arXiv:1911.05166v1 [cs.LG] 12 Nov 2019

Negative sampling in semi-supervised learning

John Chen
Rice University

Vatsal Shah
University of Texas at Austin

Anastasios Kyrillidis
Rice University

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 MixMatch, 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.

1 Introduction

Deep learning has been hugely successful in areas such as image classification (Krizhevsky et al., 2012; He et al., 2016; Zagoruyko and 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.

This form of learning is semi-supervised learning (Chapelle and 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 and Aila, 2017; Miyato et al., 2017; Tarvainen and 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 and Blum, 2005; Ben-David et al., 2008; Kääriäinen, 2005; Niyogi, 2013; Rigollet, 2007; Singh et al., 2009; Wasserman and 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 and 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 and 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, simply by adding the NS^{3}L loss in their objective.

ii) Adding the NS^{3}L loss to the state-of-the-art—non-Mixup (Zhang et al., 2017)—loss for unlabeled data, i.e., Virtual Adversarial Training (VAT) (Miyato et al., 2017), we observe superior performance compared to state-of-the-art alternatives, such as Pseudo-Label (Lee, 2013), plain VAT (Miyato et al., 2017), and VAT with Entropy Minimization (Miyato et al., 2017; Oliver et al., 2018), for the standard SSL benchmarks of SVHN and CIFAR10.

iii) Adding the NS^{3}L loss to the state-of-the-art Mixup SSL, 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 Negative Sampling in Semi-Supervised Learning

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:

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

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:

$$\begin{aligned}
&\mathbb{P}\left[\{y_i\}_{i=1}^{n+n_u} \mid \{x_i\}_{i=1}^{n+n_u}, w\right] \\
&\qquad = \prod_{i=1}^{n+n_u} \mathbb{P}\left[y_i \mid x_i, w\right] \\
&\qquad = \prod_{i=1}^{n+n_u}\prod_{k=1}^{K} \mu_{ik}^{y_{ik}}, \\
&\qquad = \left(\prod_{i_1=1}^{n}\prod_{k=1}^{K} \mu_{i_1k}^{y_{i_1k}}\right)\cdot\left(\prod_{i_2=1}^{n_u}\prod_{k=1}^{K} \mu_{i_2k}^{y_{i_2k}^u}\right)
\end{aligned}$$

Observe that,

$$\prod_{k=1}^{K} \mu_{i_2k}^{y_{i_2k}^u} = 1 - \sum_{j:y_{i_2j}\neq 1} \mu_{i_2j},$$

which follows from the definition of the quantities $\mu_{:}$ 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 unsupervised part. The log-likelihood loss function can now be written as follows:

$$\begin{aligned}
\mathcal{L}\left(\{x_i, y_i\}_{i=1}^{n+n_u}\right) = &\underbrace{-\tfrac{1}{n}\sum_{i_1=1}^{n}\sum_{k=1}^{K} y_{ik}\log\mu_{ik}}_{:=\text{supervised part}} \\
&\underbrace{-\tfrac{1}{n_u}\sum_{i_2=1}^{n_u}\log\left(1 - \sum_{j\neq\text{True label}} \mu_{i_2j}\right)}_{:=\text{unsupervised part}}
\end{aligned}$$

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 and Eisner, 2005) in our approach. I.e., for each unlabeled example in the unsupervised part, we randomly assign P labels from the set of labels. 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, with the hope that the correct label is not picked. Our idea is analogous to the `word2vec` setting (Mikolov et al., 2013), which is described in Appendix A.

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 NS^{3}L loss, where each unlabeled sample is connected with P hopefully incorrect labels, is approximated as:

$$\frac{1}{n_u}\sum_{i_2=1}^{n_u}\log\left(1-\sum_{j\neq\text{True label}}\mu_{i_2 j}\right) \approx \frac{1}{|B_2|}\sum_{i_2\in B_2}\log\left(1-\sum_{j=1}^{P}\mu_{i_2 j}\right)$$

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

$$\hat{\mathcal{L}}_{B_1,B_2}\left(\{x_i,y_i\}_{i=1}^{n+n_u}\right) = -\frac{1}{|B_1|}\sum_{i_1\in B_1}\sum_{k=1}^{K} y_{ik}\log\mu_{ik} \underbrace{-\frac{1}{|B_2|}\sum_{i_2\in B_2}\log\left(1-\sum_{j=1}^{P}\mu_{i_2 j}\right)}_{:=\text{NS}^3\text{L loss}}.$$

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, as the output of the classifier or otherwise, is given in Algorithm 1.

