Graduate School of Advanced Science and Engineering
Hiroshima University

Thesis submitted for the degree

Doctor of Philosophy in Informatics and Data Science

Methods for Robust Training of Deep Neural Networks in the Presence of Noisy Labels
誤りを含む教師信号からの深層学習の頑健な訓練法

by

Yuichiro Nomura

Supervisor: **Takio Kurita**

March 2023

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Date: March 2023

Signature: Yuichiro Nomura

Abstract

Methods for Robust Training of Deep Neural Networks in the Presence of Noisy Labels

誤りを含む教師信号からの深層学習の頑健な訓練法

In recent years, deep neural networks have achieved significant results in a variety of machine learning domains, including computer vision, natural language processing, and recommendation systems. Despite their efficacy, deep neural networks require a significant amount of data to train the models effectively. To solve this issue, some in-expensive data collection techniques such as web crawling have been developed. However, there is a risk that these data collection techniques may generate incorrect labels. When deep neural networks models for image classification are trained on datasets with such noisy labels, their generalization performance is significantly hindered. This problem is referred to as Learning with Noisy Labels (LNL).

To mitigate the degradation of generalization performance caused by noisy labels, we proposed three different methods to prevent a model from over-fitting to such noisy labels for image classification. The first approach utilizes feature vectors of training samples extracted from the hidden layer of a deep neural network model to construct an affinity graph based on the similarity between pairs of feature vectors. The model is then robustly trained by iteratively correcting noisy labels through graph label propagation on the affinity graph and updating the network parameters.

The second approach is based on one of the recent researches in LNL, known as DivideMix[33], which effectively partitions the dataset into samples with clean labels and those with noisy labels by modeling the loss distribution of all training samples with a two-component Mixture Gaussian model (GMM). The divided dataset is then treated as labeled and unlabeled samples and the classification model is trained in a semi-supervised manner. However, since the selected samples have lower loss values and are easy to classify, the training models are in risk of over-fitting to the simple patterns during training. To train the classification model without overfitting to the simple patterns, we propose to introduce *consistency regularization* on the selected samples by GMM. The consistency regularization perturbs input images and encourages the model to output consistent values to the perturbed images and original images. The classification model simultaneously receives the samples selected as clean and their perturbed ones, and it achieves higher generalization performance with less over-fitting to the selected samples.

The third approach exploits the property that deep neural networks are robust to noisy labels in the early stages of learning. Recent studies have shown that deep neural networks are robust to the noisy labels in the early stage of learning before over-fitting to noisy labels, as deep neural networks learn the simple patterns first. Therefore deep neural networks tend to output true labels for samples with noisy labels during the early stage of learning, and the number of false predictions for samples with noisy labels is higher than for samples with clean labels. Based on these observations, we propose a new sample selection approach for LNL utilizing the number of false predictions. Our method periodically collects records of false predictions during training and select samples with a low number of false predictions from recent records. Then our method iteratively performs sample selection and trains a deep neural networks model using the updated dataset. As the model is trained with more clean samples and records more accurate false predictions for sample selection, the generalization performance of the model gradually increases.

We demonstrate that our methods yield superior performance compared to the baseline and several state-of-the-art methods in image classification tasks with noisy labels

Acknowledgment

First and foremost, I would like to extend my heartfelt appreciation to Professor Takio Kurita of the Graduate School of Advanced Science and Engineering at Hiroshima University, for his invaluable guidance and support throughout my PhD journey. He guided me into the realm of pattern recognition research and gave me a comprehensive understanding of the research process and its intrinsic appeal. His expertise in pattern recognition is truly profound, and he frequently consulted with me during my doctoral studies, generously sharing his knowledge with me.

I would also like to extend my appreciation Professors Junichi Miyao, Bisser R. Raytchev, Naoki Honda, and Hiroaki Mukaidani of Hiroshima University for their insightful comments and constructive feedback on my research, which helped to refine my manuscript.

I am also grateful to my colleagues in the Pattern Recognition Laboratory at Hiroshima University, with whom I have shared research ideas and daily experiences, and have grown both as a researcher and as a person.

I would like to extend my gratitude to my friends and family for their unwavering support and encouragement during my PhD studies.

Finally, I would like to extend my gratitude to all those who participated in my research for their time and willingness to share their knowledge with me. This thesis would not have been possible without the support and assistance of all these individuals and I am deeply indebted to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Impact of Deep Neural Networks on Society

Artificial intelligence is an attempt to artificially imitate the functions of the human brain, and in recent years it has rapidly begun to proliferate in many areas of our lives. One of the fundamental technologies for achieving artificial intelligence is machine learning, which utilizes training data to discover patterns and automatically determine the parameters of a model. Machine learning is a highly valuable tool, and is applied to essential technologies and services such as cameras, smartphones, and social media [31]. Specific applications of machine learning include e-commerce recommendation systems, object recognition in images, face detection, automatic language translation, speech to text, and matching between products and users. In all of these applications, the use of deep neural networks (DNNs) which have evolved with the availability of large scale datasets and advancements in computational resources, plays an important role.

DNNs are a machine learning model that consists of multiple hierarchical processing layers, enabling the learning of representations of data at multiple levels of abstraction [31]. Each layer of the DNN tunes the parameters of the model to output the target signal based on the representation of the input data obtained from the lower layers. The enormous combination of such transformations allows DNNs to learn and represent highly complex functions that can perform pattern recognition tasks. The strength of DNNs is that it can learn such complex functions from data automatically, without requiring careful engineering and significant expertise. DNNs outperformed conventional methods in image classification accuracy in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 [27], and since then has attracted worldwide attention as a superior machine learning method. Since then, DNNs have continued to solve problems that artificial intelligence has faced over the years, such as image recognition[27, 30, 61, 64] and speech recognition [19, 40, 56], and are expected to achieve greater success in the near future.

1.2 Preparation of training dataset for DNNs

DNNs is used in diverse fields such as information retrieval [47, 48, 77], computer vision [27, 29, 52, 53], speech recognition [19, 40, 56], and natural language processing [11, 20, 57]. One drawback of DNNs is that it requires a large amount of accurately labeled training data in order to train DNNs stably. The label of a data sample refers to the category to which the sample belongs and is mainly used for classification training. Usually, in order to prepare those data sets, we ask experts to label them accurately, but the problem is that it takes an enormous amount of time and money to label all the samples. To alleviate such high costs, non-expert technologies have been developed, such as Amazon's mechanical Turk, which generates data by asking anonymous amateurs to fill out questionnaires, and web crawling, which automatically collects labels for data from keywords surrounding images on the web [63, 68, 69]. An example of a dataset collected with these techniques is ImageNet, a dataset containing 15 million high-resolution images in 22,000 categories, which was used to train AlexNet, the winner of image classification competition ILSVRC2012 [27]. T.Xiao et al.[68] collected 10 million images of clothing from various online shopping websites and automatically categorized the clothing based on keywords surrounding the images. This dataset, titled "Clothing1M," is primarily utilized

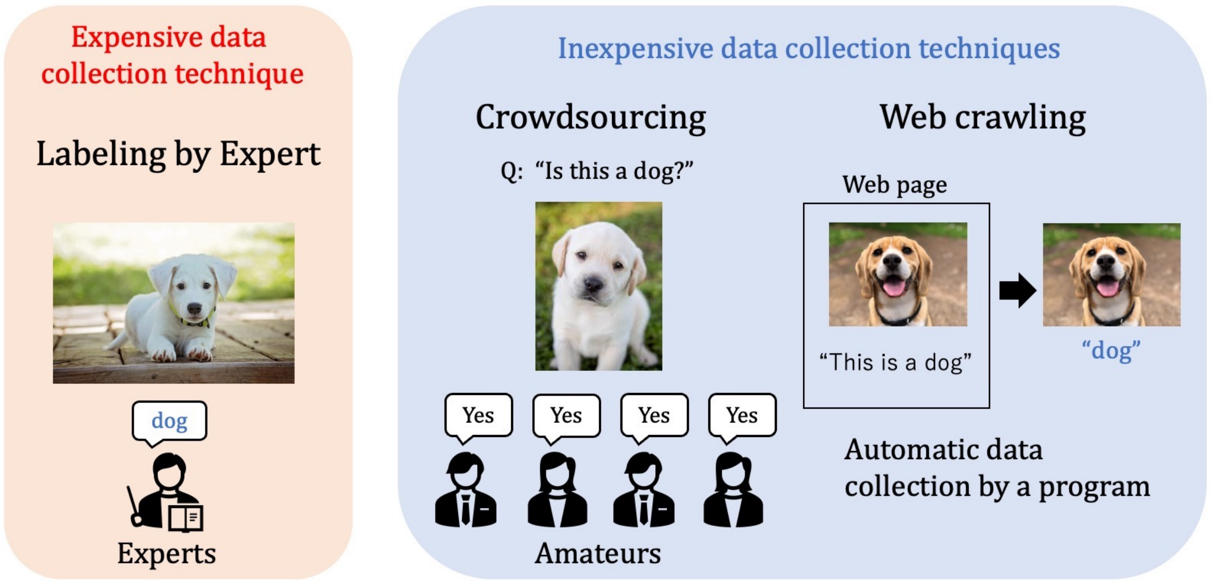


Figure 1.1: Examples of Labeling Strategies

for image classification of clothing. Fig.1.1 shows some examples of labeling strategies for data collection.

1.3 DNNs and Noisy Labels

1.3.1 Introduction to Noisy Labels

Non-expert data collection techniques such as crowdsourcing have contributed greatly to the development of DNNs by creating large datasets. However, the labels assigned by these non-experts are inaccurate compared to those assigned by experts, and the assigned labels may contain errors [6, 8, 39, 49, 59, 63, 68, 69, 72]. For example, in ImageNet collected by Amazon Mechanical Turk, incorrect labels are generated due to insufficient knowledge of workers and insufficient explanation of label assignment [74]. Even when labels are assigned by experts, mislabeling occurs when the quality of the data provided to the experts is poor or when samples are difficult to discriminate [14]. The medical imaging datasets can also have problems with mislabeling due to different criteria for judging abnormalities depending on the skill of the physicians [23]. Label errors may

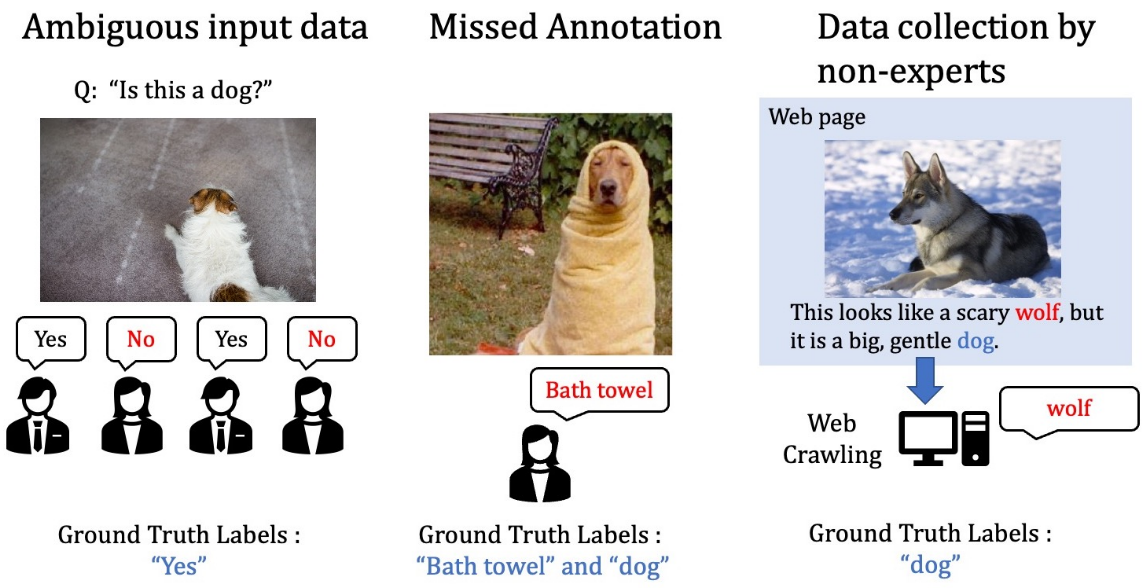

Ambiguous input data
Q: "Is this a dog?"
Yes
No
Yes
No
Ground Truth Labels : "Yes"
Missed Annotation
Bath towel
Ground Truth Labels : "Bath towel" and "dog"
Data collection by non-experts
Web page
This looks like a scary wolf, but it is a big, gentle dog.
Web Crawling
wolf
Ground Truth Labels : "dog"

Figure 1.2: Examples of Source of Noisy Labels

also occur simply due to data encoding errors or communication problems [14]. These mislabeled labels, which differ from the ground-truth, are called "noisy labels". Since it is difficult for humans to accurately label all data when they are involved in labeling, noisy labels are a practically unavoidable problem in real-world applications of machine learning. Fig.1.2 shows some examples of source of noisy labels. In fact, analyses of real-world datasets containing noisy labels have reported that the percentage of samples with noisy labels ranges from 8.0% to 38.5% [32, 34, 59, 60, 68].

1.3.2 Impact of noisy labels on classification with machine learning and DNNs

One of the tasks in which machine learning has been particularly successful is classification learning. This is the problem of training a model to learn patterns for each category from training data and to predict the category to which a sample of test data belongs. However, the performance of machine learning models depends on the quality of the labeled dataset, and if the teacher signals (labels) are incorrect, the performance will be significantly degraded. Since noisy labels are an inherent problem, learning methods robust against

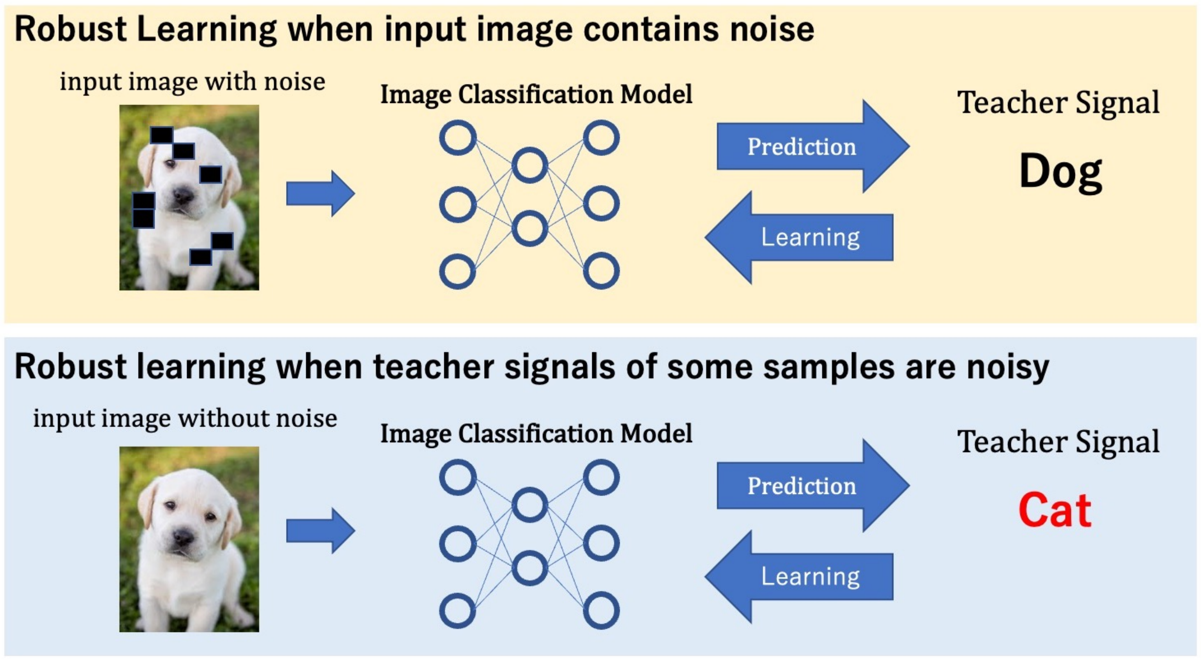

Robust Learning when input image contains noise
input image with noise
Image Classification Model
Prediction
Learning
Teacher Signal
Dog
Robust learning when teacher signals of some samples are noisy
input image without noise
Image Classification Model
Prediction
Learning
Teacher Signal
Cat

Figure 1.3: In this thesis, we focus on the case where noise is present in the teacher signal, rather than the case where noise is present in the input space. Above figure shows an example of robust learning for noisy input. The lower figure shows an example of robust learning with noise in the teacher signal, which is the subject of this thesis.

the noisy labels are crucial for machine learning. Let y be the category to which the input data sample x truly belongs. Let $\tilde{y}$ be the label incorrectly assigned to sample x. If a machine learning classifier is trained with noisy labels, it learns the wrong posterior probability $p(\tilde{y}|x)$ rather than the true posterior probability $p(y|x)$. Consequently, the classifier makes incorrect predictions on the test images. In this thesis, we assume that the training data $(x, \tilde{y})$ is sampled from the wrong distribution and there is noise in the labels, but no noise in the input data features. Fig.1.3 clearly illustrates the distinction between general robust learning for input noise and the robust learning examined in this thesis. In general robust learning, the classifier is trained to correctly output the value of the teacher signal even if there are missing or noisy samples in the input space. In this thesis, we assume that the noise is not in the input space, but rather in the teacher signals of some of the input samples. As shown in the lower part of Fig.1.3, the input image has no noise, but the teacher signal contains errors.

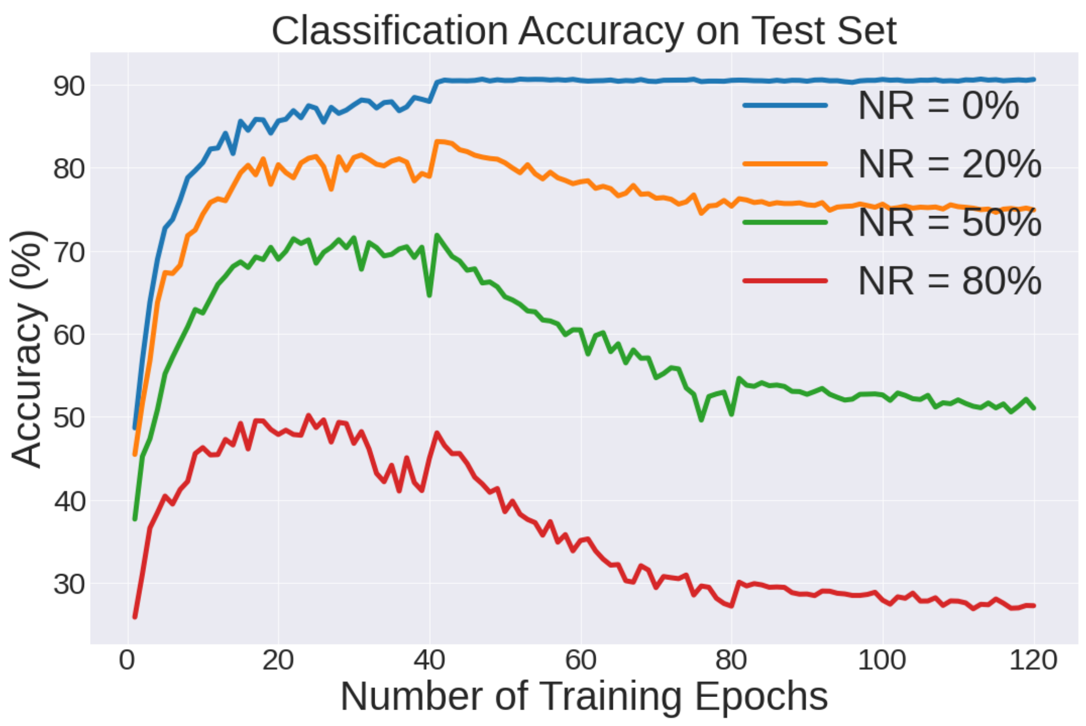


Figure 1.4: This figure shows the classification accuracy on a testset performed by DNNs models trained on a dataset with noisy labels with different noise rates. The y-axis and x-axis represent the classification accuracy on test set and the number of training epochs during training, respectively, and NR denotes the noise rate in training set.

