REPRODUCING AND EXTENDING UNSUPERVISED DATA AUGMENTATION FOR CONSISTENCY TRAINING

Brandon Ko & Erica Eaton
{bk36, eatone3}@cs.washington.edu

ABSTRACT

Deep learning often requires massive amounts of labeled data, but when the amount of labeled data is limited, semi-supervised learning can be used to effectively train a model on a combination of labeled and unlabeled data. Xie et al. (2020) introduce Unsupervised Data Augmentation, a new semi-supervised learning technique for adding noise to unlabeled samples with the goal of making the model more robust to noise in the input. We implement UDA, attempt to reproduce the experiments conducted by Xie et al. (2020), and improve upon the augmentation methods used in UDA. Overall, we observe improvement in both image and text classification tasks when employing UDA.

1 INTRODUCTION

As deep learning rises in popularity, the demand for labeled data increases tremendously. However, labeling data by hand is an expensive task that requires countless hours of labor. As a result, semi-supervised learning (SSL) is frequently used alongside deep learning to train a model on a mixture of labeled and unlabeled data. Recent SSL work has focused on consistency training, which involves injecting noise into input examples (data augmentation) to make the model more robust to slight perturbations of the input. Xie et al. (2020) introduce a novel idea in the SSL domain: substituting traditional noise injection methods with high quality data augmentation methods to improve consistency training. This method is called Unsupervised Data Augmentation (UDA). In this paper, we implement UDA, conduct similar experiments to those done by Xie et al. (2020), and extend the augmentation methods used in UDA.[1]

2 UDA IMPLEMENTATION

2.1 LOSS

The goal of UDA is to minimize the sum of the loss on labeled and unlabeled data (Xie et al., 2020). The loss on labeled data is the cross entropy between the predicted label probabilities output by the model and the actual labels for the data (Xie et al., 2020). The loss on unlabeled data (consistency loss) is the cross entropy between the predicted label probabilities for the original sample and the augmented samples (Xie et al., 2020). However, in the implementation referenced by Xie et al. (2020), the consistency loss is computed as the KL-divergence between the predicted label probabilities for the original sample and augmented samples, rather than cross entropy. In our implementation of consistency loss, we compute KL-divergence for image classification and cross entropy for text classification. The consistency loss is also weighted by a factor λ and Xie et al. (2020) use $\lambda = 1$. In our implementation, we use $\lambda = 1$ for image classification, but found $\lambda = 0.5$ yielded better results for text classification. Figure 1 depicts how the loss is computed, where the final loss is the sum of the supervised loss (loss on labeled data) and the unsupervised consistency loss (loss on unlabeled data) (Xie et al., 2020). Formally, the training objective is to minimize the following loss function (Xie et al., 2020):

$$\min_{\theta} \mathcal{J}(\theta) = \mathbb{E}_{x_1 \sim p_L(x)} \left[-\log p_\theta \left(f^*(x_1) \mid x_1 \right) \right] + \lambda \mathbb{E}_{x_2 \sim p_U(x)} \mathbb{E}_{\hat{x} \sim q(\hat{x} \mid x_2)} \left[\mathrm{CE} \left(p_{\tilde{\theta}}(y \mid x_2) \;||\; p_\theta(y \mid \hat{x}) \right) \right]$$

