

# Self-Training using Selection Network for Semi-supervised Learning

Jisoo Jeong<sup>\*a</sup>, Seungeui Lee<sup>\*b</sup> and Nojun Kwak<sup>†c</sup>

Seoul National University, Seoul, South Korea

Keywords: Semi-supervised Learning.

Abstract: Semi-supervised learning (SSL) is a study that efficiently exploits a large amount of unlabeled data to improve performance in conditions of limited labeled data. Most of the conventional SSL methods assume that the classes of unlabeled data are included in the set of classes of labeled data. In addition, these methods do not sort out useless unlabeled samples and use all the unlabeled data for learning, which is not suitable for realistic situations. In this paper, we propose an SSL method called *selective self-training* (SST), which selectively decides whether to include each unlabeled sample in the training process. It is designed to be applied to a more real situation where classes of unlabeled data are different from the ones of the labeled data. For the conventional SSL problems which deal with data where both the labeled and unlabeled samples share the same class categories, the proposed method not only performs comparable to other conventional SSL algorithms but also can be combined with other SSL algorithms. While the conventional methods cannot be applied to the new SSL problems, our method does not show any performance degradation even if the classes of unlabeled data are different from those of the labeled data.

## 1 INTRODUCTION

Recently, machine learning has achieved a lot of success in various fields and well-refined datasets are considered to be one of the most important factors (Everingham et al., 2010; Krizhevsky et al., 2012; Russakovsky et al., 2015). Since we cannot discover the underlying real distribution of data, we need a lot of samples to estimate it correctly (Nasrabadi, 2007). However, making a large dataset requires a huge amount of time, cost and manpower (Chapelle et al., 2009; Odena et al., 2018).


Semi-supervised learning (SSL) is a method relieving the inefficiencies in data collection and annotation process, which lies between the supervised learning and unsupervised learning in that both labeled and unlabeled data are used in the learning process (Chapelle et al., 2009; Odena et al., 2018). It can efficiently learn a model from fewer labeled data using a large amount of unlabeled data (Zhu, 2006). Accordingly, the significance of SSL has been studied extensively in the previous literatures (Zhu et al.,


2003; Rosenberg et al., 2005; Kingma et al., 2014; Rasmus et al., 2015; Odena, 2016; Akhmedova et al., 2017). These results suggest that SSL can be a useful approach in cases where the amount of annotated data is insufficient.

However, there is a recent research discussing the limitations of conventional SSL methods (Odena et al., 2018). They have pointed out that conventional SSL algorithms are difficult to be applied to real applications. Especially, the conventional methods assume that all the unlabeled data belong to one of the classes of the training labeled data. Training with unlabeled samples whose class distribution is significantly different from that of the labeled data may degrade the performance of traditional SSL methods. Furthermore, whenever a new set of data is available, they should be trained from the scratch using all the data including out-of-class<sup>1</sup> data.

In this paper, we focus on the classification task and propose a deep neural network based approach named as *selective self-training* (SST) to solve the limitation mentioned above. To enable learning to se-

<sup>a</sup>  <https://orcid.org/0000-0003-1154-4532>

<sup>b</sup>  <https://orcid.org/0000-0003-3560-785X>

<sup>c</sup>  <https://orcid.org/0000-0002-1792-0327>

\* Equal contribution

† Corresponding author.

<sup>1</sup>The term *out-of-class* is used to denote the situation where the new dataset contains samples originated from different classes than the classes of the old data. On the other hand, the term *in-class* is used when the new data contain only the samples belonging to the previously observed classes.

lect unlabeled data, we propose a *selection network*, which is based on the deep neural network, that decides whether each sample is to be added or not. Different from (Wang et al., 2018), SST does not directly use the classification results for the data selection. Also, we adopt an ensemble approach which is similar to the co-training method (Blum and Mitchell, 1998) that utilizes outputs of multiple classifiers to iteratively build a new training dataset. In our case, instead of using multiple classifiers, we apply a temporal ensemble method to the selection network. For each unlabeled instance, two consecutive outputs of the selection network are compared to keep our training data clean.

In addition, we have found that the balance between the number of samples per class is quite important for the performance of our network. We suggest a simple heuristics to balance the number of selected samples among the classes. By the proposed selection method, reliable samples can be added to the training set and uncertain samples including out-of-class data can be excluded. The main contributions of the proposed method can be summarized as follows:

- For the conventional SSL problems, the proposed SST method not only performs comparable to other conventional SSL algorithms but also can be combined with other algorithms.
- For the new SSL problems, the proposed SST does not show any performance degradation even with the out-of-class data.
- SST requires few hyper-parameters and can be easily implemented.

To prove the effectiveness of our proposed method, first, we conduct experiments comparing the classification errors of SST and several other state-of-the-art SSL methods (Laine and Aila, 2016; Tarvainen and Valpola, 2017; Luo et al., 2017; Miyato et al., 2017) in conventional SSL settings. Second, we propose a new experimental setup to investigate whether our method is more applicable to real-world situations. The experimental setup in (Odena et al., 2018) samples classes among in-classes and out-classes. In the experimental setting in this paper, we sample unlabeled instances evenly in all classes. We evaluate the performance of the proposed SST using three public benchmark datasets: CIFAR-10, CIFAR-100 (Krizhevsky and Hinton, 2009), and SVHN (Netzer et al., 2011).