In particular, DNNs models have been reported to overfit easily to label noise because of the large number of parameters in the model and the ability to represent any complex function [2, 25]. When DNNs models overfit to noisy labels, generalization performance deteriorates, resulting in poor performance on test data [75]. This property of DNNs that tends to overfit label noise is called the memorization effect [2, 37]. Analysis of the memorization effect revealed that DNNs learn simple patterns of the input data in the early stages of learning, resulting in high generalization performance. After learning those patterns, DNNs start overfitting to samples with noisy labels and deteriorates its performance.

Fig.1.4 shows the classification accuracy on the test set of DNNs using the CIFAR-10 dataset as training data with artificial noisy labels. NR denotes noise rate, and is the percentage of samples with noisy labels in the training dataset. The noisy labels were assigned by randomly selecting NR% of the training samples and randomly assigning new labels to them. The training model is a DNNs, and model parameters are tuned by the stochastic gradient descent. The accuracy curve with 0% NR corresponds to the classification accuracy of a model trained on a clean dataset and maintains higher classification accuracy during training. However, as the value of NR increases, the proportion of noisy labels in the training data increases, simultaneously decreasing classification accuracy. It is clear that label noise has a negative impact on the generalization performance of DNNs, indicating that a robust learning method against noisy labels is required for applications of DNNs. In this thesis, we describe our proposed methods that reduce the effects of label noise and robustly train the DNNs models.

1.4 Structure of This Thesis

Chapter 2 first provides an overview of image classification and DNNs, where the structure and learning algorithm of DNNs are explained. This is followed by a definition of noisy labels and a description of the problem setup for classification problem using DNNs. In addition, previous studies on label noise using DNNs will be discussed. Chapters 3, 4, and 5 introduce research methods for reducing gaps in generalization performance caused by the presence or absence of noisy labels in image classification problems. Finally, we conclude this thesis by summarizing the contributions of this study and discussing future issues to be addressed in Chapter 6.

1.4.1 Label Noise Removal by Graph Label Propagation

The method described in Chapter 3 [44] proposes a method to update noisy labels to clean labels during the training of DNNs model in order to alleviate the noisy labels problem. The proposed method extracts feature vectors of all training samples from the hidden layer of the DNNs model before the model starts overfitting to noisy labels, and constructs a similarity graph between training samples. Noisy labels are removed by performing label propagation on the graph. The model training is then resumed using the updated data set. By alternatively performing label updates and parameter updates, we achieved a training method that is robust to noisy labels.

We evaluated the performance of the proposed method on MNIST, a handwritten numeric character dataset, and CIFAR-10, a 10-class image dataset, by conducting comparative experiments using artificial label noise. It is found that the proposed method does not overfit to the noisy labels and maintains a high accuracy rate compared to other methods.

1.4.2 Consistency Regularization on Clean Samples

In Chapter 4, we applied the self-supervised learning technique and proposed a method for robustly learning models in the presence of noisy labels in image classification [45]. In the prior study, DivideMix [33] focused on the error value of each sample given by a DNNs model and successfully partitioned the dataset into a clean dataset and a dataset with noisy labels by training a two-component mixture Gaussian model on the distribution of error values. DivideMix treats samples with small error values as clean samples, but samples with small error values tend to have simple patterns that are easy to classify. If a training model focuses on these samples, the model will overfit to the simple patterns and ignore the samples with clean labels but difficult to classify. Therefore, we introduced Consistency Regularization (CR) for samples selected as clean by DivideMix and succeeded in learning clean samples that are difficult to classify, while preventing overfitting to simple patterns.

Artificial label noise was applied to the benchmark image classification datasets CIFAR-10 and CIFAR-100 and our method is compared with DivideMix and State-of-the-art methods. The test set achieved a better classification accuracy than both methods, indicating that CR contributes to the improvement of generalization performance. We also varied the coefficient of CR in the loss function and showed that as the coefficient of CR increased, the classification error for the selected sample increased, indicating that CR is effective in preventing model from overfitting to the clean samples.

1.4.3 Number of False Predictions for Sample Selection

Chapter 5 describes a novel sample selection method for robustly training DNNs in the presence of noisy labels in image classification [46]. DNNs are robust to noisy labels in the early stages of learning because it learns simple patterns first. In other words, since the model predicts the true label for each sample in the early stages of learning, the model makes predictions that are inconsistent with the label given to samples with label noise. For samples with label noise, the total number of false predictions that occur during training is higher than for clean samples. From prior experiments, we observed that the number of false predictions for clean samples is lower than for samples with label noise. Therefore, we proposed a new sample selection method that takes into account the number of false predictions for samples.

Comparative experiments were conducted using CIFAR-10 and CIFAR-100 with artificial label noise. The proposed method achieved a superior classification accuracy than the state-of-the-art sample selection methods, indicating that it is one of the effective sample selection methods. The quality of sample selection during training was also measured by F-score, which showed an improvement in sample selection during training.

Chapter 2

Learning with Noisy Labels

In this chapter, we describe image classification, deep neural networks (DNNs), the definition of noisy labels, and related works on the proposed method described in this thesis.

2.1 Image Classification

Image classification is the problem of predicting the category to which an object in an input image x belongs by feeding the input image x into a classification model. Image classification is one of the most popular problems in the field of machine learning, and various methods, not limited to DNNs, have been used to solve this problem. To perform image classification, a large dataset of images containing objects of the desired categories is typically prepared, and each image is labeled based on its category. The machine learning model for image classification receives an image as input and outputs a vector of posterior probabilities that the image belongs to each category. Fig.2.1 shows an example of image classification where a general machine learning model receives an image as input and outputs category prediction. The category corresponding to the highest posterior probability in the output vector should match the given label, but this is rarely achieved before training. The difference between the output predicted category and the given label is measured by an error function, and the adjustable parameters within the machine

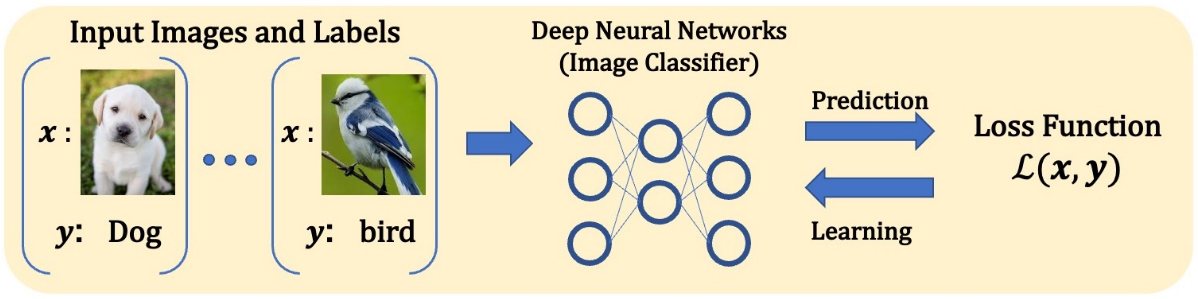


Figure 2.1: This figure shows how image classification works with a image classifier (DNNs). The image classifier receives an image as input and predicts the class to which the image belongs. The error function measures whether the prediction matches the given label, and the parameters of the model are tuned to reduce the value of the error function.

learning model are tuned to reduce the error value. This tunable parameter is a real value called a weight, which is responsible for adjusting the input-output function of the model. In the case of DNNs, there are hundreds of millions of these tunable weights inside, and a huge amount of training data is required to tune the them.

2.2 Deep Neural Networks

2.2.1 Model of a Simple Perceptron

Perceptron was proposed as a classifier of pattern recognition by learning weights of a model from training samples using linear threshold unit [28, 55]. The linear threshold unit is a model of a neuron in its simplest form and can be described as follows: A neuron receives inputs x_j $(j = 1, \cdots, M)$ from other neurons and fires when the sum of these inputs multiplied by the weight $\boldsymbol{w}$ exceeds a certain threshold value $(f(\boldsymbol{x}) = 1)$, and does not fire when it does not exceed the threshold value $(f(\boldsymbol{x}) = 0)$. Fig.2.2 visualizes the linear threshold unit. The simple perceptron computes the output $f(\boldsymbol{x})$ for input

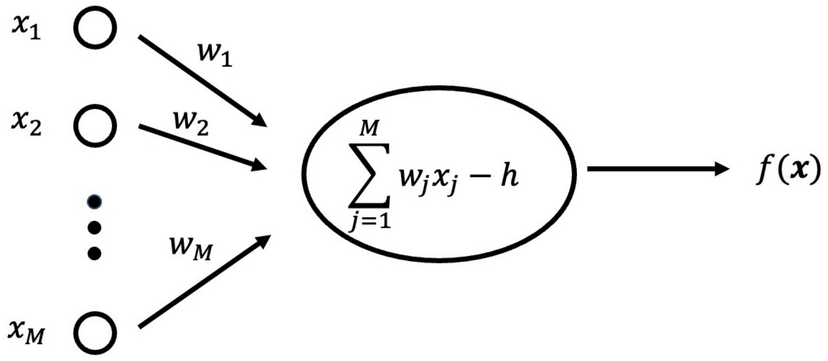


Figure 2.2: A linear threshold unit

$\boldsymbol{x} = (x_1, \cdots, x_M)^T$ as follows:

$$f(\boldsymbol{x}) = \sigma(\eta(\boldsymbol{x})) \tag{2.1}$$

$$\eta(\boldsymbol{x}) = \sum_{j=1}^{M} w_j x_j - h = \boldsymbol{w}^T \boldsymbol{x} - h \tag{2.2}$$

where σ is a threshold function, w_i is a weight from the i-th unit to the output unit and h is the bias. The value of $f(\boldsymbol{x})$ from the output unit is defined by the following threshold function σ:

$$\sigma(\eta) = \begin{cases} 1, & \text{if } \eta \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

This threshold function divides the input feature space into regions where the output of the perceptron is 1 and regions where the output of the perceptron is 0. In other words, the perceptron is a classifier that discriminates the feature space into two classes.

There is an "error correction learning" algorithm [55] for learning to estimate the parameters of a simple perceptron that performs two-class classification. In the algorithm, the simple perceptron receives one of the training data and the parameters of model are updated if its output results differ from the teacher signal. Let $\{(\boldsymbol{x}_i, t_i) | i = 1, \cdots, N\}$ be a dataset of size N, where the samples are pairs of an input vector $\boldsymbol{x}_i$ and a teacher signal

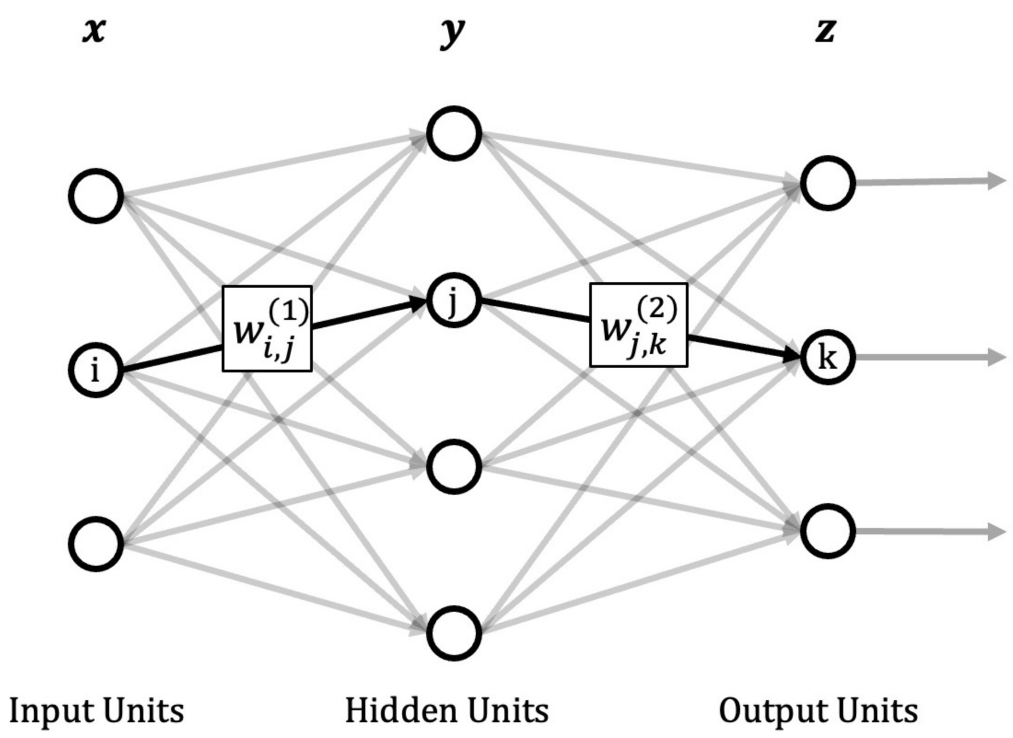


Figure 2.3: Multilayer Perceptron

t_i. The equations for updating the parameters of parameters given a randomly selected training sample $(\boldsymbol{x}_i, t_i)$ are as follows.

$$\boldsymbol{w} \Leftarrow \boldsymbol{w} + \alpha(t_i - f(\boldsymbol{x}_i))\boldsymbol{x}_i = \boldsymbol{w} + \alpha\delta_i\boldsymbol{x}_i \tag{2.4}$$

$$h \Leftarrow h - \alpha(t_i - f(\boldsymbol{x}_i)) = h - \alpha\delta_i \tag{2.5}$$

The value of $\delta_i = t_i - f(\boldsymbol{x}_i)$ is 0 if the output from perceptron matches the teacher signal and ± 1 if they differ. α is a small positive real number called the learning rate.

2.2.2 Multilayer Perceptron

Simple perceptron is effective for linearly discriminative tasks, but they must be used in combination to solve more complex tasks [28]. In fact, the brain has an enormous amount of neurons that are connected to each other to achieve complex information processing. A network that mimics this brain mechanism by combining multiple perceptrons in a hierarchical structure is called a MultiLayer Perceptron (MLP). Many networks achieve complex input-output mappings by introducing nonlinear functions called activation functions, which reproduce the firing of neurons, to the outputs of each layer. Fig.2.3 shows an example of a multilayer perceptron consisting of three layers: an input layer, an intermediate layer, and an output layer. The number of units in the input, intermediate and output layers are I, J and K, respectively. Let $\boldsymbol{z}$ be the output vector when the input vector $\boldsymbol{x}$ is input to the MLP. Each element of the vectors in the intermediate and output layers is obtained as follows:

$$z_k = o(\sum_{j=1}^{J} w_{jk}^{(2)} y_j) \tag{2.6}$$

$$y_j = h(\sum_{i=1}^{I} w_{ij}^{(1)} x_i) \tag{2.7}$$

where $w_{ij^{(1)}}$ and $w_{jk^{(2)}}$ are the weight from the i-th element of the input vector $\boldsymbol{x}$ to the j-th unit of the intermediate layer, and the weight from the j-th unit of the intermediate layer to the k-th unit of the output layer, respectively. h and o are the activation and output functions of the intermediate and output layers, respectively. When propagating information from one layer to the next, each unit computes a weighted sum of the inputs from the previous layer and passes the result through a nonlinear function. A network that propagates the information of sample $\boldsymbol{x}$ in one direction from the input layer to the intermediate layer and then to the output layer is called a feed-forward neural network. For classification problem, the output function is generally the softmax function. The multilayer perceptron is expressive enough to approximate any continuous function, however, it is difficult to tune the parameters manually due to the large number of parameters. To solve this problem, a learning method by backpropagation has been proposed as an algorithm to tune appropriate weights from training data. The backpropagation algorithm finds the gradient of the objective function with respect to the inputs to each unit in the model by the chain rule of derivatives. The backpropagation repeatedly propagates gradient from the upper layer to the lower layer to obtain the gradient for all units. Once these gradients are computed, the gradient for the weights of each unit can be computed. Let $\{(x_n, t_n)|n = 1, \cdots, N\}$ be the sample set for training, and define the objective function $\mathcal{L}$ of the network output $\boldsymbol{z}$ and the teacher signal $\boldsymbol{t}$ as follows:

$$\mathcal{L} = \frac{1}{2}\sum_{k}^{K}(t_k - z_k)^2 \tag{2.8}$$

Since the backpropagation method requires computing the partial derivative of the loss function $\mathcal{L}$, the activation function must be differentiable. The activation function frequently used in the past is a sigmoid function defined as follows:

$$h(z) = \frac{1}{1 + \exp(-z)} \tag{2.9}$$

Currently, the most used activation function is Relu, which is defined as follows:

$$h(z) = \max(0, z) \tag{2.10}$$

The partial derivative of the error function $\mathcal{L}$ with respect to the $w^{(2)}_{j,k}$ and $w^{(1)}_{i,j}$ of the model in the figure is obtained as follows:

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}_{j,k}} = \frac{\partial \mathcal{L}}{\partial z_k}\frac{\partial z_k}{\partial w^{(2)}_{j,k}} \tag{2.11}$$

$$= \delta_k \frac{\partial z_k}{\partial w^{(2)}_{i,j}} \tag{2.12}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}_{i,j}} = \sum_{k}^{K}\frac{\partial \mathcal{L}}{\partial z_k}\frac{\partial z_k}{\partial w^{(1)}_{i,j}} \tag{2.13}$$

$$= \sum_{k}^{K}\frac{\partial \mathcal{L}}{\partial z_k}\frac{\partial z_k}{\partial y_j}\frac{\partial y_j}{\partial w^{(1)}_{i,j}} \tag{2.14}$$

$$= \sum_{k}^{K}\delta_k \frac{\partial z_k}{\partial y_j}\frac{\partial y_j}{\partial w^{(1)}_{i,j}} \tag{2.15}$$

where the value of δ_k is defined as follows:

$$\delta_k = t_k - z_k \tag{2.16}$$

Using the partial derivatives of the obtained error function $\mathcal{L}$ with respect to the weights, each parameter of the model is updated as follows:

$$w^{(2)}_{j,k} \leftarrow w^{(2)}_{j,k} - \alpha \frac{\partial \mathcal{L}}{\partial w^{(2)}_{j,k}} \tag{2.17}$$

$$w^{(1)}_{i,j} \leftarrow w^{(1)}_{i,j} - \alpha \frac{\partial \mathcal{L}}{\partial w^{(1)}_{i,j}} \tag{2.18}$$

Since calculating the gradient using all training samples and updating parameters is computationally expensive, there is a method in which a single piece of data is randomly selected and the gradient is calculated. This method is called stochastic gradient descent. In actual applications, it is common to input a subset of the samples into the model as a mini-batch and calculate the gradient.