[1] Our code is available at https://github.com/brandonko/CSE599C_UDA

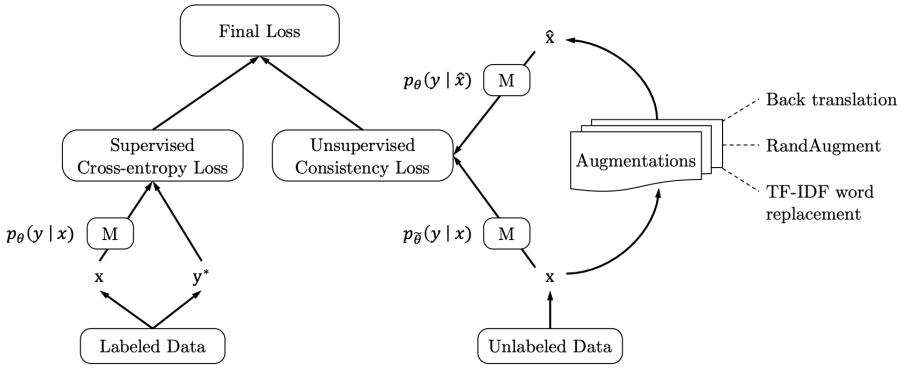

Figure 1: UDA Loss Computation (Xie et al., 2020)

2.2 AUGMENTATIONS

UDA utilizes 3 augmentation techniques: RandAugment for the computer vision domain, and TF-IDF replacement and back translation for the natural language processing (NLP) domain (Xie et al., 2020). These augmentation methods are described in more detail below.

2.2.1 RANDAUGMENT

Given an image, RandAugment randomly selects 1 of 15 transformations from the Python Image Library and applies that transformation with a probability of 0.5 to produce an augmented image (Xie et al., 2020). These transformations include invert, crop, sharpen, auto-contrast, posterize, shear in the x-direction, shear in the y-direction, translate in the x-direction, translate in the y-direction, rotate, equalize, contrast, color, solarize, and brighten (Xie et al., 2020). For transformations that require a transform magnitude, such as the number of degrees to rotate the image, Xie et al. (2020) select a random value in the range $[1, 10)$. Xie et al. (2020) generate 100 augmented images for each original, unlabeled image. We implement RandAugment using the 15 aforementioned transformations from the Python Image Library.

2.2.2 TF-IDF REPLACEMENT

Given a sentence from a corpus, TF-IDF replacement will replace unimportant words with low TF-IDF scores while keeping important words with high TF-IDF scores (Xie et al., 2020). This is done to generate variations of the same sentence that maintain the same overall meaning. To achieve this, each word is tagged with a replacement probability, denoting the chances the word will be replaced (Xie et al., 2020). If a word is to be replaced, the replacement word will be sampled from the entire corpus, where each word has a certain probability of being selected (Xie et al., 2020).

In our implementation, the sentence is broken down into a list of sub-word tokens using the BERT tokenizer from HuggingFace. First, we define the TF-IDF score of a word as the term frequency for the token in the sentence multiplied by the inverse document frequency for the token across the entire corpus: $\text{TFIDF}(w) = \text{TF}(w)\text{IDF}(w)$ (Xie et al., 2020). Next, we define C as the maximum TF-IDF score in a sentence: $C = \max_i \text{TFIDF}(x_i)$ (Xie et al., 2020). The replacement probability is then defined as $P_{\text{replace}}(x) = \min(p(C - \text{TFIDF}(x)/Z), 1)$, where p is a hyperparameter that controls the magnitude of the augmentation (set to 0.7) and Z is the average score, defined as $Z = \sum_i (C - \text{TFIDF}(x_i))/|x|$ (Xie et al., 2020). If a word is to be replaced, the replacement word is sampled from the entire corpus. We define a scoring function for each word as the following: $S(w) = \text{freq}(w)\text{IDF}(w)$ (Xie et al., 2020). With this scoring function, the selection probability for each word can be defined as $P_{\text{select}}(w) = (\max_{w'} S(w') - S(w))/Z$ where Z is a normalization term defined as $Z' = \sum_w \max_{w'} S(w') - S(w)$ (Xie et al., 2020).

2.2.3 BACK TRANSLATION

Back translation translates an existing example in some language A into another language B and then translates it back into language A (Xie et al., 2020). This can generate diverse paraphrases of the original example while maintaining the semantics. Back translation is implemented in our experiments using the PyPi BackTranslation module (Wu, 2021). Given two Google Translate API URLs, the BackTranslation module performs back translation using the two specified languages. For our implementation, we tried 5 different languages (Japanese, Korean, Chinese, French, Swedish), but the back translated sentence was nearly identical across all languages so only Korean was used for all experiments.

