

# Unsupervised Data Augmentation for Consistency Training

Qizhe Xie<sup>1,2</sup>, Zihang Dai<sup>1,2</sup>, Eduard Hovy<sup>2</sup>, Minh-Thang Luong<sup>1</sup>, Quoc V. Le<sup>1</sup>

<sup>1</sup> Google Brain, <sup>2</sup> Carnegie Mellon University

{qizhex, dzihang, hovy}@cs.cmu.edu, {thangluong, qvl}@google.com

## Abstract

Despite much success, deep learning generally does not perform well with small labeled training sets. In these scenarios, data augmentation has shown much promise in alleviating the need for more labeled data, but it so far has mostly been applied in supervised settings and achieved limited gains. In this work, we propose to apply data augmentation to unlabeled data in a semi-supervised learning setting. Our method, named Unsupervised Data Augmentation or UDA, encourages the model predictions to be consistent between an unlabeled example and an augmented unlabeled example. Unlike previous methods that use random noise such as Gaussian noise or dropout noise, UDA has a small twist in that it makes use of harder and more realistic noise generated by state-of-the-art data augmentation methods. This small twist leads to substantial improvements on six language tasks and three vision tasks even when the labeled set is extremely small. For example, on the IMDb text classification dataset, with only 20 labeled examples, UDA achieves an error rate of 4.20, outperforming the state-of-the-art model trained on 25,000 labeled examples. On standard semi-supervised learning benchmarks CIFAR-10 and SVHN, UDA outperforms all previous approaches and achieves an error rate of 2.7% on CIFAR-10 with only 4,000 examples and an error rate of 2.85% on SVHN with only 250 examples, nearly matching the performance of models trained on the full sets which are one or two orders of magnitude larger. UDA also works well on large-scale datasets such as ImageNet. When trained with 10% of the labeled set, UDA improves the top-1/top-5 accuracy from 55.1/77.3% to 68.7/88.5%. For the full ImageNet with 1.3M extra unlabeled data, UDA further pushes the performance from 78.3/94.4% to 79.0/94.5%<sup>1</sup>.

## 1 Introduction

Deep learning typically requires a lot of labeled data to succeed. Labeling data, however, is a costly process for each new task of interest. Making use of unlabeled data to improve deep learning has been an important research direction to address this costly process. On this direction, semi-supervised learning [4] is one of the most promising methods and recent works can be grouped into three categories: (1) graph-based label propagation via graph convolution [31] and graph embeddings [62], (2) modeling prediction target as latent variables [30], and (3) consistency / smoothness enforcing [2, 35, 44, 7, 58]. Among them, methods of the last category, i.e., based on smoothness enforcing, have been shown to work well on many tasks.

In a nutshell, the smoothness enforcing methods simply regularize the model's prediction to be less sensitive to small perturbations applied to examples (labeled or unlabeled). Given an observed example, smoothness enforcing methods first create a perturbed version of it (e.g., typically by adding

<sup>1</sup>Code is available at <https://github.com/google-research/uda>.

artificial noise such as Gaussian noise or dropout), and enforce the model predictions on the two examples to be similar. Intuitively, a good model should be invariant to any small perturbations that do not change the nature of an example. Under this generic framework, methods in this category differ mostly in the perturbation function, i.e., how the perturbed example is created.

In our paper, we propose to use state-of-the-art data augmentation methods found in supervised learning as the perturbation function in the smoothness enforcing framework, extending prior works by Sajjadi et al. [52], Laine and Aila [35]. We show that better augmentation methods lead to greater improvements and that they can be used on many other domains. Our method, named Unsupervised Data Augmentation or UDA, minimizes the KL divergence between model predictions on the original example and an example generated by data augmentation. Although data augmentation has been studied extensively and has led to significant improvements, it has mostly been applied in supervised learning settings [57, 34, 9, 66]. UDA, on the other hand, can directly apply state-of-the-art data augmentation methods on unsupervised data which is available at larger quantities and therefore has the potential to work much better than standard supervised data augmentation.

We evaluate UDA on a wide variety of language and vision tasks. On six text classification tasks, our method achieves significant improvements over state-of-the-art models. Notably, on IMDB, UDA with 20 labeled examples outperforms the state-of-the-art model trained on 1250x more labeled data. We also evaluate UDA on standard semi-supervised learning benchmarks CIFAR-10 and SVHN. UDA outperforms all existing semi-supervised learning methods by significant margins. On CIFAR-10 with 4,000 labeled examples, UDA achieves an error rate of 5.27, nearly matching the performance of the fully supervised model that uses 50,000 labeled examples. Furthermore, with a more advanced architecture, PyramidNet+ShakeDrop, UDA achieves a new state-of-the-art error rate of 2.7. On SVHN, UDA achieves an error rate of 2.85 with only 250 labeled examples, nearly matching the performance of the fully supervised model trained with 73,257 labeled examples. Finally, we also find UDA to be beneficial when there is a large amount of supervised data. Specifically, on ImageNet, UDA leads to improvements of top-1 accuracy and top-5 accuracy from 58.69/80.20% to 68.66/88.52% with 10% of the labeled set and from 78.28%/94.36% to 79.04%/94.45% when we use the full labeled set and an external dataset with 1.3M unlabeled examples.

Our contributions, which will be presented in the rest of the paper, are as follows:

- First, we propose a training technique called TSA that effectively prevents overfitting when much more unlabeled data is available than labeled data.
- Second, we show that targeted data augmentation methods (such as AutoAugment [9]) give a significant improvements over other untargeted augmentations.
- Third, we combine a set of data augmentations for NLPs, and show that our method works well and complements representation learning methods, such as BERT [13].
- Fourth, our paper show significant leaps in performance compared to previous methods in a range of vision and language tasks.
- Finally, we develop a method so that UDA can be applied even the class distributions of labeled and unlabeled data mismatch.

## 2 Unsupervised Data Augmentation (UDA)

In this section, we first formulate our task and then present the proposed method, UDA. Throughout this paper, we focus on classification problems and will use  $x$  to denote the input and  $y(x)$  or simply  $y^*$  to denote its ground-truth prediction target. We are interested in learning a model  $p_\theta(y | x)$  to predict  $y^*$  based on the input  $x$ , where  $\theta$  denotes the model parameters. Finally, we will use  $L$  and  $U$  to denote the sets of labeled and unlabeled examples respectively.

### 2.1 Background: Supervised Data Augmentation

Data augmentation aims at creating novel and realistic-looking training data by applying a transformation to an example, without changing its label. Formally, let  $q(\hat{x} | x)$  be the augmentation transformation from which one can draw augmented examples  $\hat{x}$  based on an original example  $x$ . For an augmentation transformation to be valid, it is required that any example  $\hat{x} \sim q(\hat{x} | x)$  drawn from the distribution shares the same ground-truth label as  $x$ , i.e.,  $y(\hat{x}) = y(x)$ . Given a valid

augmentation transformation, we can simply minimize the negative log-likelihood on augmented examples.

Supervised data augmentation can be equivalently seen as constructing an augmented labeled set from the original supervised set and then training the model on the augmented set. Therefore, the augmented set needs to provide additional inductive biases to be more effective. How to design the augmentation transformation has, thus, become critical.

In recent years, there have been significant advancements on the design of data augmentations for NLP [66], vision [34, 9] and speech [18, 47] in supervised settings. Despite the promising results, data augmentation is mostly regarded as the “cherry on the cake” which provides a steady but limited performance boost because these augmentations have so far only been applied to a set of labeled examples which is usually of a small size. Motivated by this limitation, we develop UDA to apply effective data augmentations to unlabeled data, which is often in larger quantities.

