Semi-Supervised Learning with Uncertainty

Kazuki Yoshiyama  
Graduate School of Science for Open and Environmental System, Keio University  
rkzfilter@gmail.com

Akito Sakurai  
Graduate School of Science for Open and Environmental System, Keio University  
sakurai@ae.keio.ac.jp

**keywords:** deep learning, neural network, semi-supervised learning (SSL), unsupervised learning, uncertainty.

**Summary**

The goal of semi-supervised learning is to utilize many unlabeled samples under a situation where a few labeled samples exist. Recently, researches of semi-supervised learning are evolving with deep learning technology development, because, in deep, models have powerful representation to make use of abundant unlabeled samples. In this paper, we propose a novel semi-supervised learning method with uncertainty. It naturally extends the consistency loss under the uncertainty and propose suitable regularizations for the uncertainty. Using two datasets CIFAR-10 and SVHN and with various experiments, we empirically demonstrate that the proposed method achieves competitive or higher performance in accuracy when compared to semi-supervised learning with the conventional consistency loss while our proposal can let a model generalize much faster.

1. Introduction

Many semi-supervised learning (SSL) methods have been proposed in the past decades, most of which extended existing supervised methods such as Support Vector Machine and Logistic Regression to semi-supervised ones by incorporating unlabeled samples from various perspectives [Zhu 09]: low-density separation [Joachims 99, Chapelle 05, Grandvalet 05], manifold structure [Zhu 03, Belkin 06], and generative model [Nigam 00, Castelli 95]. These methods formulate SSL as an optimization problem and solve it by proposed efficient algorithms [Xu 07, Melacci 11, Joulin 12]. [Farajtabar 11, Orbach 12, Su 11, Subramanya 11, Syed 10, Wang 13, Yoshiyama 11, Zhu 14].

Recently, deep neural networks has become popular after the great success in ImageNet Large Scale Visual Recognition Challenge [He 15, Krizhevsky 12, Szegedy 14]. The successive works in this challenge were [Szegedy 14, He 15, He 16]. These successes were not limited to the image classification problem, but also found in such as speech recognition [Amodei 15, Hannun 14, Oord 16, Sak 15] and machine translation [Bahdanau 14, Dauphin 16, Gehring 17, Kalchbrenner 16, Wu 16].

Because of the successful resurgence of neural networks, research of semi-supervised learning in the context of deep neural networks also becomes popular. Since deep neural networks have strong potential for the powerful representation from many of unlabeled samples, many semi-supervised learning methods fused with deep neural network were proposed in various perspectives, such as embeddings, generative models, using adversarial samples [Goodfellow 14a, auto-encoder [Miyato 15], graph [Miyato 16], adversarial learning [Denton 16, Goodfellow 14b, Li 17, Radford 15, Salimans 16, Springenberg 15], network architecture [Pezeshki 15, Rasmus 15, Zhao 16], loss functions [Laine 16, Sajjadi, Sajjadi 16, Tarvainen 17].

Consistency loss in [Sajjadi], that penalizes a neural network if it predicts the same unlabeled data inconsistently given different noise, e.g., dropout masks, is not only simple to implement in the existing deep learning libraries but also shows better performance in semi-supervised settings where labeled samples are scarce but unlabeled samples are abundant.

In this work, we propose a semi-supervised learning method with uncertainty, which focus on stochasticity in semi-supervised learning methods in conjunction with deep neural network and naturally integrates a recently proposed concept uncertainty [Gal 15, Gal 16, Kendall 17a, Kendall 17b]. By incorporating the uncertainty to the consistency loss, the learning can be accelerated and stabilized when compared to using only consistency loss, and the proposed semi-supervised learning method with uncertainty improves the performance in accuracy and generalizes faster.

Key contributions of this paper are:

1) proposing a novel loss function called consistency
loss with uncertainty and corresponding regularizations through uncertainty,
(2) demonstrating the competitive or higher performance of consistency loss with uncertainty in two datasets SVHN and CIFAR-10,
(3) demonstrating faster generalization.

The remainder of this paper is organized as follows. In Chapter 2, we mention the related works to semi-supervised learning in a sense of deep neural network. In Chapter 3, we summarize the closely related methods to our proposed one, then we formulate semi-supervised learning method with uncertainty in Chapter 4. In Chapter 5, we show and compare the proposed method with existing semi-supervised methods. We conclude in Chapter 6, and Chapter 7 is for discussion.

2. Related Works

There are many semi-supervised learning methods proposed recently, and many of which were proposed in context of deep learning. In this chapter, we summarize such methods as related works.

2.1 Auto Encoder

Variational Auto Encoder (VAE) was originally proposed in [Kingma 13]. It introduces a prior distribution over the latent variables, which is inferred by a neural network. [Kingma 14b] extended VAE to semi-supervised learning by placing a classifier on the top of VAE and integrating unobserved labels over the class of any unlabeled data during an inference process. In [Maaløe 16], they proposed Auxiliary Deep Generative Model (ADGM). Auxiliary variables introduced make the variational distribution more expressive without changing the generative model itself. These works will be used for comparison with our proposed method in Chapter 5.