## 2 BACKGROUND

In this section, we introduce the background of our research. First, we introduce some methods of self-training (McLachlan, 1975; Zhu, 2007; Zhu and Goldberg, 2009) on which our work is based. Then we describe consistency regularization-based algorithms such as  $\Pi$  model and temporal ensembling (Laine and Aila, 2016).

### 2.1 Self-training

Self-training method has long been used for semi-supervised learning (McLachlan, 1975; Rosenberg et al., 2005; Zhu, 2007; Zhu and Goldberg, 2009). It is a resampling technique that repeatedly labels unlabeled training samples based on the confidence scores and retrains itself with the selected pseudo-annotated data. This process can be formalized as follows. (i) Training a model with labeled data. (ii) Predicting unlabeled data with the learned model. (iii) Retraining the model with labeled and selected pseudo-labeled data. (iv) Repeating the last two steps.

However, most self-training methods assume that the labeled and unlabeled data are generated from the identical distribution. Therefore, in real-world scenarios, some instances with low likelihood according to the distribution of the labeled data are likely to be misclassified inevitably. Consequently, these erroneous samples significantly lead to worse results in the next training step. To alleviate this problem, we adopt the ensemble and balancing methods to select reliable samples.

### 2.2 Consistency Regularization

Consistency regularization is one of the popular SSL methods and has been referred to many recent researches (Laine and Aila, 2016; Miyato et al., 2017; Tarvainen and Valpola, 2017). Among them,  $\Pi$  model and temporal ensembling are widely used (Laine and Aila, 2016). They have defined new loss functions for unlabeled data. The  $\Pi$  model outputs  $f(\mathbf{x})$  and  $\hat{f}(\mathbf{x})$  for the same input  $\mathbf{x}$  by perturbing the input with different random noise and using dropout (Srivastava et al., 2014), and then minimizes the difference ( $\|f(\mathbf{x}) - \hat{f}(\mathbf{x})\|^2$ ) between these output values. Temporal ensembling does not make different predictions  $f(\mathbf{x})$  and  $\hat{f}(\mathbf{x})$ , but minimizes the difference ( $\|f_{t-1}(\mathbf{x}) - f_t(\mathbf{x})\|^2$ ) between the outputs of two consecutive iterations for computational efficiency. In spite of the improvement in performance, they require lots of things to consider for training. These methods have various hyper-parameters such as ‘ramp

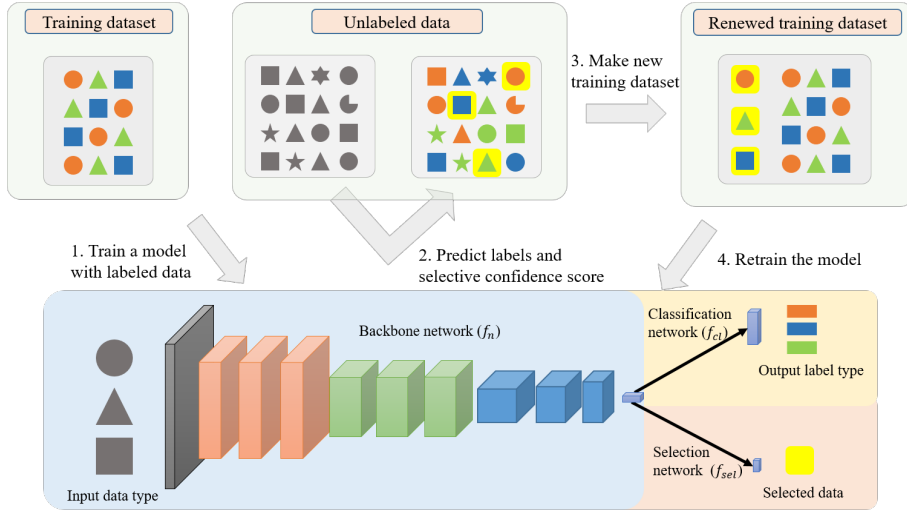


Figure 1: An overview of the proposed SST. Different shapes represent the input data with different underlying distribution, and different colors (orange, blue, and green) are for different classes. In the initial training dataset, only three classes with their corresponding distributions ( $\circ$ ,  $\square$ ,  $\triangle$ ) exist and are used for initial training. Then the unlabeled data which include unseen distribution ( $\star$ ,  $\diamond$ ) are inputted to the classification as well as the selection network. At the bottom right, unlabeled samples with higher selection network output values than a certain threshold are denoted by yellow and selected to be included in the training process for the next iteration, while the remaining are not used for training.

up’, ‘ramp down’, ‘unsupervised loss weight’ and so on. In addition, customized settings for training such as ZCA preprocessing and mean-only batch normalization (Salimans and Kingma, 2016) are also very important aspects for improving the performance (Odena et al., 2018).

