SELECTIVE SELF-TRAINING FOR SEMI-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Most of the conventional semi-supervised learning (SSL) methods assume that the classes of unlabeled data are contained in the set of classes of labeled data. In addition, these methods do not discriminate 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 or not. It is also designed to be applied to a more realistic situation where classes of unlabeled data are different from the ones of the labeled data. For the conventional SSL problems of fixed classes, the proposed method not only performs comparable to other conventional SSL algorithms, but also can be combined with other SSL algorithms. For the new SSL problems of increased classes where the conventional methods cannot be applied, the proposed 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 successes in various fields, and high-quality data is one of the most important factors for the successes (Everingham et al., 2010; Krizhevsky et al., 2012; Russakovsky et al., 2015). However, since we do not know the underlying real distribution of data, we need a lot of samples to estimate it correctly (Nasrabadi, 2007). On the other hand, creating a large amount of dataset requires huge amount of time, cost and manpower (Odena et al., 2018).

Semi-supervised learning (SSL) is a method of solving the problem of inefficiencies in data collection and annotation processes, which lies in between the supervised learning and unsupervised learning in that both the labeled and unlabeled data are used in the learning process (Chapelle et al., 2009; Odena et al., 2018). It efficiently improves the performance of a model by applying a supervised learning method to unlabeled data with an appropriate loss function. Also, it has been studied in various ways and has greatly improved performances (Zhu et al., 2003; Rosenberg et al., 2005; Kingma et al., 2014; Rasmus et al., 2015; Odena, 2016).

However, there is a recent research discussing the limitations of conventional SSL methods (Odena et al., 2018). They pointed out that conventional SSL algorithms are difficult to be applied to real applications. The conventional methods assume that all the unlabeled data belong to one of the classes of the training labeled data. Therefore, if there are unlabeled data that do not belong to the training classes, the performance of traditional SSL techniques may degrade. We could experimentally find some algorithms that do not degrade performance much in this situation, although they are not suitable for incremental learning which is a required aspect in many real applications. Also, whenever a new set of data is available, they should be trained from the start using all the data including out-of-class[1] data.

In this paper, we propose a method named as *selective self-training* (SST) to solve the limitations mentioned above. Unlike the conventional methods, our algorithm does not use all of the unlabeled

[1]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 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.

data for the training, and it rather uses them selectively. By the proposed selection method, reliable samples can be added to the training set, and uncertain samples including out-of-class data may be excluded. SST does not use the output of a classification network for data selection. Instead, we adopt a separate *selection network* so that the network decides whether to add each sample or not. Our selection network does not utilize softmax function because it outputs a relative value and it can output a high value even for an out-of-class sample. Instead, our selection network is designed to output an absolute confidence score using the sigmoid activation function.

SST requires fewer hyper-parameters and settings than other algorithms. Conventional algorithms require various hyper-parameters and detailed settings (Salimans & Kingma, 2016; Laine & Aila, 2016; Luo et al., 2017). On the other hand, in the SST, only one hyper-parameter of selection threshold is added to the ones for the supervised learning. In addition, the proposed SST is suitable for lifelong learning which make use of more knowledge from previously acquired knowledge (Thrun & Mitchell, 1995; Carlson et al., 2010; Chen & Liu, 2018). Since SSL can be learned with unlabeled data, any algorithm for SSL may seem appropriate for lifelong learning. But conventional SSL algorithms are inefficient when out-of-class samples are included in the unlabeled data. However, SST can only add samples highly related with in-class data and is suitable for lifelong learning.

SST can be categorized as a self-training method (McLachlan, 1975; Zhu, 2007; Zhu & Goldberg, 2009), but it is a new approach to add data using the selection network. With the new approach, SST has the following advantages.

- For the conventional SSL problems of fixed classes, 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 of increased classes where the conventional methods cannot be applied, the proposed SSL method does not show any performance degradation even for the out-of-class data.
- SST does not require many hyper-parameters and can be easily implemented.
- SST is more suitable for lifelong learning compared to other SSL algorithms.

2 BACKGROUND

In this section, we introduce the background of our research. First, we introduce the methods of self-training (McLachlan, 1975; Zhu, 2007; Zhu & Goldberg, 2009) on which our work is based. Then we describe consistency regularization-based algorithms such as temporal ensembling (Laine & Aila, 2016) and Smooth Neighbors on Teacher Graphs (SNTG) (Luo et al., 2017).

2.1 SELF-TRAINING

The self-training method has long been used for semi-supervised learning (McLachlan, 1975; Zhu, 2007; Zhu & Goldberg, 2009). Self-training is an iterative process, where an algorithm sets a pseudo-labels which have been classified in the previous iteration (Chapelle et al., 2009). An overview of the self-training process is as follows:

1. train a model with labeled data
2. predict the unlabeled data with the learned model
3. retrain the model with labeled data and all the predicted unlabeled data
4. repeat the above process

Since our proposed algorithm is based on self-training, it may be helpful to look at our overview in Figure 1. Self-training is easy to implement and apply to various data. However, the addition of erroneous data can spoil the network. As shown in Figure 1, there are cases where a circle is predicted as blue or green in the unlabeled dataset. In self-training, these circles will be trained incorrectly. Furthermore, other classes with different distribution such as star can be trained to green.