2.3 ENHANCEMENTS

To combat common problems encountered during training, Xie et al. (2020) propose two techniques: confidence-based masking and sharpening predictions. With confidence-based masking, any unlabeled samples whose highest predicted label probability is below a threshold, β, will not be included in the loss computation (Xie et al., 2020). As a result, the loss will consist of the loss on all labeled samples in that batch plus the loss on any unlabeled samples whose predictions the model is reasonably confident about. The second improvement to training is sharpening predictions, which regularizes the prediction probabilities for each unlabeled sample by dividing those prediction probabilities by a temperature, τ, and taking the softmax over the result for each unlabeled sample individually (Xie et al., 2020). Formally, for each unlabeled sample x, possible label y, and prediction probability z_y for label y, the sharpened prediction probabilities for that sample are (Xie et al., 2020):

$$p_{\tilde{\theta}}^{(sharp)}(y|x) = \frac{\exp\left(z_y/\tau\right)}{\sum_{y'} \exp\left(z_{y'}/\tau\right)}$$

For the temperature, Xie et al. (2020) use $\tau = 0.4$, which we use in our experiments as well. Combining confidence-based masking and sharpening predictions, the loss for unlabeled samples, where x is an unlabeled sample and $\hat{x}$ is an augmented version of that sample, becomes (Xie et al., 2020):

$$\frac{1}{|B|} \sum_{x \in B} I\left(\max_{y'} p_{\tilde{\theta}}\left(y' \mid x\right) > \beta\right) \mathrm{CE}\left(p_{\tilde{\theta}}^{(sharp)}\left(y \mid x\right) \,||\, p_\theta\left(y \mid \hat{x}\right)\right)$$

To implement these training improvements, for each unlabeled sample, we compute the softmax of the predicted label probabilities and if the maximum predicted probability is less than β, the sample is excluded from the loss computation. Otherwise, the predicted label probabilities are divided by the temperature τ, softmax is applied to the result, and the loss for that sample is computed.

3 EXPERIMENT SETUP

3.1 NLP

The dataset used for NLP experiments is the Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019). This dataset contains 9067 sentences (8551 for training, 516 for testing) from 23 linguistics publications that were labeled as grammatically acceptable or not by their original authors. Since all sentences are less than 64 words long, the encoding dimension (dimension of the sentence tensors) was set to 64. 2000 samples were selected as labeled examples, while the remaining 6551 samples were set as unlabeled (their labels were removed).

We use a pre-trained BERT model (Devlin et al., 2019) called BertForSequenceClassifiaction, provided by HuggingFace. The sequence classification variation of BERT contains a linear layer at the end for classification tasks. The model is fine-tuned using UDA on the CoLA dataset and classification accuracy is reported for the test dataset. For the augmentations, both TF-IDF replacement and back translation are experimented with.

3.2 COMPUTER VISION

For the computer vision experiments, Xie et al. (2020) perform image classification using the CIFAR-10 (Krizhevsky et al.) and SVHN (Netzer et al., 2011) datasets. CIFAR-10 contains $50,000$

labeled images for training, $10,000$ images for testing, and each image belongs to 1 of 10 classes (Krizhevsky et al.). SVHN contains $73,257$ images for training, $26,032$ images for testing, and each image belongs to 1 of 10 classes (Netzer et al., 2011). For both datasets, Xie et al. (2020) employ a Wide-ResNet-28-2 model to classify the images. This model is a version of ResNet with 28 layers and double the width. We use the Wide-ResNet-28-2 implementation from Kuen (2019). Since PyTorch does not have a pretrained version of Wide-ResNet-28-2, we pretrain the model ourselves for both CIFAR and SVHN. For classifying CIFAR images, we pretrain on an augmented version of CIFAR, where we apply the following transformations from PyTorch's torchvision library: random crop, random horizontal flip, random affine transformation with a magnitude of 3, random rotation of 3 degrees, and color jitter. For classifying SVHN images, we pretrain on the original SVHN dataset. We also tried applying augmentations to the SVHN dataset during pretraining, similar to those applied for CIFAR, but observed higher accuracy and lower loss on the test data when not augmenting SVHN. In addition, we use at most 10 augmented images for CIFAR and SVHN, compared to the 100 augmented images Xie et al. (2020) create, due to GPU limitations in Google Colaboratory.

