Negative sampling in semi-supervised learning

John Chen [1]   Vatsal Shah [2]   Anastasios Kyrillidis [1]

Abstract

We introduce Negative Sampling in Semi-Supervised Learning (NS^{3}L), a simple, fast, easy to tune algorithm for semi-supervised learning (SSL). NS^{3}L is motivated by the success of negative sampling/contrastive estimation. We demonstrate that adding the NS^{3}L loss to state-of-the-art SSL algorithms, such as the Virtual Adversarial Training (VAT), significantly improves upon vanilla VAT and its variant, VAT with Entropy Minimization. By adding the NS^{3}L loss to Mix-Match, the current state-of-the-art approach on semi-supervised tasks, we observe significant improvements over vanilla MixMatch. We conduct extensive experiments on the CIFAR10, CIFAR100, SVHN and STL10 benchmark datasets. Finally, we perform an ablation study for NS^{3}L regarding its hyperparameter tuning.

1. Introduction

Deep learning has been hugely successful in areas such as image classification (Krizhevsky et al., 2012; He et al., 2016; Zagoruyko & Komodakis, 2016; Huang et al., 2017) and speech recognition (Sak et al., 2014; Sercu et al., 2016), where a large amount of labeled data is available. However, in practice it is often prohibitively expensive to create a large, high quality labeled dataset, due to lack of time, resources, or other factors. For example, the ImageNet dataset—which consists of 3.2 million labeled images in 5247 categories—took nearly two and half years to complete with the aid of Amazon's Mechanical Turk (Deng et al., 2009). Some medical tasks may require months of preparation, expensive hardware, the collaboration of many experts, and often are limited by the number of participants (Miotto et al., 2016). As a result, it is desirable to exploit unlabeled data to aid the training of deep learning models.

[1]Department of Computer Science, Rice University, Houston, Texas USA [2]Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas USA. Correspondence to: John Chen <johnchen@rice.edu>.

This form of learning is semi-supervised learning (Chapelle & Scholkopf, 2006) (SSL). Unlike supervised learning, the aim of SSL is to leverage unlabeled data, in conjunction with labeled data, to improve performance. SSL is typically evaluated on labeled datasets where a certain proportion of labels have been discarded. There have been a number of instances in which SSL is reported to achieve performance close to purely supervised learning (Laine & Aila, 2017; Miyato et al., 2017; Tarvainen & Valpola, 2017; Berthelot et al., 2019), where the purely supervised learning model is trained on the much larger whole dataset. However, despite significant progress in this field, it is still difficult to quantify when unlabeled data may aid the performance except in a handful of cases (Balcan & Blum, 2005; Ben-David et al., 2008; Kääriäinen, 2005; Niyogi, 2013; Rigollet, 2007; Singh et al., 2009; Wasserman & Lafferty, 2008).

In this work, we restrict our attention to SSL algorithms which add a loss term to the neural network loss. These algorithms are the most flexible and practical given the difficulties in hyperparameter tuning in the entire model training process, in addition to achieving the state-of-the-art performance.

We introduce *Negative Sampling in Semi-Supervised Learning (NS^{3}L)*: a simple, fast, easy to tune SSL algorithm, motivated by negative sampling/contrastive estimation (Mikolov et al., 2013; Smith & Eisner, 2005). In negative sampling/contrastive estimation, in order to train a model on unlabeled data, we exploit *implicit* negative evidence, originating from the unlabeled samples: Using negative sampling, we seek for good models that discriminate a supervised example from its neighborhood, comprised of unsupervised examples, assigned with a random (and potentially wrong) class. Stated differently, the learner learns that not only the supervised example is good, but that the same example is locally optimal in the space of examples, and that alternative examples are inferior. With negative sampling/contrastive estimation, instead of explaining and exploiting all of the data (that is not available during training), the model implicitly must only explain why the observed, supervised example is better than its unsupervised neighbors.

Overall, NS^{3}L adds a loss term to the learning objective, and is shown to improve performance simply by doing so to other state-of-the-art SSL objectives. Since modern datasets

often have a large number of classes (Russakovsky et al., 2014), we are motivated by the observation that *it is often much easier to label a sample with a class or classes it is not, as opposed to the one class it is,* exploiting ideas from negative sampling/contrastive estimation (Mikolov et al., 2013; Smith & Eisner, 2005).

Key Contributions. Our findings can be summarized as follows:

i) We propose a new SSL algorithm, which is easy to tune, and improves SSL performance of other state of the art algorithms across a wide range of reasonable hyperparameters, simply by adding the NS^{3}L loss in their objective.

ii) Adding the NS^{3}L loss to a variety of losses, including Virtual Adversarial Training (VAT) (Miyato et al., 2017), Π model, and MixMatch (Berthelot et al., 2019), we observe improved performance compared to vanilla alternatives as well as the addition of Pseudo-Labeling or Entropy Minimization, for the standard SSL benchmarks of SVHN, CIFAR10, and CIFAR100.

iii) Adding the NS^{3}L loss to the state-of-the-art SSL algorithm, i.e., the MixMatch procedure (Berthelot et al., 2019), NS^{3}L combined with MixMatch produces superior performance for the standard SSL benchmarks of SVHN, CIFAR10 and STL-10.

Namely, adding the NS^{3}L loss to existing SSL algorithms is an easy way to improve performance, and requires limited extra computational resources for hyperparameter tuning, since it is interpretable, fast, and sufficiently easy to tune.

2. Related Work

In this paper, we restrict our attention to a subset of SSL algorithms which add a loss to the supervised loss function. *These algorithms tend to be more practical in terms of hyperparameter tuning* (Berthelot et al., 2019). There are a number of SSL algorithms not discussed in this paper, following and as mentioned in (Berthelot et al., 2019): including "transductive" models (Joachims, 1999; 2003; Gammerman et al., 1998), graph-based methods (Zhu et al., 2003; Bengio et al., 2006), and generative modeling (Joachims, 2003; Belkin & Niyogi, 2002; Salakhutdinov & Hinton, 2007; Coates & Ng, 2011; Goodfellow et al., 2011; Kingma et al., 2014; Odena, 2016; Pu et al., 2016; Salimans et al., 2016). For a comprehensive overview of SSL methods, refer to (Chapelle & Scholkopf, 2006), or (Zhu et al., 2003).