Algorithm 1 NS^{3}L

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: $\mathcal{L}_{\texttt{NS}^3\texttt{L}} = 0$.
3: **for** $b = 1,\ldots,B$ **do**
4: $\quad \mathbb{1}_{\hat{y}'_b} = \texttt{isTrue}(\hat{y}_b < T)$.
5: $\quad \mathcal{L}_{\texttt{NS}^3\texttt{L}} = \mathcal{L}_{\texttt{NS}^3\texttt{L}} - \log\left(1-\sum_{k=1}^{K}\mathbb{1}_{\hat{y}'_{bk}}\mu_{bk}\right)$.
6: **end for**
7: **return** $\frac{1}{B}\mathcal{L}_{\texttt{NS}^3\texttt{L}}$

3 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, 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 and Niyogi, 2002; Salakhutdinov and Hinton, 2007; Coates and 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 and Scholkopf (2006), or Zhu et al. (2003). We describe below the relevant categories of SSL in this paper.

3.1 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 and 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 and 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 the distance function $d(f_\theta(x), f_{\theta'}(x))$, where θ' is an exponential moving average of θ, to the supervised classification loss with some weight. However, these methods are domain specific.

3.1.1 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:

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

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.

3.2 Entropy minimization

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

3.3 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.

3.4 Mixup based SSL

Mixup (Zhang et al., 2017) combines pairs of samples and their one-hot labels $(x_1, y_1), (x_2, y_2)$ with the following operations

$$
\begin{aligned}
\lambda &\sim \texttt{Beta}(\alpha, \alpha), \\
x' &= \lambda x_1 + (1-\lambda) x_2, \\
y' &= \lambda y_1 + (1-\lambda) y_2,
\end{aligned}
$$

to produce a new sample (x', y') where α is a hyperparameter. Mixup is a form of regularization which encourages the neural network to behave linearly between training examples, justified by Occam's Razor. 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

$$
\begin{aligned}
\mathcal{X}', \mathcal{U}' &= \texttt{MixMatch}(\mathcal{X}, \mathcal{U}, E, A, \alpha), \\
\mathcal{L}_{\text{supervised}} &= \frac{1}{|\mathcal{X}'|} \sum_{i_1 \in \mathcal{X}'} \sum_{k=1}^{K} y_{i_1 k} \log \mu_{i_1 k}, \\
\mathcal{L}_{\text{unsupervised}} &= \frac{1}{K|\mathcal{U}'|} \sum_{i_2 \in \mathcal{U}'} \sum_{k=1}^{K} (y_{i_2 k} - \mu_{i_2 k})^2, \\
\mathcal{L} &= \mathcal{L}_{\text{supervised}} + \lambda_3 \mathcal{L}_{\text{ubsupervised}},
\end{aligned}
$$

where $\mathcal{X}$ is the labeled data $\{x_{i_1}, y_{i_1}\}_{i_1=1}^{n}$, $\mathcal{U}$ is the unlabeled data $\{x^u_{i_2}\}_{i_2=1}^{n_u}$, $\mathcal{X}'$ 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,

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

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

$$\begin{aligned}
\lambda &\sim \texttt{Beta}(\alpha, \alpha),\\
\lambda' &= \max(\lambda, 1-\lambda),\\
x' &= \lambda' x_1 + (1-\lambda')x_2,\\
y' &= \lambda' y_1 + (1-\lambda')y_2,
\end{aligned}$$

and 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 NS^{3}L.

4 Experiments

We separate experiments into non-Mixup-based SSL and Mixup-based SSL. For reproducibility, we follow the methodology of Oliver et al. (2018) almost exactly for our non-Mixup-based SSL experiments, and reproduce many of the key findings. In that work, they compare non-Mixup-based SSL methods and, thus, we use their consistent testbed for the same purpose. MixMatch uses an almost identical setup, but with a slightly different evaluation method, and we use the official implementation for MixMatch for our Mixup-based experiments.