2.2 CONSISTENCY REGULARIZATION

Consistency regularization is one of the popular SSL methods, and many recent researches have been conducted (Laine & Aila, 2016; Miyato et al., 2017; Tarvainen & Valpola, 2017). Among

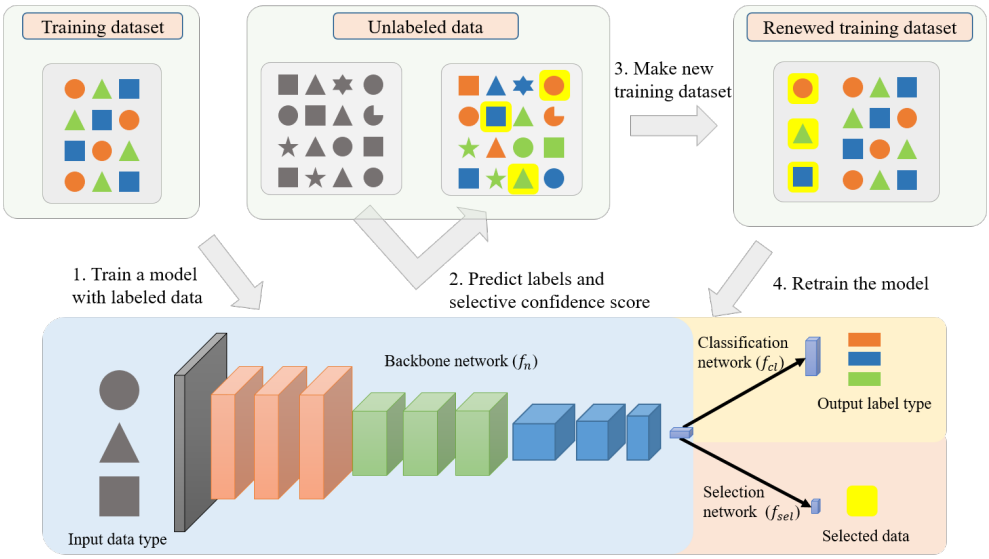


Figure 1: An overview of the proposed SST. Different shapes represent the input data with different underlying distribution, and different colors (orange, blue, and green) are for different classes. In the initial training dataset, only three classes with their corresponding distributions (○, □, △) exist and are used for initial training. Then the unlabeled data which include unseen distribution (★, ✡) 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.

them, Π model and temporal ensembling are widely used (Laine & Aila, 2016). They defined new types of loss functions for unlabeled data. The Π model outputs $f(x)$ and $f(\hat{x})$ according to input x through different random noise and dropout (Srivastava et al., 2014), and minimizes the difference between these values. Temporal ensembling does not make different predictions $f(x)$ and $f(\hat{x})$, but minimizes the difference between the previous output $f_{t-1}(x)$ and the current one $f_t(x)$ for computational efficiency. These methods have greatly improved performance. However, there are lots of things to consider for training, and it is difficult to optimize each hyper-parameter. These methods have various hyper-parameters such as 'ramp 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 & 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. Figure 1 shows an overview of the proposed SST method, which can be represented in four steps. First, SST trains the model using the labeled dataset by an ordinary supervised learning method. The next step is to predict all the unlabeled data and select a partial set of data which has high confidence with the current trained model. In doing so, the confidence is outputted by a separate selection network. Then, the selected data is added to the training dataset by labeling it as estimated. Finally, the model is re-trained with the new training dataset. The overall process of SST is described in Algorithm 1 and the details of each of the four phases will be described later.

The proposed architecture consists of a backbone network, a classification network and a selection network as shown in the lower part of Figure 1. Conventionally, the classification network is connected to the backbone network. We additionally adopt a selection network connected to the backbone network.

Algorithm 1 Pseudocode for Selective Self-Training.

procedure SST
  training with supervised learning
  for iteration $t \in \{1, \cdots, num_iterations\}$ **do**
    for sample $i \in$ unlabeled dataset **do**
      predict the unlabeled sample
      if $label^i_{t-1} \neq label^i_t$ **then**
        $ensemble_conf^i_{t-1} = 0$
      end if
      $ensemble_conf^i_t = 0.5 \times ensemble_conf^i_{t-1} + 0.5 \times selection_score$
      if $ensemble_conf^i_t > threshold$ **then**
        add the sample to the new dataset.
      end if
    end for
    data balancing & change threshold
    training with the new training dataset
  end for
end procedure