2.3 Convolutional Neural Network (CNN)

Deep Neural Networks (DNNs) is a multilayers perceptron with multiple layers and has attracted attention due to its high discrimination performance in image recognition tasks. DNNs used in ILSVRC2012 is called Convolutional Neural Network (CNN), which is based on the visual information processing of the brain and is constructed using local connections, weight sharing, pooling, and the use of multiple layers [27, 28, 31, 58].

The architecture of a typical CNN (Fig.2.4) differs from a simple multilayer perceptron in that the lower and upper layers have different processing roles. The lower layers closer to the input consist of two types of layers: convolutional layers and pooling layers. Let $\boldsymbol{X}^{(i)} \in \mathbb{R}^{c \times h \times h}$ be the input to the i-th convolutional layer and let F be the set of weight filters, called filter bank of the convolutional layer. $\boldsymbol{X}$ denotes feature maps of size $h \times h$

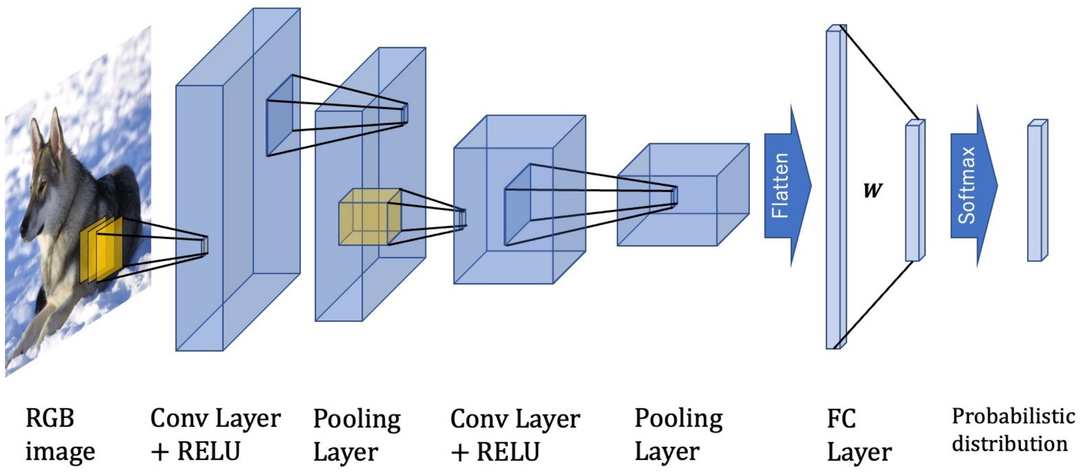


Figure 2.4: CNN

in height and width and c in number of channels. If $\boldsymbol{X}$ is an RGB image, the value of c is 3. For the output of the hidden layer, the value of c is set arbitrarily. F contains the c_{out} filters, and the size of the j-th filter $\boldsymbol{W}^{(i,j)}$ is $f \times f \times c$. $f \times f$ is the vertical and horizontal size of the filter $\boldsymbol{W}^{(i,j)}$, c_{out} is the number of channels in the output feature map from the CNN. Each unit in the convolution layer are locally connected through a filter to the feature maps output from the previous layer. The j-th filter $\boldsymbol{W}^{(i,j)}$ slides over the input feature map and performs a weighted transform for each patch region of size $f \times f$. The result of the linear transformation of the input feature map by the j-th filter is the feature map $\boldsymbol{U}^{(i,j)}$. The units in the feature map $\boldsymbol{U}^{(i,j)}$ are computed by the same filter $\boldsymbol{W}^{(i,j)}$, and a different filter is used to obtain other feature maps. The area where the filter is applied is a local area, imitating the receptive field that stimulates the neurons. Weight Sharing, in which the same filter is used in sliding fashion regardless of its position in the image, has the effect of reducing the number of parameters. The obtained feature map $\boldsymbol{U}^{(i,j)}$ is transformed by an activation function σ such as Relu to obtain the input to the next layer.

$$\boldsymbol{X}^{(i+1,j)} = \sigma(\boldsymbol{U}^{(i,j)}) \tag{2.19}$$

While the role of the convolution layer is to detect local features of the previous layer's features, the role of the pooling layer is to merge semantically similar features into one. The pooling layer downsamples the feature map by dividing the feature map into small regions and converting each local region to a scalar value. Since the relative positions of the features forming an object vary slightly, a coarse look at the position of each feature can be used to reliably detect the feature. Various pooling methods have been proposed, but a typical pooling unit is a method that outputs the maximum value of one local region. Another pooling method is average pooling, which outputs the average value of each local region.

The structure of CNNs consists of several iterations of convolution layers, activation functions, and pooling layers, followed by a fully connected layer. The full-connected layer (FC) receives the vectorized output $\boldsymbol{x}$ of the previous layer and performs a linear transformation using the weights $\boldsymbol{W}$.

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^k \tag{2.20}$$

where k is a number of classes. The posterior probability that the input data belongs to the i-th class is obtained using a softmax function as follows:

$$p_i = \frac{\exp y_i}{\sum_i^k \exp y_i} \tag{2.21}$$

When classifying classes, the index that has the largest posterior probability in the output vector from the softmax function is used as the predicted class.

When training a CNN, the gradients can be computed using the backpropagation algorithm and the weights of filter bank can be tuned using stochastic gradient descent, as in the usual multilayer perceptron. In the i-class classification problem, the cross entropy loss using the posterior probabilities of all training samples obtained by feeding

N samples into the model and the teacher signal t can be written as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{n}^{N}\sum_{i}^{k} t_{n,i}\log p_{n,i} \tag{2.22}$$

where $\boldsymbol{p}_n = (p_{n,1}, \cdots, p_{n,k}^T)$ is a posterior probability of n-th sample and $\boldsymbol{p}_n = (p_{n,1}, \cdots, p_{n,k})^T$ is a teacher signal of n-th sample. This is the error function for classification learning. $p_{n,i}$ and $t_{n,i}$ are the probability that the n-th training sample belongs to class i and the i-th element of the teacher signal $\boldsymbol{t}_n$ for that sample, respectively. The teacher signal $\boldsymbol{t}_n$ is a one-hot vector representing the class to which the training sample belongs, $t_{n,i} = 1$ if the class to which the training sample belongs is i, otherwise $t_{n,i} = 0$. As in the case of MLP, CNNs are generally SGD with mini-batches rather than optimization with all samples.

2.4 Definition and Taxonomy of Noisy Label, and Problem Setting

This section describes the definition and classification of label noise. In addition, label noise targeted by the proposed method presented in this thesis is described.

2.4.1 Definition of Noisy Labels

In this thesis, sample labels that are assigned differently from the ground-truth labels are defined as noisy labels. While noisy labels simply occur during the process of label assignment, in other cases, the relationship between the features of the input samples and the class labels affects the probability of noisy labels occurrence. This thesis focuses on noisy labels which are independent from the features of input, but noisy labels which are instance-dependent have also been studied. When we refer to noisy labels in this thesis, we imply that the noise is present only in the labels and not in the input feature.

2.4.2 Taxonomy of label noise

Instance-independent noisy labels

Label noise that is generated independently from the input features is referred to as instance-independent label noise [42, 50, 75]. This label noise is generated by the noise transition matrix with noise rate r, and is modeled as follows: In k-class classification problem, the noise transition matrix is $T \in [0,1]^{k\times k}$, where each element $T_{ij} = p(\tilde{y} = j|y = i)$ represents the probability that a sample belonging to class i incorrectly assigned to class j. The most used type of noise is symmetric label noise, which randomly reassigns true labels to one of the classes with equal probability [7]. Symmetric label noise can be of two types: one in which all classes are randomly assigned, and another in which all but the true label are randomly assigned, and the former is used in this study. Symmetric label noise is less practical label noise, but it is the label noise used as a baseline in many studies on noisy labels [60].

Another type of label noise is asymmetric label noise [50], which assigns sample label only to a particular class. For example, flip labels between classes with visually similar information, such as dogs and cats, cars and trucks. This property is similar to the label noise that occurs in the real world. Fig.2.5 visualizes examples of noise transition matrices for symmetric label noise and asymmetric label noise. In this thesis, experiments are conducted on these two types of artificial noisy labels.

Instance-dependent noisy labels

This label noise differs from the above label noise in that the input $\boldsymbol{x}$ has an effect on the probability of label noise occurrence as well. The probability of occurrence of this label noise depends on the input feature $\boldsymbol{x}$ and is closer to the label noise that occurs in the real world [16, 68].

True Label

Noisy Label	0	1	2	3
0	$1-r$	$\frac{r}{k-1}$	$\frac{r}{k-1}$	$\frac{r}{k-1}$
1	$\frac{r}{k-1}$	$1-r$	$\frac{r}{k-1}$	$\frac{r}{k-1}$
2	$\frac{r}{k-1}$	$\frac{r}{k-1}$	$1-r$	$\frac{r}{k-1}$
3	$\frac{r}{k-1}$	$\frac{r}{k-1}$	$\frac{r}{k-1}$	$1-r$

(a)

True Label

Noisy Label	0	1	2	3
0	$1-r$	r	0	0
1	0	$1-r$	r	0
2	0	0	$1-r$	r
3	r	0	0	$1-r$

(b)

Figure 2.5: k is number of classes, and r is noise rate. Each plot shows an example of the noise transition matrix where $k = 4$: (a) symmetric label noise, (b) asymmetric label noise

2.4.3 Problem Setting: Classification with Noisy Labels

Here we will introduce the problem setting and the notation of symbols used in this thesis. Column vectors and matrices are denoted in bold font (*e.g.* $\boldsymbol{x}$) and capitals (*e.g.* X), respectively. Specifically, let $\mathbf{1}$ be a vector whose elements are all 1. In supervised image classification problem with k classes, we denote a set of n training samples with clean labels as $\mathcal{D}^{\mathrm{GT}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^{GT})\}_{i=1}^{n} = \{\mathcal{X}, \mathcal{Y}^{GT}\}$ where $\boldsymbol{x}_i$ is a training image and $\boldsymbol{y}_i^{GT}$ is a one-hot vector representation of the ground truth label for the image $\boldsymbol{x}_i$. We define the set of hard-labels as $\{\boldsymbol{y}_i : \boldsymbol{y}_i \in \{0,1\}^k,\ \mathbf{1}^\top \boldsymbol{y}_i = 1\}_{i=1}^{n}$ and the set of soft-labels as $\{\boldsymbol{y}_i : \boldsymbol{y}_i \in [0,1]^k,\ \mathbf{1}^\top \boldsymbol{y}_i = 1\}_{i=1}^{n}$. The objective function to be minimized with the true labels is the cross-entropy loss defined as:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}^{GT} \log s_j(\boldsymbol{\theta}, \boldsymbol{x}_i), \tag{2.23}$$

where $\boldsymbol{\theta}$ denotes the parameters of DNNs and $\boldsymbol{s}$ is the output vector from the k-class softmax layer of the model.

In this thesis, we denote a set of n training samples with noisy labels as $\tilde{\mathcal{D}} =$

$\{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{n} = \{\mathcal{X}, \tilde{\mathcal{Y}}\}$. Training a classification model by minimizing the objective function Eq.(2.23) with $\tilde{\mathcal{D}}$ deteriorates its generalization performance and classification accuracy. Also we assume that a small set of clean samples are always available with a minimum effort by experts. We denote a small set of clean samples as $\mathcal{D}^c = \{(\boldsymbol{x}_i^c, \boldsymbol{y}_i^c)\}_{i=1}^{n^c} = \{\mathcal{X}^c, \mathcal{Y}^c\}$, where n_c is the number of clean samples and $n \gg n^c$. The small amount of clean data $\mathcal{D}^c$ is used in Chapter 3.

2.5 Related Works

2.5.1 Label Correction

This section describes prior works on noisy labels with label cleaning, which is related to the method proposed in Chapter 3. Manually correcting labels for all samples is expensive. Therefore, several algorithms have been proposed to automatically correct labels. Most label correction methods feed samples into a deep learning model and use the model's predictions during training to correct labels. By removing label noise many times during training, the robustness of the model against label noise is gradually improved. [54] substitutes the noisy labels to softmax outputs or linear combination of given labels and predicted labels after some warm-up training epochs. Since the model is robust in early learning phase, prediction probability from the softmax layer of DNNs is more reliable label than the noisy labels. The joint optimization framework [62] minimizes the error function by alternating updating labels by predicting models in training and updating models by SGD. In order to prevent the updated classes from being concentrated in one class, regularization is used to equalize the posterior probabilities of the labels. A similar method [71] adds compatibility loss to preserve information on originally clean samples while preventing the updated labels from deviating from the original labels. Han et al. [18] determines a prototype sample in each class and corrects label noise by measuring the difference between each sample and the prototype of the class to which it belongs. Yao et al. [70] introduced quality embedding, a hidden variable that expresses the reliability of the assigned labels, and an additional network to extract the true labels. When there are missing labels in multi-label classification, Durand et al. [12] prepared two networks, where one network detects and corrects the missing labels, and the second network is trained with the corrected labels.

2.5.2 Robust Loss

Robust loss function approaches design a new loss function or use regularization to robustly train classification models for noisy labels. The goal is to design an loss function that performs as well as the trained model in the noiseless case. Although there are several prior studies on robust loss, performance degradation due to label noise is inevitable even with robust loss functions [14]. The results of [38] showed that nonconvex loss functions such as 0-1 loss are robust against noise. However, since the 0-1 loss is nonconvex and non-differentiable, a surrogate loss for the 0-1 loss has been proposed [3]. However, this loss is still affected by label noise.Mean Absolute Error (MAE) is empirically shown to be more noise tolerant than widely used cross-entropy (CE) loss [15]. CE is sensitive to abnormal samples and may overfit to label noise. MAE is robust to noise because it treats all data equally, but it may underfit to training data. Then IMAE [66], which improves the weighting of the MAE for each sample, has been proposed. Generalized Cross Entropy [78] is an integrated method of MAE and CE to leverage the advantages of both losses. Early learning regularization (ELR) [36] keeps the loss value of clean samples large enough for training after the early learning phase by adding a regularization. Some other approaches utilize the estimated noise transition matrix of classes to modify the loss function to let the model recognize which classes tend to have noisy labels while training [16, 50] .

2.5.3 Sample Selection

The sample selection method selects samples to be fed into the model, allowing the model to learn robustly without having to learn samples with noisy labels. They monitor the output of the model during training and select samples to be trained at the next epoch. These methods operate outside of existing systems and can be easily incorporated into the existing systems. Curriculum learning [4] is a learning method that starts with easy samples and gradually gives harder samples to the learning model. In learning with noisy labels, clean samples are considered easy samples and samples with noisy labels are considered hard samples. The model starts learning with reliable clean samples, and when it finishes learning those samples, it gradually learns samples that are suspected to have label noise. In the teacher-student model approach [22], the teacher model selects clean samples and the student model learns them. Instead of using a predefined curriculum, the teacher constantly updates the curriculum according to the student's output.

Some methods exploit the robustness in the early learning phase to detect samples with noisy labels since the value of loss function on samples with noisy labels tend to be higher than ones with correctly labeled. Co-teaching [17] prepares two networks and each network selects samples with lower loss value as clean samples. Then each network is trained with a subset of samples selected by another network to prevent overfitting. Co-teaching+ [73] is an improved version of Co-teaching by adopting disagreements between the networks, where only samples predicted differently by the two networks as clean samples. Topofilter [67] detects noisy labels by k-nearest neighbor analysis based on the distance between pre-logits feature of each sample.

2.5.4 Other Approaches

Some other approaches perform a combination of label correction and iterative sample selection [1, 33, 59]. Most notably, DivideMix [33] uses two networks like Co-teaching and performs sample selection by fitting a two-component Gaussian Mixture Models (GMMs) at each network to the loss distribution of all training samples. Then prepares clean samples and noisy samples as labeled samples and unlabeled samples to train the networks with the semi-supervised learning technique called MixMatch [5]. Recently, AugDesc[43] has been proposed which utilizes weak augmentation of images for sample selection and strong augmentation for parameter updates.

Chapter 3

Robust Training of DNNs with Noisy Labels by Graph Label Propagation

3.1 Introduction

We proposed a method for robustly training deep neural networks (DNNs) in the presence of noisy labels in image classification problems [44]. Due to the large number of parameters in DNNs, there is a problem of overfitting to noisy labels. To solve this problem, we proposed a method to update noisy labels to clean labels by graph label propagation during training of DNNs.

Fig.3.1 describes the abstract of our method. Our method iteratively performs label update by constructing a similarity graph and parameter updates of model. At first, our classification model is trained for some epochs with cross entropy loss before over-fittiting to the noisy labels. This training period is called warmup period, during which the model obtains the ability to function as a feature extractor. The model learns the simple patterns of training samples first, and is robust to the noisy labels. At the start epoch of label update, our proposed method constructs a similarity graph between the feature vectors output from a hidden layer of DNNs of all samples. Since the model obtained robustness against the noisy labels, samples belong to the same ground-truth class have

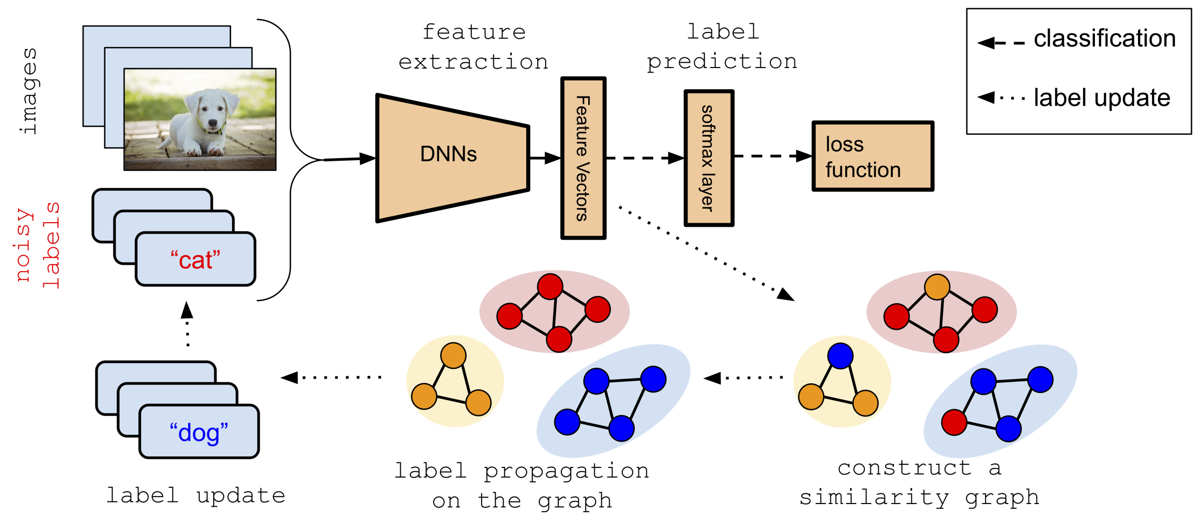


Figure 3.1: The illustration of first proposed method. Training dataset contains images and noisy labels. After training the model for some epochs, a similarity graph is constructed based on the similarity of feature vectors. Right lower graph shows the similarity graph, where each node represents a sample and each color on a node represents a noisy label. After the label propagation on the graph, training labels are updated.

higher similarity features. Then we assumed that label propagation on the graph would remove noisy labels. Then it resumes training the model with the updated label. By repeating these steps during training, the proposed method trains the DNNs robustly and cleans up the noisy labels simultaneously. In this study, we show that the proposed method successfully removes the noisy labels and robustly train the DNNs. We evaluated the performance of the proposed method by conducting comparative experiments on MNIST, a handwritten numeric character data set, and CIFAR-10, a 10-class image data set with artificial label noise. We found that the proposed method does not overfit to the noisy labels and maintains a high classification accuracy compared to other methods. We also analyze the label noise removal by visualizing the eigenvectors used for label update and the neighborhood of each sample in the feature space.