4 RESULTS

4.1 NLP

4.1.1 NO ENHANCEMENTS

Xie et al. (2020) did not mention using confidence-based masking and sharpening for NLP-related tasks, so we conduct a round of experiments without these enhancements, setting the temperature for sharpening predictions to one and the threshold for confidence-based masking to zero (Xie et al., 2020). The single augmentation with TF-IDF replacement experiment achieved a test accuracy of 62.4%, which was 1.1% higher than the 5 augmentation variation with a test accuracy of 61.3%. The single augmentation with back translation scored the highest accuracy of 63.3%, and mixing back translation with TF-IDF replacement (1 back translation + 4 TF-IDF replacement) resulted in a test accuracy of 62.3%, which was between the best back translation and TF-IDF replacement accuracies. Multiple back translations were not used because repeated back translations of the same sentence yielded the same augmented sample. Results are displayed in Table 1.

Different values for the unlabeled loss weight hyperparameter, λ, were tried for all experiments and $\lambda = 0.1$ resulted in the best performance across both types of augmentations. From the results, it seems that the model does not perform well when it relies too heavily on learning from the unsupervised loss, as lower values of λ outperform higher values, showing how the unsupervised loss prevents the model from learning optimally.

Test Accuracy	λ	Epochs	Num Agumentations	Augmentation Type
62.4	0.1	4	1	TF-IDF
61.3	0.1	8	5	TF-IDF
63.3	0.1	4	1	Back Translation
62.3	0.1	4	4 + 1	TF-IDF + Back Translation

Table 1: NLP Experiment Results With No Enhancements

4.1.2 WITH ENHANCEMENTS

Confidence-based masking and sharpening were used in the next set of experiments. Setting the threshold for confidence-based masking to 0.8 and the temperature for sharpening predictions to 0.4, the testing accuracy significantly increased for both augmentations. The testing accuracy for single augmentation with TF-IDF replacement increased by 14.5% to 76.9% accuracy, and the 5 augmentation variation increased by 18.4% to 79.7% accuracy. The single augmentation for back translation increased by 14.8%, to 78.1% accuracy while the mix of back translation and TF-IDF replacement increased by 15.4% to 77.7%. Results are displayed in Table 2.

On average, all four variations increased by 15.4% accuracy. This shows how it is better to rely heavily on the labeled samples when training. Since Xie et al. (2020) did not mention tuning the β and τ hyperparameters for any NLP datasets, different values were experimented with for each

augmentation. The highest accuracy (79.7%) was achieved with TF-IDF replacement ($\beta = 0.9$, $\tau = 0.4$, $\lambda = 0.5$ with 5 augmentations). The accuracy did not increase when we switched out one of the TF-IDF replacement augmentations with a back translation augmentation. Adding more diverse samples in the augmentation set seemed like a good idea, but it only detracted the model performance. However, when using the enhancement techniques, multiple TF-IDF replacements outperformed back translation.

Test Accuracy	λ	β	τ	Epochs	Num Agumentations	Augmentation Type
76.9	0.5	0.9	0.4	4	1	TF-IDF
79.7	0.5	0.9	0.4	15	5	TF-IDF
78.1	0.5	0.8	0.4	4	1	Back Translation
77.7	0.5	0.8	0.4	8	4 + 1	TF-IDF + Back Translation