3.1 SUPERVISED LEARNING

The SST algorithm first trains a model with supervised learning. At this time, not only the backbone network and the classification network but also the selection network are trained. The classification network f_{cl} is trained in the same way as the basic classification method. We use the softmax function for the final activations and cross entropy for the loss. The selection network learns as follows. The labels for training the selection network is obtained through an adversarial way motivated by generative adversarial networks (GAN) (Goodfellow et al., 2014; Yoo et al., 2017). An i-th sample J_i with a class label l_i is fed into the classification network f_{cl}. Then the class label for the selection network is set as:

$$s_i = \begin{cases} 1, & \text{if } r_i = l_i \\ 0, & \text{else,} \end{cases} \tag{1}$$

where r_i represents the output of the classification network, i.e., $r_i = \arg\max f_{cl}(J_i)$. The selection network is trained based on s_i, which is a fully connected network that takes outputs of the final layer of the backbone network f_n as inputs. We use the sigmoid function for the final activations and binary cross entropy for the loss. If the top-1 output value of the classification network is the same as the ground truth label i.e., $s_i = 1$, it is set to the label to be added. Otherwise, the sample is not used for the next iteration of training. Our final loss function is a sum of the classification loss (L_{cl}) and the selection loss (L_{sel}):

$$L_{total} = L_{cl} + L_{sel}. \tag{2}$$

3.2 PREDICTION AND SELECTION

After learning in a supervised manner, SST performs predictions of unlabeled dataset. While so, SST not only predicts scores for the class candidates, but also outputs a value from the selection network. If the value exceeds a certain threshold, the data is added to the training set. It is very important to set the threshold, because it determines the quality of the added unlabeled samples for the next training. During training, if the classifier network works appropriately, training accuracy will be high and the score s_i will have a value close to 1. Therefore, the threshold is set to $1 - \epsilon$ and the data quality can be controlled by changing ϵ.

SST makes use of the previous selection score by taking ensemble of the previous selection score and the current one, which can prevent sudden changes over iterations. In the ensemble process, for each sample, the predicted label of the previous iteration can be compared with the current predicted

Table 1: The number of labeled and unlabeled data in various datasets

Datasets	Synthetic dataset	CIFAR-10	SVHN	CIFAR-100
Labeled data	12	4,000	1,000	10,000
Unlabeled data	5,988	46,000	72,257	40,000
Total data	6,000	50,000	73,257	50,000

label. If this value is different, the ensemble confidence is set to 0 and the sample is excluded from the candidate training set. The detailed ensemble process is described in line 6 to 9 of Algorithm 1.

3.3 New training dataset

After performing the prediction, we add additional selected data to the new training dataset. However, just before adding selected data, we control the number of added samples in consideration of the balance among classes. The amount of added data depends on the distribution of each data. For example, a rigid body such as cars would have a smaller variation than a non-rigid body such as people. In this case, the additional dataset would contain more cars than people. If one class dominates the dataset, the model tends to overfit. Therefore, we balance the number of added samples by adjusting the number of samples for each class to be the fewest number for a class. For other classes, we randomly sample the data and add them to the additional dataset.

3.4 Re-training

After combining the additional data and the existing training dataset to a new training dataset, the model is trained with the new dataset. This process is repeated iteratively.

4 Experiments

In this section, we conduct two types of experiments. First, we evaluate the proposed SST algorithm for the conventional SSL problems of the fixed classes using in-class unlabeled data. Then, SST is tested for the new SSL problems of the increased classes using out-of-class unlabeled data. For comparison with other works, the basic experimental setup was implemented in the same way as the original works, and also the same network structure as in (Laine & Aila, 2016; Tarvainen & Valpola, 2017) was used. Detailed architectures can be found in the supplementary materials.

The basic settings of our experiments are as follows. Different from (Laine & Aila, 2016; Luo et al., 2017), we use stochastic gradient descent (SGD) with a weight decay of 0.0005 as an optimizer. Also, we do not use mean-only batch normalization layer (Salimans & Kingma, 2016) and Gaussian noise. For supervised learning with labeled data, we first trained the model for 300 epochs. After that, we trained the network for 150 epochs with a new training dataset. The learning rate is reduced by 0.1 from 10^{-1} to 10^{-3} in the step decay mode. In 300 epoch training, it decreased by 0.1 at 150 epoch and 225 epoch. In 150 epoch training, it decreased by 0.1 at 75 epoch and 113 epoch. Our supervised learning results in a slightly higher performance than the ones reported in other papers (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Luo et al., 2017) because of different settings such as Gaussian noise of inputs, optimizer, mean only batch normalization and learning rate.

4.1 Conventional SSL problems of fixed classes with in-class unlabeled data

In this setting, since all the classes of the unlabeled data are included in the labeled data, we decreased selection threshold $(1 - \epsilon)$ as the iteration goes on. The values for ϵ were increased from 10^{-5} to 10^{-2} in a log scale for all the datasets except for synthetic ones.

We experiment with a couple of simple synthetic datasets (two moons, four spins) and three popular datasets which are CIFAR-10, SVHN and CIFAR-100 (Netzer et al., 2011; Krizhevsky et al., 2014). The settings of labeled vs. unlabeled data separation for each dataset are shown in Table 1. The results of the synthetic datasets can be found in the supplementary material.