Networks of ladder architecture [Pezeshki 15, Rasmus 15, Zhao 16] were proposed as an auto encoder and used for semi-supervised learning with a few labeled samples. In [Rasmus 15], they used Denoising Auto Encoder (dAE) [Vincent 08] with a clean classifier, i.e., an encoder without noise. When in training, it adds noise to the encoder and the decoder tries to denoise the noisy input, and put cost functions between the clean classifier and the decoder. [Pezeshki 15] investigated what type of denoising functions were suitable in the ladder network and [Zhao 16] imposed a constraint that the pooling and unpooling indices in ladder architecture be the same. These works relate to ours in adding noise as randomness to a network, but they do not consider uncertainty contrary to our work.

2.2 Generative Adversarial Network

In [Goodfellow 14b], they invented Generative Adversarial Nets (GAN) where a generator network tries to produce fake samples which resemble samples drawn from the true data distribution. On the other hand, a discriminator network classifies the fake samples from true samples. This is cast as the min-max optimization problem where two players play a minimax game.

GAN itself is categorized as unsupervised learning, but there are many works to extend GAN to semi-supervised learning. In [Springenberg 15], they proposed a categorical GAN (CatGAN) which requires a discriminator network to classify samples to a certain class and applies the maximum entropy principle to outputs of the discriminator network, which makes it to be a semi-supervised learning method. [Salimans 16] did not only propose some techniques to improve GAN but also a semi-supervised learning method in GAN architecture. It is similar to CatGAN but it lets the discriminator output $C + 1$ classes where $C$ is the number of the classes and the $C + 1$st output designates if the input is real or fake. In [Denton 16], they proposed a context-conditional GAN on top of [Salimans 16], which first makes a mask on an image and a generator tries to produce an un-masked original image conditioned on the low-resolution image. Lastly, Triple GAN proposed in [Li 17] incorporates a prediction by a classifier network into GAN framework. The conventional GAN framework considers just generated fake samples, but in their Triple GAN, it also passes a predicted class into the discriminator in addition to a generated sample, so that the discriminator discriminates the pair of a generated sample and a predicted class from that of real ones. In their work, there are three type of networks: the discriminator, the generator, and the classifier, thus it is called Triple GAN. These works will be compared to ours in Chapter 5.

2.3 Adversarial Examples

Adversarial examples were first discovered by [Goodfellow 14a] and are samples perturbed by gradients of a loss function with respect to a sample itself multiplied by very small positive number. Interestingly, a well-trained classifier inadvertently classifies these perturbed samples into a dissimilar class although human cannot classify it in such a way. In [Miyato 15], they utilized this property to make a classifier robust and extended it to a semi-supervised learning method called Virtual Adversarial Training (VAT). This method relies on dense inputs, such as image, and is not applicable to sparse inputs, for example, text and categorical data. [Miyato 16] pushed the limits beyond such sparse inputs by perturbing the indeterminate
output of a neural network, such as outputs after an embedding layer, using small gradients with respect to those outputs. These works relate to our methods in a sense that perturbation introduced by adversarial examples is a kind of the transformation on input samples, although we did not use adversarial samples as one of perturbations.

2.4 Unsupervised Loss

Some loss functions were also proposed to harness unlabeled samples. [Sajjadi 16] proposed a mutual exclusivity loss which forces just one element of output vector to be one and others to be zero. It has the same property as minimum entropy regularization [Grandvalet 05]. [Sajjadi] first introduced consistency loss with which two stochastic neural networks try to predict the same class outputs. These works relate to our methods in a sense that the neural network should be stable against random transformations upon the same input samples and predict the same outputs.

3. Background

In this chapter, first we introduce the notations and definitions used through this paper then briefly summarize the existing semi-supervised learning methods related to our work and uncertainty in neural networks.

3.1 Notation and Definition

Given a labeled dataset $D = \{(x_i, y_i)\}_{i=1}^n$ where $x$ is a sample, $y$ is a label, and $n$ is the number of the labeled samples, a neural network is denoted by $f(x)$ and the output of this $f$ is denoted as $\hat{y} := f(x)$. $f$ has parameters $\theta = \{\theta_l\}_{l=1}^L$ at each depth $l$. The dimensions of $x$ and $y$ depend on the tasks at one’s hand, but we basically address an image $x \in \mathbb{R}^{n \times h \times w}$ where $n$ is the number of channels (usually 3), $h$ is the height, and $w$ is the width. When we assume a task is a classification, $y \in \mathbb{R}^C$ is the one-hot vector where a single element corresponding to a class is one and zero otherwise, and $C$ is the number of classes. In the supervised learning setting where all samples have the corresponding labels, the loss function we use is the softmax cross entropy and is defined as

$$\ell_{SCE}(y, \hat{y}) = -\sum_{c=1}^C y_c \log \hat{y}_c, \quad (1)$$

$$p_c = \frac{\exp(\hat{y}_c)}{\sum_{c=1}^C \exp(\hat{y}_c)}. \quad (2)$$

However, in the semi-supervised learning, we have a small amount of the labeled samples $D_L = \{(x_i^L, y_i)\}_{i=1}^L$ and many unlabeled samples $D_U = \{(x_i^U)_{i=\ell+1}^{\ell+u}\}$ where $\ell$ is the number of labeled samples and $u$ the number of unlabeled samples, and we generally assumes $\ell \ll u$.