## 2.2 Unsupervised Data Augmentation

As discussed in the introduction, a recent line of work in semi-supervised learning has been utilizing unlabeled examples to enforce smoothness of the model. The general form of these works can be summarized as follows:

- Given an input  $x$ , compute the output distribution  $p_\theta(y | x)$  given  $x$  and a perturbed version  $p_\theta(y | x, \epsilon)$  by injecting a small noise  $\epsilon$ . The noise can be applied to  $x$  or hidden states or be used to change the computation process.
- Minimize a divergence metric between the two predicted distributions  $\mathcal{D}(p_\theta(y | x) || p_\theta(y | x, \epsilon))$ .

This procedure enforces the model to be insensitive to the perturbation  $\epsilon$  and hence smoother with respect to changes in the input (or hidden) space.

In this work, we present a simple twist to the existing smoothness / consistency enforcing works and extend prior works on using data augmentation as perturbations [52, 35]. We propose to use state-of-the-art data augmentation targeted at different tasks as a particular form of perturbation and optimize the same smoothness or consistency enforcing objective on unlabeled examples. Specifically, following VAT [44], we choose to minimize the KL divergence between the predicted distributions on an unlabeled example and an augmented unlabeled example:

$$\min_{\theta} \mathcal{J}_{\text{UDA}}(\theta) = \mathbb{E}_{x \in U} \mathbb{E}_{\hat{x} \sim q(\hat{x}|x)} [\mathcal{D}_{\text{KL}}(p_{\tilde{\theta}}(y | x) || p_{\theta}(y | \hat{x}))], \quad (1)$$

where  $q(\hat{x} | x)$  is a data augmentation transformation and  $\tilde{\theta}$  is a *fixed* copy of the current parameters  $\theta$  indicating that the gradient is not propagated through  $\tilde{\theta}$ , as suggested by Miyato et al. [44]. The data augmentation transformation used here is the same as the augmentations used in the supervised data augmentation such as back translation for texts and random cropping for images. Since it is costly to run back translation on-the-fly during training, we generate augmented examples offline. Multiple augmented examples are generated for each unlabeled example.

To use both labeled examples and unlabeled examples, we add the cross entropy loss on labeled examples and the consistency / smoothness objective defined in Equation 1 with a weighting factor  $\lambda$  as our training objective, which is illustrated in Figure 1. Formally, the objective is defined as follows:

$$\min_{\theta} \mathcal{J} = \mathbb{E}_{x, y^* \in L} [p_{\theta}(y^* | x)] + \lambda \mathcal{J}_{\text{UDA}}(\theta), \quad (2)$$

By minimizing the consistency loss, UDA allows for label information to propagate from labeled examples to unlabeled ones. We set  $\lambda$  to 1 for most of our experiments and use different batch sizes for the supervised objective and the unsupervised objective. We found that it leads to better performances to use a larger batch size for the unsupervised objective on some datasets.

When compared to conventional perturbations such as Gaussian noise, dropout noise and simple augmentations such as affine transformations, we believe that data augmentations targeted at each task can serve as a more effective source of “noise”. Specifically, using targeted data augmentation as the perturbation function has several advantages:

- **Valid perturbations:** Data augmentation methods that achieve great performance in supervised learning have the advantage of generating realistic augmented examples that share the same ground-

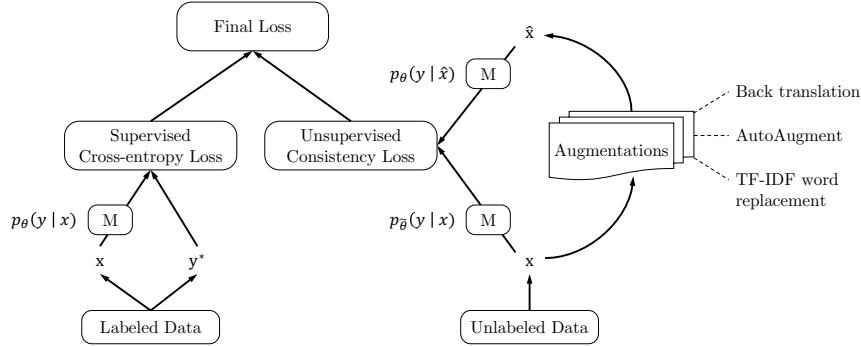


Figure 1: Training objective for UDA, where  $M$  is a model that predicts a distribution of  $y$  given  $x$ .

truth labels with the original example. Hence, it is safe to encourage the smoothness or consistency between predictions on the original unlabeled example and the augmented unlabeled examples.

- **Diverse perturbations:** Data augmentation can generate a diverse set of examples since it can make large modifications to the input example without changing its label, while the perturbations such as Gaussian or Bernoulli noise only make local changes. Encouraging smoothness on a diverse set of augmented examples can significantly improve the sample efficiency.
- **Targeted inductive biases:** Different tasks require different inductive biases. As shown in AutoAugment [9], data augmentation policy can be directly optimized towards improving validation performance on each task. Such performance-oriented augmentation policy can learn to figure out the missing or most wanted inductive biases in an original labeled set. We find that the augmentation policies found by AutoAugment work well in our semi-supervised learning setting although AutoAugment optimizes model’s performance in a supervised learning setting.

As we will show in the ablation study, diverse and valid augmentations that inject targeted inductive biases are key components that lead to significant performance improvements.

### 2.3 Augmentation Strategies for Different Tasks

As discussed in Section 2.2, data augmentation can be tailored to provide missing inductive biases specific to each task. In this section, we discuss three different augmentations used for different tasks and discuss the trade-off between diversity and validity for data augmentations. We leverage recent advancements on data augmentation and apply the following augmentation strategies:

**AutoAugment for Image Classification.** For image classification, AutoAugment [9] uses reinforcement learning to search for an “optimal” combination of image augmentation operations directly based on the validation performances, outperforming any manually designed augmentation procedure by a clear margin. We use the augmentation policies found, and opensourced by AutoAugment for experiments on CIFAR-10, SVHN and ImageNet.<sup>2</sup> We also use Cutout [14] for CIFAR-10 and SVHN since it can be composed with AutoAugment to achieve improved performance.

**Back translation for Text Classification.** Back translation [55, 15] can generate diverse paraphrases while preserving the semantics of the original sentences and has been shown to lead to significant performance improvements for QANet in question answering [66]. Hence, we employ a back translation system to paraphrase the training data, for sentiment classification datasets including IMDb, Yelp-2, Yelp-5, Amazon-2 and Amazon-5. We find that the diversity of the paraphrases is more important than the quality or the validity. Hence we employ random sampling with a tunable

<sup>2</sup><https://github.com/tensorflow/models/tree/master/research/autoaugment>

temperature instead of beam search for the generation. More specifically, we train English-to-French and French-to-English translation models using the WMT 14 corpus and perform back translation to each sentence instead of the whole paragraph, since the parallel data in WMT 14 is for sentence-level translation while the input examples in sentiment classification corpora are paragraphs. As shown in Figure 2, the paraphrases generated by back translation sentence are very diverse and have similar semantic meanings.