### 3 METHOD

In this section, we introduce our selective self-training (SST) method. The proposed model consists of three networks as shown in the bottom part of Figure 1. The output of the backbone network is fed into two sibling fully-connected layers—a classification network  $f_{cl}(\cdot; \theta_c)$  and a selection network  $f_{sel}(\cdot; \theta_s)$ , where  $\theta_c$  and  $\theta_s$  are learnable parameters for each of them. In this paper, we define the classification result and the selection score as  $r_i = f_{cl}(f_n(\cdot; \theta_n); \theta_c)$  and  $s_i = f_{sel}(f_n(\cdot; \theta_n); \theta_s)$ , respectively, where  $f_n(\cdot; \theta_n)$  denotes the backbone network with learnable parameters  $\theta_n$ . Note that we define  $r_i$  as the resultant label and it belongs to one of the class labels  $r_i \in \mathcal{Y} = \{1, 2, \dots, C\}$ . As shown in Figure 1, the proposed SST method can be represented in the following four steps. First, SST trains the network using a set of the labeled data  $\mathcal{L} = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, L\}$ , where  $\mathbf{x}_i$  and  $y_i \in \{1, 2, \dots, C\}$  denote the data and the ground truth label respectively, which is a standard supervised learning method. The next step is to predict all the unlabeled data  $\mathcal{U} = \{\mathbf{x}_i \mid i = L+1, \dots, N\}$  and select a sub-

set of the unlabeled data  $\{\mathbf{x}_i \mid i \in I_S\}$  whose data have high selection scores with the current trained model, where  $I_S$  denotes a set of selected sample indices from  $I_U = \{L+1, \dots, N\}$ . Then, we annotate the selected samples with the pseudo-categories  $\hat{y}_i$  evaluated by the  $f_{cl}(\cdot; \theta_c)$  and construct a new training dataset  $\mathcal{T}$  composed of  $\mathcal{L}$  and  $\mathcal{U}_S = \{(\mathbf{x}_i, \hat{y}_i) \mid i \in I_S\}$ . After that, we retrain the model with  $\mathcal{T}$  and repeat this process iteratively. The overall process of the SST is described in Algorithm 1 and the details of each of the four steps will be described later.

#### 3.1 Supervised Learning

The SST algorithm first trains a model with supervised learning. At this time, the entire model (all three networks) is trained simultaneously.  $f_{cl}(\cdot; \theta_c)$  is trained using the softmax function and the cross-entropy loss as in the ordinary supervised classification learning task. In case of  $f_{sel}(\cdot; \theta_s)$ , the training labels are motivated by discriminator of generative adversarial networks (GAN) (Goodfellow et al., 2014; Yoo et al., 2017). When  $\mathbf{x}_i$  with  $y_i$  is fed into the network, the target for  $f_{sel}(\cdot; \theta_s)$  is set as:

$$g_i = \begin{cases} 1, & \text{if } r_i = y_i \text{ for } i \in I_L \\ 0, & \text{if } r_i \neq y_i \text{ for } i \in I_L \end{cases} \quad (1)$$

where  $I_L = \{1, \dots, L\}$  represents a set of labeled sample indices.  $f_{sel}(\cdot; \theta_s)$  is trained with the generated target  $g_i$ . Especially, we use the sigmoid function for the final activation and the binary cross-entropy loss to

---

Algorithm 1: Training procedure of the proposed SST.

---

**Require:**  $\mathbf{x}_i, y_i$ : training data and label

**Require:**  $\mathcal{L}, \mathcal{U}$ : labeled and unlabeled datasets

**Require:**  $I_U$ : set of unlabeled sample indices

**Require:**  $f_n(\cdot; \theta_n), f_{cl}(\cdot; \theta_c), f_{sel}(\cdot; \theta_s)$ : trainable SST model

**Require:**  $\alpha, \varepsilon, K, K_{re}$ : hyper-parameters

- 1: randomly initialize  $\theta_n, \theta_c, \theta_s$
  - 2: train  $f_n(\cdot; \theta_n), f_{cl}(\cdot; \theta_c), f_{sel}(\cdot; \theta_s)$  for  $K$  epochs using  $\mathcal{L}$
  - 3: **repeat**
  - 4:   initialize  $r_i^t = -1, I_S = \emptyset$
  - 5:   **for** each  $i \in I_U$  **do**
  - 6:      $r_i^{t-1} \leftarrow r_i^t, r_i^t \leftarrow f_{cl}(f_n(\mathbf{x}_i; \theta_n); \theta_c)$
  - 7:      $s_i \leftarrow f_{sel}(f_n(\mathbf{x}_i; \theta_n); \theta_s)$
  - 8:     **if**  $r_i^{t-1} \neq r_i^t$  **then**
  - 9:        $z_i \leftarrow 0$
  - 10:     **end if**
  - 11:      $z_i \leftarrow \alpha z_i + (1 - \alpha)s_i$
  - 12:     **if**  $z_i > 1 - \varepsilon$  **then**
  - 13:        $I_S \leftarrow I_S \cup \{i\}$
  - 14:       assign label for  $\mathbf{x}_i$  using  $r_i$
  - 15:     **end if**
  - 16:   **end for**
  - 17:   update  $\mathcal{U}_S$  with data balancing
  - 18:    $\mathcal{T} \leftarrow \mathcal{L} \cup \mathcal{U}_S$
  - 19:   retrain  $f_n(\cdot; \theta_n), f_{cl}(\cdot; \theta_c), f_{sel}(\cdot; \theta_s)$  for  $K_{re}$  epochs using  $\mathcal{T}$
  - 20: **until** stopping criterion is *true*
- 