2.1. Consistency Regularization

Consistency regularization applies data augmentation to semi-supervised learning with the following intuition: Small

perturbations for each sample should not significantly change the output of the network. This is usually achieved by minimizing some distance measure between the output of the network, with and without perturbations in the input. The most straightforward distance measure is the mean squared error used by the Π model (Laine & Aila, 2017; Sajjadi et al., 2016). The Π model adds the distance term $d(f_\theta(x), f_\theta(\hat{x}))$, where $\hat{x}$ is the result of a stochastic perturbation to x, to the supervised classification loss as a regularizer, with some weight.

Mean teacher (Tarvainen & Valpola, 2017) observes the potentially unstable target prediction over the course of training with the Π model approach, and proposes a prediction function, parameterized by an exponential moving average of model parameter values. Mean teacher adds $d(f_\theta(x), f_{\theta'}(x))$, where θ' is an exponential moving average of θ, to the supervised classification loss with some weight. However, the stochastic perturbation used in these methods are domain specific.

2.2. Virtual Adversarial Training

Virtual Adversarial Training (Miyato et al., 2017) (VAT) approximates perturbations to be applied over the input to most significantly affect the output class distribution, inspired by adversarial examples (Goodfellow et al., 2015; Szegedy et al., 2014). VAT computes an approximation of the perturbation as:

$$r \sim \mathcal{N}\left(0, \frac{\xi}{\sqrt{\dim(x)}} I\right)$$
$$g = \nabla_r d\left(f_\theta(x), f_\theta(x+r)\right)$$
$$r_{\text{adv}} = \epsilon \cdot g / \|g\|_2$$

where x is an input data sample, $\dim(\cdot)$ is its dimension, d is a non-negative function that measures the divergence between two distributions, ξ and ϵ are scalar hyperparameters. Consistency regularization is then used to minimize the distance between the output of the network, with and without the perturbations in the input. Since we follow the work in (Oliver et al., 2018) almost exactly, we select the best performing consistency regularization SSL method in that work, VAT, for comparison and combination with NS^{3}L for non-Mixup SSL; Mixup procedure will be described later.

2.3. Entropy minimization

The goal of entropy minimization (Grandvalet & Bengio, 2005) is to discourage the decision boundary from passing near samples where the network produces low-confidence predictions. One way to achieve this is by adding a simple loss term to minimize the entropy for unlabeled data x with total K classes: $-\sum_{k=1}^{K} \mu_{xk} \log \mu_{xk}$. Entropy minimization on its own has not demonstrated competitive performance in SSL, however it can be combined with VAT for
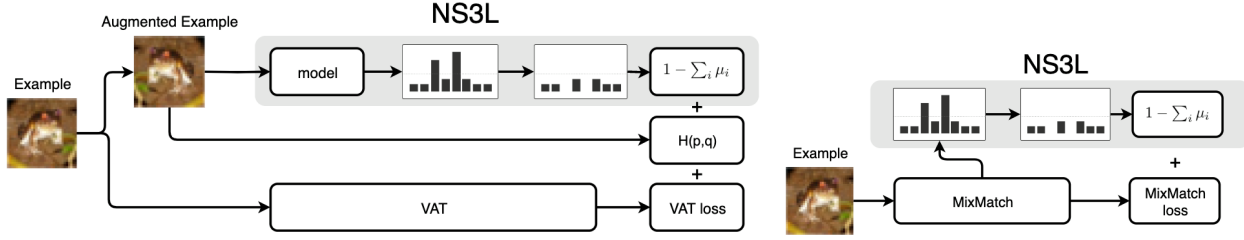
Figure 1: Left: Diagram of NS^{3}L with VAT. For NS^{3}L, an augmented example is fed into the model, which outputs a probability for each class. A threshold T is used to determine classes with sufficiently low probability, and these classes are fed into the NS^{3}L loss. The NS^{3}L loss is combined with the existing VAT loss and Cross Entropy loss. Right: Similar diagram of NS^{3}L with MixMatch; the NS^{3}L loss is combined with the existing MixMatch loss.

stronger results (Miyato et al., 2017; Oliver et al., 2018). We include entropy minimization with VAT in our experiments.

2.4. Pseudo-Labeling

Pseudo-Labeling (Lee, 2013) is a simple and easy to tune method which is widely used in practice. For a particular sample, it requires only the probability value of each class, the output of the network, and labels the sample with a class if the probability value crosses a certain threshold. The sample is then treated as a labeled sample with the standard supervised loss function. Pseudo-Labeling is closely related to entropy minimization, but only enforces low-entropy predictions for predictions which are already low-entropy. We emphasize here that the popularity of Pseudo-Labeling is likely due to its simplicity and limited extra cost for hyperparameter search.

2.5. SSL with modern data augmentation techniques

Mixup (Zhang et al., 2017) combines pairs of samples and their one-hot labels $(x_1, y_1), (x_2, y_2)$ as in: $x' = \lambda x_1 + (1 - \lambda)x_2$, $y' = \lambda y_1 + (1 - \lambda)y_2$, where $\lambda \sim \texttt{Beta}(\alpha, \alpha)$, to produce a new sample (x', y') with α being a hyperparameter. Mixup is a form of regularization which encourages the neural network to behave linearly between training examples, justified by Occam's Razor (Zhang et al., 2017). In SSL, the labels y_1, y_2 are typically the predicted labels by a neural network with some processing steps.