Table 2: Ablation study with 5 runs on CIFAR-10 dataset. 'balance' denotes the usage of data balancing scheme during data addition as described in Sec. 3.3, 'ensemble' is for the usage of previous selection scores as in the 9th line of Algorithm 1, and 'multiplication' is the scheme of multiplying top-1 softmax output of the classifier network to the selection score and use it as a new selection score.

Method	balance	ensemble	multiplication	error
Supervised learning				$18.97 \pm 0.37\%$
SST	x	x	x	$21.44 \pm 4.05\%$
	o	x	x	$14.43 \pm 0.43\%$
	o	o	x	$11.82 \pm 0.40\%$
	o	o	o	$11.86 \pm 0.15\%$

Table 3: 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)*	$18.97 \pm 0.37\%$	$13.45 \pm 0.92\%$	$40.24 \pm 0.45\%$
Supervised (all)*	$5.57 \pm 0.07\%$	$2.87 \pm 0.06\%$	$23.36 \pm 0.27\%$
Mean Teacher (Tarvainen & Valpola, 2017)	$12.31 \pm 0.28\%$	$3.95 \pm 0.21\%$	-
Π model (Laine & Aila, 2016)	$12.36 \pm 0.31\%$	$4.82 \pm 0.17\%$	$39.19 \pm 0.36\%$
TempEns (Laine & Aila, 2016)	$12.16 \pm 0.24\%$	$4.42 \pm 0.16\%$	$38.65 \pm 0.51\%$
TempEns + SNTG (Luo et al., 2017)	$10.93 \pm 0.14\%$	$3.98 \pm 0.21\%$	$40.19 \pm 0.51\%$*
VAT (Miyato et al., 2017)	$11.36 \pm 0.34\%$	$5.42 \pm 0.22\%$	-
VAT + EntMin (Miyato et al., 2017)	$10.55 \pm 0.05\%$	$3.86 \pm 0.11\%$	-
pseudo-label (Lee, 2013; Odena et al., 2018)	$17.78 \pm 0.57\%$	$7.62 \pm 0.29\%$	-
Proposed method (SST)*	$11.82 \pm 0.40\%$	$6.88 \pm 0.59\%$	$35.20 \pm 0.46\%$
SST + TempEns + SNTG*	$9.99 \pm 0.31\%$	$4.74 \pm 0.19\%$	$35.60 \pm 0.19\%$

4.1.1 Ablation Study

We performed experiments on CIFAR-10 dataset with three types of ablations. As described in Table 2, these are whether to use data balancing scheme described in Section 3.3 (balance), whether to use selection score ensemble in the 9th line of Algorithm 1 (ensemble) and whether to multiply the selection score with the top-1 softmax output of the classifier network to set a new selection score for comparison with the threshold (multiplication). First, when SST did not use all of these, the error 21.44% was lower than that of supervised learning which did not use unlabeled data. This is because the model is spoiled by the added data as described in subsection 3.3. When the data balance is used, the error is 14.43%, which is better than the baseline of supervised learning. Adding the ensemble scheme results in 11.82% error, and the multiplication scheme shows a slight drop in performance. Since all of the experiments use the same threshold, the number of candidate samples to be added is reduced by the multiplication with the top-1 softmax output and the variation becomes smaller because only confident data are added. However, we have not used the multiplication scheme in what follows because the softmax classification output is dominant in multiplication. Therefore, we used only balance and ensemble schemes in the following experiments.

4.1.2 Experimental Results

Table 3 shows the results of supervised learning, conventional SSL algorithms and the proposed SST on CIFAR-10, SVHN and CIFAR-100 datasets. For all the datasets, we have also performed experiments of combining SST with the temporal ensembling (TempEns) as well as SNTG. For the TempEns+SNTG, the pseudo-label of SST's last iteration was used as if it were 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: When supervised learning was trained with 4,000 images, the test error was 18.97%, while it was 5.58% with 50,000 labeled images. The test error in the SST algorithm is 11.82%, which is comparable to other algorithms. The combined SST+TempEns+SNTG results in a test error of 9.99% which is 1.83% better than SST only case. As stated before, TempEns+SNTG was applied once after SST had completed in our experiment. Unlike this, if SST were trained with

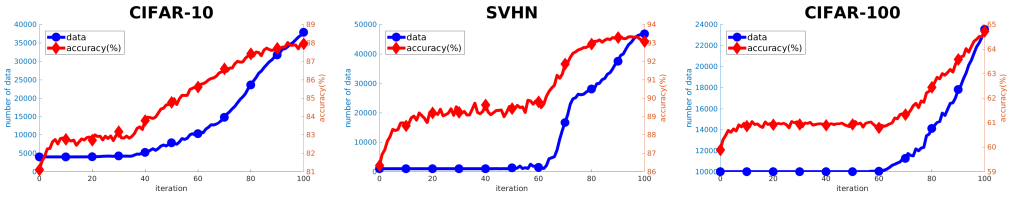
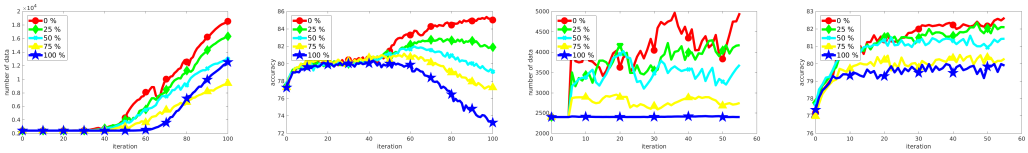


Figure 2: SST result on CIFAR-10, SVHN, and CIFAR-100 datasets with 5 runs. The x-axis is the iteration, the blue circle is the average of the number of data used for training, and the red diamond is the average accuracy.