train  $f_{sel}(\cdot; \theta_s)$ . Therefore our  $f_{sel}(\cdot; \theta_s)$  does not utilize the conventional softmax function because it produces a relative output and can induce a high value even for an out-of-class sample. Instead, our  $f_{sel}(\cdot; \theta_s)$  is designed to estimate an absolute confidence score using the sigmoid activation function. More details are provided in appendix. Consequently, our final loss function is a sum of the classification loss  $L_{cl}$  and the selection loss  $L_{sel}$ :

$$L_{total} = L_{cl} + L_{sel}. \quad (2)$$

### 3.2 Prediction and Selection

After learning the model in a supervised manner, SST takes all instances of  $\mathcal{U}$  as input and predicts  $r_i$  and  $s_i$ , for all  $i \in I_U$ . We utilize  $r_i$  and  $s_i$  to annotate and choose unlabeled samples, respectively. In the context of self-training, removing erroneously annotated samples is one of the most important things for the new training dataset. Thus, we adopt temporal co-training and ensemble methods for the selection score in order to keep our training set from contamination. First, let

Table 1: Ablation study with 5 runs on the CIFAR-10 dataset. ‘bal’ denotes the usage of data balancing scheme during data addition as described in Sec. 3.3, ‘ens’ is for the usage of previous selection scores as in the 11th line of Algorithm 1.

method	bal	ens	error
supervised learning			$18.97 \pm 0.37\%$
SST	x	x	$21.44 \pm 4.05\%$
	o	x	$14.43 \pm 0.43\%$
	o	o	$11.82 \pm 0.40\%$

$r_i^t$  and  $r_i^{t-1}$  be the classification results of the current and the previous iterations respectively and we utilize the temporal consistency of these values. If these values are different, we set the ensemble score  $z_i = 0$  to reduce uncertainty in selecting unlabeled samples. Second, inspired by (Laine and Aila, 2016), we also utilize multiple previous network evaluations of unlabeled instances by updating  $z_i = \alpha z_i + (1 - \alpha)s_i$ , where  $\alpha$  is a momentum weight for the moving average of ensemble scores. However, the aim of our ensembling approach is different from (Laine and Aila, 2016). They want to alleviate different predictions for the same input, which are resulted from different augmentation and noise to the input. However, our aim differs from theirs in that we are interested in selecting reliable (pseudo-)labeled samples. After that, we select unlabeled samples with high  $z_i$ . It is very important to set an appropriate threshold because it decides the quality of the added unlabeled samples for the next training. If  $f_{cl}(\cdot; \theta_c)$  is trained well on the labeled data, the training accuracy would be very high. Also, since  $f_{sel}(\cdot; \theta_s)$  is trained with  $g_i$  generated from  $r_i$  and  $y_i$ ,  $s_i$  will be close to 1.0. Therefore, we set the threshold to  $1 - \varepsilon$  and control it by changing  $\varepsilon$ . In this case, if  $z_i$  exceeds  $1 - \varepsilon$ , the pseudo-label of the unlabeled sample  $\hat{y}_i$  is set to  $r_i$ .

### 3.3 New Training Dataset

When we construct  $\mathcal{T}$ , we keep the number of samples of each class the same. The reason is that if one class dominates the others, the classification performance is degraded by the imbalanced distribution (Fernández et al., 2013). We also empirically found that naively creating a new training dataset fails to yield good performance. In order to fairly transfer the selected samples to the new training set, the amount of migration in each class should not exceed the number of the class having the least selected samples. We take arbitrary samples in every class as much as the maximum number satisfying this condition.  $\mathcal{T}$  is composed of both  $\mathcal{L}$  and  $\mathcal{U}_S$ . The number of selected unlabeled samples is the same for all classes.

Table 2: Classification error on CIFAR-10 (4k Labels), SVHN (1k Labels), and CIFAR-100 (10k Labels) with 5 runs using in-class unlabeled data (\* denotes that the test has been done by ourselves).

Method	CIFAR-10	SVHN	CIFAR-100
Supervised (sampled)*	18.97 ± 0.37%	13.45 ± 0.92%	40.24 ± 0.45%
Supervised (all)*	5.57 ± 0.07%	2.87 ± 0.06%	23.36 ± 0.27%
Mean Teacher (Tarvainen and Valpola, 2017)	12.31 ± 0.28%	3.95 ± 0.21%	-
$\Pi$ model (Laine and Aila, 2016)	12.36 ± 0.31%	4.82 ± 0.17%	39.19 ± 0.36%
TempEns (Laine and Aila, 2016)	12.16 ± 0.24%	4.42 ± 0.16%	38.65 ± 0.51%
TempEns + SNTG (Luo et al., 2017)	10.93 ± 0.14%	3.98 ± 0.21%	40.19 ± 0.51%*
VAT (Miyato et al., 2017)	11.36 ± 0.34%	5.42 ± 0.22%	-
VAT + EntMin (Miyato et al., 2017)	10.55 ± 0.05%	3.86 ± 0.11%	-
pseudo-label (Lee, 2013; Odena et al., 2018)	17.78 ± 0.57%	7.62 ± 0.29%	-
Proposed method (SST)*	11.82 ± 0.40%	6.88 ± 0.59%	34.89 ± 0.75%
SST + TempEns + SNTG*	9.99 ± 0.31%	4.74 ± 0.19%	34.94 ± 0.54%

### 3.4 Re-training

After combining the labeled and selected pseudo-labeled data, the model is retrained with the new dataset for  $K_{re}$  epochs. In this step, the label for the  $f_{sel}(\cdot; \theta_s)$  is obtained by a process similar to Eq. (1). Above steps (except for Section 3.1) are repeated for  $M$  iterations until (near-) convergence.