3.2 Related Works of Label Propagation

One of the representative methods of machine learning with label propagation based on the similarity between samples is Zhou et al. [79]. The label propagation was used for the semi-supervised learning problem with labeled and unlabeled samples. There are several derivatives of Zhou et al. [79] that use label propagation on graphs in image classification problem with noisy labels [13, 65]. These methods differ from the proposed method in that they are trained using handmade feature extraction methods such as Bag-of-Visual-Words, rather than deep learning features during the training process. A semi-supervised learning method for label propagation on the feature space of deep learning is proposed in [21], but this method does not consider the noisy labels.

3.3 Proposed method

In this chapter, an one-hot vector representation of a noisy training label is denoted as $\tilde{\boldsymbol{y}}$, and the matrix representation of all training noisy labels is denoted as $\tilde{Y}$. In the same way, the matrix representation of clean labels is denoted as $\tilde{Y}^c$

3.3.1 Iterative Label Correcting Algorithm

In this subsection, we describe how to train the parameters of the DNNs model robustly against the noisy labels. The noisy labels are corrected by using graph label propagation on the feature space. To make the model more robust to noisy labels, we add a small subset of clean samples to each mini-batch during training of the model with the stochastic gradient descent (SGD) on the loss function. The overall procedure of the proposed method is summarized in Algorithm 1.

Algorithm 1 Training DNNs by iteratively updating $\boldsymbol{\theta}$ and Y

```
    Input: $\mathcal{D} = \{\mathcal{X}, \tilde{\mathcal{Y}}\}$, $\mathcal{D}^c = \{\mathcal{X}^c, \mathcal{Y}^c\}$, network parameter $\boldsymbol{\theta}$, epoch $t_{\text{start}}$, $t_{\max}$;
 1: for $t = 1, 2, \dots t_{\max}$ do
 2:     Update $\boldsymbol{\theta}^{(t+1)}$ by SGD on $\mathcal{L}(\boldsymbol{\theta}^{(t)} | \mathcal{X}, \mathcal{X}^c, \tilde{\mathcal{Y}}^{(t)}, \mathcal{Y}^c)$;
 3:     if $t \geq t_{start}$ then
 4:         Extract all feature vectors $\{\boldsymbol{h}_i\}_{i=1}^{n'}$ from $\mathcal{X}$ and $\mathcal{X}^c$;
 5:         Construct the similarity matrix $W^{(t)}$ of the feature vectors;
 6:         Compute the Graph Laplacian $L = I - D^{-1/2} W D^{-1/2}$ where $D_{ii} = \sum_{j=1}^{n'} W_{ij}$
 7:         Finds $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots, \boldsymbol{\phi}_p$, the $p$ eigenvectors with smallest eigenvalue of $L^{(t)}$;
 8:         Update $\tilde{\mathcal{Y}}^{(t+1)}$ by a linear combination of the eigenvectors.
 9:     end if
10: end for
    Output: $\boldsymbol{\theta}^{(t_{\max})}$, $Y^{(t_{\max})}$
```

Training DNNs with Noisy Labels for Some Epochs

Usually DNNs model in the early stage of training in classification problem is robust against the noisy labels because it only learns simple patterns before it over-fits to the noisy labels. Therefore our proposed scheme trains DNNs model on training dataset with the noisy labels for some epochs ($t < t_{\text{start}}$), and starts correcting the labels before it over-fits ($t \geq t_{\text{start}}$).

Graph Label Propagation on Feature Space

This graph-based label propagation approach is inspired by the semi-supervised method [13]. After the label update starts ($t \geq t_{\text{start}}$), we extract the feature vectors of all training and clean images from the hidden layer of DNNs, specifically the fully-connected layer. The set of feature vectors is denoted as $H = \{\boldsymbol{h}_i\}_{i=1}^{n'}$ where $\boldsymbol{h}_i$ is the extracted feature of i-th image and $n' = n + n^c$.

Then we compute a sparse affinity graph $A \in \mathbb{R}^{n' \times n'}$ whose elements A_{ij} are non-negative pairwise similarities between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ as follows:

$$A_{ij} = \begin{cases} \exp\left(-\|\boldsymbol{h}_i - \boldsymbol{h}_j\|/\sigma^2\right) & \text{if } i \neq j \wedge \boldsymbol{h}_j \in \mathrm{NN}_m(i) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where NN_m denotes a set of m nearest neighbors in H, and σ is the hyper-parameter of RBF kernel. Then we define $W = A^{\mathrm{T}} + A$ as a sparse symmetric adjacency matrix with zero diagonal. The normalized graph Laplacian L is defined as $L = I - D^{\frac{1}{2}} W D^{\frac{1}{2}}$ with diagonal matrix D whose diagonal elements are $D_{ii} = \sum_j W_{ij}$.

Let $\boldsymbol{f} \in \mathbb{R}^{n'}$ be any real valued function on the graph defined above. The smoothness of the function is measured by the following loss function:

$$\boldsymbol{f}^T L \boldsymbol{f} = \frac{1}{2} \sum_{i,j} W_{ij} \left(\frac{1}{\sqrt{D_{ii}}} f_i - \frac{1}{\sqrt{D_{jj}}} f_j\right)^2. \tag{3.2}$$

Let $\boldsymbol{\psi} \in \mathbb{R}^{n'}$ be any real valued target function on the graph. Then we minimize the following criterion to get $\boldsymbol{f}$ which is a smooth function with respect to graph and agrees with the target signal $\boldsymbol{\psi}$:

$$J(\boldsymbol{f}) = \boldsymbol{f}^T L \boldsymbol{f} + \left(\sum_{i=1}^{n'} \mu_i (f_i - \psi_i)^2\right) \tag{3.3}$$

$$= \boldsymbol{f} L \boldsymbol{f} + (\boldsymbol{f} - \boldsymbol{\psi})^T M (\boldsymbol{f} - \boldsymbol{\psi}) \tag{3.4}$$

where M is a diagonal matrix whose diagonal elements $M_{ii} = \mu_i$ assign the importance of i-th sample. In the experiment, we assign larger value to the μ of clean sample.

Solving the above problem (Eq.3.4) requires expensive computational costs for large n. We can reduce the computational costs by working with a small number of eigenvectors of the graph Laplacian. We denote the sets of p eigenvectors and eigenvalues of the graph Laplacian as $\{\boldsymbol{\phi}_i, \lambda_i\}_{i=1}^p$. Note that the smoothness of the eigenvector $\boldsymbol{\phi}_i$ on the graph is measured by $\boldsymbol{\phi}_i^T L \boldsymbol{\phi}_i = \lambda_i$. Therefore the eigenvector with smaller eigenvalue is smoother function on the graph. Since any function $\boldsymbol{f} \in \mathbb{R}^{n'}$ can be written as $\boldsymbol{f} = \sum_i \alpha_i \boldsymbol{\phi}_i$, the function will be a linear combinations of the eigenvectors and with smallest eigenvalue.

Therefore we can reduce the dimension of the function by a linear combination of the eigenvectors $\boldsymbol{f} = U\boldsymbol{\alpha}$ where U is a $n' \times p$ matrix whose columns are p eigenvectors with smallest eigenvalues. Then we substitute the reduced function into Eq.(3.4):

$$J(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \Lambda \boldsymbol{\alpha} + (U\boldsymbol{\alpha} - \boldsymbol{\psi})^T M (U\boldsymbol{\alpha} - \boldsymbol{\psi}) \tag{3.5}$$

where $\Lambda_{ii} = \lambda_i$. The optimal $\boldsymbol{\alpha}$ solves the following $p \times p$ system of equations:

$$(\Lambda + U^T M U)^T \boldsymbol{\alpha} = U^T M \boldsymbol{\psi} \tag{3.6}$$

Then the prediction signals are computed by $\boldsymbol{f} = U\boldsymbol{\alpha}$.

In our experimental setting, we denote the concatenated matrix of the training and clean one-hot labels as $(Y')^T = (\tilde{Y}^T | (Y^c)^T)$. Each column of Y' is a real valued vector whose elements are in $[0, 1]$. Then we solve the following criterion for matrix A to obtain the smooth functions on the graph for each class j:

$$(\Sigma + U^T M U)^T A = U^T M Y' \tag{3.7}$$

After we obtain the k smooth functions $F \in \mathbb{R}^{n' \times k}$ on the graph by $F = UA$, each row of training labels $\tilde{Y}$ is updated as follows:

$$\tilde{y}_{ij} = \begin{cases} 1, & \text{if} \quad j = \arg\max_{j'} F_{ij'} \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

While updating the noisy label $\tilde{\boldsymbol{y}}_i$, we used the average predicted labels of the past some epochs as the final updated labels $\tilde{\boldsymbol{y}}_i$ to stabilize the variability of the update labels. The averaged labels are soft-labels and capture the degree of confidence of each sample belongs to a certain class.

Loss function

The proposed method trains the DNNs model with the following loss function constructed by three terms:

$$\mathcal{L} = \mathcal{L}_n(\boldsymbol{\theta}|\mathcal{X}, \tilde{\mathcal{Y}}) + \alpha\mathcal{L}_c(\boldsymbol{\theta}|\mathcal{X}^c, \mathcal{Y}^c) + \beta\mathcal{L}_e(\boldsymbol{\theta}|\mathcal{X}) \tag{3.9}$$

where $\mathcal{L}_n$, $\mathcal{L}_c$, $\mathcal{L}_e$ denote the two classification losses and regularization loss respectively, and α and β are hyper-parameters. In this study, we use the Kullbuck-Leibler(KL) divergence for $\mathcal{L}_n$ and $\mathcal{L}_c$ as follows:

$$\mathcal{L}_n(\boldsymbol{\theta}|\mathcal{X}, \tilde{\mathcal{Y}}) = \frac{1}{n}\sum_{i=1}^{n} D_{KL}(\tilde{\boldsymbol{y}}_i \| \boldsymbol{s}(\boldsymbol{x}_i, \boldsymbol{\theta})) \tag{3.10}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} \tilde{y}_{ij} \log\left(\frac{\tilde{y}_{ij}}{s_j(\boldsymbol{\theta}, \boldsymbol{x}_i)}\right) \tag{3.11}$$

$$\mathcal{L}_c(\boldsymbol{\theta}|\mathcal{X}^c, \mathcal{Y}^c) = \frac{1}{n^c}\sum_{i=1}^{n^c} D_{KL}(\boldsymbol{y}_i^c \| \boldsymbol{s}(\boldsymbol{x}_i^c, \boldsymbol{\theta})) \tag{3.12}$$

$$= \frac{1}{n^c}\sum_{i=1}^{n^c}\sum_{j=1}^{k} y_{ij}^c \log\left(\frac{y_{ij}^c}{s_j(\boldsymbol{\theta}, \boldsymbol{x}_i^c)}\right) \tag{3.13}$$

$\mathcal{L}_e$ is the regularization term that concentrates the probability distribution of each soft-label to a single class as follows:

$$\mathcal{L}_e(\boldsymbol{\theta}|\mathcal{X}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} s_j(\boldsymbol{\theta}, \boldsymbol{x}_i) \log s_j(\boldsymbol{\theta}, \boldsymbol{x}_i) \tag{3.14}$$

Because the regularization enforces the soft-label of each image to belong to a single class, each feature vector also tends to belong to a single class and increases the similarity between samples belonging to the same class.

3.4 Experiments

3.4.1 Datasets and Settings

Table 3.1: Summary of datasets used in the experiments.

	# of training	# of clean	# of validation	# of test	# of class
Two-Moon	1000	10	200	200	2
MNIST	20000	100	900	10000	10
CIFAR10	45000	1000	4000	10000	10

We use Two-moon dataset, MNIST and CIFAR10 with noisy labels to evaluate the proposed method. The sizes of training, clean, validation, and test set for each dataset are shown in Table3.1.

We added artificial label noise to these datasets for evaluation. To add 40% label noise to the Two-Moon dataset, we randomly select 40% training samples of the dataset and flip their labels to another class. The left and right images in Fig.3.2 are the original Two-Moon and the noisy version of Two-Moon, respectively. For MNIST and CIFAR10 dataset, we select r% training samples from each dataset and randomly assign their labels to one of the ten classes. We added 40% and 80% label noise to each dataset and the label accuracies before label update are summarized in Table 3.4.

3.4.2 Experiments with Two-Moon dataset

In the experiment with Two-Moon, we used a 4-layers MLP with 100 units in each hidden layer. To prevent the overfitting, weight decay is introduced and dropout with rate of 0.2 is applied to each hidden layer. The objective function (Eq.3.9) is optimized by SGD with the learning rate 0.01 for 50 epochs. The hyper-parameters of the label propagation and the loss function are set as follows: $\{\sigma^2 = 0.02, m = 10, p = 10, \alpha = 1.0, \beta = 0.4, t_{\text{start}} = 15\}$. As a comparison method, we used the 4-layers MLP trained on the loss function (Eq.3.9) without the label update. Our proposed method and the baseline method achieved 99.5% and 95.5% accuracy on the test set, respectively. The pseudo

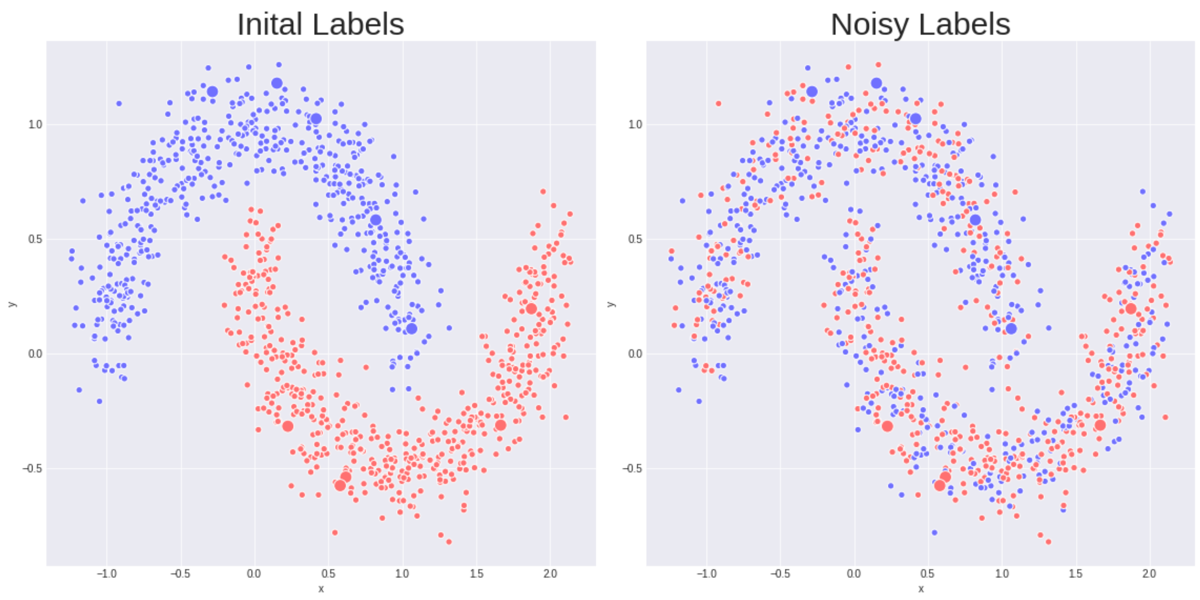


Figure 3.2: Images of Two-Moon experiments. **Left**: Original Two-Moon without noisy labels. The large dot point indicate where the cleans samples are present. **Right**: Two-Moon dataset with noisy labels

labels output by our proposed method during training are shown in the plots of Fig.3.3 and we can verify that our method successfully clean the noisy labels in the noisy version of Two-Moon dataset. Each plot in Fig.3.3 shows the pseudo labels given by our method during 14^{th} epoch and 25^{th} epoch. At the 15^{th} epoch, the first pseudo labels are given and they are incorrect near the decision boundary between the two moons as shown in Fig.3.3. However, the pseudo labels near the decision boundary gradually get corrected as training progresses.