(a) (b) (c) (d)

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

TempEns+SNTG in every iteration, SST could have obtained better performance. However, this method requires enormous training time and we could not experiment this setting.

SVHN: When supervised learning was trained with 1,000 images, the test error was 13.45%. SST has an error of 6.88% which is relatively higher than those of other SSL algorithms. This is due to the inherent characteristics of the dataset. Unlike other datasets, SVHN dataset is not well balanced. On the other hand, the SST algorithm increases the volume of data with balance as mentioned in Section 3.3. The smallest class contains about 5,000 images in SVHN, and SST has not learned more than 50,000 images during learning. For this reason, the performance is lower than other algorithms. SST with TempEns+SNTG shows 4.74% error, which is better than that of SST-only case but is worse than TempEns+SNTG. Since SVHN is a relatively easy dataset, the accuracy seems to be easily saturated. As shown in Figure 2, in CIFAR-10, as data increases, the accuracy is not fully saturated. On the other hand, SVHN is saturated, and there is no performance improvement when data are added at last iterations. We think this is owing to the use of hard labels in SST where incorrectly estimated samples deteriorate the performance.

CIFAR-100 : We had an error of 40.24% with supervised learning only, and an error of 35.2% when SST was performed. The performance improvement was about 5%. As one can see in the Figure 2, the CIFAR-100 dataset has not been saturated yet. If SST is trained more with a lower threshold, SST may get better performance. We observed that performance was lower than TempEns when we used TempEns+SNTG together. Also, when TempEns+SNTG were added to SST, performance dropped slightly [2]. The reason for this can be conjectured that the hyper-parameter in the current temporal ensembling and SNTG may not have been optimized.

4.2 New SSL problems of increased classes with out-of-class unlabeled data

We experimented with the following settings for real applications. In CIFAR-10, each 400 images of 6 animal classes are used as the labeled data and the other 20,000 data are used as unlabeled ones. In CIFAR-100, each 100 images of 50 animal classes are used as the labeled data and 20,000 unlabeled data are used. Unlike the experimental setting in (Odena et al., 2018), we experiment according to the ratio (%) of out-of-class data in the unlabeled set. More details are in supplementary material.

[2] We reproduced tempEns+SNTG with Pytorch, and showed similar performance to CIFAR-10 and SVHN. In CIFAR-100 case, there is no result using data augmentation for CIFAR-100 in (Luo et al., 2017), and we filled in our own results.

Table 4: Classification error for new SSL problems on CIFAR-10 and CIFAR-100 dataset with 5 runs. '%' means the ratio of non-animal classes.

dataset	CIFAR-10		CIFAR-100	
method	SST(decay)	SST(fixed)	SST(decay)	SST(fixed)
supervised	$22.27 \pm 0.47\%$		$34.62 \pm 1.14\%$	
0%	$14.99 \pm 0.54\%$	$17.33 \pm 0.47\%$	$28.01 \pm 0.44\%$	$32.11 \pm 0.41\%$
25%	$17.93 \pm 0.33\%$	$18.14 \pm 0.47\%$	$29.94 \pm 0.45\%$	$32.11 \pm 0.58\%$
50%	$20.91 \pm 0.53\%$	$18.57 \pm 0.64\%$	$31.78 \pm 0.62\%$	$32.31 \pm 0.59\%$
75%	$22.72 \pm 0.42\%$	$19.73 \pm 0.60\%$	$34.44 \pm 0.85\%$	$32.58 \pm 0.22\%$
100%	$26.78 \pm 1.35\%$	$20.11 \pm 0.53\%$	$37.17 \pm 1.08\%$	$32.84 \pm 0.41\%$

Figure 3 shows the average of the results obtained by performing SST 5 times for each ratio in CIFAR-10. As shown in Figure 3(a), as the number of iterations increases, the threshold decreases and data increases gradually. Obviously, $0 \sim 75\%$ of the non-animal data and 100% of the non-animal data show different trends. In case of $0 \sim 75\%$, the data shows a slight increase from $30 \sim 40$ iterations. On the other hand, in the case of 100%, data is added little by little from $55 \sim 65$ iterations. This tendency has been observed in many previous researches on classification (Viola & Jones, 2001; Zhang & Viola, 2008) and we applied a fixed threshold in a similar way. We have set the fixed threshold to the threshold at 47th iteration. To prevent data being added suddenly, no data was added until 5 iterations.