Applying Mixup to SSL led to Interpolation Consistency Training (ICT) (Verma et al., 2019) and MixMatch (Berthelot et al., 2019), which significantly improved upon previous results with SSL on the standard benchmarks of CIFAR10 and SVHN. ICT trains the model f_θ to output predictions similar to a mean-teacher $f_{\theta'}$, where θ' is an exponential moving average of θ. Namely, on unlabeled data, ICT encourages $f_\theta(\texttt{Mixup}(x_i, x_j)) \approx \texttt{Mixup}(f_{\theta'}(x_i), f_{\theta'}(x_j))$.

MixMatch applies a number of processing steps for labeled

and unlabeled data on each iteration and mixes both labeled and unlabeled data together. The final loss is given by $\mathcal{L} = \mathcal{L}_{\text{supervised}} + \lambda_3 \mathcal{L}_{\text{ubsupervised}}$, where

$$\mathcal{X}', \mathcal{U}' = \texttt{MixMatch}(\mathcal{X}, \mathcal{U}, E, A, \alpha)$$

$$\mathcal{L}_{\text{supervised}} = \frac{1}{|\mathcal{X}'|} \sum_{i_1 \in \mathcal{X}'} \sum_{k=1}^{K} y_{i_1 k} \log \mu_{i_1 k}$$

$$\mathcal{L}_{\text{unsupervised}} = \frac{1}{K|\mathcal{U}'|} \sum_{i_2 \in \mathcal{U}'} \sum_{k=1}^{K} (y_{i_2 k} - \mu_{i_2 k})^2$$

where $\mathcal{X}$ is the labeled data $\{x_{i_1}, y_{i_1}\}_{i_1=1}^{n}$, $\mathcal{U}$ is the unlabeled data $\{x_{i_2}^u\}_{i_2=1}^{n_u}$, $\mathcal{X}'$ and $\mathcal{U}'$ are the output samples labeled by MixMatch, and E, A, α, λ_3 are hyperparameters. Given a batch of labeled and unlabeled samples, MixMatch applies A data augmentations on each unlabeled sample x_{i_2}, averages the predictions across the A augmentations,

$$p = \frac{1}{A} \sum_{a=1}^{A} f_\theta(\texttt{Augment}(x_{i_2}^u))$$

and applies temperature sharpening,

$$\texttt{Sharpen}(p, E)_k := \frac{p_k^{1/E}}{\sum_{k=1}^{K} p_k^{1/E}},$$

to the average prediction. A is typically 2 in practice, and E is 0.5. The unlabeled data is labeled with this sharpened average prediction.

Let the collection of labeled unlabeled data be $\widehat{\mathcal{U}}$. Standard data augmentation is applied to the originally labeled data and let this be denoted $\widehat{\mathcal{X}}$. Let $\mathcal{W}$ denote the shuffled collection of $\widehat{\mathcal{U}}$ and $\widehat{\mathcal{X}}$. MixMatch alters Mixup by adding a max operation: $\lambda \sim \texttt{Beta}(\alpha, \alpha)$, $\lambda' = \max(\lambda, 1 - \lambda)$; it then produces $\mathcal{X}' = \texttt{Mixup}(\widehat{\mathcal{X}}_{i_1}, W_{i_1})$ and $\mathcal{U}' = \texttt{Mixup}(\widehat{\mathcal{U}}_{i_2}, W_{i_2 + |\widehat{\mathcal{X}}|})$.

Since MixMatch performs the strongest empirically, we select MixMatch as the best performing Mixup-based SSL method for comparison and combination with NS³L. We make a note here that more recently there is also work on applying stronger data augmentation (Xie et al., 2019).

3. Negative Sampling in Semi-Supervised Learning

In this section, we provide the pseudo-code for the Negative Sampling with Semi-Supervised Learning (NS³L) algorithm in Algorithm 1. NS³L assigns a random label to an unsupervised sample as long as the probability of that random label being correct is low. Adding NS³L to any existing algorithms allows us to achieve significant performance improvements. We first provide the mathematical motivation behind NS³L followed by intuition of why NS³L works using a simple toy example in $1D$.

3.1. Mathematical Motivation

Let the set of labeled samples be denoted as $\{x_i, y_i\}_{i=1}^{n}$, x_i being the input and y_i being the associated label, and the set of unlabeled samples be denoted as $\{x_i^u\}_{i=1}^{n_u}$, each with unknown correct label y_i^u. For the rest of the text, we will consider the cross-entropy loss, which is one of the most widely used loss functions for classification. The objective function for cross entropy loss over the labeled examples is:

$$\mathcal{L}\left(\{x_i, y_i\}_{i=1}^{n}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log \mu_{ik},$$

where there are n labeled samples, K classes, $y_{ik} = \mathbb{1}_{k=y_i}$ is the identity operator that equals 1 when $k = y_i$, and μ_{ik} is the output of the classifier for sample i for class k.

For the sake of simplicity, we will perform the following relabeling: for all $i \in [n_u]$, $x_{i+n} = x_i^u$ and $y_{i+n} = y_i^u$. In the hypothetical scenario where the labels for the unlabeled data are known and for w the parameters of the model, the likelihood would be:

$$\mathbb{P}\left[\{y_i\}_{i=1}^{n+n_u} \mid \{x_i\}_{i=1}^{n+n_u}, w\right]$$
$$= \prod_{i=1}^{n+n_u} \mathbb{P}\left[y_i \mid x_i, w\right] = \prod_{i=1}^{n+n_u}\prod_{k=1}^{K} \mu_{ik}^{y_{ik}},$$
$$= \left(\prod_{i_1=1}^{n}\prod_{k=1}^{K} \mu_{i_1 k}^{y_{i_1 k}}\right) \cdot \left(\prod_{i_2=1}^{n_u}\prod_{k=1}^{K} \mu_{i_2 k}^{y_{i_2 k}^u}\right)$$

Observe that, $\prod_{k=1}^{K} \mu_{i_2 k}^{y_{i_2 k}^u} = 1 - \sum_{j:y_{i_2 j}\neq 1} \mu_{i_2 j}$, which follows from the definition of the quantities $\mu_{\cdot}$ that represent a probability distribution and, consequently, sum up to one.