3.2 Consistency Loss

Consistency Loss was originally proposed in [Sajjadi]. In their framework, they impose transformation stability on a neural network and define as,

$$\ell_{cl}(\hat{y}^{(1)}, \hat{y}^{(2)}) = \frac{1}{C} \|f^{(1)}(T(x^U)) - f^{(2)}(T(x^U))\|_2^2. \quad (3)$$

where $T$ is some transformation on an image, such as random rotation, random horizontal flipping, and random pixel translation, and $f^{(1)}$ and $f^{(2)}$ is the same neural network but have different stochasticity which is introduced by normalization layers, such as a simple Gaussian noise layer to add noise and/or dropout [Hinton 12] to make the intermediate hidden output sparse by randomly omitting features. The intuition behind the consistency loss is that the neural network should be stable against random transformations upon the same input samples and predict the same outputs.

3.3 Entropy Minimization

In [Grandvalet 05], they proposed entropy minimization. It enforces a decision boundary of $f$ to be confined in regions where there are a few samples because they assume the decision boundary should lie in a low-density regions. Mathematically, it lets the output $\hat{y}$ be the one-hot vector and is defined as,

$$\min_{\hat{y}} -\frac{1}{C} \sum_{c=1}^C p_c \log p_c, \quad (4)$$

where $p_c$ is the same as Eq.2.

3.4 Uncertainty in Neural Network

Recently, in [Gal 15, Gal 16, Kendall 17a, Kendall 17b], they introduced uncertainty into neural networks. There are two type of uncertainties called epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty is modeled by placing a prior distribution over a model weights, and the aleatoric uncertainty on the other hand is modeled by placing a distribution over the output of the model [Kendall 17a].

In epistemic uncertainty, they put a prior distribution over parameters of a neural network, for example, using Gaussian prior,

$$W \sim \mathcal{N}(0, I), \quad (5)$$

where $W$ is a weight matrix of a neural network, $I$ is the identity matrix, and $\mathcal{N}$ is Gaussian distribution which has 0 mean and unit variance. This modeling is also referred
to as a Bayesian Neural Network (BNN), BNN is easy to formulate but difficult to implement. Also it is difficult to perform exact inferences because we have to compute the posterior distribution over parameters conditioned on data, thus an approximation is necessary. Dropout variational inference is a practical approach for approximate inference in large and complex model such as a neural network [Gal 15, Kendall 17a]. This inference is very simple to implement because we just need to add dropout before every weight layer.

When considering epistemic uncertainty in context of consistency loss, a neural network itself has epistemic uncertainty, i.e., the output of the neural network for the same sample is not deterministic because we add transformations, the dropout, and/or some noises to the neural network.

The aleatoric uncertainty is modeled by placing a distribution over the output of a model,

\[ \hat{y}_c \sim N(\mu_c, \sigma_c), \]

where \( \mu_c \) and \( \sigma_c \) are the mean and variance of Gaussian distribution for each element of an output.

To capture aleatoric uncertainty in regression, we would have to tune the observation noise parameter. There are two types of aleatoric uncertainties: homoscedastic and heteroscedastic. Homoscedastic regression assumes constant observation noise for every inputs. Heteroscedastic regression, on the other hand, assumes that observation noise can vary with input. In non-Bayesian neural networks, this observation noise parameter is often fixed as part of the model \[ k \] weight decay, and ignored. However, when made data-dependent, it can be learned as a function of the data [Kendall 17a].

In order to infer the posterior distribution for a BNN, we let a neural network predict a mean and variance for the posterior distribution. Thus, \( \mu_c \) and \( \sigma_c \) are the outputs of a single neural network \( f \) with its head split which means the output of the second last layer is branched and it is fed into the two different convolutions (Figure 1 and Table 1). When considering Gaussian likelihood, the aleatoric uncertainty is defined as,

\[ \ell(y, f(x)) = \frac{1}{C} \sum_{c=1}^{C} \left( \frac{(y_c - \mu_c)^2}{2\sigma_c^2} + \log \sigma_c^2 \right). \]

\[ \ell = \frac{1}{C} \sum_{c=1}^{C} \left( \frac{(y_c - \mu_c)^2}{2\sigma_c^2} + \log \sigma_c^2 \right). \]

4. Proposed Method

In this chapter, we naturally integrate uncertainty to semi-supervised learning, and formulate our proposed method.