Table 2: NLP Experiment Results With Enhancements

4.2 Computer Vision

Following the experiments done by Xie et al. (2020), the threshold for confidence-based masking is set to 0.8, the temperature for sharpening predictions is set to 0.4, and the same number of labeled versus unlabeled training images for each dataset are used ($4,000$ labeled CIFAR images and $1,000$ labeled SVHN images). Test accuracy and loss for the pretrained models without using UDA are shown in Table 3 and results when incorporating UDA are in Table 4. When applying UDA, Xie et al. (2020) achieve approximately 96% and 97% test accuracy for CIFAR and SVHN, respectively, but we achieve 10% and 20% test accuracy, respectively. During training, we observed that upon fine-tuning the pretrained models for CIFAR and SVHN by incorporating UDA, test accuracy immediately decreased to 10% and 20%, respectively, and remained constant for subsequent epochs, while test loss increased. Tuning the hyperparameters for both models also produced no change in the test accuracy.

The main differences between our implementation and that of Xie et al. (2020), which could account for the discrepancy between our results and those in the paper, are pretraining and the number of augmented images. Although Xie et al. (2020) do not explicitly discuss pretraining the Wide-ResNet-28-2 model, they do state this model is from Zagoruyko & Komodakis (2016), which pretrains the model on ImageNet. PyTorch does not offer a pretrained Wide-ResNet-28-2 model, but does offer a Wide-ResNet-50-2 model pretrained on ImageNet. However, we were unable to use the pretrained Wide-ResNet-50-2 or pretrain on ImageNet ourselves due to resource limitations. Thus, we chose to pretrain on CIFAR and SVHN, separately. We also ran the same experiments, applying UDA with no pretraining, but results were the same or worse than with pretraining. After pretraining, the maximum prediction probability for unlabeled CIFAR images was typically between 0.45 and 0.65, falling below the threshold of 0.8 specified by Xie et al. (2020). Thus, nearly all unlabeled CIFAR images were ignored when fine-tuning with UDA, likely causing the model to overfit on the small labeled dataset, yielding only 10% accuracy, which is the approximate number of labeled CIFAR images used for training out of the total number of CIFAR images. In contrast, the maximum prediction probability for unlabeled SVHN images was typically between 0.8 and 0.9. However, the test accuracy when using only $1,000$ labeled images was 20%. It is possible that using only 10 augmented images instead of 100, which is the other main difference between our implementation and that of Xie et al. (2020), could also be contributing to the low accuracy, as we are using far fewer augmented images ($460,000$ compared to $4,600,000$ for CIFAR and $722,570$ compared to $7,225,700$ for SVHN). Further, since RandAugment chooses from 15 transformations and we use only 10 augmented images, the augmented images could be more biased towards certain transformations, causing the model to learn the transformations, however extreme they may be, with respect to that particular image, yielding a gradient step in the wrong direction and deterioration over time.

Dataset	Test Images Classified Correctly	Test Accuracy	Test Loss
CIFAR	8,097 / 10,000	81%	1.65
SVHN	24,516 / 26,032	94%	1.52

Table 3: Test Accuracy and Loss for Pretrained Wide-ResNet Models

Dataset	Num Augmented Images	Test Images Classified Correctly	Test Accuracy	Test Loss
CIFAR	5	1,000 / 10,000	10%	2.36
	10	1,000 / 10,000	10%	2.36
SVHN	5	5,099 / 26,032	20%	2.27
	10	5,099 / 26,032	20%	2.27

Table 4: Results for Image Classification with UDA

5 EXTENSIONS

5.1 NLP

For this section, all listed experiments are conducted with the unlabeled loss weight set to 0.5, the threshold for confidence-based masking set to 0.9, and the temperature for sharpening predictions set to 0.4.

5.1.1 WORD-LEVEL TF-IDF REPLACEMENT

Xie et al. (2020) specify the use of TF-IDF replacement with the BERT tokenizer which tokenizes words into sub-words. Using these tokens, replacements are conducted, which means a sub-word can be replaced with a full word. The idea of replacing part of a word seemed strange, so for this extension, the TF-IDF replacement algorithm was adjusted so that tokenization occurs at the word level and not the sub-word level. The NLTK RegExp tokenizer was used to tokenize sentences at the word level.