4.1 Non-Mixup-based SSL

Following Oliver et al. (2018), the model employed is the standard Wide ResNet (WRN) (Zagoruyko and Komodakis, 2016) with depth 28 and width 2, batch normalization (Ioffe and Szegedy, 2015), and leaky ReLU activations (Maas and Ng, 2013). The optimizer is the Adam optimizer (Kingma and 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 training data/validation 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 almost the exact 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^{3}L and NS^{3}L + 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 and Valpola, 2017) to 50,000, and iteration of learning rate decay to 130,000. All other methodology follows that work (Oliver et al., 2018).

4.1.1 Baseline Methods

For baseline methods, we consider Pseudo-Labeling, due to its simplicity on the level of NS^{3}L, and VAT for its performance, in addition to VAT + Entropy Minimization. We omit the Π model and Mean Teacher, since 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.

4.1.2 Implementation of NS^{3}L

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

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

The performance of NS^{3}L with random negative sampling assignment or Nearest Neighbor-based assignment is given in Section 5. We label a sample with negative labels for the classes whose probability value falls below a certain threshold. We then simply add the NS^{3}L loss to the existing SSL loss function. Using NS^{3}L on its own gives

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

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

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

for some weighting $\lambda_i, i \in \{1, 2\}$. The weighting is a common practice in SSL, also used in MixMatch. 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.

4.1.3 Results

We follow the practice in Oliver et al. (2018) and use the same hyperparameters for plain NS^{3}L and NS^{3}L in 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.

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. For NS^{3}L, we use a threshold $T = 0.04$, learning rate of 6e-4, and $\lambda_1 = 1$. 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. All other settings remain as is optimized individually.

We created 5 splits of 4,000 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. 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. We use the same hyperparameters for NS^{3}L and VAT + NS^{3}L as in CIFAR10.

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

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. *In our experiments, NS^{3}L alone runs more than 2x faster than VAT.*

4.2 Mixup-based SSL

We follow the methodology of Berthelot et al. (2019) and continue to use the same model described in section 4.1. In the previous section, we use the standard training data/validation data split for SVHN and CIFAR10, with all but 1,000 labels and all but 4,000 labels discarded respectively. Since the performance of MixMatch is particularly strong using only a small number of labeled samples, we include experiments for SVHN with all but 250 labels discarded, and CIFAR10 with all but 250 labels discarded. 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.

Dataset	Supervised	Pseudo-Label	VAT	VAT + EntMin	NS^{3}L	VAT + NS^{3}L
CIFAR10	20.76 ± .28	17.56 ± .29	14.72 ± .23	14.34 ± .18	16.03 ± .05	**13.94** ± .10
SVHN	12.39 ± .53	7.70 ± .22	6.20 ± .11	6.10 ± .02	6.52 ± .22	**5.51** ± .14
CIFAR100	48.26 ± .25	46.91 ± .31	44.38 ± .56	43.92 ± .44	46.34 ± .37	**43.70** ± .19

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 and EntMin refers to Virtual Adversarial Training and Entropy Minimization respectively.

Hyperparameters are tuned individually for each dataset, and the median of the last 20 checkpoints' test error is reported, following Berthelot et al. (2019). We run 5 seeds.

Again, we reduce training epochs to 300 epochs for both SVHN and CIFAR10, which is a typical training time for fully supervised models. We reduce the training epochs of STL10 significantly in interest of training time. All other methodology follows the work of MixMatch. We note here that 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.

4.2.1 Baseline Methods

We run MixMatch with the official implementation, and use the parameters recommended in the original work for each dataset.

4.2.2 Implementation of NS^{3}L

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,

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

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

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

$$\mathcal{L}_{\texttt{NS}^3\texttt{L}} = \texttt{NS}^3\texttt{L}(\mathcal{X}' \bigcup \mathcal{U}', T)$$

$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \lambda_3 \mathcal{L}_{\text{unsupervised}} + \lambda_1 \mathcal{L}_{\texttt{NS}^3\texttt{L}}$$

4.2.3 Results

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, and 250 labeled samples and 44,750 unlabeled samples, where MixMatch performs much stronger than other SSL methods. The results are given in Table 2.