Taking negative logarithms allows us to split the loss function into two components: i) the supervised part and ii) the

---

Algorithm 1 NS³L

1: **Input**: Mini batch size B, batch of examples x_b and their predicted vector of label probabilities $\hat{y}_b$ using the output of the classifier $\{x_b, \hat{y}_b\}_{b=1}^{B}$, threshold T.
2: $\mathcal{L}_{\mathrm{NS^3L}} = 0$.
3: **for** $b = 1, \ldots, B$ **do**
4:     $\mathbb{1}_{\hat{y}_b'} = \mathtt{isTrue}(\hat{y}_b < T)$.
5:     $\mathcal{L}_{\mathrm{NS^3L}} = \mathcal{L}_{\mathrm{NS^3L}} - \log\left(1 - \sum_{k=1}^{K} \mathbb{1}_{\hat{y}_{bk}'}\mu_{bk}\right)$.
6: **end for**
7: **Return** $\frac{1}{B}\mathcal{L}_{\mathrm{NS^3L}}$

---

unsupervised part. The log-likelihood loss function can now be written as follows:

$$\mathcal{L}\left(\{x_i, y_i\}_{i=1}^{n+n_u}\right) = \underbrace{-\frac{1}{n}\sum_{i_1=1}^{n}\sum_{k=1}^{K} y_{ik} \log \mu_{ik}}_{:=\text{supervised part}}$$
$$\underbrace{-\frac{1}{n_u}\sum_{i_2=1}^{n_u} \log\left(1 - \sum_{j\neq\text{True label}} \mu_{i_2 j}\right)}_{:=\text{unsupervised part}}$$

While the true labels need to be known for the unsupervised part to be accurate, we draw ideas from negative sampling/contrastive estimation (Mikolov et al., 2013; Smith & Eisner, 2005): i.e., for each unlabeled example in the unsupervised part, we randomly assign P labels from the set of labels; see also Appendix A. These P labels indicate classes that *the sample does not belong to*: as the number of labels in the task increase, the probability of including the correct label in the set of P labels is small. The way labels are selected could be uniformly at random or by using Nearest Neighbor search, or even based on the output probabilities of the network, where with high probability the correct label is not picked.

The approach above assumes the use of the full dataset, both for the supervised and unsupervised parts. In practice, more often than not we train models based on stochastic gradient descent, and we implement a mini-batch variant of this approach with different batch sizes B_1 and B_2 for labeled and unlabeled data, respectively. Particularly, for the supervised mini-batch of size B_1 for labeled data, the objective term is approximated as:

$$\frac{1}{n}\sum_{i_1=1}^{n}\sum_{k=1}^{K} y_{ik} \log \mu_{ik} \approx \frac{1}{|B_1|}\sum_{i_1\in B_1}\sum_{k=1}^{K} y_{ik} \log \mu_{ik}.$$

The unsupervised part with mini-batch size of B_2 and NS³L loss, where each unlabeled sample is connected with P_{i_2}

a) Data is identically distributed

b) Few supervised samples lead to selection bias

c) Allow unsupervised data into the mix. Consider negative samples based on a threshold T (i.e. in shaded yellow region)

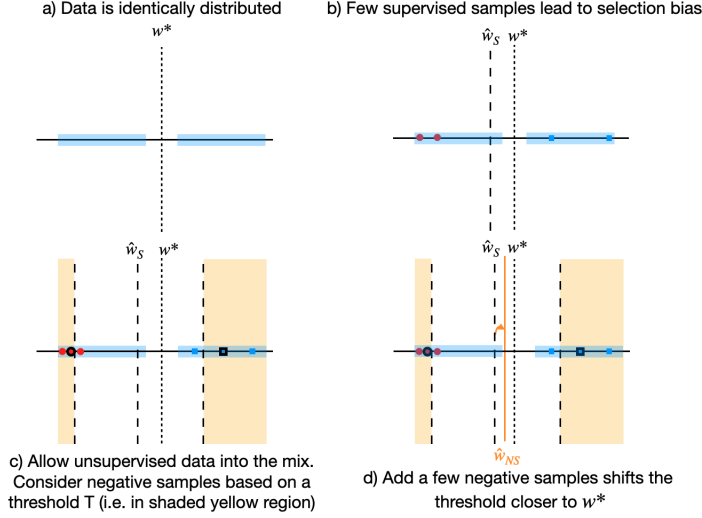d) Add a few negative samples shifts the threshold closer to w^*

Figure 2: A toy example illustrating the effectiveness of Negative Sampling in Semi supervised learning

hopefully incorrect labels, is approximated as:

$$\frac{1}{n_u} \sum_{i_2=1}^{n_u} \log\left(1 - \sum_{j \neq \text{True label}} \mu_{i_2 j}\right)$$

$$\approx \frac{1}{|B_2|} \sum_{i_2 \in B_2} \log\left(1 - \sum_{j=1}^{P_{i_2}} \mu_{i_2 j}\right)$$

Based on the above, our NS^{3}L loss looks as follows:

$$\hat{\mathcal{L}}_{B_1, B_2}\left(\{x_i, y_i\}_{i=1}^{n+n_u}\right) = -\frac{1}{|B_1|} \sum_{i_1 \in B_1} \sum_{k=1}^{K} y_{ik} \log \mu_{ik}$$

$$\underbrace{- \frac{1}{|B_2|} \sum_{i_2 \in B_2} \log\left(1 - \sum_{j=1}^{P_{i_2}} \mu_{i_2 j}\right)}_{:= \text{NS}^3\text{L loss}}$$

Thus, the NS^{3}L loss is just an additive loss term that can be easily included in many existing SSL algorithms, as we show next. For clarity, a pseudocode implementation of the algorithm where negative labels are identified by the label probability being below a threshold T, as the output of the classifier or otherwise, is given in Algorithm 1.