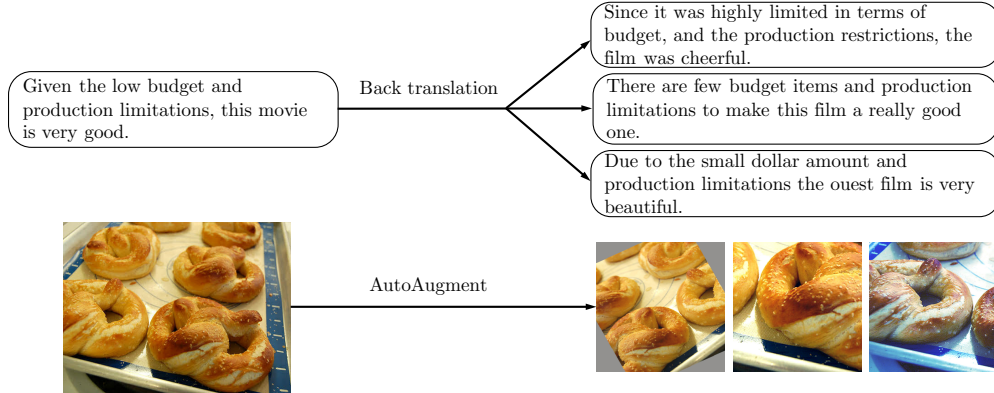


Figure 2: Augmented examples using back translation and AutoAugment

**TF-IDF based word replacing for Text Classification.** While back translation is good at maintaining a global semantics of the original sentence, there is no guarantee that it will keep certain words. However, on DBPedia where the task is to predict the category of a Wikipedia page, some keywords are more informative than other words in determining the category. Therefore, we propose an augmentation method called TF-IDF based word replacing that replaces uninformative words that usually have low TF-IDF scores while keeping keywords which have high TF-IDF scores. We refer readers to Appendix B for a detailed description.

## 2.4 Trade-off Between Diversity and Validity for Data Augmentation

Despite that state-of-the-art data augmentation methods can generate diverse and valid augmented examples as discussed in section 2.2, there is a trade-off between diversity and validity since diversity is achieved by changing a part of the original example, naturally leading to the risk of altering the ground-truth label. We find it beneficial to tune the trade-off between diversity and validity for data augmentation methods.

For image classification, AutoAugment automatically finds the sweet spot between diversity and validity since it is optimized according to the validation set performances in the supervised setting. For text classification, we tune the temperature of random sampling. On the one hand, when we use a temperature of 0, decoding by random sampling degenerates into greedy decoding and generates perfectly valid but identical paraphrases. On the other hand, when we use a temperature of 1, random sampling generates very diverse but barely readable paraphrases. We find that setting the Softmax temperature to 0.7, 0.8 or 0.9 leads to the best performances.

## 3 Additional Training Techniques

In this section, we introduce additional techniques for applying UDA in different scenarios. First, to allow the model to be trained on more unlabeled data without overfitting, we introduce a technique called Training Signal Annealing in Section 3.1. Then, to make the training signal stronger when the predictions are over-flat, we present three intuitive methods to sharpen the predictions in Section 3.2. Lastly, to apply UDA on out-of-domain unlabeled data, we introduce a simple method called Domain-relevance Data Filtering in Section 3.3.

### 3.1 Training Signal Annealing

Since it is much easier to obtain unlabeled data than labeled data, in practice, we often encounter a situation where there is a large gap between the amount of unlabeled data and that of labeled data. To enable UDA to take advantage of as much unlabeled data as possible, we usually need a large enough model, but a large model can easily overfit the supervised data of a limited size. To tackle this difficulty, we introduce a new training technique called Training Signal Annealing (TSA).

The main intuition behind TSA is to gradually release the training signals of the labeled examples without overfitting them as the model is trained on more and more unlabeled examples. Specifically, for each training step  $t$ , we set a threshold  $\frac{1}{K} \leq \eta_t \leq 1$ , with  $K$  being the number of categories. When the probability of the correct category  $p_\theta(y^* | x)$  of a labeled example is higher than the threshold  $\eta_t$ , we remove this example from the loss function and only train on other labeled examples in the minibatch. Formally, given a minibatch of labeled examples  $B$ , we replace the supervised objective with the following objective:

$$\min_{\theta} \frac{1}{Z} \sum_{x, y^* \in B} [-I(p_\theta(y^* | x) < \eta_t) \log p_\theta(y^* | x)],$$

where  $I(\cdot)$  is the indicator function and  $Z = \sum_{x, y^* \in B} I(p_\theta(y^* | x) < \eta_t)$  is simply a re-normalization factor. Effectively, the threshold  $\eta_t$  serves as a ceiling to prevent the model from over-training on examples that the model is already confident about. When we gradually anneal  $\eta_t$  from  $\frac{1}{K}$  to 1 during training, the model can only slowly receive supervisions from the labeled examples, largely alleviating the overfitting problem. Suppose  $T$  is the total number of training steps and  $t$  is the current training step. To account for different ratios of unlabeled data and labeled data, we consider three particular schedules of  $\eta_t$ , as shown in Figure 3.

- **log-schedule** — The threshold  $\eta_t$  is increased most rapidly at the beginning of the training:  $\eta_t = (1 - \exp(-\frac{t}{T} * 5)) * (1 - \frac{1}{K}) + \frac{1}{K}$ ;
- **linear-schedule** — The threshold  $\eta_t$  is increased linearly as training progresses:  $\eta_t = \frac{t}{T} * (1 - \frac{1}{K}) + \frac{1}{K}$ ;
- **exp-schedule** — The threshold  $\eta_t$  is increased most rapidly at the end of the training:  $\eta_t = \exp((\frac{t}{T} - 1) * 5) * (1 - \frac{1}{K}) + \frac{1}{K}$ .

Intuitively, when the model is prone to overfit, e.g., when the problem is relatively easy or the number of labeled examples is very limited, the exp-schedule is the most suitable one as the supervised signal is mostly released at the end of training. Following a similar logic, when the model is less likely to overfit (e.g., when we have abundant labeled examples or when the model employs effective regularizations), the log-schedule can serve well.

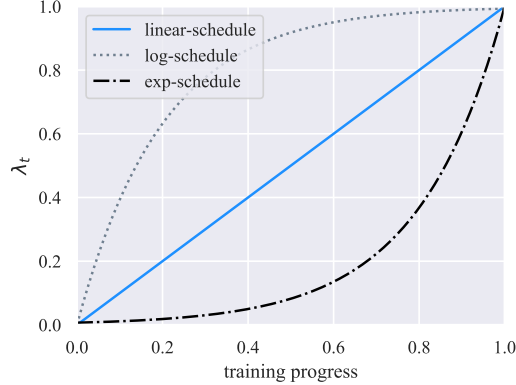


Figure 3: Three schedules of TSA. We set  $\eta_t = \frac{1}{K} + \lambda_t * (1 - \frac{1}{K})$ , so that  $\eta_t$  is increased from  $\frac{1}{K}$  to 1 when  $\lambda_t$  goes from 0 to 1.

### 3.2 Sharpening Predictions

We observe that the predicted distributions on unlabeled examples and augmented unlabeled examples tend to be over-flat across categories, in cases where the problem is hard and the number of labeled examples is very small. Consequently, the unsupervised training signal from the KL divergence is relatively weak and thus gets dominated by the supervised part. For example, on ImageNet, when we use 10% of the labeled set, the predicted distributions on unlabeled examples are much less sharp than the distributions on labeled examples. Therefore, we find it helpful to sharpen the predicted distribution produced on unlabeled examples and employ the following three intuitive techniques:

- **Confidence-based masking** — We find it to be helpful to mask out examples that the current model is not confident about. Specifically, in each minibatch, the consistency loss term is computed only on examples whose highest probability, i.e.,  $\max_y p_{\hat{\theta}(y|x)}$ , is greater than a threshold.
- **Entropy minimization** — Entropy minimization [17] regularizes the predicted distribution on augmented examples  $p_{\theta}(y | \hat{x})$  to have a low entropy. To employ this technique, we add an entropy term to the overall objective.
- **Softmax temperature controlling** — We tune the Softmax temperature when computing the predictions on original examples  $p_{\hat{\theta}(y|x)}$ . Specifically,  $p_{\hat{\theta}(y|x)}$  is computed as  $\text{Softmax}(l(x)/\tau)$  where  $l(x)$  denotes the logits and  $\tau$  is the temperature. A lower temperature corresponds to a sharper distribution.