As shown in Table 5, the word-level TF-IDF replacement with one augmentation outperformed its sub-word variation by only 0.4% with an accuracy of 77.3%, but the word-level TF-IDF replacement with 5 augmentations underperformed its sub-word variation by 0.6% with an accuracy of 79.1%. Overall, it did not seem like using word-level TF-IDF replacement provides significantly better results. Even though we build our understanding of a sentence word-by-word, the model can build its own understanding of a sentence through sub-words. From this experiment, we learned that changing the way the model interprets the sentence so that it aligns with our interpretation does not necessarily mean the model will perform better.

Test Accuracy	Epochs	Num Agumentations	Augmentation Type
77.3	4	1	$\text{TFIDF}_{\text{WordLevel}}$
79.1	12	5	$\text{TFIDF}_{\text{WordLevel}}$

Table 5: Word-Level TF-IDF Replacement

5.1.2 BACK TRANSLATION INTO TF-IDF REPLACEMENT

Xie et al. (2020) touch on the idea of using both augmentations together to generate more diverse samples, but never mention using both augmentations on the same sample. By creating a back translation of a sample and then building multiple augmentations of that sample using TF-IDF replacement, variations of the paraphrase of the original sample can be generated, potentially helping the model learn patterns that are quite different from those found in the original corpus. In our implementation, new words (generated from back translation) that are encountered during the TF-IDF replacement phase are skipped over and never replaced.

There are three variations of this experiment: a single augmentation of TF-IDF replacement on the back translated sentence, multiple augmentations of TF-IDF replacement on the back translated sen-

tence, and a mix of TF-IDF replacement on the back translated sentence and TF-IDF replacement on the original sentence. These three variations are run using sub-word level and word level TF-IDF replacement. On average, the word level TF-IDF replacement experiments were 1.2% more accurate than the sub-word level TF-IDF replacement experiments. The word level TF-IDF replacement experiments (with three augmentations of back translation into TF-IDF replacement and three augmentations of just TF-IDF replacement) also outperformed the highest scoring experiment with enhancements (five augmentations of TF-IDF replacement) by 0.3% with a test accuracy of 80.0%. Results are displayed in Table 6.

Test Accuracy	Epochs	Num Agumentations	Augmentation Type
76.7	4	1	$\text{BT} \rightarrow \text{TFIDF}$
77.8	12	5	$\text{BT} \rightarrow \text{TFIDF}$
79.3	20	3 + 3	$(\text{BT} \rightarrow \text{TFIDF}) + (\text{TFIDF})$
78.2	4	1	$\text{BT} \rightarrow \text{TFIDF}_{\text{WordLevel}}$
78.9	12	5	$\text{BT} \rightarrow \text{TFIDF}_{\text{WordLevel}}$
80.0	15	3 + 3	$(\text{BT} \rightarrow \text{TFIDF}_{\text{WordLevel}}) + (\text{TFIDF}_{\text{WordLevel}})$

Table 6: Back Translation into TF-IDF Replacement

5.2 Computer Vision

5.2.1 Number of Labeled Images and Confidence Threshold

Since the maximum prediction probability for unlabeled CIFAR images typically fell in the range 0.45 to 0.65, below the original confidence threshold of 0.8, we reduce the confidence threshold for CIFAR to 0.6, in order to include more unlabeled images in the consistency loss computation. In addition, since Xie et al. (2020) see at least a slight reduction in error rate as they increase the number of labeled images in their UDA experiments, we increase the number of labeled CIFAR and SVHN images as well. We also try treating all images as both labeled and unlabeled, computing the supervised cross entropy loss and unsupervised consistency loss on all images. Results when reducing the confidence threshold for CIFAR to 0.6 and increasing the number of labeled images, as well as results when not pretraining, are shown in Table 7 for CIFAR and Table 8 for SVHN.