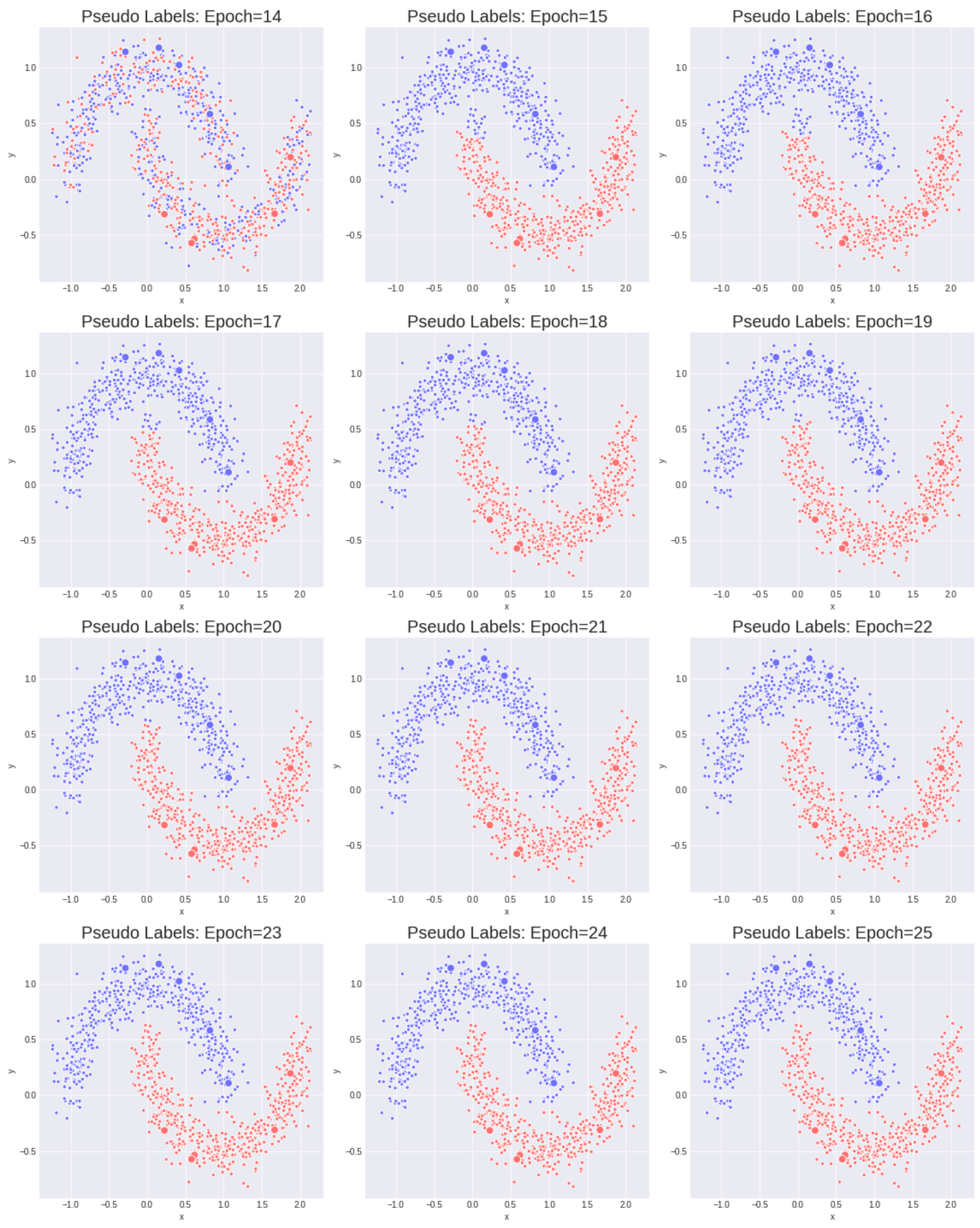


Figure 3.3: Each plot shows pseudo labels at each epoch after label update starts.

Visualizing Eigenvectors on the Input

In this experiment, we have visualized the eigenvectors $\boldsymbol{\phi}_i \in \mathbb{R}^{n'}$ output during the process of our method to understand how their linear combination cleans up the noisy labels. The eigenvectors in Fig.3.4 and Fig.3.5 were obtained at the beginning of the label update and end of training, respectively. Each figure shows the first 10 eigenvectors with smallest eigenvalue and the eigenvector with the second smallest eigen value divides the dataset into two classes in the input space. Comparing the two figures, we see that the second eigenvector in Fig.3.4 ambiguously classifies the two classes at the boundary of the two clusters, while the second eigenvector in Fig.3.5 clearly separates the two clusters. This suggests that the two clusters are clearly separated in the feature space of the MLP at the end of training.

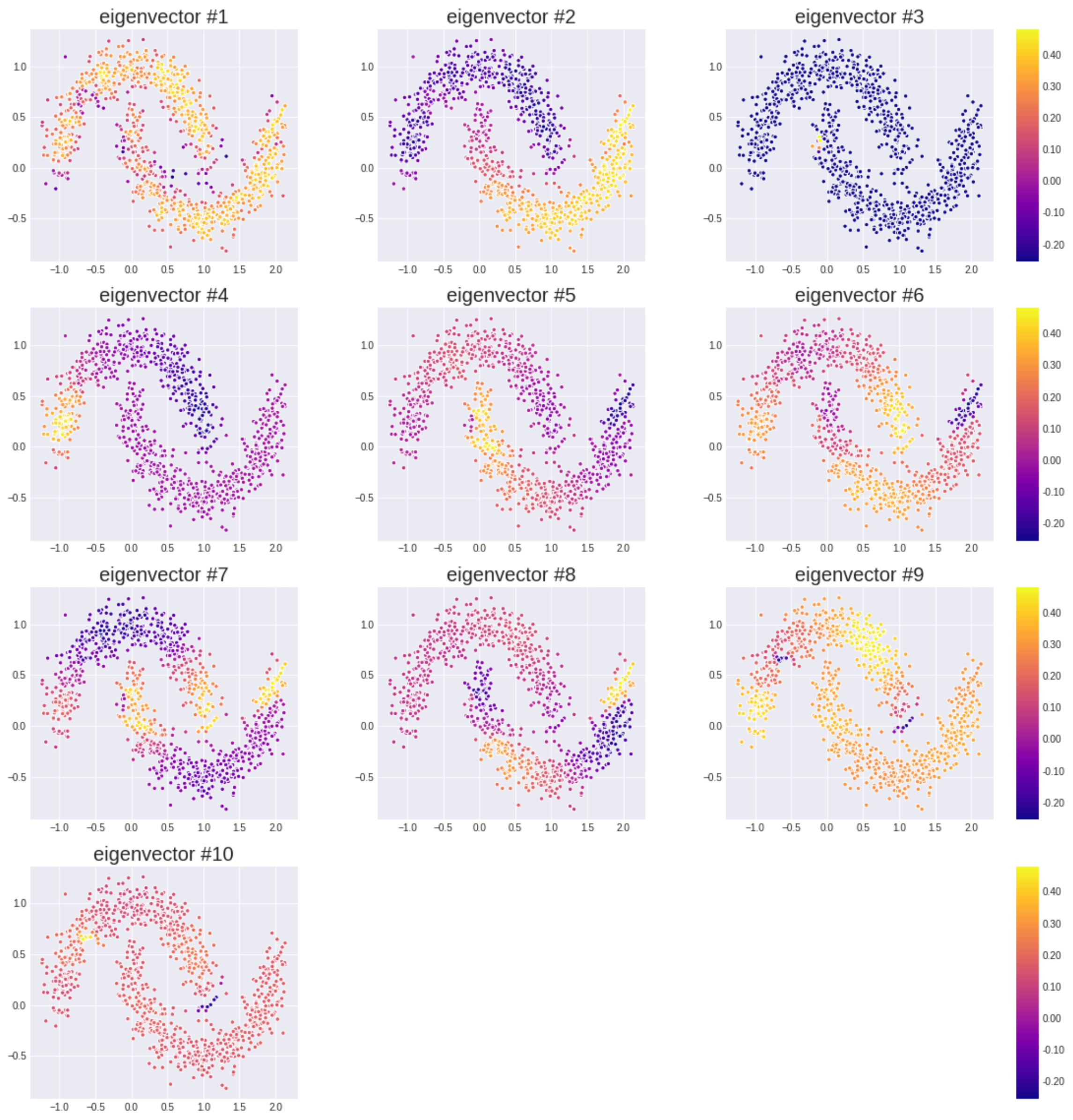


Figure 3.4: The first 10 eigenvectors with smallest eigenvalues of the graph laplacian visualized on the input at the start of the label update.

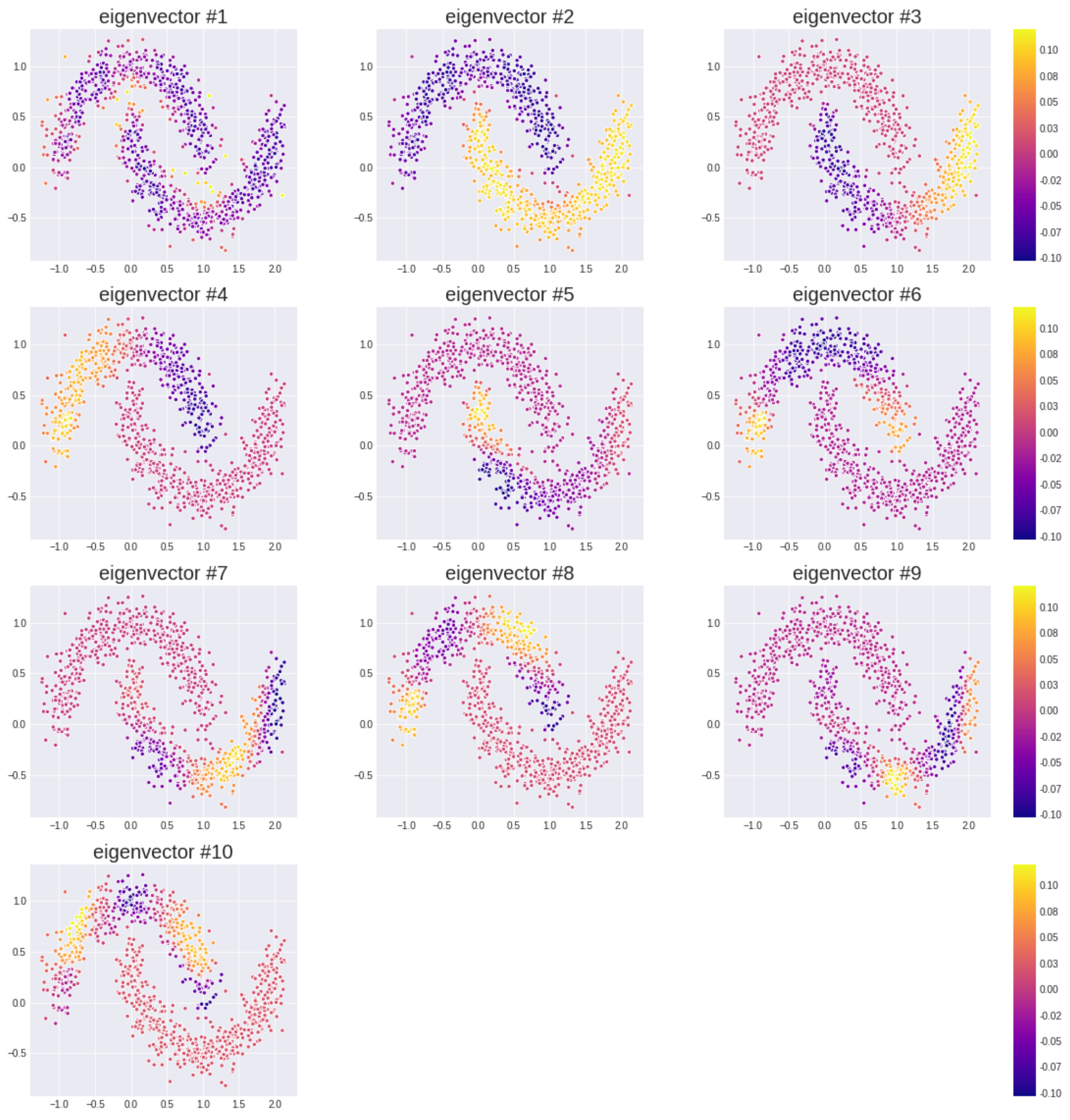


Figure 3.5: The first 10 eigenvectors with smallest eigenvalues of the graph laplacian visualized on the input at the end of training.

3.4.3 Experiments with MNIST and CIFAR10

The MNIST digit dataset consists of 28×28 gray-scale images and the CIFAR-10 dataset consists of 32×32 color images. We randomly split training set of each dataset into training, clean, and validation set as shown in Table 3.1.

We implement the 4-layers MLP for MNIST and the 9-layers CNN for CIFAR-10. We compare our proposed method (Ours) with the following approaches that have the same architecture as the proposed method and do not update the labels: a model trained on $\mathcal{L}_n$ only (Baseline), a model trained on $\mathcal{L}_n + \mathcal{L}_c$ (Baseline+clean), and a model trained on $\mathcal{L}_n + \mathcal{L}_c + \mathcal{L}_e$ (Baseline+clean+reg).

In the both datasets, all images are normalized to in range of $[-1, 1]$ and data augmentation by vertical and horizontal random flip and random crops after padding 4 pixels. The size of random crop is 28×28 for MNIST and 32×32 for CIFAR-10.

The loss function for DNNs is optimized by SGD with a momentum of 0.9, a weight decay of 10^{-4}, a learning rate of 0.01. In the both datasets, the batch size of the training samples with noisy labels was 100, and the batch size of clean samples were set to 5 and 30 for MNIST and CIFAR-10, respectively. The models were trained for 100 epochs for MNIST, and 400 for CIFAR-10. In the both datasets, we set 0.2 for σ^2, 100 for m, 50 for p. The values of α, β, t_{start} are given by the grid search over the following hyper-parameter spaces: $\alpha = \{0.0, 0.1, 1.0, 10.0\}$, $\beta = \{0.0, 0.1, 1.0, 10.0\}$, $t_{\text{start}} = \{\text{no update}, 20, 40, 60\}$ for MNIST, $t_{\text{start}} = \{\text{no update}, 100, 200, 300\}$ for CIFAR-10. The obtained values are summarized in Table.3.2.

Table.3.3 shows the classification accuracy of the proposed method and the other methods for the test set. These results are calculated from the average of five trials of the experiment. It was confirmed that the proposed method achieves better accuracy than the other methods for any noisy rate in all datasets. Table.3.4 reports the label accuracy before and after updating the labels. It shows that the proposed method succeeds in removing the label noise in the both datasets. Therefore the proposed method is trained on the more accurate labels after the label update and achieves higher classification accuracy on the test set.

Table 3.2: The hyperparameters for experiments.

Dataset	MNIST		CIFAR-10	
noise rate (%)	40%	80%	40%	80%
α	0.0	1.0	0.1	0.1
β	1.0	1.0	1.0	1.0
t_{start}	20	20	300	200

Table 3.3: The classification accuracy on the test set.

#	Method	Test Accuracy (%)					
Dataset		MNIST			CIFAR10		
noise rate (%)		0	40	80	0	40	80
#1	Baseline	97.61	83.58	44.34	88.95	66.94	29.70
#2	Baseline+clean	97.60	81.53	42.34	89.39	68.39	50.57
#3	Baseline+clean+reg	-	84.67	59.28	-	76.84	64.04
#4	Ours	-	**96.26**	**89.58**	-	**86.05**	**77.78**

Table 3.4: Label accuracy before and after label update.

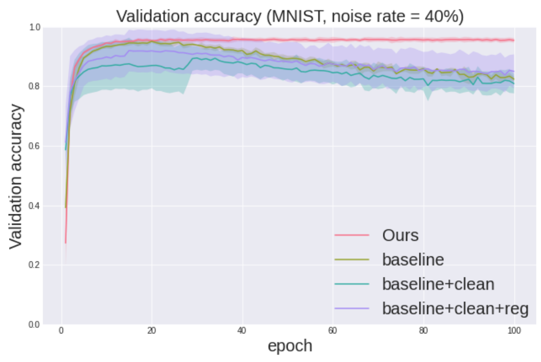
#	noise rate (%)	Label Accuracy (%)			
Dataset		MNIST		CIFAR10	
Label update		before	after	before	after
#1	40%	64.05	97.12	64.01	90.03
#2	80%	28.05	89.34	28.02	80.57

Accuracy curves on the validation set, train set with true labels, and pseudo labels

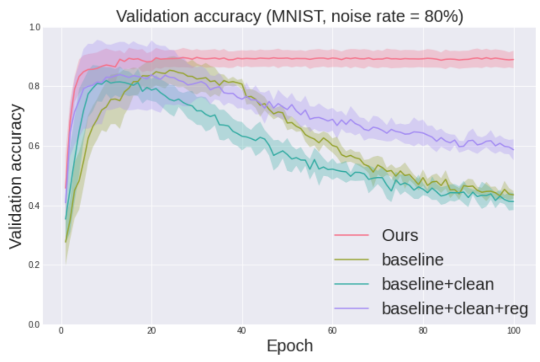
We plot the validation accuracy of the both datasets for each epoch in Fig.3.6a, 3.6b, 3.7a and 3.7b. The transparent colored area is the standard deviation of the validation accuracy at each epoch. In the case of MNIST, the Fig.3.6a and Fig.3.6b plot the validation accuracy with noise rate of 40% and 80%, respectively. While other baseline approaches cause overfitting and gradually reduce the validation accuracy as the training progresses, the validation accuracy of the proposed method remains high after label update. In the case of CIFAR-10, the Fig.3.7a and Fig.3.7b plot the validation accuracy with noise rate of 40% and 80%, respectively. The proposed method achieves higher accuracy than the other approaches at the end of training. In the both figures, we can confirm that the validation accuracy spikes immediately after the label update. It was confirmed that the classification accuracy in the validation data was recovered by correcting the label noise. We also plot the training accuracy with respect to the true labels of training samples in Fig.3.6c, 3.6d, 3.7c and 3.7d. In the all cases, the training accuracy with true labels spikes immediately after label update as shown in validation accuracy curve. This means that our method allows the DNNs model to learn the correct, labels rather than overfitting the noisy labels of training samples. In addition, we plot the accuracy of updated labels given by the graph label propagation during training on the both dataset in Fig.3.6e, 3.6e, 3.7e and 3.7f. It is observed that the accuracy of pseudo labels increases immediately after the start of label updating, and high accuracy of pseudo labels is maintained until the end of training, except in the case of CIFAR-10 with noise rate of 40% in Fig.3.7e. In Fig.3.7e, the label accuracy gradually decreases after the start of label update. We deduce that this is the result of an accumulation of errors in updated labels and overfitting of training model to the errors.