3.2. Intuition

Our aim is to illustrate how our simple idea aids the task of learning with unlabeled data. We will consider a simple example in 1D (Figure 2), where we assume binary classification with cross-entropy loss for simplicity.

Let $w^\star$ denote the separating hyperplane and assume that the data lies uniformly on either side of $w^\star$, indicated by the

shaded blue region (Figure 2a). Without loss of generality, let the points on the left and right of the hyperplane have the labels 1 and 0, respectively. Our aim is to recover $w^\star$.

It is possible for the labeled examples to have a selection bias (Chawla & Karakoulas, 2005) (for example certain images of cats are easier to label than others); assume that this property leads the algorithm to converge to $\hat{w}$; Figure 2b. However, in the SSL setting, we do have access to a large number of unlabeled examples. How can we utilize it to improve our prediction?

Consider one of the highlighted samples (x_u) (red dot with black boundary in Figure 2c). Let us assume its underlying true label is 1. The key difference in both approaches is that in inductive SSL (Chapelle & Scholkopf, 2006; Zhu et al., 2003) we make a gradient update by labeling any point in the shaded yellow region as the predicted label while in negative sampling we make a gradient update by labeling the same point as not 0. Both these algorithms only perform updates only if we are certain about the label.

Now, let us compare the gradients of a sample using the classical inductive SSL approach and negative sampling.

$$\text{Inductive SSL}: \nabla \mathcal{L}\left(\{x_u\}\right) = -(1 - \mu_u)x_u$$

$$\text{NS}^3\text{L}: \nabla \mathcal{L}\left(\{x_u\}\right) = \mu_u x_u$$

From the equations above, it is clear that NS^{3}L and Inductive SSL push the gradients in opposite directions. The gradient updates of supervised samples align with the gradient updates of the unsupervised samples labeled using Inductive SSL. However, that is not the case for NS^{3}L. Since the unsupervised data samples come from a uniform distri-

bution, it is more likely that we will pick more "negative" samples from the class on the right (intersection of yellow and blue shaded regions). These negative samples have a bias to the right side of the plane ultimately bringing back the separating hyper-plane closer to $w^\star$ (Figure 2d).

4. Experiments

We use the codebase from (Berthelot et al., 2019) for experiments involving MixMatch, and otherwise use the codebase from (Oliver et al., 2018). We make the distinction due to the existence of some experimental differences, and this is the best way to reproduce the reported performances. Namely, (Berthelot et al., 2019) differs from (Oliver et al., 2018) in that it evaluates an exponential moving average of the model parameters, as opposed to using a learning rate decay schedule, and uses weight decay.

4.1. Experimental Setup

Following (Oliver et al., 2018), the model employed is the standard Wide ResNet (WRN) (Zagoruyko & Komodakis, 2016) with depth 28 and width 2, batch normalization (Ioffe & Szegedy, 2015), and leaky ReLU activations (Maas & Ng, 2013). The optimizer is the Adam optimizer (Kingma & Ba, 2014). The batch size is 100, half of which are labeled and half are unlabeled. Standard procedures for regularization, data augmentation, and preprocessing are followed.

We use the standard training data/validation data split for SVHN, with 65,932 training images and 7,325 validation images. All but 1,000 examples are turned "unlabeled". Similarly, we use the standard training/data validation data split for CIFAR10, with 45,000 training images and 5,000 validation images. All but 4,000 labels are turned "unlabeled". We also use the standard data split for CIFAR100, with 45,000 training images and 5,000 validation images. All but 10,000 labels are turned "unlabeled".

Hyperparameters are optimized to minimize validation error; test error is reported at the point of lowest validation error. We select hyperparameters which perform well for both SVHN and CIFAR10. After selecting hyperparameters on CIFAR10 and SVHN, we run the same hyperparameters with practically no further tuning on CIFAR100 to determine the ability of each method to generalize to new datasets. Since VAT and VAT + EntMin use different hyperparameters for CIFAR10 and SVHN, we use those tuned for CIFAR10 for the CIFAR100 dataset. For NS^3L, $NS^3L + \Pi$ model, $NS^3L + VAT$, we divide the threshold T by 10 since there are 10x classes in CIFAR100. We run 5 seeds for all cases.

Since models are typically trained on CIFAR10 (Krizhevsky, 2009) and SVHN (Netzer et al., 2011) for fewer than the 500,000 iterations (1,000 epochs) (Oliver et al., 2018), we make the only changes of reducing the total iterations to 200,000, warmup period (Tarvainen & Valpola, 2017) to 50,000, and iteration of learning rate decay to 130,000. All other methodology follows that work (Oliver et al., 2018).

For MixMatch experiments, we follow the methodology of (Berthelot et al., 2019) and continue to use the same model described above. Since the performance of MixMatch is particularly strong using only a small number of labeled samples, we also include experiments for SVHN with all but 250 labels discarded, and CIFAR10 with all but 250 labels discarded, in addition to the previously mentioned experiments. We also include experiments on STL10, a dataset designed for SSL, which has 5,000 labeled images and 100,000 unlabeled images drawn from a slightly different distribution than the labeled data. All but 1,000 labels are discarded for STL10. The median of the last 20 checkpoints' test error is reported, following (Berthelot et al., 2019). Note that we reduce the training epochs of STL10 significantly in interest of training time. All other methodology follows the work of MixMatch.

4.2. Baseline Methods

For baseline methods, we consider Pseudo-Labeling, due to its simplicity on the level of NS^3L, and MixMatch and VAT for its performance, in addition to VAT + Entropy Minimization and VAT + Psuedo-Labeling. We also include Π model and omit Mean Teacher, although we follow the experiments of (Oliver et al., 2018) and both produce worse performance than VAT. The supervised baseline is trained on the remaining labeled data after some labels have been removed. We generally follow the tuned hyperparameters in the literature and do not observe noticeable gains from further hyperparameter tuning.