CIFAR-10 : Table 4 shows performance according to non-animal ratio. In the case of SST (decay), the performance is improved at 0%, but at 100%, the performance is degraded as in other algorithms. On the other hand, in the case of SST (fixed), samples are not added much and performance was improved at 100% non-animal ratio as shown in Figure 3(c). Furthermore, at 0%, there was a more performance improvement than 100 %, but less performance improvement than decay. Because less but stable data are added by fixed SST, 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 unknown.

CIFAR-100 : The fixed threshold at CIFAR-100 is the threshold value at 35 iterations. In the decay mode, the performance is much improved at 0%, and at 100%, the performance is degraded. On the other hand, in the fixed mode, there was no performance degradation from 0% to 100%. In CIFAR-100, the difference between 0% and 100 % is less than CIFAR-10, because the gap between animal and non-animal is small and additional data is small. More details are in supplementary material.

5 Conclusion

We proposed selective self-training (SST) for semi-supervised learning (SSL) problem. Unlike conventional methods, 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 have shown performance improvements as 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 like other algorithms. 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 have 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.

REFERENCES

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, pp. 3. Atlanta, 2010.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

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

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, pp. 2, 2013.

Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. *arXiv preprint arXiv:1711.00258*, 2017.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

Geoffrey J McLachlan. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

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.

Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16 (4):049901, 2007.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5, 2011.

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

Augustus Odena, Avital Oliver, Colin Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of semi-supervised learning algorithms. 2018.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pp. 3546–3554, 2015.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *WACV/MOTION*, pp. 29–36, 2005.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 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*, pp. 1195–1204, 2017.

Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pp. 165–196. Springer, 1995.

Paul Viola and Michael Jones. 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, pp. I–I. IEEE, 2001.

YoungJoon Yoo, Seonguk Park, Junyoung Choi, Sangdoo Yun, and Nojun Kwak. Butterfly effect: Bidirectional control of classification performance by small additive perturbation. *arXiv preprint arXiv:1711.09681*, 2017.

Cha Zhang and Paul A Viola. Multiple-instance pruning for learning efficient cascade detectors. In *Advances in neural information processing systems*, pp. 1681–1688, 2008.

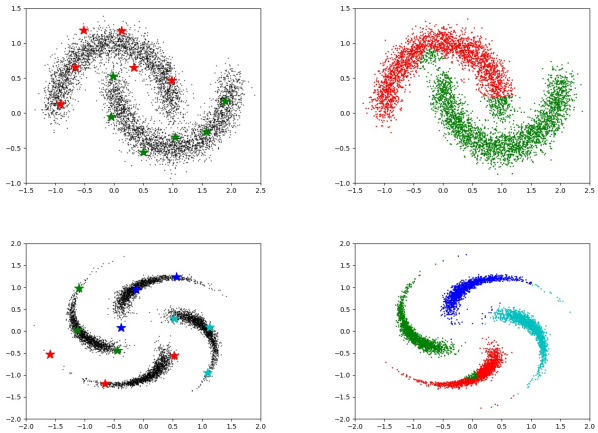
Xiaojin Zhu. Semi-supervised learning tutorial. In *International Conference on Machine Learning (ICML)*, pp. 1–135, 2007.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

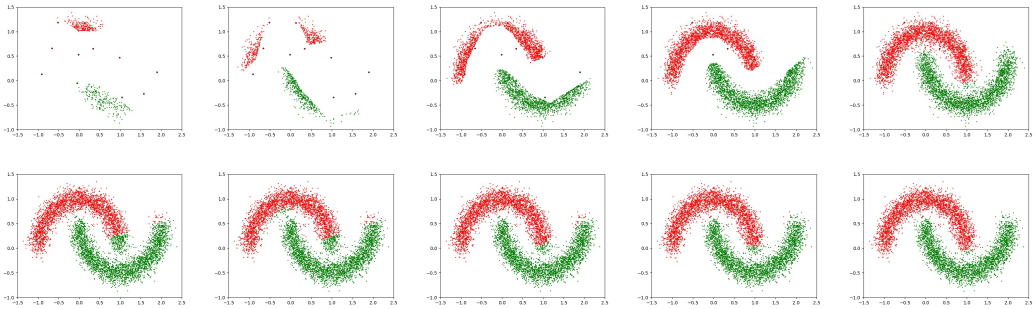
6 SUPPLEMENTARY MATERIAL

6.1 SYNTHETIC DATASETS



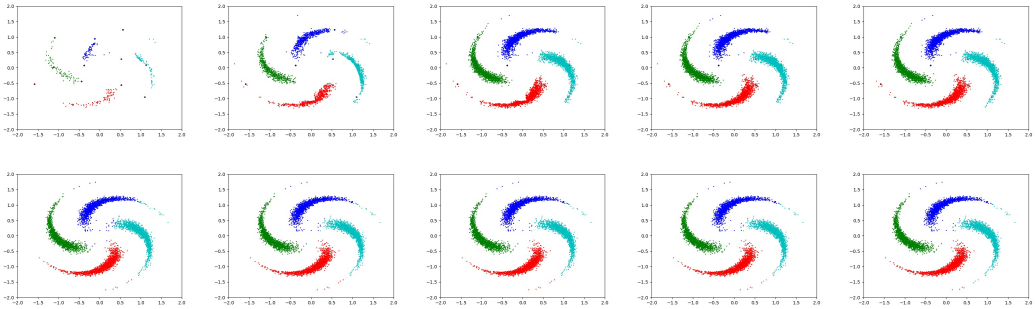
(a) (b)

