are accompanied by a large decrease in overall accuracy.

Interestingly, the use of label softening has a noticeable and significant impact on model performance, resulting in different behavior for models trained with Soft E-Mixup and Soft E-Stitchup when compared to models trained with the other augmentation methods. This difference in performance results in the unique shape of the accuracy curves for Soft E-Mixup and Soft E-Stitchup, as seen in Fig. 3. The accuracy curves of Soft E-Mixup and Soft E-Stitchup have a very similar appearance and have a different shape when compared to accuracy curves produced by the other augmentation methods, thus highlighting the visible impact of label softening on the downstream classification model's performance.

5.3 Confidence Regularization

Label softening has the effect of decreasing the confidence of predictions (i.e., the magnitude of outputted probabilities from the model) made by a deep learning model. When comparing the magnitude of outputted probabilities produced by a network trained using Soft E-Mixup or Soft E-Stitchup to those of models trained without label softening (i.e., E-Mixup, E-Stitchup, and the control experiment), the prediction probabilities are of a significantly lower magnitude. Furthermore, prediction probabilities produced by models trained with E-Mixup and E-Stitchup were also found to be of lesser magnitude in comparison to models trained in the control experiment. High prediction magnitudes, which are a characteristic of models trained with one-hot labels, can be observed in the behavior of models trained for the control experiment. For example, the ROC curve for the control experiment in Fig. 4 has a sharp edge at the upper left corner of the curve, which does not occur any of the other ROC curves. This sharp corner is present in the control experiment's ROC curve because almost all of the model's predictions have extremely high confidence (i.e., confidence of 1.0). Such high confidence in all of the model's predictions causes the measures taken for the ROC curve to change drastically at a single threshold while remaining almost constant when other measures are taken, thus creating the sharp corner that is seen in Fig. 4. This sharp corner is not present in the other ROC curves because their predictions are of lower confidence and better reflect the true accuracy of the model, thus eliminating the sudden change in ROC metrics at a single threshold that is seen in the control.

From the above observations, it becomes evident that soft labels regularize the confidence of the network, leading to lower-probability predictions in comparison to networks trained without label softening. This confidence regularization effect occurs because the model, when soft labels are used, is no longer forced to make predictions with perfect confidence. Although a prediction confidence that is slightly below 1.0 is sufficient in most cases, the model will typically be trained to make perfect predictions, which allows marginal errors (e.g., a prediction with a confidence of 0.95 instead of 1.0) to dominate the gradient during training and leads the model to overfit at the cost of perfectly matching one-hot targets. Because the target probability of each class is decreased when label softening is used, the model is no longer forced to perfectly replicate one-hot targets, thus avoiding this issue. Such confidence regularization improves a network's ability to identify data in the "none" category and prevents the model from making confident, incorrect predictions on data that it does not understand well, as is shown by the improved "none" category performance of models trained with embedding augmentation and softened labels. Although all of the proposed augmentation methods create soft labels during training (i.e., all forms of embedding augmentation eliminate one-hot labels), Soft E-Mixup and Soft E-Stitchup use added softening to increase the amount of confidence regularization that occurs during training.

To more closely examine the proposed methods' performance and their ability to regulate the confidence of a model, accuracy metrics were recorded throughout training for models using each type of embedding augmentation, as well as without any augmentation. The results for the control experiment in comparison to E-Mixup are illustrated in Fig. 5. At the beginning of training, the "none" category accuracy for both methods is quite high because most predictions are near 0, thus causing the majority of products to be predicted in the "none" category. Over time, the model begins to make correct predictions with higher confidence, thus leading to slightly lower accuracy in the "none" category as the overall accuracy increases. During training, the network must find a balance between making correct, confident predictions and avoiding overfitting such that the "none" category accuracy is not decreased more than necessary. If the "none" category accuracy continues to decrease with marginal gains in overall accuracy, then the model is becoming overconfident, causing some products that belong in the "none" category to be incorrectly predicted with high confidence with minimal gains in overall accuracy.