The network overview used in our proposed method are illustrated in Figure 1, and our final objective are formulated in Eq.13. Here we derive it step-by-step.

\[ x^l \rightarrow T(x^l) \rightarrow f \rightarrow \hat{y} \rightarrow \ell_{sKL}(y;\hat{y}) \]

\[ x^u \rightarrow T^0(x^u) \rightarrow f^{(1)} \rightarrow \hat{y}^{(1)} \rightarrow \ell_{sKL}(y;\hat{y}) \]

\[ x^u \rightarrow T^0(x^u) \rightarrow f^{(2)} \rightarrow \hat{y}^{(2)} \rightarrow \ell_{sKL}(y;\hat{y}) \]

Fig. 1: Network overview. \( x^l \) is a labeled sample, and \( x^u \) is an unlabeled sample. \( T, T^{(1)}, \) and \( T^{(2)} \) are random transformations. \( f, f^{(1)}, \) and \( f^{(2)} \) are neural networks sharing parameters and its head is split. \( y \) is a label, and \( \hat{y} \) is an output of the network. \( \mu^{(1)}, \sigma^{(1)}, \mu^{(2)}, \) and \( \sigma^{(2)} \) are the network outputs but treated as means and standard deviations. These outputs are fed into the corresponding losses, the softmax cross entropy \( \ell_{SCE} \), and the symmetric KL divergence \( \ell_{s-KLD} \).

Consistency loss Eq.3 has inherently stochastic since a neural network \( f \) has dropout layers, i.e., it has the property of the epistemic uncertainty in inference in the training phase. Thus what we have to consider is the aleatoric uncertainty. Because it imposes a distribution over outputs of a neural network, we place Gaussian distribution over \( \hat{y}^{(1)} \) and \( \hat{y}^{(2)} \),

\[ \hat{y}^{(1)}_c \sim N(\mu^{(1)}_c, \sigma^{(1)}_c), \hspace{0.5cm} \hat{y}^{(2)}_c \sim N(\mu^{(2)}_c, \sigma^{(2)}_c). \]

In consistency loss, we measure consistency in Euclidean space as the mean squared error; however, here we measure consistency over the two Gaussian distributions since two Gaussian distributions are placed as in Eq.8. One natural way to measure the similarity over two distributions is symmetric KL-divergence (s-KLD). s-KLD between the two distributions Eq.8 is computed as,

\[ \ell_{s-KLD}(N(\mu^{(1)}_c, \sigma^{(1)}_c), N(\mu^{(2)}_c, \sigma^{(2)}_c)) \]

\[ = \frac{\log \sigma^{(2)}_c}{\sigma^{(1)}_c} + \frac{(\sigma^{(1)}_c)^2}{2(\sigma^{(2)}_c)^2} + \frac{(\mu^{(1)}_c - \mu^{(2)}_c)^2}{2(\sigma^{(2)}_c)^2} - \frac{1}{2} \]

\[ \log \frac{\sigma^{(1)}_c}{\sigma^{(2)}_c} + \frac{(\sigma^{(2)}_c)^2}{2(\sigma^{(1)}_c)^2} + \frac{(\mu^{(2)}_c - \mu^{(1)}_c)^2}{2(\sigma^{(1)}_c)^2} - \frac{1}{2} \]
we call this loss as consistency loss with uncertainty. In the second and forth terms of Eq. 10, the numerators can be considered as the simple consistency loss but scaled observation noises which were introduced when taking the epistemic and heteroscedastic aleatoric uncertainties into account.

When we compare Eq.7, Eq.9, and Eq.10, in Eq.7, $\log \sigma^2_c$ works as a regularization to prevent $\sigma_c$ from becoming too large, but in Eq.9, such regularizations appear as $\log \sigma^{(2)}_c$ and $\log \sigma^{(2)}_c$ in part. However, we consider s-KLD, thus in Eq.10, $\log \sigma^{(2)}_c$ and $\log \sigma^{(2)}_c$ cancel out each other; instead, $\frac{\sigma^{(2)}_c}{2 \sigma^{(2)}_c}$ and $\frac{\sigma^{(2)}_c}{2 \sigma^{(2)}_c}$ appear. These two terms alone do not work as regularization to prevent $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ from becoming too large. Therefore, we have to add another regularization.

In consistency loss Eq.3, we expect a neural network to predict the same output even if it has some stochasticity and there are some random transformation on an input sample, we follow and apply this to the observation noise and constraint $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ to be the same values. This derives the following loss,

$$\min_y \left( \sigma_c^{(1)} - \sigma_c^{(2)} \right)^2.$$  \hspace{1cm} (11)

We call this uncertainty consistency loss. When the training of a neural network continues, eventually $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ become the same, meaning that heteroscedastic aleatoric uncertainty introduced in this proposed methods will become homoscedastic aleatoric uncertainty at the end.

However, this uncertainty consistency loss alone is not enough to obtain a meaningful solution in Eq.10 since if Eq.11 holds, it could be possible that we let $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ be too large while minimizing Eq.10 to 0. Thus, we regularize $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ to be constant. It is natural that both $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ are one, because when we compare Eq.3 and Eq.10, s-KLD results in Eq.3 but is multiplied by $\frac{1}{2}$ if both $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ are one. We introduce, what we call, the unit variance regularization in consistency loss with uncertainty.

$$\min_y \left( \sigma_c^{(1)} - 1 \right)^2 + \left( \sigma_c^{(2)} - 1 \right)^2.$$  \hspace{1cm} (12)

Using both uncertainty consistency loss Eq.11 and unit variance regularizations Eq.12 forces the objective to resemble the consistency loss Eq.3. However during the training process, in semi-supervised learning with uncertainty, a neural network has to consider uncertainty. If there are uncertainty in a prediction, the variances scale the squared error terms in Eq.10 and if the variance is unit, it does not scale; s-KLD Eq.10 reduces to the consistency loss Eq.3 multiplied by $\frac{1}{2}$.