As in Berthelot et al. (2019), we use $\alpha = 0.75$ and $\lambda_3 = 75$. For NS^{3}L, 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.

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.

Similar to the previous section, we 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.

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

Table 2: 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.

SVHN. We evaluate the accuracy of each method with 1,000 labeled samples and 64,932 unlabeled samples, as is standard practice, and 250 labeled samples and 65,682 unlabeled samples. The results are shown

in Table 3.

Following the literature, we use $\alpha = 0.75$ and $\lambda_3 = 250$. For NS^{3}L, 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.

We created 5 splits with 5 different seeds, where each model is trained on a different split and test error is reported with mean and standard deviation.

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.

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

Table 3: 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.

STL10. We evaluate the accuracy of each method with 1,000 labeled samples and 100,000 unlabeled samples. The results are given in Table 4.

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.

STL10	1,000
MixMatch	22.20 ± 0.89
Mixmatch + NS^{3}L	**21.74** ± 0.33

Table 4: Test errors achieved by MixMatch and MixMatch + NS^{3}L on the standard benchmark of STL10, with all but 1,000 labels removed.

5 Alternative NS^{3}L methods

With computational efficiency in mind, we compare several methods of implementing NS^{3}L in Table 5 on the F-MNIST dataset with a small Convolutional Neural Network. We split the F-MNIST dataset into a 2,000/58,000 labeled/unlabeled split and report validation error at the end of training. Specifically, we compare:

- `Supervised:` trained only on the 2,000 labeled samples.
- `Uniform:` negative labels are selected uniformly over all classes.
- `NN:` We use the Nearest Neighbor (NN) method to the exclude the class of the NN, exclude four classes with the NNs, or to label with the class with the furthest NN.
- `Threshold:` refers to the method of section 4.1.2
- `Oracle:` negative labels are selected uniformly over all wrong classes.

F-MNIST	2,000
Supervised	17.25 ± .22
Uniform - 1	18.64 ± .38
Uniform - 3	19.35 ± .33
Exclude class of NN - 1	17.12 ± .15
Exclude 4 nearest classes with NN - 1	17.13 ± .21
Furthest class with NN - 1	16.76 ± .15
Threshold $T = 0.03$	16.47 ± .18
Threshold $T = 0.05$	16.59 ± .19
Oracle - 1	16.37 ± .12
Oracle - 3	15.20 ± .66

Table 5: Test error achieved by various NS^{3}L techniques on F-MNIST with all but 2,000 labels removed. We use a small CNN trained for 50 epochs. Where applicable, the number after the dash indicates the number of negative labels per sample selected.

Selecting negative labels uniformly over all classes appears to hurt performance, suggesting that negative labels must be selected more carefully in the classification setting. `NN` methods appear to improve over purely supervised training, however the effectiveness is limited by long preprocessing times and the high dimensionality of the data.

The method described in section 4.1.2, listed here as `Threshold`, achieves superior test error in comparison to `NN` and `Uniform` methods. In particular, it is competitive with `Oracle - 1`, an oracle which labels each unlabeled sample with one negative label which the sample is not a class of.

It is no surprise that `Oracle - 3` improves substantially over `Oracle - 1`, and it is not inconceivable to develop methods which can accurately select a small number of negative labels, and these may lead to even better results when combined with other SSL methods.

We stress that this is not a definitive list of methods to implement negative sampling in SSL, and our fast

proposed method, when combined with other SSL, already improves over the state-of-the-art.

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

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

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

Shai Ben-David, Tyler Lu, and Dávid Pál. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. In *COLT*, pages 33–44, 2008.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. *MIT Press*, 2006.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Avital Papernot, Nicolas Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

Olivier Chapelle and Bernhard Scholkopf. Semi-supervised learning. *MIT Press*, 2006.

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

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.