Figure 4: Synthetic datasets (Top : Two moons, Bottom : 4 spins) (a): training dataset (black point : unlabeled data point, color point : labeled data point), (b) test dataset result



$iter = 31$ $iter = 36$ $iter = 41$ $iter = 46$ $iter = 50$

Figure 5: Synthetic datasets (Two moons) Top : training dataset, Bottom : test result



$iter = 36$ $iter = 39$ $iter = 41$ $iter = 46$ $iter = 50$

Figure 6: Synthetic datasets (Four spin) Top : training dataset, Bottom : test result

Table 5: Network structure for synthetic datasets

Type	Filter Shape	Output Size
input	(x,y) point	2
fully connected	2×30	30
fully connected	30×30	30
classification outputs	$30 \times$ classes (softmax)	classes
selecting outputs	30×1 (sigmoid)	1

Table 6: Network structure for CIFAR-10, SVHN, and CIFAR-100 datasets

Type	Filter Shape / pad	Output Size
input	RGB channel images	$32 \times 32 \times 3$
conv2d	$3 \times 3 \times 3 \times 128$ / same	$32 \times 32 \times 128$
conv2d	$3 \times 3 \times 128 \times 128$ / same	$32 \times 32 \times 128$
conv2d	$3 \times 3 \times 128 \times 128$ / same	$32 \times 32 \times 128$
pool	maxpool 2×2	$16 \times 16 \times 128$
dropout	p = 0.5	$16 \times 16 \times 128$
conv2d	$3 \times 3 \times 128 \times 256$ / same	$16 \times 16 \times 256$
conv2d	$3 \times 3 \times 256 \times 256$ / same	$16 \times 16 \times 256$
conv2d	$3 \times 3 \times 256 \times 256$ / same	$16 \times 16 \times 256$
pool	maxpool 2×2	$8 \times 8 \times 256$
dropout	p = 0.5	$8 \times 8 \times 256$
conv2d	$3 \times 3 \times 256 \times 512$ / valid	$6 \times 6 \times 512$
conv2d	$3 \times 3 \times 512 \times 256$ / same	$6 \times 6 \times 256$
conv2d	$3 \times 3 \times 256 \times 128$ / same	$6 \times 6 \times 128$
pool	average pool 6×6	$1 \times 1 \times 128$
classification outputs	$128 \times$ classes (softmax)	$1 \times 1 \times$ classes
selecting outputs	128×1 (sigmoid)	$1 \times 1 \times 1$

As synthetic datasets, two moons and 4 spins were tested in the same manner as SNTG(Luo et al., 2017). Each dataset has 6,000 training and 6,000 test samples. In the case of two moons, there are two classes $y \in \{0, 1\}$, and in case of 4 spins, $y \in \{0, 1, 2, 3\}$. In 6,000 training data, there are 12 labeled data and 5,988 unlabeled data. Thus, for two moons, each class has 6 points and for 4 spins, each class has 3 points. Because the number of labeled datapoints are too small, random sampling can lead to sample similar points. Therefore, we randomly sampled the labeled data with a constraint that the Euclidian distance of each data point is greater than 0.7. For these datasets, total iteration was performed 50 times, and the ϵ was increased from 10^{-7} to $10^{-4.5}$ on a log scale.

Figure 4 shows the basic setting of the synthetic dataset, and Figure 5 and 6 show the progress of the SST algorithm. The SST algorithm improves performance by gradually expanding certain data in a synthetic dataset.

6.2 Network structure

We used two types of networks. The network for training the synthetic dataset is shown in Table 5 and consists of two hidden layers with 30 nodes. The network structure for CIFAR-10, SVHN, and CIFAR-100 consists of convolutions, and its structure is shown in Table 6. We used basic batch normalization (Ioffe & Szegedy, 2015) and Leaky ReLU (Maas et al., 2013) with α = 0.1.

6.3 Data setting for New SSL problem of increased classes

(Odena et al., 2018) adds only four unlabeled classes and tests according to the radio of unlabeled class. For example, at 50%, two classes are in-class, and two classes are out-of-class. However, we experimented with the ratio of non-animal data. Thus at 50% in CIFAR-10, unlabeled data consists of 50% in-class and 50% out-of-class. The data for each ratio are shown in Table 7, and the data category for animal and non-animal is shown in Table 8.