Num Labeled	Pretrained?	Num Augmented	Test Imgs. Correct	Test Accy.	Test Loss
25,000	No	5	1,000/10,000	10%	2.36
25,000	Yes	5	1,000/10,000	10%	2.36
40,000	No	5	1,000/10,000	10%	2.36
40,000	Yes	5	1,000/10,000	10%	2.36
50,000	No	5	7,287/10,000	73%	1.75
50,000	Yes	5	8,649/10,000	86%	1.61
50,000	Yes	10	8,577/10,000	86%	1.62

Table 7: Results for CIFAR with UDA and Varying the Number of Labeled Images

Num Labeled	Pretrained?	Num Augmented	Test Imgs. Correct	Test Accy.	Test Loss
50,000	No	5	24,555 / 26,032	94%	1.52
50,000	No	10	24,550 / 26,032	94%	1.52
50,000	Yes	5	24,776 / 26,032	95%	1.51
50,000	Yes	10	24,799 / 26,032	95%	1.51

Table 8: Results for SVHN with UDA and Varying the Number of Labeled Images

For CIFAR and SVHN, treating all images as both labeled and unlabeled and reducing the confidence threshold for CIFAR to 0.6 resulted in a substantial improvement in test accuracy compared to using only $4,000$ labeled CIFAR images and $1,000$ labeled SVHN images. As shown in Tables 7 and 8, pretraining the models yields better results than no pretraining. When using the pretrained

model, incorporating UDA, and treating all images as both labeled and unlabeled, test accuracy is approximately 5% higher for CIFAR and 1% higher for SVHN than the initial test accuracy after pretraining. Treating all images as both labeled and unlabeled likely helps counteract the bias introduced, as previously described, when using at most 10 augmented images per unlabeled image because both the supervised loss and unsupervised consistency loss are computed for each image.

5.2.2 MODIFYING RANDAUGMENT

For RandAugment, Xie et al. (2020) randomly select the transformation magnitude, if applicable, from the range $[1, 10)$, but this range is not appropriate for every transformation. For example, a brightness value above 5 creates a nearly all white image, making the image unrecognizable and hence impossible to learn the correct label. Thus, we create a custom magnitude range for each transformation that requires a magnitude. To determine each magnitude range, we experiment on actual images to ensure the image is recognizable. Results when using this new version of RandAugment with pretraining and 5 augmented images are shown in Table 9.

Dataset	Num Labeled Images	Test Images Classified Correctly	Test Accuracy	Test Loss
CIFAR	4,000	1,000 / 10,000	10%	2.36
	50,000	8,674 / 10,000	87%	1.63
SVHN	1,000	5,099 / 26,032	20%	2.27
	73,257	24,855 / 26,032	95%	1.51

Table 9: Results for Image Classification with Modified Version of RandAugment

Although our modification to RandAugment did not improve the results when using only $4,000$ labeled CIFAR images and $1,000$ labeled SVHN images, it did improve the results when treating all images as both labeled and unlabeled. For CIFAR, the test accuracy increased by 1% and the number of correctly classified test images increased by 97, compared to the best result with the original version of RandAugment. For SVHN, the number of correctly classified test images increased by 56, compared to the best result with the original version of RandAugment.

6 CRITICAL REVIEW

6.1 RESOURCE LIMITATIONS

Due to Google Colaboratory resource limitations, we ran into multiple issues that made it more difficult to reproduce the work of Xie et al. (2020). For the NLP experiments, we were limited to an encoding dimension of 64, which is far too small to encode the datasets used in the paper (the Yelp review dataset contains many reviews with more than 500 words). As a result, we were forced to use datasets with short sentences. For the computer vision experiments, we were unable to pretrain on ImageNet ourselves or use a larger pretrained Wide-ResNet model. We were also limited to using at most 10 augmented images for each unlabeled image, compared to 100 images used by Xie et al. (2020).