4.3. Implementation of NS^3L

We implement NS^3L using the output probabilities of the network with the unlabeled samples, namely

$$\mathcal{L}_{NS^3L} = NS^3L(\{x_{i_2}, \mu_{i_2}\}_{i_2=1}^B, T).$$

The performance of NS^3L with random negative sampling assignment or Nearest Neighbor-based assignment is given in Section **??** in the appendix. We label a sample with negative labels for the classes whose probability value falls below a certain threshold. We then simply add the NS^3L loss to the existing SSL loss function. Using NS^3L on its own gives

$$\mathcal{L} = \mathcal{L}_{supervised} + \lambda_1 \mathcal{L}_{NS^3L}$$

for some weighting λ_1. For adding NS^3L to VAT, this gives

$$\mathcal{L} = \mathcal{L}_{supervised} + \lambda_2 \mathcal{L}_{VAT} + \lambda_1 \mathcal{L}_{NS^3L}$$

for some weighting $\lambda_i, i \in \{1, 2\}$. This is applied similarly to the Π model. The weighting is a common practice in SSL,

Table 1: Test errors achieved by various SSL approaches on the standard benchmarks of CIFAR10, with all but 4,000 labels removed, SVHN, with all but 1,000 labels removed, and CIFAR100, with all but 10,000 labels removed. "Supervised" refers to using only 4,000, 1,000, and 10,000 labeled samples from CIFAR10, SVHN, and CIFAR100 respectively without any unlabeled data. VAT refers to Virtual Adversarial Training.

Dataset	Supervised	PL	NS^{3}L	VAT	VAT + EntMin	Π model	Π + NS^{3}L	VAT + NS^{3}L
CIFAR10	$20.76 \pm .28$	$17.56 \pm .29$	$16.03 \pm .05$	$14.72 \pm .23$	$14.34 \pm .18$	$17.12 \pm .19$	$16.06 \pm .21$	$13.94 \pm .10$
SVHN	$12.39 \pm .53$	$7.70 \pm .22$	$6.52 \pm .22$	$6.20 \pm .11$	$6.10 \pm .02$	$8.48 \pm .15$	$7.98 \pm .18$	$5.51 \pm .14$
CIFAR100	$48.26 \pm .25$	$46.91 \pm .31$	$46.34 \pm .37$	$44.38 \pm .56$	$43.92 \pm .44$	$47.87 \pm .34$	$46.98 \pm .41$	$43.70 \pm .19$

also used in MixMatch and VAT + Entropy Minimization. This is the simplest form of NS^{3}L and we believe there are large gains to be made with more complex methods of choosing the negative labels.

Recall that MixMatch outputs $\mathcal{X}', \mathcal{U}' = \texttt{MixMatch}(\mathcal{X}, \mathcal{U}, T, A, \alpha)$ collections of samples with their generated labels. We label each sample $x_i \in \mathcal{X}' \bigcup \mathcal{U}'$ with negative labels for the classes whose generated probability value falls below a certain threshold. We then simply add the NS^{3}L loss to the existing SSL loss function, computing the NS^{3}L loss using the probability outputs of the network as usual. Namely,

$$\mathcal{X}', \mathcal{U}' = \texttt{MixMatch}(\mathcal{X}, \mathcal{U}, E, A, \alpha)$$

$$\mathcal{L}_{\text{supervised}} = \frac{1}{|\mathcal{X}'|} \sum_{i_1 \in \mathcal{X}'} \sum_{k=1}^{K} y_{i_1 k} \log \mu_{i_1 k}$$

$$\mathcal{L}_{\text{unsupervised}} = \frac{1}{K|\mathcal{U}'|} \sum_{i_2 \in \mathcal{U}'} \sum_{k=1}^{K} (y_{i_2 k} - \mu_{i_2 k})^2$$

$$\mathcal{L}_{\text{NS}^3\text{L}} = \text{NS}^3\text{L}(\mathcal{X}' \bigcup \mathcal{U}', T)$$
$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \lambda_3 \mathcal{L}_{\text{unsupervised}} + \lambda_1 \mathcal{L}_{\text{NS}^3\text{L}}$$

4.4. Results

We follow the practice in (Oliver et al., 2018) and use the same hyperparameters for plain NS^{3}L and NS^{3}L as added to other losses, e.g. NS^{3}L + VAT, for both CIFAR10 and SVHN. After selecting hyperparameters on CIFAR10 and SVHN, we run almost the exact same hyperparameters with little further tuning on CIFAR100, where the threshold T is divided by 10 since there are 10x classes in CIFAR100.

For MixMatch experiments, we follow the practice of (Berthelot et al., 2019) and tune NS^{3}L separately for each dataset. MixMatch + NS^{3}L only takes marginally longer runtime than MixMatch on its own. The learning rate is fixed.

CIFAR10: We evaluate the accuracy of each method with 4,000 labeled samples and 41,000 unlabeled samples, as is

standard practice. The results are given in Table 1. Further results comparing the addition of Entropy Minimization, Pseudo-Labeling and NS^{3}L are given in Table 2. MixMatch results are given in Table 3. For NS^{3}L, we use a threshold $T = 0.04$, learning rate of 6e-4, and $\lambda_1 = 1$. Identical hyperparameters are used for Π model + NS^{3}L. For VAT + NS^{3}L, we use a shared learning rate of 6e-4 and reduce λ_1 from 1 to 0.3, which is identical to λ_2. We perform extensive hyperparameter tuning for VAT + PL. For MixMatch, as in (Berthelot et al., 2019), we use $\alpha = 0.75$ and $\lambda_3 = 75$. For NS^{3}L + MixMatch, we use a threshold of $T = 0.05$ and a coefficient of $\lambda_1 = 5$ for 250 labeled samples and $\lambda_1 = 10$ for 4,000 labeled samples. All other settings remain as is optimized individually.