## 4 EXPERIMENTS

To evaluate our proposed SST algorithm, we conduct two types of experiments. First, we evaluate the proposed SST algorithm for the conventional SSL problem where all unlabeled data are in-class. Then, SST is evaluated with the new SSL problem where some of the unlabeled data are out-of-class.

In the case of in-class data, gradually gathering highly confident samples in  $\mathcal{U}$  can help improve the performance. On the other hand, in the case of out-of-class data, a strict threshold is preferred to prevent uncertain out-of-class data from being involved in the new training set. Therefore, we have experimented with *decay* mode that decreases the threshold in log-scale and *fixed* mode that fixes the threshold in the way described in Section 4.2. We have experimented our method with 100 iterations and determined  $\epsilon$  by cross-validation in *decay* modes. In case of *fixed* modes,  $\epsilon$  is fixed and the number of iteration is determined by cross-validation. The details about the experimental setup is presented in appendix.

### 4.1 Conventional SSL Problems with In-class Unlabeled Data

We experiment with three popular datasets which are SVHN, CIFAR-10, and CIFAR-100 (Netzer et al.,

2011; Krizhevsky et al., 2014). The settings of labeled versus unlabeled data separation for each dataset are the same with (Laine and Aila, 2016; Miyato et al., 2017; Tarvainen and Valpola, 2017). More details are provided in appendix. Also, the network is the same with (Laine and Aila, 2016).

#### 4.1.1 Ablation Study

We have performed experiments on CIFAR-10 dataset with the combination of two types of components. As described in Table 1, these are whether to use data balancing scheme described in Section 3.3 (balance), whether to use selection score ensemble in the 11th line of Algorithm 1 (ensemble). First, when SST does not use all of these, the error 21.44% is higher than that of the supervised learning which does not use any unlabeled data. This is due to the problem of unbalanced data mentioned in subsection 3.3. When the data balance is used, the error is 14.43%, which is better than the baseline 18.97%. Adding the ensemble scheme results in 11.82% error. Therefore, we have used only balance and ensemble schemes in the following experiments.

#### 4.1.2 Experimental Results

Table 2 shows the experiment results of supervised learning, conventional SSL algorithms and the proposed SST on CIFAR-10, SVHN and CIFAR-100 datasets. Our baseline model with supervised learning performs slightly better than what has been reported in other papers (Laine and Aila, 2016; Tarvainen and Valpola, 2017; Luo et al., 2017) because of our different settings such as Gaussian noise on inputs, optimizer selection, the mean-only batch normalizations and the learning rate parameters. For all the datasets, we have also performed experiments with a model of SST combined with the temporal ensembling (Tem-

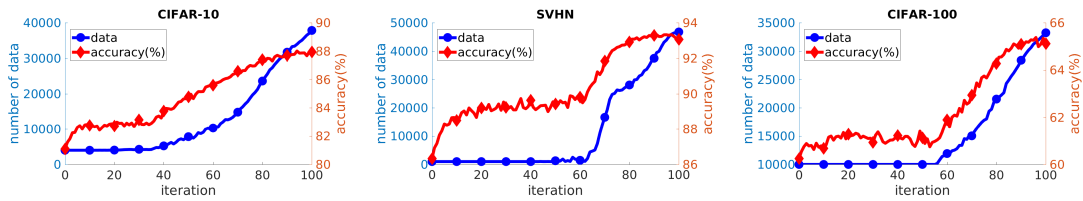


Figure 2: SST result on CIFAR-10, SVHN, and CIFAR-100 datasets with 5 runs. The  $x$ -axis is the iteration, the blue circle is the average of the number of data used for training, and the red diamond is the average accuracy.

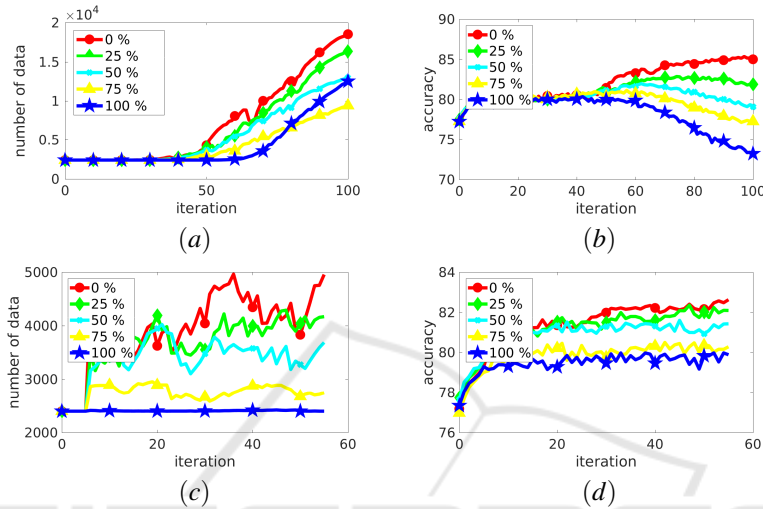


Figure 3: Result of new SSL problems on CIFAR-10 dataset with 5 runs. (a) number of data with iteration in decay mode (b) accuracy with iteration in decay mode (c) number of data with iteration in fixed mode (d) accuracy with iteration in fixed mode. % means the ratio of the number of non-animal classes in the unlabeled data.

pEns) and SNTG, labeled as SST+TempEns+SNTG. For the model, the pseudo-labels of SST at the last iteration is considered as the true class label. Figure 2 shows the number of samples used in the training and the corresponding accuracy on the test set for each dataset.