Sensitivity of our methods with hyper-parameters

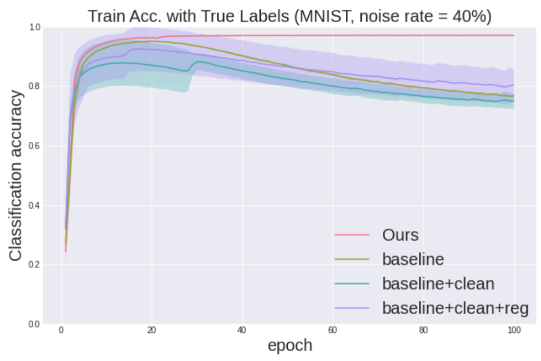
In our experiments, the value of each hyper-parameter is set as follows: $\alpha = \{0.0, 0.1, 1.0, 10.0\}$, $\beta = \{0.0, 0.1, 1.0, 10.0\}$, $t_{\text{start}} = \{\text{no update}, 20, 40, 60\}$ for MNIST, and $t_{\text{start}} = \{\text{no update}, 100, 200, 300\}$ for CIFAR-10. Fig.3.8a, 3.8b, 3.9a, and 3.9b show the sensitivity of our method to α, β and t_{start} by visualizing validation accuracy at the end of training in various noise settings on each dataset. There are four bar plots in the four figures and the value of t_{start} is different in each plot. Each plot shows a grouped bar chart and each group represents the value of β, and each color of bar represents the value of α. In each figure, our method is less sensitive to the value of t_{start} and produces excellent results, except when there is no label update. In Fig.3.8a and 3.8b, for larger values of α, the



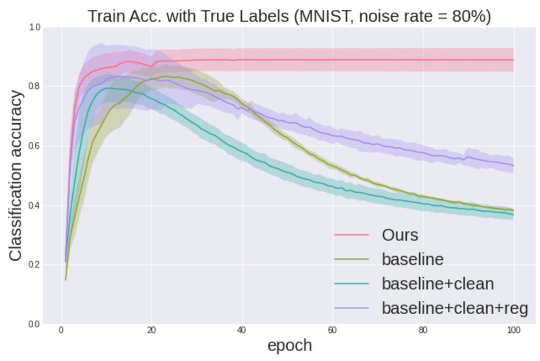
(a)



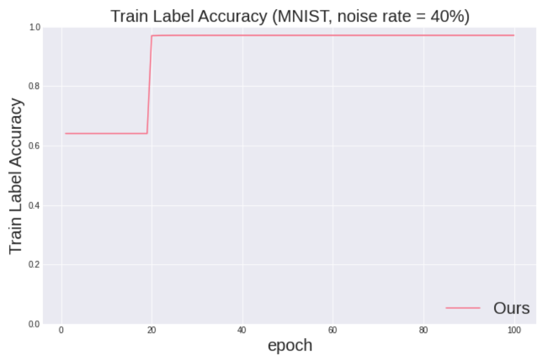
(b)



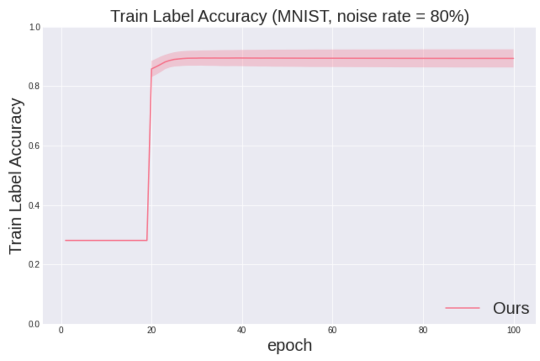
(c)



(d)

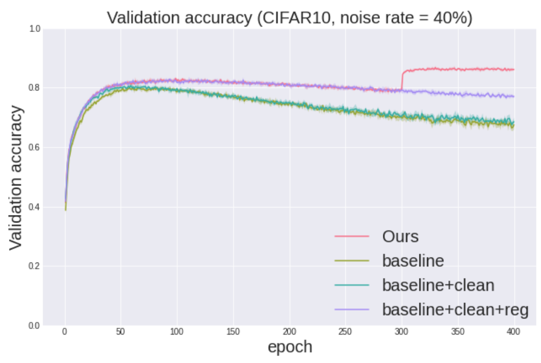


(e)

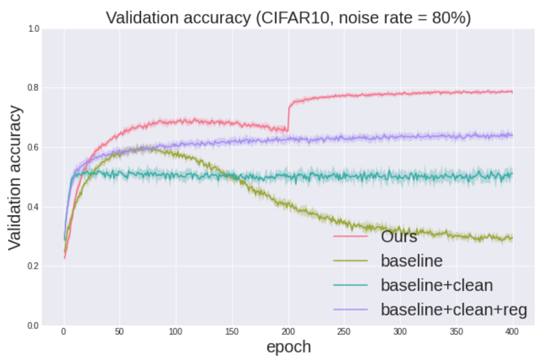


(f)

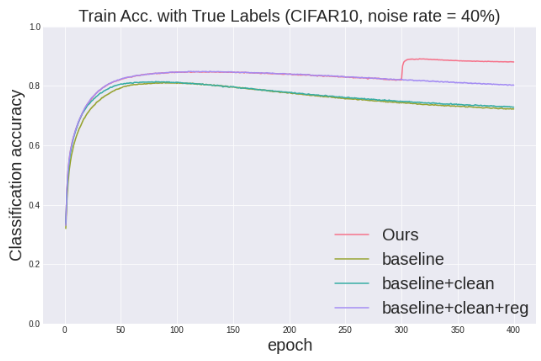
Figure 3.6



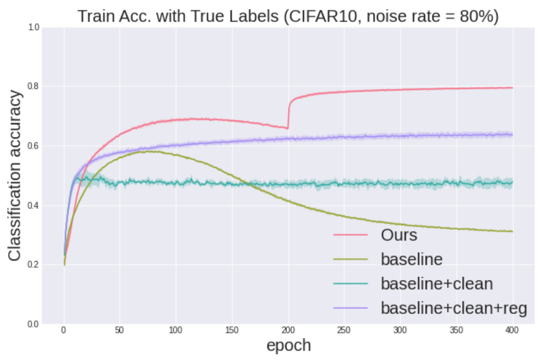
(a)



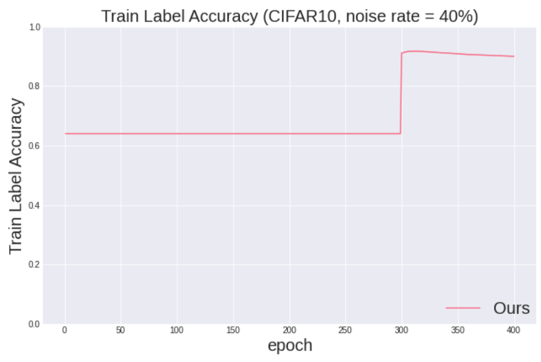
(b)



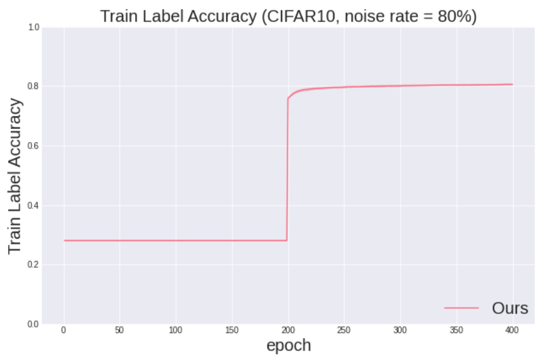
(c)



(d)



(e)



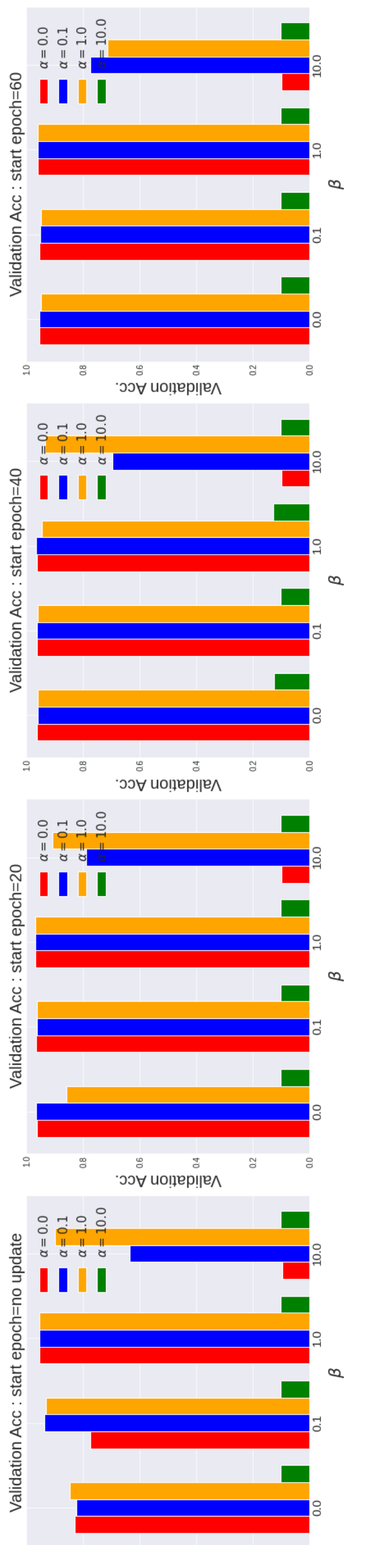
(f)

Figure 3.7

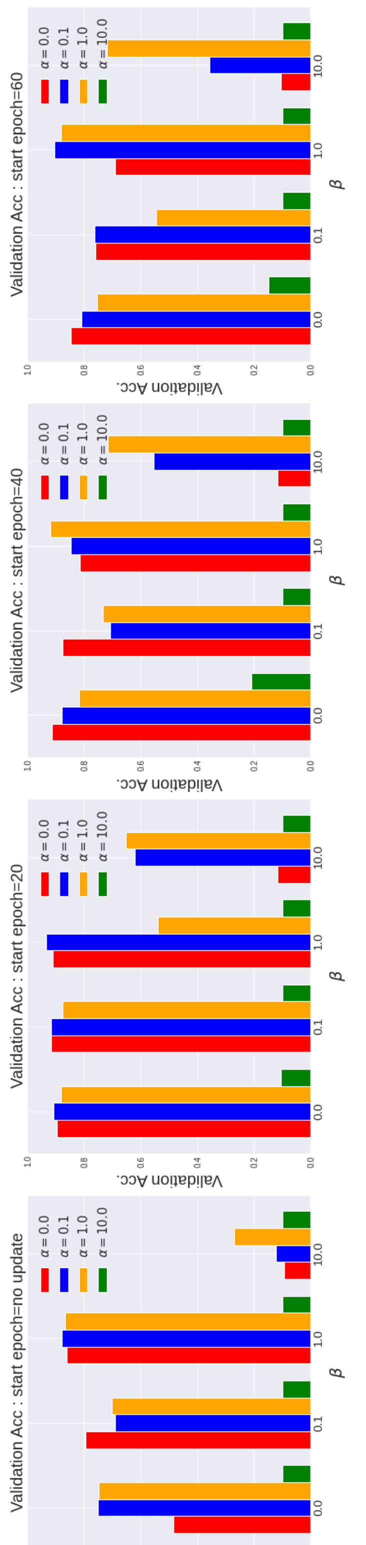
performance of our method deteriorates, and there is no critical difference in performance for $\alpha = 0.0$, $\alpha = 0.1$ and $\alpha = 1.0$. This indicate that the value of α is not important to achieve the superior results in the case of MNIST. This result may be due to the fact that MNIST is an easy dataset for image classification and some of the clean samples are not critical to the performance of our method. When $\alpha = 10.0$, the training model overfits to the small portion of dataset, and deteriorates the performance. The value of β is not important for performance of our method, except for $\beta = 10.0$. Higher value of $\beta = 10.0$ may prevent model from learning the label information of MNIST dataset. In Fig.3.9a and 3.9b, $\alpha = 10.0$ and $\alpha = 0.0$ lower the performance of our method in CIFAR10. In all experimental settings, the best performance is obtained when the value of $\alpha = 0.1$ except when there is no label update. This indicates that for the CIFAR10 data set, a small amount of clean data set is effective in improving the performance of this method, but a large value of α causes overfitting to the clean set and degrades generalization performance. The value of β is also not important for performance of our method, except for $\beta = 10.0$. As in the MNIST case, higher value of $\beta = 10.0$ may prevent model from learning the label information of CIFAR10.

Visualizing the neighborhoods of samples in the feature space

To see what samples each vertex on the graph has as its neighbors, we visualize the closest neighbors of each sample in the feature space. we used CIFAR-10 dataset which has the noisy labels with a noisy rate of 80%. At the end of training, we saved the graph W and randomly picked up 6 samples as queries. In Fig.3.10, we visualize the top 7 neighbors closest to the each query on W. The color of bounding box of each image represents its true label. We can confirm that the true class of the neighbors is almost same as the corresponding query and that the visual appearance of the neighbors is similar to the corresponding query at the end of training.

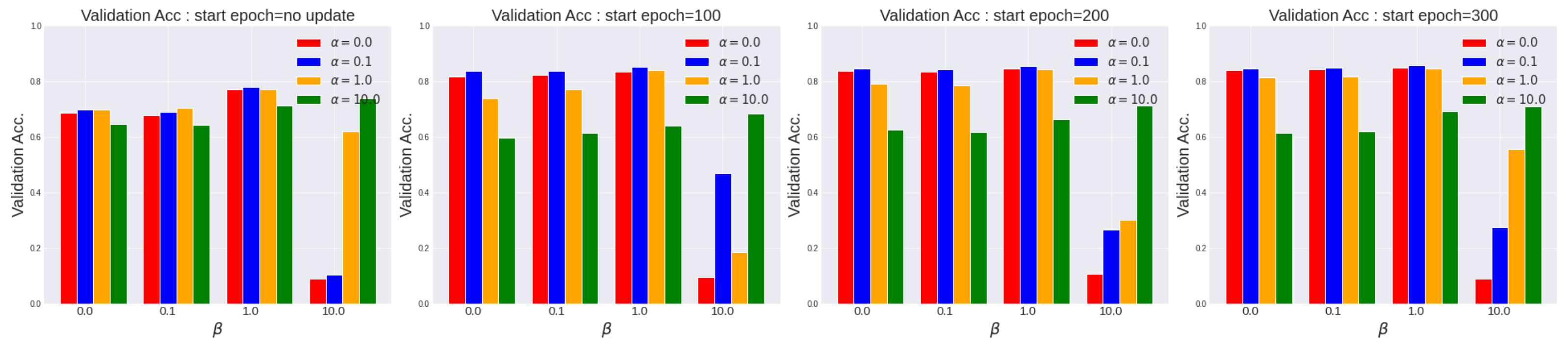


(a) Validation accuracy on MNIST with noise rate = 40%, $t_{\text{start}} = 20$

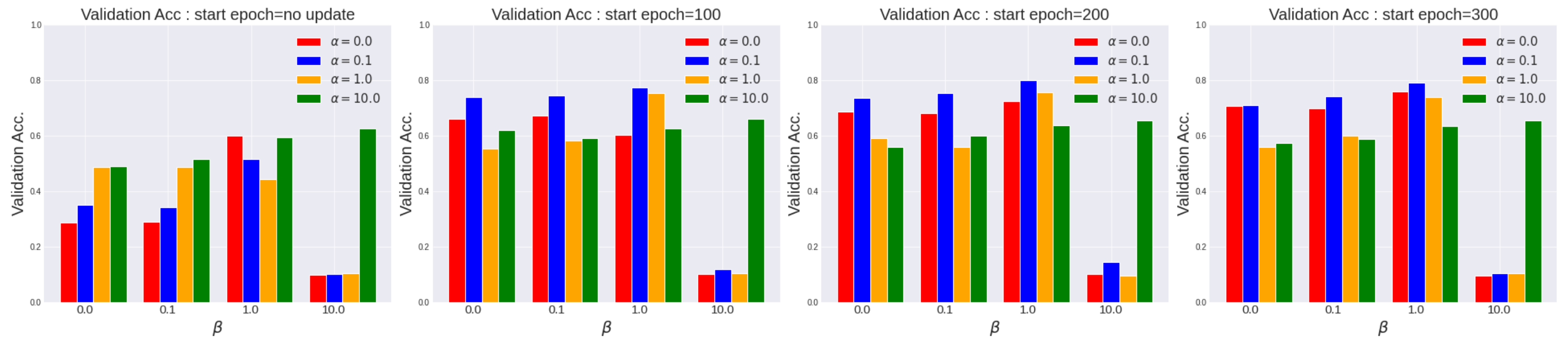


(b) Validation accuracy on MNIST with noise rate = 80%, $t_{\text{start}} = 20$

Figure 3.8



(a) Validation accuracy on CIFAR-10 with noise rate = 40%



(b) Validation accuracy on CIFAR-10 with noise rate = 80%

Figure 3.9

3.5 Discussion

Table.3.3 shows that the proposed method achieves superior classification accuracy than baseline methods. The accuracy curves in Fig.3.6 and 3.7 show that the proposed method maintains higher classification acuracy during training, while the other methods overfit to noisy labels and test accuracy decreases. This indicates the effectiveness of the algorithm for label correction during training. In other words, label propagation on the graph based on similarity between feature vectors extracted from DNNs was effective in label correction. In the datasets used in this study, it was confirmed that the similarity of samples that truly belong to the same class was high in the early stages of learning. The label noise elimination of the proposed method depends on the robustness of DNNs in the early stages of learning, it is not clear whether DNNs can achieve robustness on other datasets, such as datasets with a large number of classes or imbalanced classes, and this needs to be verified.

In Fig.3.8 and 3.9, each plot shows a decrease in classification accuracy for large values of α. This is a result of the model overfitting to a small amount of clean data. The best accuracy for both datasets and noise ratios are achieved for non-negative values of β, suggesting that regularization is effective in robustness to noisy labels. In addition, the best starting epoch of label update is different for each data set and noise ratio, requiring careful hyperparameter tuning. When the data set or model is large, the computational cost is expensive.

Fig.3.10 visualizes the samples that exist in the neighborhood in the feature space at the end of training. It was observed that samples with similar visual features exist in the neighborhood regardless of whether they truly belong to the same class or not. The fact that the label accuracy after label update is not 100% suggests that the constructed graph does not correctly reflect the similarity between samples belong to same class. Since the similarity of features of samples that truly belong to the same class should be close even if the visual similarity is different, it is necessary to devise a way to reduce the intra-class variance of the features of samples belong to the same ground-truth class.