Table 7: Number of each class data for new SSL problems

Dataset	Ratio	Labeled data	Unlabeled data	
		Animal	Animal	Non-Animal
CIFAR-10	0%	400	3334 or 3333	0
	25%	400	2500	1250
	50%	400	1667 or 1666	2500
	75%	400	834 or 833	3750
	100%	400	0	5000
CIFAR-100	0%	100	400	0
	25%	100	300	100
	50%	100	200	200
	75%	100	100	300
	100%	100	0	400

Table 8: Data category for new SSL problems

Type	Animal	Non-Animal
CIFAR-10 (CLASS)	bird, cat, deer, dog, frog, horse	airplane, automobile, ship, truck
CIFAR-100 (SUPERCLASS)	aquatic mammals, fish, insects, large carnivores, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals	flowers, food containers, fruit and vegetables, household electrical devices, household furniture, large man-made outdoor things, large natural outdoor scenes, trees, vehicles 1, vehicles 2

6.4 Other algorithms for new SSL problems of increased classes

Table 9 shows the results of a general test on other algorithms. First, self-training (McLachlan, 1975; Zhu, 2007; Zhu & Goldberg, 2009) does not improve performance even at 0%, and performance at 100% is low. When SST is applied to the softmax output as a threshold without selection network, the performance is improved at 0%, but the performance is degraded at 100%. There is a confidence of 99% or more, but because it is a relative value, a lot of bad data is added. In the case of temporal ensembling, performance is not compromised by adding non-animal data. However, as mentioned in this paper, it is inefficient because all the other classes need to be trained.

6.5 New SSL problems of increased classes in CIFAR-100

In case of $0 \sim 75\%$, number of data shows a slight increase from about 30 iterations. On the other hand, in the case of 100%, selected samples are added little by little from about 40 iterations. The fixed threshold set to threshold of 35 iterations. To prevent data being added suddenly, we do not add samples until 5 iterations. Figure 7 shows the experimental results. As mentioned in our paper, at 100% with fixed threshold, the data is not added much and the performance does not degrade.

Table 9: Other algorithms for new SSL problems of increased classes with 5 runs

method	self-training	softmax output(0.99)	Temporal ensembling
supervised	$22.27 \pm 0.47\%$	$22.27 \pm 0.47\%$	$24.10 \pm 0.66\%$
0%	$21.97 \pm 0.24\%$	$15.26 \pm 0.64\%$	$14.85 \pm 0.42\%$
25%	$22.80 \pm 0.39\%$	$19.21 \pm 0.71\%$	$16.02 \pm 0.43\%$
50%	$23.93 \pm 0.71\%$	$21.43 \pm 0.67\%$	$17.16 \pm 0.70\%$
75%	$25.45 \pm 0.47\%$	$24.38 \pm 0.76\%$	$18.58 \pm 0.49\%$
100%	$27.31 \pm 0.57\%$	$27.72 \pm 0.71\%$	$20.42 \pm 0.93\%$

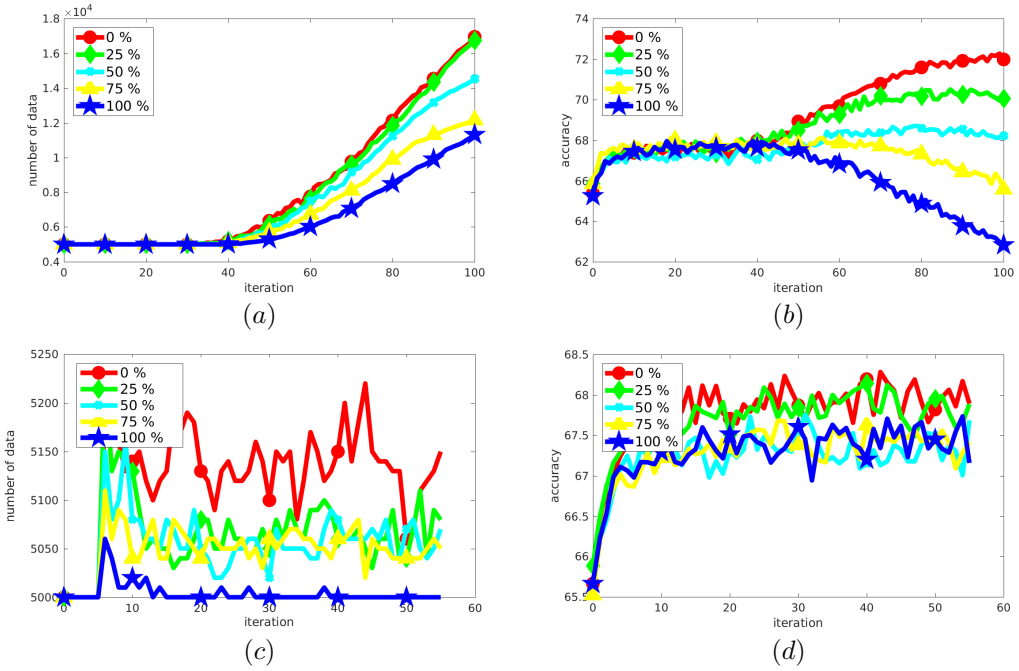


Figure 7: Result of new SSL problems on CIFAR-100 dataset with 5 runs. (a) number of data with iteration in decaying threshold (b) accuracy with iteration in decaying threshold (c) number of data with iteration in fixed threshold (d) accuracy with iteration in fixed threshold. % means the ratio of non-animal classes in the unlabeled data.