**CIFAR-10:** The baseline network yields the test error of 18.97% and 5.57% when trained with 4,000 (sampled) and 50,000 (all) labeled images respectively. The test error of our SST method reaches 11.82% which is comparable to other algorithms while SST+TempEns+SNTG model results 1.83% better than the SST-only model.

**SVHN:** The baseline model for SVHN dataset is trained with 1,000 labeled images and yields the test error of 13.45%. Our proposed method has an error of 6.88% which is relatively higher than those of other SSL algorithms. Performing better than SST, SST+TempEns+SNTG reaches 4.74% of error which is worse than that of TempEns+SNTG model. We suspect two reasons for this. The first is that SVHN dataset is not well balanced, and the second is that SVHN is a relatively easy dataset, so it seems to be easily added to the hard labels. With data balancing,

the SST is still worse than other algorithms. We think this phenomenon owes to the use of hard labels in SST where incorrectly estimated samples deteriorate the performance.

**CIFAR-100:** While the baseline model results in 40.24% of error rate through supervised learning with sampled data, our method performs with 34.89% of error, enhancing the performance by 5.3%. We have observed that the performance of TempEns+SNTG is lower than TempEns, and when TempEns+SNTG is added to SST, performance is degraded slightly. Although TempEng+SNTG shows better performance than TempEng without augmentation in (Luo et al., 2017), its performance is worse than that of TempEng with augmentation in our experiment. The reason for this can be conjectured that the hyper-parameter in the current temporal ensembling and SNTG may not have been optimized<sup>2</sup>.

<sup>2</sup>We have reproduced tempEns+SNTG model with a Pytorch implementation, and have verified of its performance on CIFAR-10 and SVHN akin to what is reported in (Luo et al., 2017). However, for CIFAR-100 dataset, since the experimental results without data augmentation are not reported, we thus report our reproduced results.

Table 3: Classification error for new SSL problems on CIFAR-10 and CIFAR-100 dataset with 5 runs. '%' means the ratio of the number of non-animal classes.

dataset	CIFAR-10		CIFAR-100	
	SST(decay)	SST(fixed)	SST(decay)	SST(fixed)
supervised	$22.27 \pm 0.47\%$		$34.62 \pm 1.14\%$	
0%	$14.99 \pm 0.54\%$	$17.84 \pm 0.39\%$	$28.01 \pm 0.44\%$	$32.16 \pm 0.64\%$
25%	$17.93 \pm 0.33\%$	$18.38 \pm 0.52\%$	$29.94 \pm 0.45\%$	$32.28 \pm 0.58\%$
50%	$20.91 \pm 0.53\%$	$19.04 \pm 0.63\%$	$31.78 \pm 0.62\%$	$32.60 \pm 0.67\%$
75%	$22.72 \pm 0.42\%$	$20.07 \pm 0.98\%$	$34.44 \pm 0.85\%$	$32.32 \pm 0.52\%$
100%	$26.78 \pm 1.35\%$	$20.24 \pm 0.15\%$	$37.17 \pm 1.08\%$	$32.62 \pm 0.63\%$

## 4.2 New SSL Problems with Out-of-class Unlabeled Data

We have experimented with the following settings for real-world applications. The dataset is categorized into six animal and four non-animal classes as similarly done in (Odena et al., 2018). In CIFAR-10, 400 images per animal class are used as the labeled data (total 2,400 images for 6 animal classes) and a pool of 20,000 images with different mixtures of both animal and non-animal classes are experimented as an unlabeled dataset. In CIFAR-100, 5,000 labeled data (100 images per animal class) and a total of 20,000 unlabeled images of both classes with different mixed ratios are utilized. Unlike the experimental setting in (Odena et al., 2018), we have experimented according to the ratio (%) of the number of out-of-class data in the unlabeled dataset.

As mentioned above, in the presence of out-of-class samples, a strict threshold is required. If all of the unlabeled data is assumed to be in-class, the decay mode may be a good choice. However, in many real-applications, out-of-class unlabeled data is also added to the training set in the decay mode and causes poor performance. In avoidance of such matter, we have experimented on a *fixed mode* of criterion threshold on adding the unlabeled data. Unlike the decay mode that decrements the threshold value, SST in the fixed mode sets a fixed threshold at a reasonably high value throughout the training. Our method in the fixed mode should be considered more suitable for real-applications but empirically shows lower performances in Figure 3 and Table 3 than when running in the decay mode. The difference between the decay mode and the fixed mode are an unchangeable  $\epsilon$  and the initial ensemble.