We created 5 splits of the number of labeled samples, each with a different seed. Each model is trained on a different split and test error is reported with mean and standard deviation. We find that NS^{3}L performs reasonably well and significantly better than Pseudo-Labeling, over a 1.5% improvement. A significant gain over all algorithms is attained by adding the NS^{3}L loss to the VAT loss. VAT + NS^{3}L achieves almost a 1% improvement over VAT, and is about 0.5% better than VAT + EntMin and VAT + PL. We also find that adding NS^{3}L immediately improves the performance of MixMatch, with a 2% improvement with 250 labeled samples and a small improvement for 4,000 samples. The 250 labeled samples case may be the more interesting case since it highlights the sample efficiency of the method. This underscores the flexibility of NS^{3}L to improve existing methods.

SVHN: We evaluate the accuracy of each method with 1,000 labeled samples and 64,932 unlabeled samples, as is standard practice. The results are shown in Table 1. MixMatch results are shown in Table 4. We use the same hyperparameters for NS^{3}L, Π model + NS^{3}L and VAT + NS^{3}L as in CIFAR10. For MixMatch ollowing the literature, we use $\alpha = 0.75$ and $\lambda_3 = 250$. For NS^{3}L + MixMatch, we again use a threshold of $T = 0.05$ and a coefficient of $\lambda_1 = 2$ for both 250 labeled samples and 1,000 labeled samples.

Again, 5 splits are created, each with a different seed. Each model is trained on a different split and test error is reported

Table 2: Test errors achieved by various SSL approaches on top of VAT on the standard benchmarks of CIFAR10, with all but 4,000 labels removed, and CIFAR100, with all but 10,000 labels removed. VAT, EntMin and PL refer to Virtual Adversarial Training, Entropy Minimization, and Pseudo-Labeling respectively.

Dataset	VAT	VAT + EntMin	VAT + PL	VAT + NS^{3}L
CIFAR10	$14.72 \pm .23$	$14.34 \pm .18$	$14.15 \pm .14$	$\mathbf{13.94} \pm .10$
CIFAR100	$44.38 \pm .56$	$43.92 \pm .44$	$43.93 \pm .33$	$\mathbf{43.70} \pm .19$

Table 3: Test errors achieved by MixMatch and MixMatch + NS^{3}L on the standard benchmark of CIFAR10, with all but 250 labels removed and all but 4,000 labels removed.

CIFAR10	250	4,000
MixMatch	14.49 ± 1.60	7.05 ± 0.10
Mixmatch + NS^{3}L	$\mathbf{12.48} \pm 1.21$	$\mathbf{6.92} \pm 0.12$

Table 5: Test errors achieved by MixMatch and MixMatch + NS^{3}L on the standard benchmark of STL10, with all but 1,000 labels removed.

STL10	1,000
MixMatch	22.20 ± 0.89
Mixmatch + NS^{3}L	$\mathbf{21.74} \pm 0.33$

with mean and standard deviation. Here, NS^{3}L achieves competitive learning rate with VAT, 6.52% versus 6.20%, and is significantly better than Pseudo-Labeling, at 7.70%. By combining NS^{3}L with VAT, test error is further reduced by a notable margin, almost 1% better than VAT alone and more than 0.5% better than VAT + EntMin.

By adding NS^{3}L to MixMatch, the model achieves almost the same test error with 250 labeled samples than it does using only MixMatch on 1,000 labeled samples. In other words, in this case applying NS^{3}L improves performance almost equivalent to having 4x the amount of labeled data. In the cases of 250 labeled samples and 1,000 labeled samples, adding NS^{3}L to MixMatch improves performance by 0.4% and 0.15% respectively, achieving state-of-the-art results.

Table 4: Test errors achieved by MixMatch and MixMatch + NS^{3}L on the standard benchmark of SVHN, with all but 250 labels removed and all but 1,000 labels removed.

SVHN	250	1,000
MixMatch	3.75 ± 0.09	3.28 ± 0.11
Mixmatch + NS^{3}L	$\mathbf{3.38} \pm 0.08$	$\mathbf{3.14} \pm 0.11$

STL10: We evaluate the accuracy of MixMatch and Mix-Match + NS^{3}L with 1,000 labeled samples and 100,000 unlabeled samples. The results are given in Table 5. Following the literature, we use $\alpha = 0.75$ and $\lambda_3 = 50$. For NS^{3}L, we again use a threshold of $T = 0.05$ and $\lambda_1 = 2$. We trained the model for a significantly fewer epochs than in (Berthelot et al., 2019), however even in this case NS^{3}L can improve upon MixMatch, reducing test error slightly.

CIFAR100; We evaluate the accuracy of each method with 10,000 labeled samples and 35,000 unlabeled samples, as is standard practice. The results are given in Table 1. For NS^{3}L, we use a threshold $T = 0.04/10 = 0.004$, learning rate of 6e-4, and $\lambda_1 = 1$, following the settings in CIFAR10 and SVHN. For VAT + NS^{3}L in CIFAR100, we use a shared learning rate of 3e-3 and $\lambda_1 = 0.3$, $\lambda_2 = 0.6$.