The final objective function to be minimized is

$$\mathcal{L}(x; y; x^U; \theta) = -\sum_{c=1}^C y_c \log p_c + \frac{\lambda_c}{C} \sum_{c=1}^C \left( \frac{(\sigma_c^{(1)})^2}{2(\sigma_c^{(2)})^2} + \frac{(\sigma_c^{(1)} - \mu_c^{(1)})^2}{2(\sigma_c^{(1)})^2} + \frac{(\sigma_c^{(2)})^2}{2(\sigma_c^{(2)})^2} + \frac{(\mu_c^{(2)} - \mu_c^{(1)})^2}{2(\sigma_c^{(2)})^2} \right)$$

$$+ \frac{\lambda_u}{C} \sum_{c=1}^C \left( (\sigma_c^{(1)} - 1)^2 + (\sigma_c^{(2)} - 1)^2 \right)$$

$$- \frac{\lambda_m}{C} \sum_{c=1}^C (p_c^{(1)} \log p_c^{(1)} + p_c^{(1)} \log p_c^{(1)} + p_c^{(2)} \log p_c^{(2)}).$$  \hspace{1cm} (13)

where $p_c^{(1)}$, $p_c^{(1)}$, and $p_c^{(2)}$ are the softmax output with different transformations applied to $x^U$, $x$, and $x^U$, and $\lambda_c$, $\lambda_u$, and $\lambda_m$ are the hyperparameters for each loss term. In the second term of Eq.13, $\sigma_c^{(1)}$ and $\sigma_c^{(2)}$ act as learned scaling coefficients and automatically scales the squared error terms.

We use mini-batch stochastic gradient descent over a batch of samples of size $B$, so reformulate Eq.13 over $B$ samples,

$$\mathcal{L}_B(x_b; y_b, x^U_b; \theta) = \frac{1}{B} \sum_{b=1}^B \mathcal{L}(x_b^U, y_b, x_b^U; \theta).$$  \hspace{1cm} (14)

The proposed algorithm in semi-supervised learning with uncertainty is described in Algorithm 1.

5. Experiments

In this chapter, we report our experimental settings including datasets, a network architecture, and training details we used and show experimental results.

Datasets

We used CIFAR-10 and SVHN datasets. CIFAR-10 dataset contains 50000 training samples and 10000 test samples. We tested our proposed methods with 4000 labeled samples and treated all 50000 training samples. Labeled samples were drawn from all 50000 training samples randomly, we did not constraints the number of samples in each classes to be equal. Our experiments were
done for 10 runs, and we report the average of classification validation errors and its standard deviations. SVHN dataset contains 73257 training samples and 10000 validation samples. We ran two experiments where we used 500 and 1000 labeled samples respectively, and all the training samples were used as unlabeled samples. Experimental datasets for SVHN in semi-supervised learning settings were created as done in CIFAR-10.

**Network Architecture**

Our network architecture used in the experiments was highly inspired by [Springenberg 14]. [Springenberg 15, Rasmus 15] also used almost the same network architecture and some other used the smaller number of maps with the same architecture, but the head was not split. Table 1 shows the network architecture we used in our experiments. When the proposed method is used, the network is split in the head, such that in Table 1, at the last conv4 layer, the network is split into two sub-networks, and a convolution added on top of conv3c followed by a global average pooling, and a batch normalization in each sub-network.

**Training Details**

The batch size was 32 throughout the experiments. We did not use Zero Component Analysis (ZCA) to input images since we could not obtain better performance in our preliminary experiments even if we used ZCA. Adam optimizer [Kingma 14a] was used as an optimizer, and its hyper parameters, \(\alpha, \beta_1,\) and \(\beta_2,\) are 0.001, 0.9, and 0.999 respectively since these are the default values for hyper-

---

**Algorithm 1** semi-supervised learning with uncertainty

**Require:** Labeled samples \(D_L = \{ (x_i^L, y_i) \}_{i=1}^ \) and unla-
abeled samples \(D_U = \{ x_i^U \}_{i=1}^ \) a neural network \(f \) with parameters \(\theta = [\theta]^L_{i=1} \) where \(L \) is the number of param-
metric layers, random transformations \(T,\) batch size \(B,\) learning rate \(\eta,\) iteration counter \(k = 1 \) to maximum of iteration counts \(K,\)

**repeat**

randomly select \(B \) labeled samples in \(D_L \) and \(B \) unlabeled samples in \(D_U.\)

evaluate \(f(T(x_b^L)), f^{(1)}(T^{(1)}(x_b^U)), \) and \(f^{(2)}(T^{(2)}(x_b^U))\)

for all \(b = 1, \ldots, B,\)

compute the loss with Eq.14.

update \(\theta^{k+1}_b = \theta_b^k - \eta \times \partial L_b(\theta^k) / \partial \theta^k_b \) for all \(\ell = 1, \ldots, L,\)