Setting a threshold value for the fixed mode is critical for a feasible comparison against the decay mode. Figure 3 shows the average of the results obtained when performing SST five times for each ratio in CIFAR-10. As shown in Figure 3(a), as the number of iteration increases, the threshold in the decay mode decreases and the number of additional unlabeled data increases. Obviously, while the different percentage

of the non-animal data inclusion show different trends of training, in the cases of 0 ~ 75% of non-animal data included in the unlabeled dataset, the additionally selected training data shows an initial increase at 30<sup>th</sup> ~ 40<sup>th</sup> iteration. Also, when the unlabeled dataset is composed of only the out-of-class data, selective data addition of our method initiates at 55<sup>th</sup> ~ 65<sup>th</sup> training iteration. This tendency has been observed in previous researches on classification problems and we have set the threshold value fixed at a value between two initiating points of data addition as similarly done in the works of (Viola and Jones, 2001; Zhang and Viola, 2008). We have set the fixed threshold based on 47th iteration (between 40 and 55). For a more reliable selection score, we have not added any unlabeled data to the new training set and have trained our method with the labeled data only for 5 iterations.

As it can be seen in Table 3, in the case of SST in the decay mode, the performance has been improved when the unlabeled dataset consists only in-class animal data, but when the unlabeled pool is filled with only out-of-class data, the performance is degraded. For the case of SST with a fixed threshold value, samples are not added and the performance was not degraded at 100% non-animal ratio as shown in Figure 3(c). Furthermore, at 0% of out-of-class samples in the pool, there is more improvement in the performance than at 100 % of out-of-class samples while still being inferior to the improvement than the decay mode. Because less but stable data samples are added by SST with a fixed threshold, the performance is improved for all the cases compared to that of supervised learning. Therefore, it is more suitable for real applications where the origin of data is usually unknown.

## 5 CONCLUSION

We proposed selective self-training (SST) for semi-supervised learning (SSL) problem. SST selectively samples unlabeled data and trains the model with a subset of the dataset. Using selection network, reliable samples can be added to the new training dataset. In this paper, we conduct two types of experiments.

First, we experiment with the assumption that unlabeled data are in-class like conventional SSL problems. Then, we experiment how SST performs for out-of-class unlabeled data.

For the conventional SSL problems, we achieved competitive results on several datasets and our method could be combined with conventional algorithms to improve performance. The accuracy of SST is either saturated or not depending on the dataset. Nonetheless, SST has shown performance improvements as a number of data increases. In addition, the results of the combined experiments of SST and other algorithms show the possibility of performance improvement.

For the new SSL problems, SST did not show any performance degradation even if the model is learned from in-class data and out-of-class unlabeled data. Decreasing the threshold of the selection network in new SSL problem, performance degrades. However, the output of the selection network shows different trends according to in-class and out-of-class. By setting a threshold that does not add out-of-class data, SST has prevented the addition of out-of-class samples to the new training dataset. It means that it is possible to prevent the erroneous data from being added to the unlabeled dataset in a real environment.

## ACKNOWLEDGEMENTS

This work was supported by IITP grant funded by the Korea government (MSIT) (No.2019-0-01367).

## REFERENCES

- Akhmedova, S., Semenkin, E., and Stanovov, V. (2017). Semi-supervised svm with fuzzy controlled cooperation of biology related algorithms. In *ICINCO (1)*, pages 64–71.
- Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.
- Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.
- Fernández, A., López, V., Galar, M., Del Jesus, M. J., and Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-based systems*, 42:97–110.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.
- Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.
- Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.
- Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: <http://www.cs.toronto.edu/kriz/cifar.html>*.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.
- Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2.
- Luo, Y., Zhu, J., Li, M., Ren, Y., and Zhang, B. (2017). Smooth neighbors on teacher graphs for semi-supervised learning. *arXiv preprint arXiv:1711.00258*.
- McLachlan, G. J. (1975). Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369.
- Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.
- Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.
- Odena, A. (2016). Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*.
- Odena, A., Oliver, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. (2018). Realistic evaluation of semi-supervised learning algorithms.
- Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.



- Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *WACV/MOTION*, pages 29–36.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.
- Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.
- Wang, K., Lin, L., Yan, X., Chen, Z., Zhang, D., and Zhang, L. (2018). Cost-effective object detection: Active sample mining with switchable selection criteria. *IEEE Transactions on Neural Networks and Learning Systems*, (99):1–17.
- Yoo, Y., Park, S., Choi, J., Yun, S., and Kwak, N. (2017). Butterfly effect: Bidirectional control of classification performance by small additive perturbation. *arXiv preprint arXiv:1711.09681*.
- Zhang, C. and Viola, P. A. (2008). Multiple-instance pruning for learning efficient cascade detectors. In *Advances in neural information processing systems*, pages 1681–1688.
- Zhu, X. (2006). Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4.
- Zhu, X. (2007). Semi-supervised learning tutorial. In *International Conference on Machine Learning (ICML)*, pages 1–135.
- Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.
- Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

## APPENDIX

### The Basic Settings of Our Experiments

The basic settings of our experiments are as follows. Different from (Laine and Aila, 2016; Luo et al., 2017), we use stochastic gradient descent (SGD) with a weight decay of 0.0005 as an optimizer. The momentum weight for the ensemble of selection scores is set to  $\alpha = 0.5$ . Also, we do not apply mean-only batch normalization layer (Salimans and Kingma, 2016) and Gaussian noise. We follow the same data augmentation scheme in (Laine and Aila, 2016) consisting of horizontal flips and random translations. However, ZCA whitening is not used. In the supervised learning phase, we train our model using batch size 100 for 300 epochs. After that, in the retraining phase, we train using the same batch size for 150 epochs with the new training dataset. The learning rate starts from 0.1. In the supervised learning phase, it is divided by 10 at the 150-th and 225-th epoch. In the retraining phase, it is divided by 10 at the 75-th and 113-th epoch.