In practice, we find combining confidence-based masking and softmax temperature controlling to be most effective for settings with very small amount of labeled data, while entropy minimization works well for cases with relatively larger amount of labeled data.

### 3.3 Domain-relevance Data Filtering

Ideally, we would like to make use of out-of-domain unlabeled data since it is usually much easier to collect, but the class distributions of out-of-domain data are usually mismatched with those of in-domain data. Due to the mismatched class distributions, using out-of-domain unlabeled data can hurt the performance than not using it [46]. To obtain data relevant to the domain for the task at hand, we adopt a common technique for detecting out-of-domain data. We use our baseline model trained on the in-domain data to infer the labels of data in a large out-of-domain dataset and pick out the examples (equally distributed among classes) that the model is most confident about. Specifically, for each category, we sort all out-of-domain examples based on the classified probabilities of being in that category and select the examples with the highest probabilities.

## 4 Experiments

We apply UDA to a variety of language and vision tasks. Specifically, we show experiments on six text classification tasks in Section 4.1. Then, in Section 4.2, we compare UDA with other semi-supervised learning methods on standard vision benchmarks, CIFAR-10 and SVHN. Lastly, we evaluate UDA on ImageNet in Section 4.3 and provide ablation studies for TSA and augmentation methods in Section 4.4. We only present the information necessary to compare the empirical results here and refer readers to Appendix D and the code for implementation details.

### 4.1 Text Classification Experiments

**Datasets.** We conduct experiments on six language datasets including IMDb, Yelp-2, Yelp-5, Amazon-2, Amazon-5 and DBPedia [40, 70], where DBPedia contains Wikipedia pages for category classifications and all other datasets are about sentiment classifications on different domains. In our semi-supervised setting, we set the number of supervised examples to 20 for binary sentiment classification tasks including IMDb, Yelp-2 and Amazon-2. For the five-way classification datasets Yelp-5 and Amazon-5, we use 2,500 examples (i.e., 500 examples per class). Finally, although DBPedia has 14 categories, the problem is relatively simple. Hence, we set the number of training examples per class to 10. For unlabeled data, we use the whole training set for DBPedia and the

concatenation of the training set and the unlabeled set for IMDb. We obtain large datasets of Yelp reviews and Amazon reviews [41] as the unlabeled data for Yelp-2 Yelp-5, Amazon-2 and Amazon-5.<sup>3</sup>

**Experiment settings.** We adopt the Transformer model [60] used in BERT [13] as our baseline model due to its great performances on many tasks. Then, we consider four initialization schemes. Specifically, we initialize our models either with (a) random Transformer, (b) BERT<sub>BASE</sub>, (c) BERT<sub>LARGE</sub> or (d) BERT<sub>FINETUNE</sub>: BERT<sub>LARGE</sub> fine-tuned on in-domain unlabeled data. The latter finetuning strategy is motivated by the fact that ELMo [49] and ULMFiT [26] show that finetuning language models on domain specific data can lead to performance improvements. We do not pursue further experiments for BERT<sub>FINETUNE</sub> on DBpedia since fine-tuning BERT on DBpedia does not result in better performance than BERT<sub>LARGE</sub> in our preliminary experiments. This is probably due to the fact that DBpedia is on the Wikipedia domain and BERT is already trained on the whole Wikipedia corpus. In all these four settings, we compare the performance with and without UDA.

**Main results.** The results for text classification are shown in Table 1 with three key observations.

- Firstly, UDA consistently improves the performance regardless of the model initialization scheme. Most notably, even when BERT is further finetuned on in-domain data, UDA can still significantly reduce the error rate from 6.50% to 4.20% on IMDb. This result shows that the benefits UDA provides are complementary to that of representation learning.
- Secondly, with a significantly smaller amount of supervised examples, UDA can offer decent or even competitive performances compared to the SOTA model trained with full supervised data. In particular, on binary sentiment classification tasks, with only 20 supervised examples, UDA outperforms the previous SOTA trained on full supervised data on IMDb and gets very close on Yelp-2 and Amazon-2.
- Finally, we also note that five-category sentiment classification tasks turn out to be much more difficult than their binary counterparts and there still exists a clear gap between UDA with 500 labeled examples per class and BERT trained on the entire supervised set. This suggests a room for further improvement in the future.

Fully supervised baseline							
Datasets (# Sup examples)		IMDb (25k)	Yelp-2 (560k)	Yelp-5 (650k)	Amazon-2 (3.6m)	Amazon-5 (3m)	DBpedia (560k)
Pre-BERT SOTA		4.32	2.16	29.98	3.32	34.81	0.70
BERT <sub>LARGE</sub>		4.51	1.89	29.32	2.63	34.17	0.64
Semi-supervised setting							
Initialization	UDA	IMDb (20)	Yelp-2 (20)	Yelp-5 (2.5k)	Amazon-2 (20)	Amazon-5 (2.5k)	DBpedia (140)
Random	✗	43.27	40.25	50.80	45.39	55.70	41.14
	✓	25.23	8.33	41.35	16.16	44.19	7.24
BERT <sub>BASE</sub>	✗	27.56	13.60	41.00	26.75	44.09	2.58
	✓	5.45	2.61	33.80	3.96	38.40	1.33
BERT <sub>LARGE</sub>	✗	11.72	10.55	38.90	15.54	42.30	1.68
	✓	4.78	2.50	33.54	3.93	37.80	1.09
BERT <sub>FINETUNE</sub>	✗	6.50	2.94	32.39	12.17	37.32	-
	✓	<b>4.20</b>	<b>2.05</b>	<b>32.08</b>	<b>3.50</b>	<b>37.12</b>	-

Table 1: Error rates on text classification datasets. In the fully supervised settings, the pre-BERT SOTAs include ULMFiT [26] for Yelp-2 and Yelp-5, DPCNN [29] for Amazon-2 and Amazon-5, Mixed VAT [51] for IMDb and DBpedia.

**Results with different labeled set sizes.** We also evaluate the performance of UDA with different numbers of supervised examples. As shown in Figure 4, UDA leads to consistent improvements for all labeled set sizes. In the large-data regime, with the full training set of IMDb, UDA also provides robust gains. On Yelp-2, with 2,000 examples, UDA outperforms the previous SOTA model trained with 560,000 examples.

<sup>3</sup><https://www.kaggle.com/yelp-dataset/yelp-dataset>, <http://jmcauley.ucsd.edu/data/amazon/>



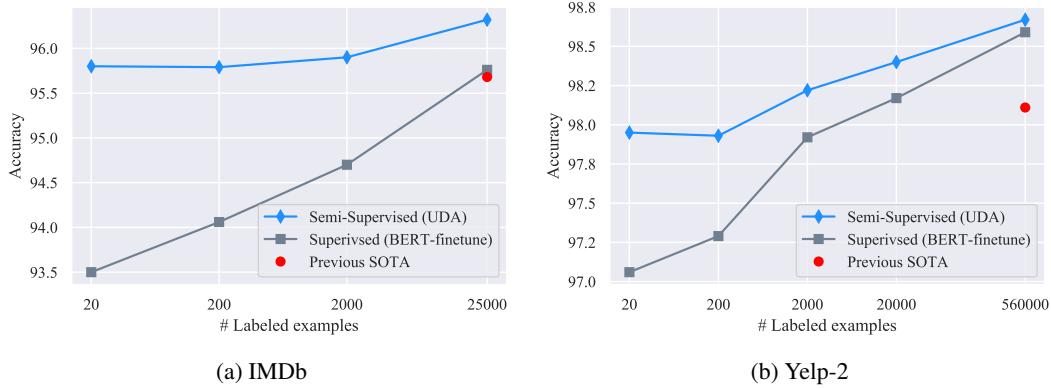


Figure 4: Accuracy on IMDb and Yelp-2 with different number of labeled examples.