6.2 AMBIGUITIES

When reproducing Xie et al. (2020), we encountered two main ambiguities. The first is the loss formula, as Xie et al. did not specify how expectation should be calculated (i.e. whether to simply compute the mean). Also, Xie et al. (2020) state that the consistency loss is the cross entropy between the predicted label probabilities for unlabeled samples and augmented samples, but in their implementation, KL-divergence is used rather than cross entropy. The second ambiguity is whether the Wide-ResNet model used for image classification should be pretrained, and if so, what dataset the model should be pretrained on. Given the improvement in results we observed for image classification, particularly on CIFAR, when pretraining the model, one could argue that it is important to discuss what pretraining is done, if any.

6.3 COMPARISON TO ORIGINAL VERSION

Compared to the original version of the paper (Xie et al., 2019), the current version (Xie et al., 2020) presents UDA in a more clear and structured manner and makes changes to the loss function as well as the augmentation method used for image classification.

The explanation of augmentation techniques in the original paper is quite vague. Back translation is not clearly defined and there is no explanation of the mechanics behind TF-IDF replacement. The current version introduces an appendix for explaining TF-IDF replacement along with text examples of back translation and visual examples of RandAugment (Xie et al., 2020). Also, the experiment section in the current version covers a variety of different models and unsupervised learning method combinations, while the original version only uses a Wide-ResNet-28-2 (Xie et al., 2019).

In the original version of the paper, the equations for supervised loss and unsupervised consistency loss are presented separately, but the complete UDA objective, which is to minimize the sum of these losses, is not presented (Xie et al., 2019). However, in the current version of the paper, the full objective is provided (Xie et al., 2020), which is clearer and easier to understand than presenting pieces of the objective separately as in the original version (Xie et al., 2019). In addition, the current version of the paper computes the consistency loss using cross entropy (Xie et al., 2020), while the original version (Xie et al., 2019) and Xie et al.'s implementation (referenced in Xie et al. (2020)) use KL-divergence. It is not clear why Xie et al. changed consistency loss to be computed with cross entropy instead of KL-divergence in the current version of the paper.

Another difference between the original and current versions of the paper is that the original version uses AutoAugment (Xie et al., 2019) and the current version uses RandAugment (Xie et al., 2020), which is a variation of AutoAugment. Using RandAugment is faster and more efficient than AutoAugment because it only requires selecting a random transformation rather than searching for a transformation and does not require the data to be labeled (Xie et al., 2020).

7 CONCLUSION

Deep learning is a rapidly growing field that requires massive amounts of labeled data for training models. Due to this demand, many SSL techniques have been introduced to take advantage of unlabeled data. Xie et al. (2020) introduce Unsupervised Data Augmentation, which shows promising results for the value of data augmentation in deep learning models. We implemented and experimented with UDA on NLP and computer vision tasks, and provided extensions to the augmentation techniques employed in UDA.

For the NLP experiments, UDA helps slightly with the model's learning, but relying too much on the unsupervised loss can yield poor results. Given our resource limitations, it is difficult to tell if UDA is able to provide as much value as stated by Xie et al. (2020). However, the extensions we implemented showed a slight improvement in comparison to the original UDA techniques. Unfortunately, the CoLA dataset is not the best for showcasing UDA's potential, since the sentences are very short and the classification task is binary. There are other domains that need more diverse augmentations to better understand real-world data, and we believe that UDA can thrive in these situations.

For the computer vision experiments, UDA does result in an improvement in accuracy and loss over the pretrained model, particularly with our modification to RandAugment, but only when treating all images as both labeled and unlabeled. Since we were limited to using only 10 augmented images for each unlabeled image, it is possible that using 100 augmented images instead, as in Xie et al. (2020), would yield further improvement in the results and potentially even when using fewer labeled images as well, proving the value of UDA, since the augmented images would be far less likely to be skewed towards a subset of the 15 transformations in RandAugment.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Jason Kuen. Wideresnet-pytorch. https://github.com/xternalz/WideResNet-pytorch, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments, 2019.

Zhiqiang Wu. Backtranslation. https://pypi.org/project/BackTranslation/, 2021.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. 2019.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL http://arxiv.org/abs/1605.07146.