When E-Mixup augmentation is utilized during training, it can be seen in Fig. 5 that the "none" category accuracy decreases for

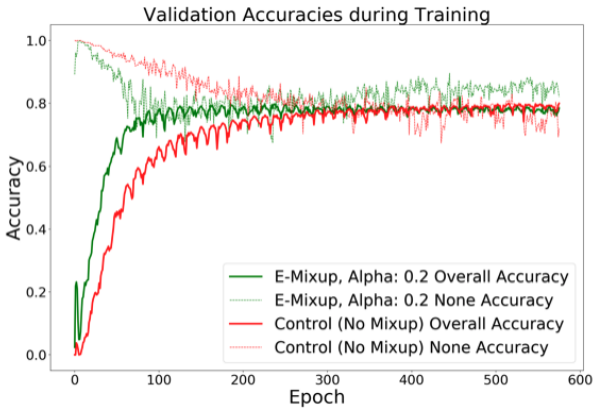

Figure 5: Overall and "none" category accuracies throughout training of the downstream classification model for the Control and with E-Mixup augmentation. The "none" category accuracy continues to decrease in the Control but plateaus after decreasing initially with E-Mixup, thus illustrating that embedding augmentation regulates model overconfidence.

several epochs but quickly plateaus and stops decreasing. Additionally, as training continues, the "none" category accuracy increases, converging at a higher value than the overall accuracy. After this point, both overall accuracy and "none" category accuracy remain stable for the rest of training. However, without any embedding augmentation, the "none" category accuracy continues to decrease throughout training, even after the overall validation accuracy of the model converges. This trend reveals that, without augmenting the embedding inputs, the model can become overconfident during training, causing it to become less effective at identifying products in the "none" category and, in turn, make incorrect predictions for such products. In contrast, the "none" category and overall accuracy of the model trained with E-Mixup remain stable as the "none" category accuracy of the control experiment decreases. This observation illustrates that such augmentation methods regularize the model's tendency to become overconfident during training, resulting in a stable convergence both overall and for the "none" category.

Such an ability to regularize the overconfidence of a model is one of the main benefits of the proposed augmentation methods, as it leads to higher performance in identifying the "none" category and, in many cases, increased overall performance (see Table 1). As a result, such methods are quite useful for deep learning models that are deployed into production, as they allow a model to dynamically identify data that belongs to an unseen class. As a result, an informed decision can then be made regarding data that should be labeled before the model is retrained (i.e., data within the "none" category should be labeled because the model does not know how to classify it). Additionally, as seen in Fig. 5, models trained with the proposed augmentation methods converge to their maximum overall accuracy much quicker than those trained without embedding augmentation. Therefore, the proposed augmentation methods can be used to easily recommend new data that should be labeled and

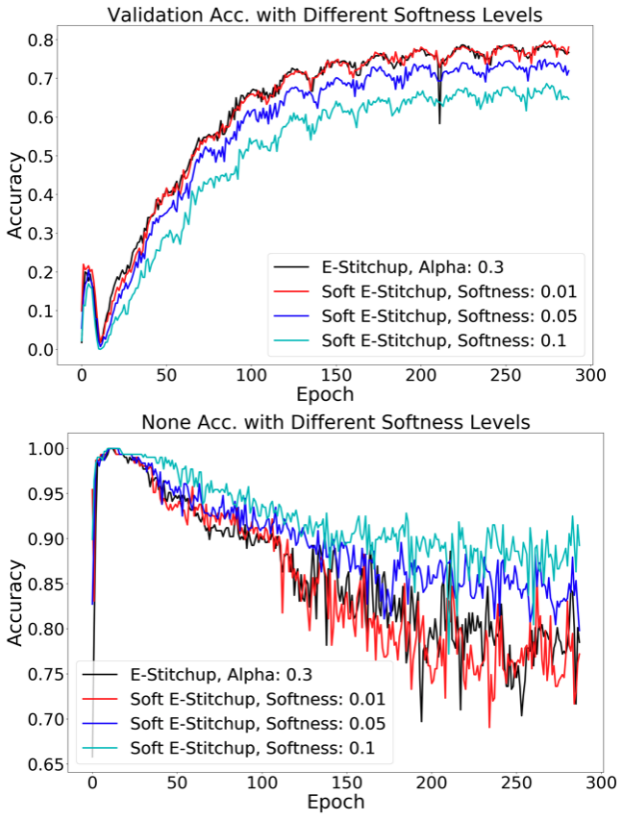

Figure 6: Overall Accuracy (top) and "none" accuracy (bottom) during training for both Soft E-Stitchup trained with different levels of label softness and E-Stitchup without any extra label softening. The alpha value for E-Stitchup was kept constant at 0.3 for each of these experiments.