## 4.2 Comparison with semi-supervised learning methods

**Experiment settings.** Following the standard semi-supervised learning setting, we compare UDA with prior works on CIFAR-10 [33] and SVHN [45]. We follow the settings in [46] and employ Wide-ResNet-28-2 [67, 21] as our baseline model. We compare UDA with Pseudo-Label [36], an algorithm based on self-training, Virtual adversarial training (VAT) [44], an algorithm that generates adversarial Gaussian perturbations on input, II-Model [35], which combines simple input augmentation with hidden state perturbations, Mean Teacher [58], which enforces smoothness on model parameters and MixMatch [3], a concurrent work that unifies several prior works on semi-supervised learning.

**Comparisons with existing semi-supervised learning methods.** In Figure 5, we compare UDA with existing works with varied sizes of labeled examples. UDA outperforms all existing methods with a clear margin, including MixMatch [3], a concurrent work on semi-supervised learning. For example, with 250 examples, UDA achieves an error rate of 8.41 on CIFAR-10 and 2.85 on SVHN while MixMatch has an error rate of 11.08 on CIFAR-10 and 3.78 on SVHN. More interestingly, UDA matches the performance of models trained on the full supervised data when AutoAugment is not employed. In particular, UDA achieves an error rate of 5.27 on CIFAR-10 with 4,000 labeled examples and an error rate of 2.85 on SVHN with 250 labeled examples matching the performance of our fully supervised model without AutoAugment, which achieves an error rate of 5.36 on CIFAR-10 with 50,000 labeled examples and an error rate of 2.84 on SVHN with 73,257 labeled examples.<sup>4</sup> Note that the difference of UDA and VAT is essentially the perturbation process. While the perturbations produced by VAT often contain high-frequency artifacts that do not exist in real images, data augmentation mostly generates diverse and realistic images. The performance difference between UDA and VAT shows the superiority of data augmentation based perturbation.

In Appendix C, we also compare UDA with recently proposed methods including ICT [61] and mixmixup [19] which enforce interpolation smoothness similar to mixup [69] and LGA + VAT [28], an algorithm based on gradient similarity. UDA outperforms all previous approaches and reduces more than 30% of the error rates of state-of-the-art methods.

**Results with more advanced architectures.** We also test whether UDA can benefit from more advanced architectures by using Shake-Shake (26 2x96d) [16] and PyramidNet+ShakeDrop [63] instead of Wide-ResNet-28-2. As shown in Table 2, UDA achieves an error rate of 2.7 when we employ PyramidNet+ShakeDrop, matching the performance of the fully supervised model without AutoAugment, and outperforming the best semi-supervised learning result with an error rate of 4.95 achieved by MixMatch.

<sup>4</sup>The fully supervised baseline performance of Wide-ResNet-28-2 in MixMatch is higher than ours, although the performance of our implementation matches the performance reported in prior works. We hypothesize that the difference is due to the fact that MixMatch employs Exponential Moving Average of model parameters.

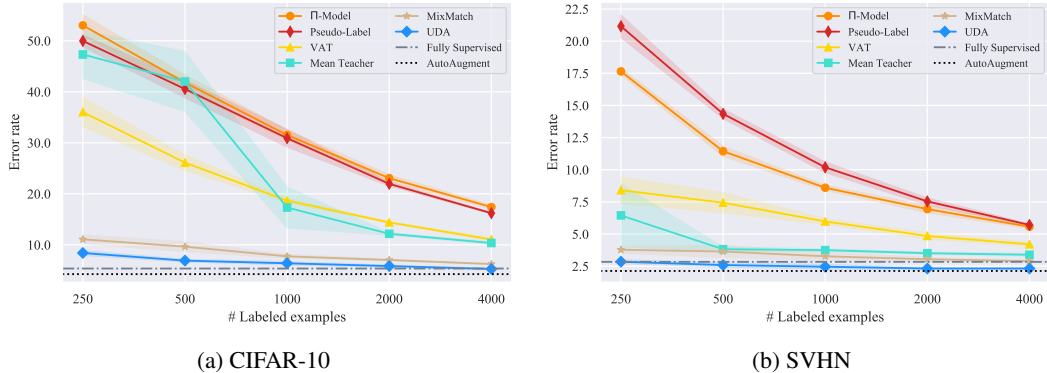


Figure 5: Comparison with semi-supervised learning methods on CIFAR-10 and SVHN with varied number of labeled examples. The performances of  $\Pi$ -Model, Pseudo-Label, VAT and Mean Teacher are reported in [3].

Methods	# Sup	Wide-ResNet-28-2	Shake-Shake	ShakeDrop
Supervised	50k	5.4	2.9	2.7
AutoAugment	50k	4.3	2.0	1.5
UDA	4k	5.3	3.6	2.7

Table 2: Error rates on CIFAR-10 with different models. # Sup denotes the number of supervised examples. ShakeDrop denotes PyramidNet+ShakeDrop. With only 4,000 labeled examples, UDA matches the performance of fully supervised models without AutoAugment for Wide-ResNet-28-2 and PyramidNet+ShakeDrop.

### 4.3 ImageNet experiments

In previous sections, all datasets we consider have a relatively small number of training examples and classes. In addition, we only use in-domain unlabeled data in previous experiments, where the class distribution of the unlabeled data always match with that of labeled data. In order to further test whether UDA can still excel on larger and more challenging datasets, we conduct experiments on ImageNet [12]. We also develop a method to apply UDA on out-of-domain unlabeled data, which leads to performance improvements when we use the whole ImageNet as the supervised data.

**Experiment settings.** To provide an informative evaluation, we conduct experiments on two settings with different numbers of supervised examples: (a) We use 10% of the supervised data of ImageNet while using all other data as unlabeled data, (b) Secondly, we consider the fully supervised scenario where we keep all images in ImageNet as supervised data and obtain extra unlabeled data from the JFT dataset [25, 6]. We use the domain-relevance data filtering method to filter out 1.3M images from the JFT dataset.

**Results.** As shown in Table 3, for the 10% supervised data setting, when compared to the supervised baseline, UDA improves the top-1 and top-5 accuracy from 55.09% to 68.66% and from 77.26% to 88.52% respectively. When compared with VAT + EntMin, a prior work on semi-supervised learning, UDA improves the top-5 accuracy from 83.39% to 88.52%. As for the full ImageNet setting shown in Table 4, when compared with AutoAugment, UDA improves the baseline top-1 accuracy from 78.28% to 79.04% and improves the top-5 accuracy from 94.36% to 94.45%, with only 1.3M more unlabeled data. We expect that there will be further improvements with more unlabeled data, which we leave as future works.

### 4.4 Ablation Studies

In this section, we provide an analysis of when and how to use TSA and effects of different augmentation methods for researchers and practitioners.

Methods	top-1 acc	top-5 acc
Supervised	55.09	77.26
Pseudo-Label [36] <sup>‡</sup>	-	82.41
VAT [44] <sup>‡</sup>	-	82.78
VAT + EntMin [44] <sup>‡</sup>	-	83.39
UDA	<b>68.66</b>	<b>88.52</b>

Table 3: Accuracy on ImageNet with 10% of the labeled set. We set the image size to 224. The results with <sup>‡</sup> is reported in [68].