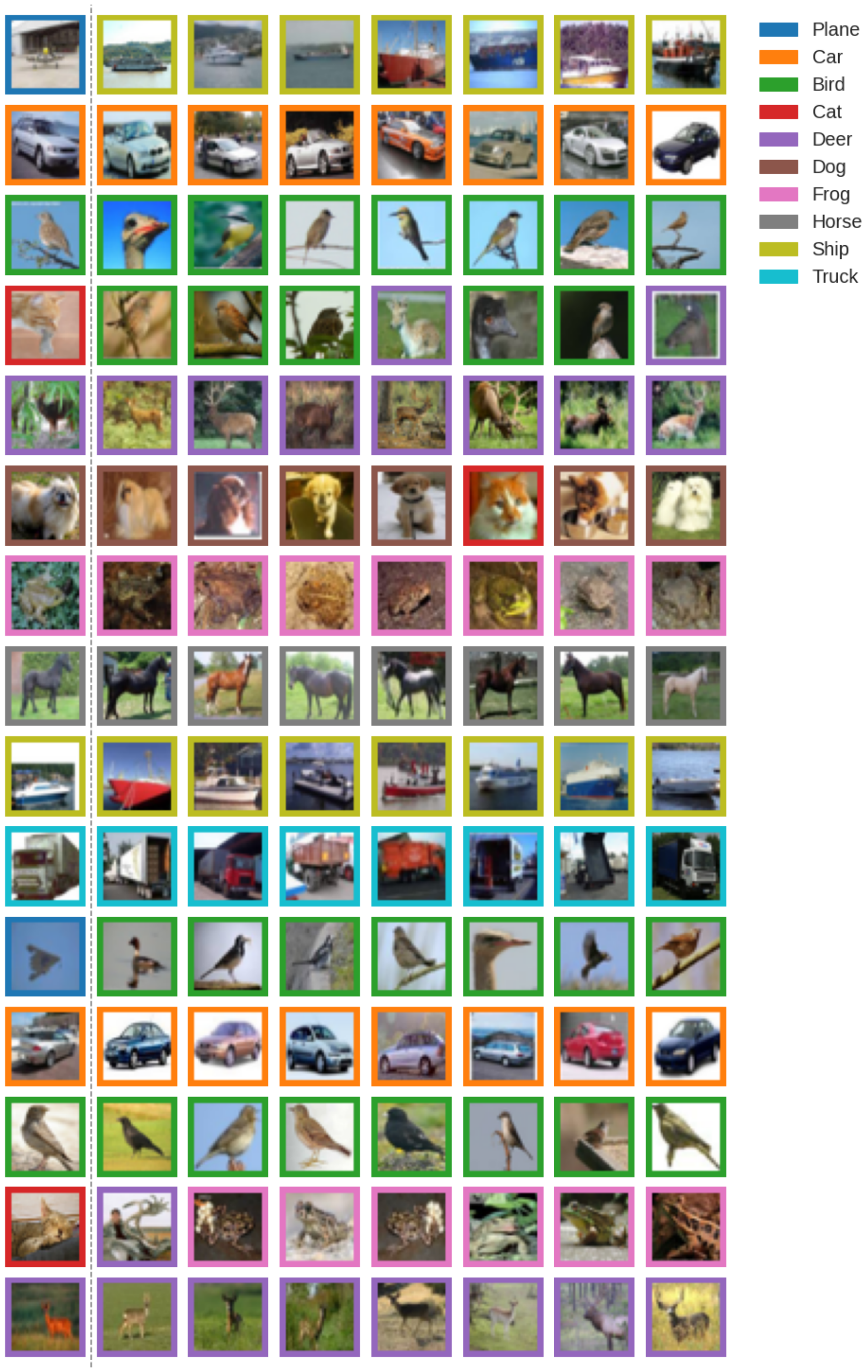


Figure 3.10: The leftmost image in each row is the query and the other seven images are neighbors on the feature space of the query. The color of the bounding box of each image represents the true class.

Chapter 4

Consistency Regularization on Clean Samples

4.1 Introduction

Analysis of the memorization effects revealed that deep learning model trained on dataset with noisy labels first learns simple patterns then gradually overfits to the training samples with noisy labels, resulting in good generalization performance in the early learning phase [2]. Taking advantage of this property, several approaches have been proposed to tackle the problem of learning with noisy labels (LNL) [17, 50, 62]. In particular, DivideMix [33] achieves excellent results on baseline datasets with noisy labels by modeling the loss distribution to select clean samples and training a classification model in a semi-supervised manner.

Following the results of DivideMix's successful selection of clean samples, we propose to adopt a consistency regularization on the selected samples during training for preventing a classification model from overfitting to the simple patterns of the selected samples [45]. The consistency regularization is one of the machine learning techniques developed in the field of semi-supervised learning. In our problem setting, the consistency regularization encourages model to make consistent predictions on the perturbed images that match the predictions to the original images. Since the selected samples tend to be easy to classify, the sample selection may overlook samples that have the correct label but are difficult to classify. The consistency regularization on clean samples mitigates this drawback by adding noise to samples with simple patterns to reduce overfitting. In recent years, several approaches for data augmentation have been proposed, and we use one of them, RandAugment[10], to perturb the selected samples. An overview of our method is shown in Fig.4.1.

The contributions of this study are summarized as follows. (1) We introduce the consistency regularization on the samples selected as clean as an extension of DivideMix. (2) Extensive evaluation on CIFAR-10 and CIFAR-100 with synthetically generated label noise is performed to confirm that DivideMix with the consistency regularization yields comparably or better than state-of-the-art methods. (3) We performed the ablation study on the value of hyperparameter for consistency regularization based on dataset with noisy labels and confirmed that consistency regularization contributes to the generalization performance of model.

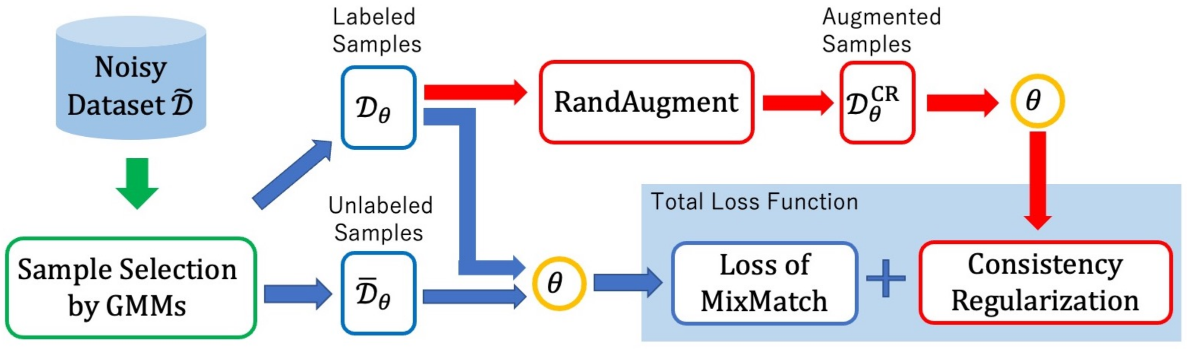


Figure 4.1: Diagram of DivideMix with consistency regularization. After the samples are divided into a set of labeled samples $\mathcal{D}_{\boldsymbol{\theta}}$ and a set of unlabeled samples $\bar{\mathcal{D}}_{\boldsymbol{\theta}}$ by GMM (green box), each set is fed into the model $\boldsymbol{\theta}$ (bottom) and compute the loss function of MixMatch following DivideMix [33]. At the same time, RandAugment is applied only to the labeled samples $\mathcal{D}_{\boldsymbol{\theta}}$ (top) to obtain modified samples $\mathcal{D}_{\boldsymbol{\theta}}^{CR}$ and is fed into θ to compute the cross entropy loss as a consistency regularization. The model θ is trained on the total loss consists of the loss of MixMatch and the consistency regularization.

4.2 Related works

4.2.1 Dividemix

Dividemix[33] selects training samples with lower loss value as a set of labeled data and the rest of the samples are treated as a set of unlabeled data, and train the model in a semi-supervised learning manner. Dividemix fits a two-components Gaussian Mixture Model (GMM) [51] to the loss distribution of all training samples to find the clean probability of each sample then the samples, and divides the samples based on that probability.

Let $\mathcal{D} = \{\mathcal{X}, \tilde{\mathcal{Y}}\} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{N}$ denote the training samples with noisy labels where $\boldsymbol{x}_i$ is an image and $\tilde{\boldsymbol{y}}_i$ is an one-hot vector represents label over k classes. Suppose the parameters of a deep learning model are denoted as $\boldsymbol{\theta}$ and the objective function for training is the cross entropy loss $\ell(\boldsymbol{\theta})$ as follows:

$$\ell(\boldsymbol{\theta}) = \{\ell_i\}_{i=1}^{N} = \big\{ -\sum_{l=1}^{k} \tilde{y}_i^k \log(\mathrm{p}_{\mathrm{model}}^k(x_i, \theta))\big\}_{i=1}^{N} \tag{4.1}$$

where $\mathrm{p}_{\mathrm{model}}^k$ is a softmax output from the model for class k. After ℓ is computed for all training samples, a two-component GMM is fitted to ℓ using the Expectation-Maximization algorithm. Probability that a sample being clean is defined as w_i and equals to the posterior probability $p(g|\ell_i)$ for each sample from the Gaussian component g with smaller mean.

At each epoch, training data is divided into a labeled set $\mathcal{X}$ and an unlabeled set $\mathcal{U}$ by setting a threshold τ on w_i given by GMM of one network, and the other network is trained on the divided set in a semi-supervised manner. By alternating the roles of the two networks, they teach an estimated set of clean samples each other and avoid accumulating confirmation bias.

After dataset is divided into two sets, and a mini-batch of labeled set $\{(\boldsymbol{x}_b, \tilde{\boldsymbol{y}}_b, w_b); b \in$

$(1, \cdots, B)\}$ is given, label refinement is performed as follows:

$$\bar{\boldsymbol{y}}_b = w_b \tilde{\boldsymbol{y}}_b + (1 - w_b)\boldsymbol{p}_b \tag{4.2}$$

where $\bar{\boldsymbol{y}}_b$ is a refined label, $\boldsymbol{p}_b$ is a network's prediction (averaged across multiple augmentations of $\boldsymbol{x}_b$) and w_b is a clean probability given by the other network. And given a mini-batch of unlabeled set $\{\boldsymbol{u}_b; b \in (1, \cdots, B)\}$, predictions on unlabeled samples from two networks are averaged to estimate their labels $\hat{\boldsymbol{q}}_b$. Each $\bar{\boldsymbol{y}}_b$ and $\bar{\boldsymbol{q}}_b$ are transformed by sharpening function to reduce their temperature and obtain $\hat{\boldsymbol{y}}_b$ and $\hat{\boldsymbol{q}}_b$.

Given $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$, MixUp[76] is applied to them where each sample is interpolated with another sample randomly chosen from the combined mini-batch of $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$. The transformed sets are denoted as $\mathcal{X}'$ and $\mathcal{U}'$. Finally, semi-supervised method called MixMatch[5] is applied to the augmented dataset $\mathcal{X}'$ and $\mathcal{U}'$.

The loss on $\mathcal{X}'$ is the cross entropy loss $\mathcal{L}_{\mathcal{X}}$ and the loss on $\mathcal{U}'$ is the mean squared error $\mathcal{L}_{\mathcal{U}}$ as follows:

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{X}'|} \sum_{\boldsymbol{x},\boldsymbol{p} \in \mathcal{X}'} \sum_{k} p^k \log(\mathrm{p}^k_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta})) \tag{4.3}$$

$$\mathcal{L}_{\mathcal{U}} = -\frac{1}{|\mathcal{U}'|} \sum_{\boldsymbol{x},\boldsymbol{p} \in \mathcal{U}'} \|\boldsymbol{p} - \mathbf{p}_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta})\|_2^2 \tag{4.4}$$

where $\boldsymbol{p}$ is a mixed label for input $\boldsymbol{x}$. With the addition of another regularization term $\mathcal{L}_{\mathrm{reg}}$ which prevents assigning all samples to a single class, the final total error is:

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\mathrm{reg}} \tag{4.5}$$

where λ_r is set to 1 for all experiments and $\lambda_{\mathcal{U}}$ is set to the same value as used in the experiment of DivideMix [33].

4.2.2 RandAugment

Data augmentation is a widely used technique to increase the number of training samples to enhance the generalization performance of image classification model. Typical data augmentation methods for images include rotation, flipping, cropping, etc. In general, data augmentation methods require expertise in each domain to apply plausible transformations to each sample. Recently, several approaches [9, 35] have been developed that learn optimal policies on a small proxy task for automatically designing augmentation strategies without prior knowledge of each domain. However, these approaches require a huge computational cost to find the optimal hyperparameters in their search space.

Instead of searching for hyperparameters in a proxy task, RandAugment [10] performs a grid search on the validation set to determine the best hyperparameters with drastically reduced computational cost. RandAugment is a simple data augmentation method using hyperparameters n and m, where n controls the number of transformations to be applied to a single sample and m controls the magnitude of each transformation. RandAugment selects n transformation from the following transformations with uniform probability for every image in every minibatch.

- identity
- autoContrast
- equalize
- rotate
- solarize
- color
- posterize
- contrase
- brightness
- sharpness
- shear-x
- shear-y
- translate-x
- translate-y

4.3 Consistency Regularization on Selected Samples

Our proposed method is based on DivideMix [33] which is an excellent approach for learning with noisy labels by modeling the loss distribution and selecting clean samples. Since recent studies show that a deep learning model in the early learning phase is robust against the noisy labels[2, 36], DivideMix selects samples with small loss values that are easy to classify as clean samples. In other words, DivideMix may overfit to the samples with simple patterns and fail to select samples with true labels but hard to classify.

Our method prevents DivideMix from overfitting to the samples with simple patterns by introducing a consistency regularization, which is widely used in semi-supervised learning (SSL). In SSL, consistency regularization encourages the training model to output the same values to the perturbed version of the unlabeled sample as to the original sample. In our method, consistency regularization is applied to selected samples by GMM and it encourages predictions on perturbed selected samples to be consistent with predictions on the original ones. This prevents training model from overfitting to the simple patterns, and encourages the model to learn samples with true labels but hard to classify by transforming easier samples by perturbation. While DivideMix uses only random cropping and horizontal flipping as data augmentation methods, our method uses RandAugment [10] as a method for adding perturbation to training images. The abstract of our method is summarized in Fig.4.1 .

At first, we train two networks $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ on the original noisy dataset $\tilde{\mathcal{D}} = \{\mathcal{X}, \tilde{\mathcal{Y}}\} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{N}$ for a few epochs. This training period comes from the belief about robustness of deep learning in the early learning phase. After the *warmup* period, all training samples are fed into $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ and a two-components GMM is fit to these loss distributions. Then select a set of clean samples as a set of labeled samples $\mathcal{D}_{\boldsymbol{\theta}_1}, \mathcal{D}_{\boldsymbol{\theta}_2}$ and define a complementary set as a set of unlabeled samples $\bar{\mathcal{D}}_{\boldsymbol{\theta}_1}, \bar{\mathcal{D}}_{\boldsymbol{\theta}_2}$ based on the output from a two-components GMM. If a clean probability w_i of each sample i exceeds τ, i is selected as clean. As shown in DivideMix, compute the refined labels $\bar{\boldsymbol{y}}_b$ and $\bar{\boldsymbol{q}}_b$ of each sample from $\mathcal{D}_{\boldsymbol{\theta}_l}$ and $\bar{\mathcal{D}}_{\boldsymbol{\theta}_l}$ for $l \in \{1, 2\}$. After transforming each label, MixUp and MixMatch transforms each dataset into $\mathcal{D}'_{\boldsymbol{\theta}_l}$ and $\bar{\mathcal{D}}'_{\boldsymbol{\theta}_l}$. The loss on $\mathcal{D}'_{\boldsymbol{\theta}_l}$ is the cross entropy loss $\mathcal{L}_{\mathcal{X}}$ and the loss on $\bar{\mathcal{D}}'_{\boldsymbol{\theta}_l}$ is the mean squared error $\mathcal{L}_{\mathcal{U}}$ as follows:

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{D}'_{\boldsymbol{\theta}_l}|} \sum_{\boldsymbol{x}, \boldsymbol{p} \in \mathcal{D}'_{\boldsymbol{\theta}_l}} \sum_{k} p^k \log(\mathrm{p}^k_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta}_l)) \tag{4.6}$$

$$\mathcal{L}_{\mathcal{U}} = -\frac{1}{|\bar{\mathcal{D}}'_{\boldsymbol{\theta}_l}|} \sum_{\boldsymbol{x},\boldsymbol{p} \in \bar{\mathcal{D}}'_{\boldsymbol{\theta}_l}} \|\boldsymbol{p} - \mathrm{p}_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta}_l)\|_2^2 \tag{4.7}$$

where $\boldsymbol{p}$ is a mixed label for input $\boldsymbol{x}$. In addition to these two terms, the loss for MixMatch consists of a regularization term as in Eq.4.5

Given a set of selected samples $\mathcal{D}_{\boldsymbol{\theta}_l} = \{\mathcal{X}_{\boldsymbol{\theta}_l}, \mathcal{Y}_{\boldsymbol{\theta}_l}\}$ where $l = \{1, 2\}$ be a set of clean samples for each network $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ at each epoch, each label $\boldsymbol{y}_b$ is refined to $\hat{\boldsymbol{y}}_b$ by label refinement and sharpening of DivideMix. Then for each $(\boldsymbol{x}_b, \hat{\boldsymbol{y}}_b) \in \mathcal{D}_{\boldsymbol{\theta}_l}$ for $b \in (1, \cdots, B)$, RandAugment is applied to $\boldsymbol{x}_b$ to convert the easy to classify samples into the hard to classify samples and a modified sample and modified sets of samples are denoted as $\boldsymbol{x}_b^{CR}$ and $\mathcal{D}_{\boldsymbol{\theta}_l}^{CR} = \{\mathcal{X}_{\boldsymbol{\theta}_l}^{CR}, \mathcal{Y}_{\boldsymbol{\theta}_l}^{CR}\} = \{\mathcal{X}_{\boldsymbol{\theta}_l}^{CR}, \hat{\mathcal{Y}}_{\boldsymbol{\theta}_l}\}$ where $\boldsymbol{y}_b^{CR} \in \mathcal{Y}_{\boldsymbol{\theta}_l}^{CR}$ of each sample is equal to the $\hat{\boldsymbol{y}}_b$ of original samples $\mathcal{D}_{\boldsymbol{\theta}_l}$. In terms of the consistency regularization, a sample $(\boldsymbol{x}_b, \hat{\boldsymbol{y}}_b) \in \mathcal{D}_{\boldsymbol{\theta}_l}$ corresponds to original sample and $(\boldsymbol{x}_b^{CR}, \hat{\boldsymbol{y}}_b) \in \mathcal{D}_{\boldsymbol{\theta}_l}^{CR}$ corresponds to the perturbed sample. Then the consistency on selected samples for l-th network is defined as follows:

$$\mathcal{L}_c = -\frac{1}{|\mathcal{D}_{\boldsymbol{\theta}_l}^{CR}|} \sum_{\boldsymbol{x}^c, \hat{\boldsymbol{y}} \in \mathcal{D}_{\boldsymbol{\theta}_l}^{CR}} \sum_k \hat{y}^k \cdot \log(\mathrm{p}_{\mathrm{model}}^k(\boldsymbol{x}^{CR}; \boldsymbol{\theta}_l)) \tag{4.8}$$

For the consistency regularization, we did not apply MixUp for further modification of inputs. $\mathcal{L}_{\mathcal{X}}$ modifies $\mathcal{D}_{\boldsymbol{\theta}_l}$ by MixUp, where MixUp is a linear interpolation between a sample of $\mathcal{D}_{\boldsymbol{\theta}_l}$ and the other sample. Therefore $\mathcal{L}_{\mathcal{X}}$ maintains information that the label of $\boldsymbol{x}_b$ is $\hat{\boldsymbol{y}}_b$. Then the total loss for each network is a sum of the loss of MixMatch and the consistency regularization:

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\mathrm{reg}} + \lambda_c \mathcal{L}_c \tag{4.9}$$

where λ_c is a hyperparameter.

In Algorithm 2, we summarized the entire computational procedure. In later chapters, in addition to the comparison between the proposed method and existing methods, we will discuss the change in the value of the error function for the selected samples and the change in the accuracy of sample selection by consistency regularization.