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

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

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

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

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

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

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

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

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

Diederik P Kingma, Shakir Mohamed, Danilo Jimenes Rezende, and Max Welling. Semisupervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014.

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

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.

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

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neu-

ral networks. *ICML Workshop on Challenges in Representation Learning*, 2013.

Hannun Awni Y. Maas, Andrew L. and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffery Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

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

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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

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

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv preprint arXiv:1804.09170*, 2018.

Yunchen Pu, Gan Zhe, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems*, 2016.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2014.

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*, 2016.

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

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alex Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for LVCSR. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4955–4959. IEEE, 2016.

Aarti Singh, Robert Nowak, and Jerry Zhu. Unlabeled data: Now it helps, now it doesn't. In *Advances in neural information processing systems*, pages 1513–1520, 2009.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

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*, 2017.

Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Pas. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

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

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Pas. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

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

A Negative Sampling

We present the case of `word2vec` for negative sampling where the number of words and contexts is such that picking a random pair of (word, context) is with high probability not related. To make the resemblance, let us describe the intuition behind `word2vec`. Here, the task is to relate words –represented as w– with contexts –represented as c. We can theoretically conceptualize words w being related with x, and contexts being related to labels y. The negative sampling by Mikolov et al., considers the following objective function: consider a pair (w, c) of a word and a context. If this pair comes from valid data that correctly connects these two, then we can say that the data pair (w, c) came from the true data distribution; if this pair does otherwise, then we claim that (w, c) does not come from the true distribution.

In math, we will denote by $\mathbb{P}[D = 1 \mid w, c]$ as the probability that (w, c) satisfies the first case, and $\mathbb{P}[D = 0 \mid w, c]$ otherwise. The paper models these probabilities as:

$$\mathbb{P}[D = 1 \mid w, c] = \frac{1}{1 + e^{-v_c^\top v_w}},$$

where v_c, v_w correspond to the vector representation of the context and word, respectively.

Now, in order to find good vector representations $\theta := \{v_c, v_w\}$ (we naively group all variables into θ), given the data, we perform maximum log-likelihood as follows:

$$\begin{aligned}
\theta &= \arg\max_\theta \left(\prod_{(w,c)\in\mathcal{D}} \mathbb{P}[D = 1 \mid w, c, \theta] \right) \cdot \left(\prod_{(w,c)\notin\mathcal{D}} \mathbb{P}[D = 0 \mid w, c, \theta] \right) \\
&= \arg\max_\theta \left(\prod_{(w,c)\in\mathcal{D}} \mathbb{P}[D = 1 \mid w, c, \theta] \right) \cdot \left(\prod_{(w,c)\notin\mathcal{D}} (1 - \mathbb{P}[D = 1 \mid w, c, \theta]) \right) \\
&= \arg\max_\theta \left(\sum_{(w,c)\in\mathcal{D}} \log(\mathbb{P}[D = 1 \mid w, c, \theta]) \right) + \left(\sum_{(w,c)\notin\mathcal{D}} \log(1 - \mathbb{P}[D = 1 \mid w, c, \theta]) \right) \\
&= \arg\max_\theta \left(\sum_{(w,c)\in\mathcal{D}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}} \right) \right) + \left(\sum_{(w,c)\notin\mathcal{D}} \log\left(\frac{1}{1 + e^{v_c^\top v_w}} \right) \right)
\end{aligned}$$

Of course, we never take the whole dataset (whole corpus $\mathcal{D}$) and do gradient descent; rather we perform SGD by considering only a subset of the data for the first term:

$$\sum_{(w,c)\in\mathcal{D}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}} \right) \approx \sum_{(w,c)\in\text{mini-batch}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}} \right);$$

Also, we cannot consider *every* data point not in the dataset; rather, we perform *negative sampling* by selecting random pairs (according to some probability - this is important)—say P pairs:

$$\left(\sum_{(w,c)\notin\mathcal{D}} \log\left(\frac{1}{1 + e^{v_c^\top v_w}} \right) \right) \approx \sum_{p=1}^{P} \log\left(\frac{1}{1 + e^{-\tilde{v}_c^\top \tilde{v}_w}} \right),$$

where the tildes represent the "non-valid" data.