Methods	top-1 / top-5 accuracy
Supervised	77.28 / 93.73
AutoAugment	78.28 / 94.36
UDA	<b>79.04 / 94.45</b>

Table 4: Accuracy on the full ImageNet with image size 331. We use the ImageNet dataset and another 1.3M unlabeled images from the JFT dataset as the unlabeled data.

**Ablations on Training Signal Annealing (TSA).** We study the effect of TSA on two tasks with different amounts of unlabeled data: (a) Yelp-5: on this text classification task, we have about 6M unlabeled examples while only having 2.5k supervised examples. We do not initialize the network by BERT in this study to rule out factors of having a pre-trained representation, (b) CIFAR-10: we have 50k unlabeled examples while having 4k labeled examples.

As shown in Table 5, on Yelp-5, where there is a lot more unlabeled data than supervised data, TSA reduces the error rate from 50.81% to 41.35% when compared to the baseline without TSA. More specifically, the best performance is achieved when we choose to postpone releasing the supervised training signal to the end of the training, i.e, exp-schedule leads to the best performance. On the other hand, linear-schedule is the sweet spot on CIFAR-10 in terms of the speed of releasing supervised training signals, where the amount of unlabeled data is not a lot larger than that of supervised data.

**Ablations on augmentation methods.** Targeted augmentation methods such as AutoAugment have been shown to lead to significant performance improvements in supervised learning. In this study, we would like to investigate whether targeted augmentations are effective when applied to unlabeled data and whether improvements of augmentations in supervised learning can lead to improvements in our semi-supervised learning setting.

Firstly, as shown in Table 6, if we apply the augmentation policy found on SVHN by AutoAugment to CIFAR-10 (denoted by Switched Augment), the error rate increases from 5.10 to 5.59, which demonstrates the effectiveness of targeted data augmentations. Further, if we remove AutoAugment and only use Cutout, the error rate increases to 6.42. Finally, the error rate increases to 16.17 if we only use simple cropping and flipping as the augmentation. On SVHN, the effects of different augmentations are similar. These results show the importance of applying augmentation methods targeted at each task to inject the most needed inductive biases.

We also observe that the effectiveness of augmentation methods on supervised learning settings transfers to our semi-supervised settings. Specifically, in the fully supervised learning settings, Cubuk et al. [9] also show that AutoAugment improves upon Cutout and that Cutout is more effectively than basic augmentations, which aligns well with the observations in semi-supervised settings. In our preliminary experiments for sentiment classifications, we have also found that, both in supervised learning settings and unsupervised learning settings, back-translation works better than simple word dropping or word replacing although word dropping or replacing can lead to improved performance on the purely supervised baseline.

TSA schedule	Yelp-5	CIFAR-10
$\times$	50.81	5.67
log-schedule	49.06	5.41
linear-schedule	45.41	<b>5.10</b>
exp-schedule	<b>41.35</b>	7.25

Table 5: Ablation study for Training Signal Annealing (TSA) on Yelp-5 and CIFAR-10. The shown numbers are error rates.

Augmentation	CIFAR-10	SVHN
Cropping & flipping	16.17	8.27
Cutout	6.42	3.09
Switched Augment	5.59	2.74
AutoAugment	<b>5.10</b>	<b>2.22</b>

Table 6: Ablation study for data augmentation methods. Switched Augment means to apply the policy found by AutoAugment on SVHN to CIFAR-10 and vice versa.

## 5 Related Work

Due to the space limits, we only discuss the most relevant works here and refer readers to Appendix A for the complete related work. Most related to our method is a line of work that enforces classifiers to be smooth with respect to perturbations applied to the input examples or hidden representations. As explained earlier, works in this family mostly differ in how the perturbation is defined: Pseudo-ensemble [2] directly applies Gaussian noise;  $\Pi$ -Model [35] combines simple input augmentation with hidden state noise; VAT [44, 43] defines the perturbation by approximating the direction of change in the input space that the model is most sensitive to; Cross-view training [7] masks out part of the input data; Sajjadi et al. [52] combines dropout and random max-pooling with affine transformation applied to the data as the perturbations. Apart from enforcing smoothness on the input examples and the hidden representations, another line of research enforces smoothness on the model parameter space. Works in this category include Mean Teacher [58], fast-Stochastic Weight Averaging [1] and Smooth Neighbors on Teacher Graphs [38].

## 6 Conclusion

In this paper, we show that data augmentation and semi-supervised learning are well connected: better data augmentation can lead to significantly better semi-supervised learning. Our method, UDA, employs highly targeted data augmentations to generate diverse and realistic perturbations and enforces the model to be smooth with respect to these perturbations. We also propose a technique called TSA that can effectively prevent UDA from overfitting the supervised data, when a lot more unlabeled data is available. For text, UDA combines well with representation learning, e.g., BERT, and is very effective in low-data regime where state-of-the-art performance is achieved on IMDB with only 20 examples. For vision, UDA reduces error rates by more than 30% in heavily-benchmarked semi-supervised learning setups. Lastly, UDA can effectively leverage out-of-domain unlabeled data and achieve improved performances on ImageNet where we have a large amount of supervised data.

## Acknowledgements

We want to thank Hieu Pham, Adams Wei Yu and Zhilin Yang for their tireless help to the authors on different stages of this project and thank Colin Raffel for pointing out the connections between our work and previous works. We also would like to thank Olga Wichrowska, Ekin Dogus Cubuk, Jiateng Xie, Guokun Lai, Yulun Du, Trieu Trinh, Ran Zhao, Ola Spyra, Brandon Yang, Daiyi Peng, Andrew Dai, Samy Bengio, Jeff Dean and the Google Brain team for insightful discussions and support to the work.

## References

- [1] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. 2018.
- [2] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.
- [3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.
- [4] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.
- [5] Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*, 2016.
- [6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.
- [7] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*, 2018.

- [8] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.
- [10] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.
- [11] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6510–6520, 2017.
- [12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
- [15] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.
- [16] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.
- [17] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.
- [18] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.
- [19] Ryuichiro Hataya and Hideki Nakayama. Unifying semi-supervised and robust learning by mixup. *ICLR The 2nd Learning from Limited Labeled Data (LLD) Workshop*, 2019.
- [20] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828, 2016.
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [22] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. Sequence to sequence mixture model for diverse machine translation. *arXiv preprint arXiv:1810.07391*, 2018.
- [23] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.
- [24] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.
- [25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [26] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339, 2018.
- [27] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1558–1567. JMLR.org, 2017.
- [28] Jacob Jackson and John Schulman. Semi-supervised learning by label gradient alignment. *arXiv preprint arXiv:1902.02336*, 2019.

- [29] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570, 2017.
- [30] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.
- [31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [32] Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. *arXiv preprint arXiv:1903.06059*, 2019.
- [33] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [35] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.
- [36] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.
- [37] Davis Liang, Zhiheng Huang, and Zachary C Lipton. Learning noise-invariant representations for robust speech recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 56–63. IEEE, 2018.
- [38] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8896–8905, 2018.
- [39] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.
- [40] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.
- [41] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.
- [42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [43] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.
- [44] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [45] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [46] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.
- [47] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

- [48] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [49] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.
- [50] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. URL [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language_understanding_paper.pdf), 2018.
- [51] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. Revisiting lstm networks for semi-supervised text classification via mixed objective function. 2018.
- [52] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.
- [53] Julian Salazar, Davis Liang, Zhiheng Huang, and Zachary C Lipton. Invariant representation learning for robust deep networks. In *Workshop on Integration of Deep Learning Theories, NeurIPS*, 2018.
- [54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.
- [55] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.
- [56] Tianxiao Shen, Myle Ott, Michael Auli, and Marc’Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. *arXiv preprint arXiv:1902.07816*, 2019.
- [57] Patrice Y Simard, Yann A LeCun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274. Springer, 1998.
- [58] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.
- [59] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.
- [60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [61] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.
- [62] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.
- [63] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedown regularization for deep residual learning. *arXiv preprint arXiv:1802.02375*, 2018.
- [64] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.
- [65] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*, 2017.
- [66] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.
- [67] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- [68] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S<sup>4</sup>l: Self-supervised semi-supervised learning. *arXiv preprint arXiv:1905.03670*, 2019.