\(k \leftarrow k + 1.\)

**until** \(k < K.\)

---

**Table 1:** Network Architecture used in our experiments. This architecture was inspired by ConvPool-CNN-C [Springenberg 14]. Conv is a convolution. BN is a batch normalization [Ioffe 15], and ELU is an exponential linear unit [Clevert 15]. The \(\times 2\) of conv4 and pool3 means that the output of the conv3c is forked, then that is fed into the two different convolutions.

| Name      | Description                                      |
|-----------|--------------------------------------------------|
| input     | 32x32 RGB image                                   |
| transformations | 32x32 perturbed RGB image                      |
| conv1a    | 128 filters, 3x3, pad=1, BN, ELU                 |
| conv1b    | 128 filters, 3x3, pad=1, BN, ELU                 |
| conv1c    | 128 filters, 3x3, pad=1, BN, ELU                 |
| pool1     | Maxpool 2x2 pixels                                |
| drop1     | Dropout, p=0.5                                    |
| conv2a    | 256 filters, 3x3, pad=1, BN, ELU                 |
| conv2b    | 256 filters, 3x3, pad=1, BN, ELU                 |
| conv2c    | 256 filters, 3x3, pad=1, BN, ELU                 |
| pool2     | Maxpool 2x2 pixels                                |
| drop2     | Dropout, p=0.5                                    |
| conv3a    | 512 filters, 3x3, pad=0, BN, ELU                 |
| conv3b    | 256 filters, 1x1, BN, ELU                        |
| conv3c    | 128 filters, 1x1, BN, ELU                        |
| conv4 x 2 | 10 filters, 1x1                                  |
| pool3 x 2 | Global average pool BN, (6x6 → 1x1)              |

parameters. For transformation \(T,\) random rotation within 15 degrees and random horizontal flipping were used. As weights for each loss, \(\lambda_c\) and \(\lambda_{em}\) were set to 1, and \(\lambda_{uc}\) and \(\lambda_{uu}\) were set to 0.1.

**Results**

Table 2 and Table 3 show the comparison of performance in averages and standard deviations of validation errors among semi-supervised learning methods using the two datasets: CIFAR-10 and SVHN. In Table 2, Consistency Loss + Entropy Minimization (CL + EM) and ours clearly outperform the other methods, our proposed method is competitive with CL + EM, and ours achieved worse in validation error than CL + EM with statistically non-significant difference. In Table 3, CL + EM and ours clearly outperforms the other methods, which is the same as in Table 2, but ours achieved the least validation error among the methods where the number of labeled samples was as small as 500.

Figure 2 shows the learning curve of the average and standard deviations of validation accuracies over the learning epochs. Here, one epoch is defined as passed when the number of input samples that we fed the number of
Table 2: Comparison of classification error (validation error) using CIFAR-10 among semi-supervised learning methods. † is our implementation. Our experiments for both of Consistency Loss + Entropy Minimization (CL + EM) and ours were done 10 runs.

| Methods                  | Error Rate (4000 labeled samples) |
|--------------------------|-----------------------------------|
| Conv-Large, -model [Rasmus 15] | 20.40 ± 0.47                      |
| CatGAN [Springenberg 15]   | 19.58 ± 0.58                      |
| Improved GAN [Salimans 16] | 18.63 ± 2.32                      |
| Triple GAN [Li 17]         | 18.82 ± 0.32                      |
| CL [Sajjadi] + EM †        | 14.43 ± 0.42                      |
| Ours                     | 15.12 ± 0.33                      |

Table 3: Comparison of classification error (validation error) using SVHN among semi-supervised learning methods. † is our implementation. Our experiments for both of CL + EM and ours were done 10 runs.

| Methods                  | Error Rate (500 labeled samples) | Error Rate (1000 labeled samples) |
|--------------------------|----------------------------------|-----------------------------------|
| DGN [Kingma 14b]         | 36.02 ± 0.10                     |                                   |
| Virtual Adversarial [Miyato 15] | 24.63                          |                                   |
| ADGM [Maaløe 16]        | 22.86                            |                                   |
| SDGM [Maaløe 16]        | 16.61 ± 0.24                     |                                   |
| Improved GAN [Salimans 16] | 18.44 ± 4.80                    | 8.11 ± 1.3                        |
| Triple GAN [Li 17]       | 5.83 ± 0.20                      |                                   |
| CL [Sajjadi] + EM †      | 5.50 ± 0.40                      | 4.88 ± 0.29                       |
| Ours                     | 4.97 ± 0.69                      | 3.79 ± 0.23                       |

all unlabeled samples. Therefore there were some replications of labeled samples fed in one epoch. These learning curves for CIFAR-10 and SVHN show that models trained with our methods apparently generalized faster than CL + EM. The reason might be that our method treats explicitly uncertainty as a standard deviation which appears in weights of consistency loss as in Eq.10. In Figure 3, at the very beginning of the training, the standard deviations are very large. However, the standard deviation suddenly goes down and increases while the average of hits (0 or 1) in classification rapidly increases. After that, the average of hits in classification stably goes up and the standard deviation fluctuates but steadily decreases.