included in the training set and to retrain the downstream model quickly and at minimal computational cost, completely avoiding any expensive fine-tuning of pre-trained embedding models.

5.4 Manipulating Label Softness

In addition to the experiments with the default label softness of 0.05 (i.e., Soft E-Mixup and Soft E-Stitchup), several experiments were performed to determine if altering the amount of label softness has any impact on models trained with Soft E-Mixup and Soft E-Stitchup. As seen in Fig. 6, increasing the amount of label softness follows a predictable pattern of decreasing overall accuracy, but increasing "none" category accuracy. Although only Soft E-Stitchup is displayed in Fig. 6, an almost identical pattern was observed when running the same test using Soft E-Mixup. Additionally, as the amount of softness is decreased, the performance of the model begins to resemble that of the model trained without label softening, as is seen in the similar performance of the models trained with E-Stitchup and Soft E-Stitchup with softness of 0.01.

Despite the patterns that are observed while changing the amount of label softness, there was no significant difference between the "none" category and overall AUC scores of models trained with

different levels of softness, thus revealing that the models can yield identical performance if the confidence threshold is set accordingly. All models trained with added label softness of 0.05 or greater performed almost identically in terms of overall and "none" category AUC scores, while models trained with less added softness began to yield AUC scores similar to E-Stitchup or E-Mixup (i.e., the methods that do not use added label softening). Because of these observations, the amount of label softness for Soft E-Mixup and Soft E-Stitchup was set to 0.05 by default, as models trained with this amount of softness yields identical performance even when the amount of label softening is increased.

6 DISCUSSION AND FUTURE WORK

The proposed augmentation methods are shown to be effective in boosting validation performance of the deep classification model by improving both the validation accuracy of the model as well as its ability to identify data belonging to unseen classes(i.e., the "none" category). Furthermore, these methods result in faster convergence for downstream models, can be used for any type of embedding representation of data, and do not require any fine-tuning of larger transfer learning models. We believe these proposed methods form a foundation for improving deep transfer learning with embeddings, as such methods are compatible with all types of data and can increase model performance with minimal added complexity.

There are various avenues that can be explored to further expand upon this work. First, it would be useful to run a wider range of tests with embedding augmentations using many different types of embeddings methods, such as XLNet [21] or FastText [8] embeddings for text or VGG [15] for producing image embeddings. Such a study would allow the effectiveness of these augmentation methods to be observed across many different methods of producing embeddings, thus leading to a better understanding of when such augmentation methods are most effective. Second, E-Mixup could be compared to the performance of Mixup, as it was originally proposed, on raw image inputs. In this study, one could retrain a Residual Network using normal Mixup augmentation, train a separate classification model using embedding augmentation, and compare the results from each of the two methods. If embedding augmentation is shown to produce comparable results to fine-tuning the Residual network directly with Mixup, is would be proven that embedding augmentations can create such performance improvements at significantly lower computational cost. Finally, it would be useful to further explore the impact of label softening in other areas of deep learning. For example, soft labeling could be combined with the original Mixup data augmentation to see if it improves model performance or creates different behavior for the resulting model.