The number of training iteration and thresholding  $\epsilon$  are very important parameters in our algorithm and have a considerable correlation with each other. In the first experiment, the iteration number remains fixed and the growth rate of  $\epsilon$  is adjusted so that the validation accuracy saturates near the settled iteration number. While the validation accuracy is evaluated using the cross-validation, we set the number of training iteration to be 100 so that the model is trained enough until it saturates.  $\epsilon$  is increased in log-scale and begins at a very small value ( $10^{-5}$ ) where no data is added. The growth rate of  $\epsilon$  is determined according to when the validation accuracy saturates. The stopping criterion is that the accuracy of the current iteration reaches the average accuracy of the previous 20 steps. If the stopping iteration is much less than 100 times, the  $\epsilon$  growth rate should be reduced so that the data is added more slowly. If the stopping iteration significantly exceeds 100 iterations, the  $\epsilon$  growth rate should be increased so that the data is added more easily. We allow 5 iterations as a deviation from 100 iterations and the growth rate of  $\epsilon$  is left unchanged in this interval. As a result, the  $\epsilon$  is gradually increased in log-scale by 10 times every 33 iterations in CIFAR-10 and SVHN. In the case of CIFAR-100, the  $\epsilon$  is increased by 10 times in log-scale every 27 iterations. In the second experiment, we leave the  $\epsilon$  fixed and simply train the model until the stopping criteria are satisfied. Other details are the same as those of the first experiment.

Table 4: Classification error and the number of added unlabeled data of softmax and sigmoid for new SSL problems on CIFAR-10 with 5 runs.

method	Error		Added data	
	softmax	sigmoid	softmax	sigmoid
supervised	22.27 ± 0.47%		-	
0%	18.27 ± 0.52%	17.84 ± 0.39%	4,306	2,338
25%	18.35 ± 0.86%	18.38 ± 0.52%	3,350	1,470
50%	18.72 ± 0.36%	19.04 ± 0.63%	2,580	811
75%	20.33 ± 0.82%	20.07 ± 0.98%	1,711	315
100%	20.71 ± 0.19%	20.24 ± 0.15%	864	1

## Data Details

We have experimented with CIFAR-10, SVHN, and CIFAR-100 datasets that consist of  $32 \times 32$  pixel RGB images. CIFAR-10 and SVHN have 10 classes and CIFAR-100 has 100 classes. Overall, standard data normalization and augmentation scheme are used. For data augmentation, we used random horizontal flipping and random translation by up to 2 pixels. In the case of SVHN, random horizontal flipping is not used. To show that the SST algorithm is comparable to the conventional SSL algorithms, we experimented with the popular setting (Laine and Aila, 2016; Miyato et al., 2017; Tarvainen and Valpola, 2017). The validation set in the cross-validation to obtain the reduction rate of  $\epsilon$  is extracted from the training set by 5000 images. After the  $\epsilon$  is obtained, all the training datasets are used. The following is the standard labeled/unlabeled split.

**CIFAR-10:** 4k labeled data (400 images per class), 46k unlabeled data (4,600 images per class), and 10k test data.

**SVHN:** 1k labeled data (100 images per class), 72,257 unlabeled data (it is not well balanced), and 26,032 test data.

**CIFAR-100:** 10k labeled data (100 images per class), 40k unlabeled data (400 images per class), and 10k test data.

## Comparison of Softmax with Sigmoid

Eq 3 and Eq 4 are the formula of softmax function and sigmoid function, respectively. Eq 3 can be represented in the form shown in Eq 5.

$$\text{softmax}_i(v) = \frac{e^{v_i}}{\sum_j e^{v_j}} \quad (3)$$

$$\text{sigmoid}(v) = \frac{1}{1 + e^{-v}} \quad (4)$$

$$\text{softmax}_i(v) = \frac{1}{1 + (e^{-v_i}) \times (\sum_{j=1}^{i-1} e^{v_j} + \sum_{j=i+1}^J e^{v_j})} \quad (5)$$

If  $v_i$  is comparably larger than the other  $v_j$ , the softmax function performs like a sigmoid function. Also, even if  $v_i$  with moderately high values, the softmax output still becomes close to 1 when having relatively and extremely small values of  $v_j$ . Eq 6 represents such case.

$$\text{softmax}_i(v) \approx \frac{1}{1 + (e^{-v_i}) \times 0} = 1 \quad (6)$$

We experiment and compare softmax outputs of  $f_{cl}(\cdot; \theta_c)$  against sigmoid outputs of  $f_{sel}(\cdot; \theta_s)$  when sampling unlabeled data. Table 4 shows the classification error and the number of added unlabeled data with sampling based on outputs of the softmax and the sigmoid. The threshold for sampling in softmax output is set same to the sigmoid threshold. Although the thresholds are very high enough, in softmax case, an average of 864 unlabeled datas are added for the case of 100% of the non-animal data. Furthermore, with 0% of the non-animal data, error rate of using softmax is larger than that of using sigmoid even when the added data is larger. To the best of our knowledge, the addition of data with high softmax outputs does not affect the loss much, which leads to the small performance improvement. This shows the limitation of thresholding with softmax.