We also did the sensitivity analysis for $\lambda_{uc}$ and $\lambda_{uu}$ using CIFAR-10 dataset. Figure 4 shows that when the lower values for hyperparameters were used, higher accuracy was obtained.

The role of regularizations for $\sigma$ in Eq.11 and Eq.12 are important. Table 4 indicates that without these regularization, we could not obtain better results compared to ones when we used the uncertainty consistency loss and the unit variance regularizations.

![Fig. 2: Learning curve of averages with standard deviations over 10 runs using CIFAR-10 and SVHN datasets. Models trained by our proposed method generalize faster than one trained by consistency loss only. This is because our proposed method considers standard deviations which acts as trainable coefficients so automatically tunes coefficients of squared error terms. From around 120 epochs, in CL + EM the learning curve diverged, indicating CL + EM is unstable in learning while our proposed method is much more stable.](image-url)
Fig. 3: Curves of standard deviations and averages of hits (0 or 1) in classification over iterations in training using CIFAR-10 dataset. We picked up a standard deviation and a hit in classification of a same sample from a batch of samples over iterations since we should compare these values for the same sample in a batch. At the very beginning of the training, the standard deviations are very large. However, the standard deviation suddenly goes down and increases while the average of hits in classification rapidly increases. After that, the average of hits in classification stably goes up and the standard deviation steadily decreases. It indicates that our proposed method automatically computes better coefficients to squared error terms.

Fig. 4: Sensitivity analysis with CIFAR-10 for $\lambda_{uc}$ and $\lambda_{uu}$. We used the same values for both $\lambda_{uc}$ and $\lambda_{uu}$. Roughly speaking, smaller $\lambda$ gives higher accuracy.

Table 4: Comparison between Eq.13 and that without the uncertainty consistency loss and the unit variance regularizations. † means we ran one experiment.

| Methods                          | Error Rate (4000 labeled samples) |
|----------------------------------|-----------------------------------|
| Ours with both $\lambda_{uc}$ and $\lambda_{uu}$ being zero † | 18.16                             |
| Ours                            | 15.12 ± 0.33                      |

6. Conclusion

We proposed a novel semi-supervised learning method with uncertainty where we introduced uncertainty for consistency loss and regularizations for such uncertainty in Chapter 4. We conducted various experiments and two commonly used datasets. Our proposed method let learning models generalize faster than one trained using the consistency loss and obtained accuracies competitive and/or better.

7. Discussion

We think there are mainly two directions of research in semi-supervised learning methods using consistency loss. Applications to much larger problem like [Russakovsky 14] and to different field such as signal processing and natural language processing are necessary to demonstrate usefulness of semi-supervised learning. The other direction is to search for layers to incorporate stochasticity without drastically changing model behaviours to better generalization since it naturally fits into our proposed method.

8. Acknowledgement

This works were supported by the KLL PhD. Program Research Grant and the Grant of Kenkyu-no-Susume. We would like to express our sincerest gratitude to the anonymous reviewers and editors for their insightful comments.

References

[Amodei 15] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A. Y., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Sestapun, D., Son Gupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., and Zhu, Z.: Deep speech 2: End-to-End speech recognition in English and Mandarin, CoRR, Vol. abs/1512.02595, (2015)

[Bahdanau 14] Bahdanau, D., Cho, K., and Bengio, Y.: Neural machine translation by jointly learning to align and translate, CoRR, Vol. abs/1409.0473, (2014)

[Belkin 06] Belkin, M., Niyogi, P., and Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, Journal of Machine Learning Research, Vol. 7,
Semi-Supervised Learning with Uncertainty

...ing using uncertainty to weigh losses for scene geometry and semantics, CoRR, Vol. abs/1705.07115, (2017)

[Kingma 13] Kingma, D. P. and Welling, M.: Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013)

[Kingma 14a] Kingma, D., Rezende, D. J., Mohamed, S., and Welling, M.: Semi-supervised learning with deep generative models, CoRR, Vol. abs/1406.5298, (2014)

[Krizhevsky 12] Krizhevsky, A.: Convolutional neural networks on tiny imagenet, CoRR, Vol. abs/1404.5561, (2014)

[Krizhevsky 15] Krizhevsky, A., Sutskever, I., and Hinton, G. E.: ImageNet classification with deep convolutional neural networks, in Perrett, B., Burgess, C. J., Daunh, L., and Weinberger, K. Q. eds., Advances in Neural Information Processing Systems 25, pp. 1097–1105, Curran Associates, Inc. (2012)

[Laine 16] Laine, S. and Aila, T.: Temporal ensembling for semi-supervised learning, CoRR, Vol. abs/1610.02242, (2016)

[Li 17] Li, C., Xu, K., Zhu, J., and Zhang, B.: Triple generative adversarial nets, CoRR, Vol. abs/1703.02291, (2017)

[Maaløe 16] Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O.: Auxiliary deep generative models, arXiv preprint arXiv:1602.05473 (2016)

[Melacci 11] Melacci, S. and Belkin, M.: Laplacian support vector machines trained in the primal, Journal of Machine Learning Research, Vol. 12, pp. 1149–1184 (2011)