7 CONCLUSIONS

In this work, four new types of data augmentation are presented; E-Mixup, E-Stitchup, Soft E-Mixup, and Soft E-Stitchup; that are generally applicable to embedding representations of data. Because these embedding augmentation methods are generally applicable to embedding representations, they can be applied to embeddings produced by many different types of data, including both textual and image data. Each of these augmentation methods is evaluated both in terms of generalization performance and ability to identify data that belongs to an unseen class, referred to as the "none" category accuracy. The proposed methods are shown to increase validation performance and "none" category accuracy when compared to a model that does not use any embedding augmentation during training. In comparison to models trained without embedding augmentation, the AUC scores of resulting models improved from a score of 0.9582 to 0.9812 when Soft E-Stitchup is included in training and "none" category accuracy improves by over 25 percent in certain cases by using any form of embedding augmentation. Additionally, embedding augmentation is shown to lead to faster convergence in downstream models and introduces minimal extra cost into the training process, requiring only a few lines of code and a constant factor of added complexity in the processing of each mini-batch. These proposed methods hold promise in the area of transfer learning, as they allow unseen classes of data to be dynamically identified, thus highlighting data that needs to be labeled and included in the training set for retraining, and improves downstream model performance without modifying any pre-trained networks. Therefore, these methods provide a viable alternative for improving model performance when resources are not available to incur the cost of fine-tuning pre-trained embedding models.

8 ACKNOWLEDGMENTS

The authors would like to acknowledge the University of Texas at Austin; especially Dr. Cem C. Tutum and Dr. Risto Miikkulainen; for supervision and useful discussion provided during this project.

REFERENCES

[1] 2019. Fashion Product Images Dataset. https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset

[2] Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. SciBERT: Pretrained Contextualized Embeddings for Scientific Text. *CoRR* abs/1903.10676 (2019). arXiv:1903.10676 http://arxiv.org/abs/1903.10676

[3] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. 2018. AutoAugment: Learning Augmentation Policies from Data. *CoRR* abs/1805.09501 (2018). arXiv:1805.09501 http://arxiv.org/abs/1805.09501

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[6] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data Augmentation for Low-Resource Neural Machine Translation. *arXiv e-prints*, Article arXiv:1705.00440 (May 2017), arXiv:1705.00440 pages. arXiv:cs.CL/1705.00440

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv e-prints*, Article arXiv:1512.03385 (Dec 2015), arXiv:1512.03385 pages. arXiv:cs.CV/1512.03385

[8] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *CoRR* abs/1607.01759 (2016). arXiv:1607.01759 http://arxiv.org/abs/1607.01759

[9] Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 36–45. https://doi.org/10.3115/v1/D14-1005

[10] Sosuke Kobayashi. 2018. Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. *arXiv e-prints*, Article arXiv:1805.06201 (May 2018), arXiv:1805.06201 pages. arXiv:cs.CL/1805.06201

[11] Daojun Liang, Feng Yang, Tian Zhang, and Peter Yang. 2018. Understanding Mixup Training Methods. *IEEE Access* PP (10 2018), 1–1. https://doi.org/10.1109/ACCESS.2018.2872698

[12] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When Does Label Smoothing Help? *CoRR* abs/1906.02629 (2019). arXiv:1906.02629 http://arxiv.org/abs/1906.02629

[13] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), 5149–5152.

[15] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, Article arXiv:1409.1556 (Sep 2014), arXiv:1409.1556 pages. arXiv:cs.CV/1409.1556

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). arXiv:1512.00567 http://arxiv.org/abs/1512.00567

[17] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR* abs/1905.11946 (2019). arXiv:1905.11946 http://arxiv.org/abs/1905.11946

[18] Luke Taylor and Geoff Nitschke. 2017. Improving Deep Learning using Generic Data Augmentation. *arXiv e-prints*, Article arXiv:1708.06020 (Aug 2017), arXiv:1708.06020 pages. arXiv:cs.LG/1708.06020

[19] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks. *arXiv e-prints*, Article arXiv:1905.11001 (May 2019), arXiv:1905.11001 pages. arXiv:stat.ML/1905.11001

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[21] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv e-prints*, Article arXiv:1906.08237 (Jun 2019), arXiv:1906.08237 pages. arXiv:cs.CL/1906.08237

[22] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2017. mixup: Beyond Empirical Risk Minimization. *CoRR* abs/1710.09412 (2017). arXiv:1710.09412 http://arxiv.org/abs/1710.09412