- [69] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.
- [70] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.
- [71] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.



## A Complete Related Work

Due to the long history of semi-supervised learning (SSL), we refer readers to [4] for a general review. More recently, many efforts have been made to renovate classic ideas into deep neural instantiations. For example, graph-based label propagation [71] has been extended to neural methods via graph embeddings [62, 64] and later graph convolutions [31]. Similarly, with the variational auto-encoding framework and reinforce algorithm, classic graphical models based SSL methods with target variable being latent can also take advantage of deep architectures [30, 39, 65]. Besides the direct extensions, it was found that training neural classifiers to classify out-of-domain examples into an additional class [54] works very well in practice. Later, Dai et al. [11] shows that this can be seen as an instantiation of low-density separation.

Apart from enforcing smoothness on the input examples and the hidden representations, another line of research enforces smoothness of model parameters, which is complementary to our method. For example, Mean Teacher [58] maintains a teacher model with parameters being the ensemble of a student model’s parameters and enforces the consistency between the predictions by the teacher model and the student model. Recently, Athiwaratkun et al. [1] propose fast-Stochastic Weight Averaging that improves  $\Pi$ -Model and Mean Teacher by encouraging the model to explore a diverse set of plausible parameters. Smooth Neighbors on Teacher Graphs [38] construct a similarity graph between unlabeled examples and combines input-level smoothness with model-level smoothness. It is worth noting that input-level smoothness and model-level smoothness usually jointly contribute to the SOTA performance in supervised learning and hence combining them might lead to further performance improvements in the semi-supervised setting.

Also related to our work is the field of data augmentation research. Besides the conventional approaches and two data augmentation methods mentioned in Section 2.1, a recent approach MixUp [69] goes beyond data augmentation from a single data point and performs interpolation of data pairs to achieve augmentation. Recently, Hernández-García and König [24] have shown that data augmentation can be regarded as a kind of explicit regularization methods similar to Dropout. Back translation [55, 15] and dual learning [20, 5] can be regarded as performing data augmentation on monolingual data and have been shown to improve the performance for machine translation. Hu et al. [27] applies the consistency loss on augmented examples and achieve great performances on clustering and unsupervised hash learning. UDA is also well-connected to invariant representation learning [37, 53] where the consistency loss is not only applied to the output layer but is also used for feature matching.

Diverse paraphrases generated by back translation has been a key component in the improved performance of our text classification experiments. We use random sampling instead of beam search for decoding similar to the work by Edunov et al. [15]. There are also recent works on generating diverse translations [22, 56, 32] that might lead to further improvements when used as data augmentations.

Apart from semi-supervised learning, unsupervised representation learning offers another way to utilize unsupervised data. Collobert and Weston [8] demonstrated that word embeddings learned by language modeling can improve the performance significantly on semantic role labeling. Later, the pre-training of word embeddings was simplified and substantially scaled in Word2Vec [42] and Glove [48]. More recently, Dai and Le [10], Peters et al. [49], Radford et al. [50], Howard and Ruder [26], Devlin et al. [13] have shown that pre-training using language modeling and denoising auto-encoding leads to significant improvements on many tasks in the language domain. There is also a growing interest in self-supervised learning for vision [68, 23, 59]. In Section 4.1, we show that the proposed method and unsupervised representation learning can complement each other and jointly yield the state-of-the-art results.

## B Details for TF-IDF based word replacing for Text Classification

We describe the TF-IDF based word replacing data augmentation method in this section. Ideally, we would like the augmentation method to generate both diverse and valid examples. Hence, the augmentation is designed to retain keywords and replace uninformative words with other uninformative words.

Specifically, Suppose  $IDF(w)$  is the IDF score for word  $w$  computed on the whole corpus, and  $TF(w)$  is the TF score for word  $w$  in a sentence. We compute the TF-IDF score as  $TFIDF(w) = TF(w)IDF(w)$ . Suppose the maximum TF-IDF score in a sentence  $x$  is  $C = \max_i TFIDF(x_i)$ . The probability for having word  $x_i$  replaced is set to  $p(C - TFIDF(x_i))/Z_1$ , where  $p$  is a per-token replacement probability hyperparameter and  $Z_1$  is a normalization term. We would like each word  $x_1$  to have a probability of  $p$  of being replaced in expectation. Hence, we set  $Z$  to  $\sum_i (C - TFIDF(x_i))/|x|$  where  $|x|$  is the length of sentence  $x$ . We clip  $p(C - TFIDF(x_i))/Z$  to be 1 when it is greater than 1.

When a word is replaced, we sample another word from the whole vocabulary for the replacement. Intuitively, the new words should not be keywords to prevent changing the ground-truth labels of the sentence. To measure if a word is keyword, we compute a score of each word on the whole corpus. Specifically, we compute the score as  $S(w) = \text{freq}(w)IDF(w)$  where  $\text{freq}(w)$  is the frequency of word  $w$  on the whole corpus. We set the probability of sampling word  $w$  as  $(\max_{w'} S(w') - S(w))/Z_2$  where  $Z_2 = \sum_w \max_{w'} S(w') - S(w)$  is a normalization term.

## C Additional Results on CIFAR-10 and SVHN

### C.1 CIFAR-10 with 4,000 examples and SVHN with 1,000 examples

We present comparisons with the results reported by Oliver et al. [46] and three recent works [61, 19, 28]. The results are shown in Table 7. When compared with the previous SOTA model ICT [61], UDA reduces the error rate from 7.66% to 5.27% on CIFAR-10 and from 3.53% to 2.46% on SVHN, marking a relative reduction of 31.2% and 30.3%, respectively.

Methods	CIFAR-10 (4k)	SVHN (1k)
Supervised <sup>‡</sup>	20.26 ± 0.38	12.83 ± 0.47
AutoAugment [9] <sup>◊</sup>	14.1	8.2
Pseudo-Label [36] <sup>‡</sup>	17.78 ± 0.57	7.62 ± 0.29
Π-Model [35] <sup>‡</sup>	16.37 ± 0.63	7.19 ± 0.27
Mean Teacher [58] <sup>‡</sup>	15.87 ± 0.28	5.65 ± 0.47
VAT [44] <sup>‡</sup>	13.86 ± 0.27	5.63 ± 0.20
VAT + EntMin [44] <sup>‡</sup>	13.13 ± 0.39	5.35 ± 0.19
LGA + VAT [28]	12.06 ± 0.19	6.58 ± 0.36
mixmixup [19]	10	-
ICT [61]	7.66 ± 0.17	3.53 ± 0.07
UDA	<b>5.27 ± 0.11</b>	<b>2.46 ± 0.17</b>

Table 7: Comparison with existing methods on CIFAR-10 and SVHN with 4, 000 and 1, 000 examples respectively. Results marked with <sup>‡</sup> are reported by [46]. All compared methods use a common architecture WRN-28-2 with 1.46M parameters except AutoAugment<sup>◊</sup>. AutoAugment also does not use unlabeled data.