[Miyato 15] Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S.: Distributional smoothing with virtual adversarial training, arXiv preprint arXiv:1507.00677 (2015)

[Miyato 16] Miyato, T., Dai, A. M., and Goodfellow, I.: Adversarial training methods for semi-supervised text classification, arXiv preprint arXiv:1605.07725 (2016)

[Nigam 00] Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. M.: Text classification from labeled and unlabeled documents using EM, Machine Learning, Vol. 39, No. 2/3, pp. 103–134 (2000)

[Oord 16] Oord, A. V. M., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K.: WaveNet: A generative model for raw audio, CoRR, Vol. abs/1609.03499, (2016)

[Orbach 12] Orbach, M. and Crammer, K.: Graph-based transduction with confidence, in Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II, pp. 323–338 (2012)

[Pezeshki 15] Pezeshki, M., Fan, L., Brakel, P., Courville, A. C., and Bengio, Y.: Deconstructing the ladder network architecture, CoRR, Vol. abs/1511.06340, (2015)

[Radford 15] Radford, A., Metz, L., and Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks, CoRR, Vol. abs/1511.06434, (2015)

[Rasmus 15] Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T.: Semi-supervised learning with ladder networks, CoRR, Vol. abs/1507.02672, (2015)

[Russakovsky 14] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, Fei-Fei: ImageNet large scale visual recognition challenge, CoRR, Vol. abs/1409.0575, (2014)

[Sajadi 17] Sajjadi, M., Javanmardi, M., and Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning, CoRR, Vol. abs/1606.04586, (2016)

[Sajadi 16] Sajjadi, M., Javanmardi, M., and Tasdizen, T.: Mutual exclusivity loss for semi-supervised deep learning, CoRR, Vol. abs/1606.03141, (2016)

[Sak 15] Sak, S., Senior, A. W., Rao, K., and Beaufays, F.: Fast and accurate recurrent neural network acoustic models for speech recognition, CoRR, Vol. abs/1507.06947, (2015)

[Salimans 16] Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X.: Improved techniques for training gans, CoRR, Vol. abs/1606.03498, (2016)

[Springenberg 14] Springenberg, J. T.: Dowoski, A., Brox, T., and Riedmiller, M. A.: Striving for simplicity: The all convolutional net, CoRR, Vol. abs/1412.6806, (2014)

[Springenberg 15] Springenberg, J. T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks,
Zhu 14] Zhu, J., Mao, J., and Yuille, A. L.: Learning from weakly supervised data by the expectation loss svm (e-svm) algorithm, in Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 1125–1133 (2014)

[Su 11] Su, J., Shirah, J. S., and Matwin, S.: Large scale text classification using semisupervised multinomial naive Bayes, in Proceedings of the 25th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, pp. 97–104 (2011)

[Subramanya 11] Subramanya, A. and Bilmes, J. A.: Semi-supervised learning with measure propagation, Journal of Machine Learning Research, Vol. 12, pp. 3311–3370 (2011)

[Syed 10] Syed, U. and Taskar, B.: Semi-supervised learning with adversarially missing label information, in Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada., pp. 2244–2252 (2010)

[Szegedy 14] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A.: Going deeper with convolutions, CoRR, Vol. abs/1409.4842, (2014)

[Tarvainen 17] Tarvainen, A. and Valpola, H.: Weight-averaged consistency targets improve semi-supervised deep learning results, CoRR, Vol. abs/1703.01780, (2017)

[Vincent 08] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders, pp. 1096–1103 (2008)

[Wang 13] Wang, J., Jehbara, T., and Chang, S.: Semi-supervised learning using greedy max-cut, Journal of Machine Learning Research, Vol. 14, No. 1, pp. 771–800 (2013)

[Wu 16] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J.: Google’s neural machine translation system: bridging the gap between human and machine translation, CoRR, Vol. abs/1609.08144, (2016)

[Xu 07] Xu, Z., Jin, R., Zhu, J., King, I., and Lyu, M. R.: Efficient convex relaxation for transductive support vector machine, in Advances in Neural Information Processing Systems 20, Proceedings of the 21 Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, pp. 1641–1648 (2007)

[Yoshiyama 11] Yoshiyama, K. and Sakurai, A.: Manifold-regularized minimax probability machine, in Partially Supervised Learning - First IAPR TC3 Workshop, PSL 2011, Ulm, Germany, September 15-16, 2011, Revised Selected Papers, pp. 42–51 (2011)

[Zhao 16] Zhao, J., Mathieu, M., Goroshin, R., and LeCun, Y.: Stacked what-where auto-encoders (2016)

[Zhu 03] Zhu, X., Ghahramani, Z., and Lafferty, J. D.: Semi-supervised learning using gaussian fields and harmonic functions, in Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, pp. 912–919 (2003)

[Zhu 09] Zhu, X. and Goldberg, A. B.: Introduction to semi-supervised learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2009)

[Zhu 14] Zhu, J., Mao, J., and Yuille, A. L.: Learning from weakly supervised data by the expectation loss svm (e-svm) algorithm, in Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 1125–1133 (2014)

Received October 17, 2017.