As before, we created 5 splits of 10,000 labeled samples, each with a different seed, and each model is trained on a different split. Test error is reported with mean and standard deviation. NS^{3}L is observed to improve 0.6% test error over Pseudo-Labeling and adding NS^{3}L to VAT reduces test error slightly and achieves the best performance. This suggests that EntMin and NS^{3}L boosts VAT even with little hyperparameter tuning, and perhaps should be used as default. We note that the performance of SSL methods can be sensitive to hyperparameter tuning, and minor hyperparameter tuning may improve performance greatly. *Due to VAT performing additional forward and backwards passes, NS^{3}L alone runs more than 2x faster than VAT.*

5. Parameter Sensitivity

We provide experimental results on the sensitivity of NS^{3}L with respect to the threshold parameter T and the weighting parameter λ_1. We use the CIFAR10 dataset with all but 4,000 labels removed for NS^{3}L and VAT + NS^{3}L. We use the SVHN dataset with all but 250 labels removed for Mix-Match + NS^{3}L. We fix all other optimal parameters given in Section 4. Results are given in Figure 3, where 4 values of threshold T and 3 values of weighting parameter λ_1 are selected. We interpolated the result for better readability.

Referring to Figure 3, the optimal λ_1 depends on the setting and is affected when used simultaneously with VAT
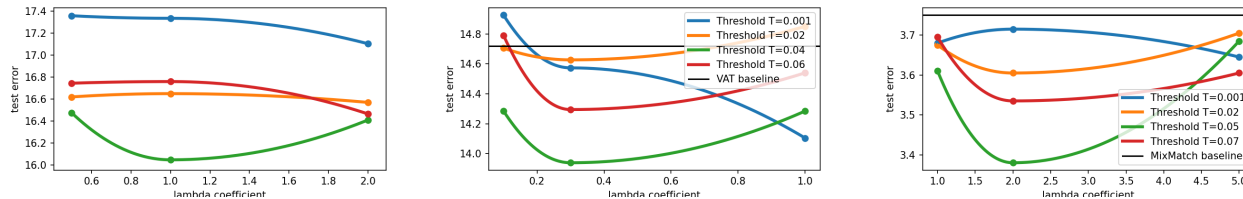
Figure 3: Parameter sensitivity study. Left: Test errors achieved by NS^{3}L on the standard benchmark of CIFAR10, with all but 4,000 labels removed. Middle: Test errors achieved by VAT + NS^{3}L on the standard benchmark of CIFAR10, with all but 4,000 labels removed. Right row: Test errors achieved by Mixmatch + NS^{3}L on the standard benchmark of SVHN, with all but 250 labels removed.

or Mixmatch. E.g., the optimal λ_1 for NS^{3}L on CIFAR10 with all but 4,000 labels removed varies from approximately 1, when added to the existing cross entropy loss alone, to 0.3, when added to the cross entropy loss and VAT with a coefficient of 0.3. When added to Mixmatch on SVHN with all but 250 labels removed, the optimal λ_1 is closer to 2.

The performance is more sensitive to the threshold T, and an optimal threshold $T \approx 0.04$ appears to hold empirically across settings, and we note that the datasets are all of 10 classes. Referring to Table 1 and Table 4, we see a clear improvement by adding NS3L, even when it is poorly tuned.

6. Conclusion

With simplicity, speed, and ease of tuning in mind, we proposed *Negative Sampling in Semi-Supervised Learning* (NS^{3}L), a semi-supervised learning method inspired by negative sampling, which simply adds a loss function. We demonstrate the effectiveness of NS^{3}L when combined with existing SSL algorithms, producing the overall best result for non-Mixup-based SSL, by combining NS^{3}L with VAT, and Mixup-based SSL, by combining NS^{3}L with MixMatch. We show improvements across a variety of tasks with only a minor increase in training time.

References

Balcan, M.-F. and Blum, A. A pac-style model for learning from labeled and unlabeled data. In *International Conference on Computational Learning Theory*, pp. 111–126. Springer, 2005.

Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, 2002.

Ben-David, S., Lu, T., and Pál, D. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. In *COLT*, pp. 33–44, 2008.

Bengio, Y., Delalleau, O., and Le Roux, N. Label propagation and quadratic criterion. *MIT Press*, 2006.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, Nicolas Oliver, A., and Raffel, C. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

Chapelle, O. and Scholkopf, B. Semi-supervised learning. *MIT Press*, 2006.

Chawla, N. V. and Karakoulas, G. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

Coates, A. and Ng, A. Y. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Gammerman, A., Vovk, V., and Vapnik, V. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

Goodfellow, I. J., Courville, A., and Bengio, Y. Spike-and-slab sparse coding for unsupervised feature discovery. *NIPS Workshop on Challenges in Learning Hierarchical Models*, 2011.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, 2005.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training. In *International Conference on Machine Learning*, 2015.

Joachims, T. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

Joachims, T. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, 2003.

Kääriäinen, M. Generalization error bounds using unlabeled data. In *International Conference on Computational Learning Theory*, pp. 127–142. Springer, 2005.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semisupervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *ICML Workshop on Challenges in Representation Learning*, 2013.

Maas, Andrew L., H. A. Y. and Ng, A. Y. Rectifier non-linearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Niyogi, P. Manifold regularization and semi-supervised learning: Some theoretical analyses. *The Journal of Machine Learning Research*, 14(1):1229–1250, 2013.

Odena, A. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv preprint arXiv:1804.09170*, 2018.

Pu, Y., Zhe, G., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems*, 2016.

Rigollet, P. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, 8(Jul):1369–1392, 2007.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Li, F.-F. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2014.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

Sak, H., Senior, A., and Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

Salakhutdinov, R. and Hinton, Geoffrey, H. E. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in Neural Information Processing Systems*, 2007.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Sercu, T., Puhrsch, C., Kingsbury, B., and LeCun, Y. Very deep multilingual convolutional neural networks for LVCSR. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4955–4959. IEEE, 2016.

Singh, A., Nowak, R., and Zhu, J. Unlabeled data: Now it helps, now it doesn't. In *Advances in neural information processing systems*, pp. 1513–1520, 2009.

Smith, N. A. and Eisner, J. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 354–362. Association for Computational Linguistics, 2005.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, 2017.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Pas, D. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

Wasserman, L. and Lafferty, J. D. Statistical analysis of semi-supervised regression. In *Advances in Neural Information Processing Systems*, pp. 801–808, 2008.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

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

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003.