### C.2 Results with different labeled set sizes

**CIFAR-10** In Table 8, we show results for compared methods of Figure 5a. Pure supervised learning using 50,000 examples achieves an error rate of 5.36 without AutoAugment and 4.26 with AutoAugment. The performance of the baseline models are reproduced by MixMatch [3].

Methods / # Sup	250	500	1,000	2,000	4,000
Π-Model	53.02 ± 2.05	41.82 ± 1.52	31.53 ± 0.98	23.07 ± 0.66	17.41 ± 0.37
Pseudo-Label	49.98 ± 1.17	40.55 ± 1.70	30.91 ± 1.73	21.96 ± 0.42	16.21 ± 0.11
VAT	36.03 ± 2.82	26.11 ± 1.52	18.68 ± 0.40	14.40 ± 0.15	11.05 ± 0.31
Mean Teacher	47.32 ± 4.71	42.01 ± 5.86	17.32 ± 4.00	12.17 ± 0.22	10.36 ± 0.25
MixMatch	11.08 ± 0.87	9.65 ± 0.94	7.75 ± 0.32	7.03 ± 0.15	6.24 ± 0.06
UDA	<b>8.41 ± 0.46</b>	<b>6.91 ± 0.23</b>	<b>6.39 ± 0.32</b>	<b>5.84 ± 0.17</b>	<b>5.27 ± 0.11</b>

Table 8: Error rate (%) for CIFAR10.

**SVHN** In Table 9, we show results for compared methods of Figure 5b. Pure supervised learning using 73,257 examples achieves an error rate of 2.84 without AutoAugment and 2.13 with AutoAugment.

Methods / # Sup	250	500	1,000	2,000	4,000
II-Model	$17.65 \pm 0.27$	$11.44 \pm 0.39$	$8.60 \pm 0.18$	$6.94 \pm 0.27$	$5.57 \pm 0.14$
Pseudo-Label	$21.16 \pm 0.88$	$14.35 \pm 0.37$	$10.19 \pm 0.41$	$7.54 \pm 0.27$	$5.71 \pm 0.07$
VAT	$8.41 \pm 1.01$	$7.44 \pm 0.79$	$5.98 \pm 0.21$	$4.85 \pm 0.23$	$4.20 \pm 0.15$
Mean Teacher	$6.45 \pm 2.43$	$3.82 \pm 0.17$	$3.75 \pm 0.10$	$3.51 \pm 0.09$	$3.39 \pm 0.11$
MixMatch	$3.78 \pm 0.26$	$3.64 \pm 0.46$	$3.27 \pm 0.31$	$3.04 \pm 0.13$	$2.89 \pm 0.06$
UDA	<b><math>2.85 \pm 0.15</math></b>	<b><math>2.59 \pm 0.15</math></b>	<b><math>2.46 \pm 0.17</math></b>	<b><math>2.32 \pm 0.08</math></b>	<b><math>2.32 \pm 0.05</math></b>

Table 9: Error rate for SVHN.

## D Experiment Details

In this section, we provide experiment details for the considered experiment settings.

### D.1 Text Classifications

**Preprocessing.** For all text classification datasets, we truncate the input to 512 subwords since BERT is pretrained with a maximum sequence length of 512, which leads to better performances than using a sequence length of 256 or 128. Further, when the length of an example is greater than 512, we keep the last 512 subwords instead of the first 512 subwords as keeping the latter part of the sentence lead to better performances on IMDb.

**Fine-tuning BERT on in-domain unsupervised data.** We fine-tune the BERT model on in-domain unsupervised data using the code released by BERT. We try learning rate of  $2e-5$ ,  $5e-5$  and  $1e-4$ , batch size of 32, 64 and 128 and number of training steps of 30k, 100k and 300k. We pick the fine-tuned models by the BERT loss on a held-out set instead of the performance on a downstream task.

**Random initialized Transformer.** For the experiments with randomly initialized Transformer, we adopt hyperparameters for BERT base except that we only use 6 hidden layers and 8 attention heads. We also increase the dropout rate on the attention and the hidden states to 0.2. When we train UDA with randomly initialized architectures, we train UDA for 500k or 1M steps on Amazon-5 and Yelp-5 where we have abundant unlabeled data.

**BERT hyperparameters.** Following the common BERT fine-tuning procedure, we keep a dropout rate of 0.1, and try learning rate of  $1e-5$ ,  $2e-5$  and  $5e-5$  and batch size of 32 and 128. We also tune the number of steps ranging from 30 to 100k for various data sizes.

**UDA hyperparameters.** We set the weight on the unsupervised objective  $\lambda$  to 1 in all of our experiments. We use a batch size of 32 for the supervised objective since 32 is the smallest batch size on v3-32 Cloud TPU Pod. We use a batch size of 224 for the unsupervised objective when the Transformer is initialized with BERT so that the model can be trained on more unlabeled data. We find that generating one augmented example for each unlabeled example is enough for BERT<sub>FINETUNE</sub>.

All experiments in this part are performed on a v3-32 Cloud TPU Pod.

### D.2 CIFAR-10 and SVHN

For hyperparameter tuning, we follow Oliver et al. [46] and only tune the learning rate and hyperparameters for our unsupervised objective. Other hyperparameters follow those of the released AutoAugment code. Since there are many more unlabeled examples than labeled examples, we use a larger batch size for the unsupervised objective. For example, in our CIFAR-10 experiments on TPUs, we use a batch size of 32 for the supervised loss and use a batch size of 960 for the unsupervised loss. We train the model for 100k steps and the weight on the unsupervised objective  $\lambda$  is set to 1. On GPUs, we find it work well to use a batch size of 64 and 320 for the supervised loss and the unsupervised loss respectively and train for 400k steps. We generate 100 augmented examples for each unlabeled example. For the benchmark with 4,000 examples on CIFAR-10 and 1,000 examples on SVHN, we

use the same examples which AutoAugment finds its optimal policy on, since AutoAugment finds the optimal policies using 4,000 supervised examples in CIFAR-10 and 1,000 supervised examples in SVHN. We report the average performance and the standard deviation for ten runs.

For the experiments with Shake-Shake, we train UDA for 300k steps and use a batch size of 128 for the supervised objective and use a batch size of 512 for the unsupervised objective. For the experiments with PyramidNet+ShakeDrop, we train UDA for 700k steps and use a batch size of 64 for the supervised objective and a batch size of 128 for the unsupervised objective. For both models, we use a learning rate of 0.03 and use a cosine learning decay with one annealing cycle following AutoAugment.

All experiments in this part are performed on a v3-32 Cloud TPU v3 Pod.

### D.3 ImageNet

Unless otherwise stated, we follow the standard hyperparameters used in an open-source implementation of ResNet.<sup>5</sup> For the 10% labeled set setting, we use a batch size of 512 for the supervised objective and a batch size of 15,360 for the unsupervised objective. We use a base learning rate of 0.3 that is decayed by 10 for four times and set the weight on the unsupervised objective  $\lambda$  to 20. We set the threshold to 0.5 for the confidence-based filtering and set the Softmax temperature  $\tau$  to 0.4. The model is trained for 40k steps. Experiments in this part are performed on a v3-64 Cloud TPU v3 Pod.

For experiments on the full ImageNet, we use a batch size of 8,192 for the supervised objective and a batch size of 16,384 for the unsupervised objective. The weight on the unsupervised objective  $\lambda$  is set to 1. We use entropy minimization to sharpen the prediction. We use a base learning rate of 1.6 and decay it by 10 for four times. Experiments in this part are performed on a v3-128 Cloud TPU v3 Pod.

---

<sup>5</sup><https://github.com/tensorflow/tpu/tree/master/models/official/resnet>