4.3.1 Extension of Consistency Regularization

For further improvement of our method, we introduced a consistency regularization averaged over multiple inputs. The new consistency regularization is defined as follows:

$$\mathcal{L}_c = -\frac{1}{I \cdot |\mathcal{D}^{CR}_{\boldsymbol{\theta}_l}|} \sum_{i=1}^{I} \sum_{x^{CR}, \hat{y} \in \mathcal{D}^{CR,(i)}_{\boldsymbol{\theta}_l}} \sum_{l} \hat{y}^k \log(\mathrm{p}^k_{\mathrm{model}}(\boldsymbol{x}^{CR}; \boldsymbol{\theta})) \tag{4.10}$$

where I is the number of trials that create augmented version of $\mathcal{D}_{\boldsymbol{\theta}_l}$ by RandAugment. $\mathcal{D}^{CR,(i)}_{\boldsymbol{\theta}_l}$ is the augmented dataset at i-th trial. $\mathcal{L}_c$ encourages model to receive different representations of $\boldsymbol{x}^c$ with consistent label $\hat{\boldsymbol{y}}$ over I trials.

4.4 Experiments

First, we performed an evaluation on a dataset containing synthetically generated label noise, and then conducted a comparison experiment with state-of-the-art approaches. In later subsections, we evaluate the sensitivity of the model to the hyperparameter λ_c in test accuracy and check the effect of consistency regularization on the value of the loss function for the selected sample.

4.4.1 Datasets and Implementation Details

We used CIFAR-10 and CIFAR-100 [26] dataset for training and validating our proposed method. Both dataset contains 50K training images and 10K test images of size 32×32. We extract 5K samples from training samples as a validation set. Following the prior works[33][17], we add synthetically generated label noise with various noise rate to each dataset for evaluating our method. The type of label noise is symmetric noise, which randomly flips the labels of the training samples to one of the categories with a certain probability.

We used a 18-layers PreAct Resnet for CIFAR-10 and CIFAR-100 experiments. We trained each model using SGD with a moment of 0.9, a weight decay of $1.0e-4$, and a batch size of 128. The total number of training epochs is 300. The initial learning rate is 0.02 and it's multiplied by 0.1 at every 10 epochs after 150-th epoch. According to DivideMix, the other hyperparameters are set as follows: $J = 2$, $T = 0.5$, $\alpha = 4$ and $\tau = 0.5$. α and τ are the hyperparameter for beta distribution of MixMatch and clean probability threshold for GMM, respectively. λ_U is set as 25 except for 20% noise ratio when it is set as 0 in the CIFAR10 experiment, and is set as 150 except for 20% noise ratio when it is set at 25 in the CIFAR-100 experiment. The warmup period is set to 10 for CIFAR-10 and set to 30 for CIFAR-100. The hyperparameters of RandAugment is determined by the classification accuracy on the validation set and summarized in Table.4.2.

4.4.2 Comparison with State-of-the-art Methods

We compared our method with DivideMix[33] and Augment Descent (AugDesc)[43], which is the current state-of-the-art for learning with noisy labels through sample selection and data augmentation, using the same network architecture. AugDesc defines the common random flip and crop image augmentation as weak data augmentation, and AutoAugment [9] as strong data augmentation. AugDesc models loss distribution on weakly or strongly augmented training samples by a two-component GMM to divide the dataset into a labeled set and an unlabeled set. Then AugDesc trains a classification model on strongly augmented training samples in a semi-supervised manner following training procedures of DivideMix. AugDesc-SAW (AugDesc-WAW) trains model with strong (weak) data augmentation during warmup period. The proposed method and AugDesc are quite similar in terms of sample selection and data augmentation strategy, but AugDesc is a method that learns only with perturbed samples for parameter updates, whereas our proposed method imposes constraints on classification model to ensure that the outputs for perturbed samples are similar to ones for original samples. This means that AugDesc does not obtain information of original samples and our proposed method receives samples selected as clean more frequently than AugDesc. In Table.4.1, Cross-Entropy denotes baseline model trained with Cross-Entropy loss using the same network architecture. We report the best test accuracy obtained during training and mean test accuracy averaged across the last 10 training epochs. The results with different levels of symmetric label noise on CIFAR-10 and CIFAR-100 are summarized in the Table.4.1 and the results given by our method (Ours ($I = 1$)) is compared with the above state-of-the-art approaches, where I is the number of augmented samples used in the consistency regularization. We also report the results from a variants of our method: consistency regularization with 2 trials of data augmentation (Ours ($I = 2$)). As the number of I increases, the model receives more augmented samples for stronger effect on preventing overfitting. At the same time, learning the original samples in the classification loss ($\mathcal{L}_{\mathcal{X}}$) becomes more difficult. We prepared these variants to explore the appropriate number of augmented samples in the consistency regularization. Our method (Ours ($I = 1$)) achieves superior results on CIFAR-10 with 50% label noise and CIFAR-100 with 20% label noise than the state-of-the-art approaches. When the noise rate of CIFAR-100 is 80%, AugDesc-WAW achieves better results than the other methods. Our method (Ours ($I = 2$)), which performs multiple trials of data augmentation to input, yields results comparative to or lower than ours with $I = 1$.

4.4.3 Analysis of the Effectiveness of Consistency Regularization

Table.4.3 shows the results of the analysis of the effect of the hyperparameter λ_c on the performance of the model(Ours($I = 1$)) on datasets with label noise. The value of λ_c is one of $\{0.1, 0.5, 1.0, 1.5\}$ in the experiment and Table.4.3 shows that the performance of model is sensitive to λ_c when the noise rate is high (80%) on both datasets. For higher noise rate (80%), the model with smaller λ_c yields better results. On the contrary, the results on the different noise rate are less sensitive to the value of λ_c.

Fig.4.3 shows how the value of $\mathcal{L}_{\mathcal{X}}$ with different value of λ_c changes as learning progresses on both datasets. We observe the value of $\mathcal{L}_{\mathcal{X}}$ to confirm the effect of the consistency regularization by changing the value of λ_c. Y-axis of each plot shows the average value of $\mathcal{L}_{\mathcal{X}}$ for each epoch. The value of λ_c is one of $\{0.0, 0.1, 0.5, 1.0, 1.5\}$, and the top row is the result of CIFAR-10 and the bottom row is the result of CIFAR-100. $\mathcal{L}_{\mathcal{X}}$ is the loss of MixMatch on selected sample by GMM, and measures how well the model fits those samples. In all plots, the value of $\mathcal{L}_{\mathcal{X}}$ with larger λ_c is greater than the ones with smaller λ_c. Since the best classification accuracy of our method is given when the value of λ_c is nonzero, we can deduce that the larger value of $\mathcal{L}_{\mathcal{X}}$ is better for generalization performance, and the consistency regularization prevents model from overfitting to the selected samples. However, if the value of λ_c is too large, $\mathcal{L}_{\mathcal{X}}$ also becomes large. This causes the learning model to underfit the selected sample, resulting in poor generalization performance as shown in Table.4.3.

Fig.4.4 and 4.5 shows the number of selected samples as clean and Area Under the Curve (AUC) for clean/noisy sample classification at each epoch on CIFAR-10 and CIFAR-100 training data with various $\lambda_c \in \{0.0, 0.1, 0.5, 1.0, 1.5\}$ when the noise rate is 20% or 80%. In results on both CIFAR-10 and CIFAR-100, if $\lambda_c > 0$, the number of selected samples is greater than the case with $\lambda_c = 0$ while AUC is comaparative or better. This means that the loss distribution given by model trained with consistency regularization selects the more samples with correct labels more accurately. One possible explanation for less accurate sample selection with $\lambda_c = 0$ is that the model overfits to the samples with simple patterns and regards the samples with correct labels but hard to classify as corrupted data.

4.4.4 Discussion

Table.4.1 shows that our method achieves superior results than DivideMix[33] (baseline model) in all cases, and better than or comparative to the results given by AugDesc[43] except for 80% symmetric label noise on CIFAR-100 dataset. One possible explanation for this shortage is that the number of selected sample for each class is small by GMM for high noise rate, while AugDesc applies strong data augmentation to all training samples. Fig.4.6 shows that the number of selected samples by GMM at each epoch and the number of selected samples decreases as the noise rate in the dataset increases. Hyper-parameter setting used in this experiments are shown in Table.4.2. Therefore our method does not consider the consistency on the enormous unlabeled (not selected) samples during training when the noise rate is high. For further improvements of our methods, we should address the consistency on the unlabeled (not selected) samples during training. And we compared two variants of our method (Ours($I = 1$) and Ours($I = 1$)) in Table.4.1. As the value of I increases, the model receives more augmented samples, which is expected to have a stronger effect on preventing overfitting, but on the other hand, learning the original samples in $\mathcal{L}_X$ becomes more difficult. Experimental results show that $I = 1$ is sufficient for learning.

Training model with consistency regularization receives the perturbed selected samples and their original ones at the same time. Then model does not overfit to the selected samples. As shown in Fig.4.3 for both datasets with various noise rates, the classification loss on the selected samples $\mathcal{L}_{\mathcal{X}}$ increases as the λ_c increases in both datasets with various noise rates, indicating that the model using consistency regularization does not overfit to the selected samples during training. Also our method with nonzero λ_c achieved the best classification accuracy, it's provably showed that preventing the model from fitting to the selected samples is benefiting for higher generalization performance. Since the samples selected by fitting GMM to the loss distribution have simple patterns and tend to be easy to classify [2], our consistency regularization contributes to preventing the model from overfitting to the simple patterns and results in higher generalization performance.

Fig.4.4 and 4.5 showed that the number of selected samples and the value of AUC of classification accuracy for noisy/clean labels given by our method is larger than the baseline ($\lambda_c = 0$) during training when noise rate is small (20%) in both datasets. This means that the model correctly outputs high loss value to the samples with noisy labels and the model was able to be trained with more correctly labeled samples during training. When the noise rate is 80% in CIFAR-100, the results are also better than the baseline. However, when the noise rate is 80% in CIFAR-10, the number of selected samples and AUC given by our method is almost equal to the baseline. As shown in Table.4.1, the classification accuracy is also equal to the baseline. These results also support the claim that our method is less effective when the noise rate is large, because the number of samples to which consistency regularization is applied is small. One possible improvement is to generate accurate pseudo-labels for unlabeled samples, and apply consistency regularization to them as well.

Algorithm 2 Training with Consistency Regularization

Input: Dataset with noisy labels $\tilde{\mathcal{D}} = \{\mathcal{X}, \tilde{\mathcal{Y}}\} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^N$, Two networks $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, sharpening temperature T, unsupervised loss weight λ_U, regularization term weight λ_c, $\lambda_r = 1$, Beta distribution α for MixMatch, Number of augmentations J

```
 1: // Training stage
 2: $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 = \text{WarmUp}(\tilde{\mathcal{D}}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$
 3: while $e <$ MaxEpoch do
 4:     $\mathcal{W}_2 = \text{GMM}(\tilde{\mathcal{D}}, \boldsymbol{\theta}_1)$
 5:     $\mathcal{W}_1 = \text{GMM}(\tilde{\mathcal{D}}, \boldsymbol{\theta}_2)$
 6:     for $l = 1, 2$ do
 7:         $\mathcal{D}^{(l)}_{\boldsymbol{\theta}_l,e} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i, w_i) | w_i \geq \tau, \forall(\boldsymbol{x}_i, \boldsymbol{y}_i, w_i) \in (\tilde{\mathcal{D}}, \mathcal{W}_l)\}$
 8:         $\bar{\mathcal{D}}^{(l)}_{\boldsymbol{\theta}_l,e} = \{\boldsymbol{x}_i | w_i < \tau, \forall(\boldsymbol{x}_i, w_i) \in (\tilde{\mathcal{D}}, \mathcal{W}_l)\}$
 9:         for iter = 1 to num_iter do
10:             Draw a mini-batch $\{(\boldsymbol{x}_b, \boldsymbol{y}_b, w_b)\}_{b=1}^B$ from $\mathcal{D}_{\boldsymbol{\theta}_l,e}$
11:             Draw a mini-batch $\{(\boldsymbol{u}_b)\}_{b=1}^B$ from $\bar{\mathcal{D}}_{\boldsymbol{\theta}_l,e}$
12:             for $b = 1$ to $B$ do
13:                 $\boldsymbol{x}_b^{CR} = \text{RandAugment}(\boldsymbol{x}_b | N, M)$
14:                 for $j = 1$ to $J$ do
15:                     $\hat{\boldsymbol{x}}_{b,j} = \text{Augment}(\boldsymbol{x}_b)$
16:                     $\hat{\boldsymbol{u}}_{b,j} = \text{Augment}(\boldsymbol{u}_b)$
17:                 end for
18:                 $\boldsymbol{p}_b = \frac{1}{J}\sum_j \mathbf{p}_{\text{model}}(\hat{\boldsymbol{x}}_{b,j}; \boldsymbol{\theta}_k)$
19:                 $\hat{\boldsymbol{y}}_b = w_b \boldsymbol{y}_b + (1 - w_b)\boldsymbol{p}_b$
20:                 $\hat{\boldsymbol{y}}_b = \text{Sharpen}(\bar{\boldsymbol{y}}_b, T)$
21:                 $\bar{\boldsymbol{q}}_b = \frac{1}{2J}\sum_j (\mathbf{p}_{\text{model}}(\hat{\boldsymbol{u}}_{b,j}; \boldsymbol{\theta}_1) + \mathbf{p}_{\text{model}}(\hat{\boldsymbol{u}}_{b,j}; \boldsymbol{\theta}_2))$
22:                 $\hat{\boldsymbol{q}}_b = \text{Sharpen}(\bar{\boldsymbol{q}}_b, T)$
23:             end for
24:             $\hat{\mathcal{X}} = \{(\hat{\boldsymbol{x}}_{b,j}, \hat{\boldsymbol{y}}_b); b \in (1, \cdots, B), j \in (1, \cdots, J)\}$
25:             $\hat{\mathcal{U}} = \{(\hat{\boldsymbol{u}}_{b,j}, \hat{\boldsymbol{q}}_b); b \in (1, \cdots, B), j \in (1, \cdots, J)\}$
26:             $\mathcal{L}_{\mathcal{X}}, \mathcal{L}_{\mathcal{U}} = \text{MixMatch}(\hat{\mathcal{X}}, \hat{\mathcal{U}})$
27:             $\mathcal{D}^{(l),CR}_{\boldsymbol{\theta}_l,e} = \{(\boldsymbol{x}, \boldsymbol{y}) | \boldsymbol{x} \in \boldsymbol{x}^{CR}, \boldsymbol{y} \in \hat{\boldsymbol{y}}\}$
28:             $\mathcal{L}_c = \text{ConsistencyRegularization}(\mathcal{D}^{(l),CR}_{\boldsymbol{\theta}_l,e})$
29:             $\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_U \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\text{reg}} + \lambda_c \mathcal{L}_c$
30:             $\boldsymbol{\theta}_l = \text{SGD}(\mathcal{L}, \boldsymbol{\theta}_l)$
31:         end for
32:     end for
33: end while
```

Table 4.1: Comparison with baseline methods and current state-of-the-art approaches on CIFAR-10 and CIFAR-100 with symmetric label noise in test accuracy (%). DivideMix [33] and AugDesc-(SAW/WAW) [43] was reimplemented using public code. The mean accuracy and its standard deviation are computed over five noise realizations.

		CIFAR-10			CIFAR-100		
Model	Noise	20%	50%	80%	20%	50%	80%
Cross Entropy	Best	86.39 ± 0.51	79.49 ± 0.48	59.48 ± 1.19	61.99 ± 0.39	47.26 ± 0.59	22.35 ± 0.80
	Last	83.63 ± 0.17	58.49 ± 0.31	26.44 ± 0.53	61.73 ± 0.37	38.58 ± 0.47	10.72 ± 0.20
DivideMix[33]	Best	96.07 ± 0.07	94.72 ± 0.06	92.72 ± 0.27	76.38 ± 0.16	72.98 ± 0.13	56.02 ± 0.62
	Last	95.79 ± 0.07	94.44 ± 0.11	92.47 ± 0.31	75.93 ± 0.14	72.46 ± 0.07	55.65 ± 0.64
AugDesc-SAW[43]	Best	96.13 ± 0.07	94.55 ± 0.43	93.50 ± 0.40	78.86 ± 0.04	76.72 ± 0.17	55.44 ± 3.66
	Last	95.96 ± 0.09	94.39 ± 0.44	93.33 ± 0.38	78.61 ± 0.04	76.44 ± 0.22	55.20 ± 3.65
AugDesc-WAW[43]	Best	96.17 ± 0.08	95.24 ± 0.13	93.56 ± 0.48	78.86 ± 0.09	76.67 ± 0.20	$\mathbf{64.53 \pm 0.50}$
	Last	95.99 ± 0.09	95.04 ± 0.16	93.38 ± 0.49	78.61 ± 0.12	76.38 ± 0.20	$\mathbf{64.30 \pm 0.50}$
Ours (I=1)	Best	$\mathbf{96.73 \pm 0.08}$	96.20 ± 0.09	$\mathbf{93.99 \pm 0.13}$	80.83 ± 0.09	$\mathbf{77.17 \pm 0.27}$	61.19 ± 0.58
	Last	$\mathbf{96.47 \pm 0.12}$	95.96 ± 0.09	$\mathbf{93.78 \pm 0.14}$	80.28 ± 0.14	$\mathbf{76.73 \pm 0.41}$	60.81 ± 0.68
Ours (I=2)	Best	96.67 ± 0.05	$\mathbf{96.37 \pm 0.07}$	93.83 ± 0.11	$\mathbf{81.11 \pm 0.17}$	76.50 ± 0.38	61.05 ± 0.45
	Last	96.39 ± 0.04	$\mathbf{96.11 \pm 0.06}$	93.64 ± 0.09	$\mathbf{80.51 \pm 0.12}$	76.20 ± 0.35	60.68 ± 0.51

Table 4.2: The value of RandAugment hyperparameters N and M used in Table.4.1 with CIFAR-10 and CIFAR-100 at different levels of noise rate.

	CIFAR-10			CIFAR-100		
Noise Rate	20%	50%	80%	20%	50%	80%
N	2	1	2	1	1	2
M	2	4	4	2	2	2
λ_c	0.5	0.5	0.1	0.5	0.5	0.1
$\lambda_{\mathcal{U}}$	0	25	25	25	150	150

Table 4.3: Test accuracy on CIFAR-10 and CIFAR-100 with noisy labels with different various on Attention module. The mean accuracy and its standard deviation are computed over three noise realizations

		CIFAR-10			CIFAR-100		
	Noise	20%	50%	80%	20%	50%	80%
$\lambda_c = 0.1$	Best	96.67	95.86	**94.17**	79.90	76.31	60.94
	Last	96.36	95.51	**93.91**	79.40	75.89	60.57
$\lambda_c = 0.5$	Best	**96.79**	**96.35**	93.77	80.69	77.18	60.14
	Last	**96.48**	**96.09**	93.52	80.38	76.79	60.01
$\lambda_c = 1.0$	Best	96.30	95.92	91.59	79.88	75.49	56.63
	Last	96.09	95.74	91.21	79.48	75.83	56.29
$\lambda_c = 1.5$	Best	96.16	95.79	90.38	79.14	74.51	54.87
	Last	95.76	95.52	90.19	78.73	74.02	54.45

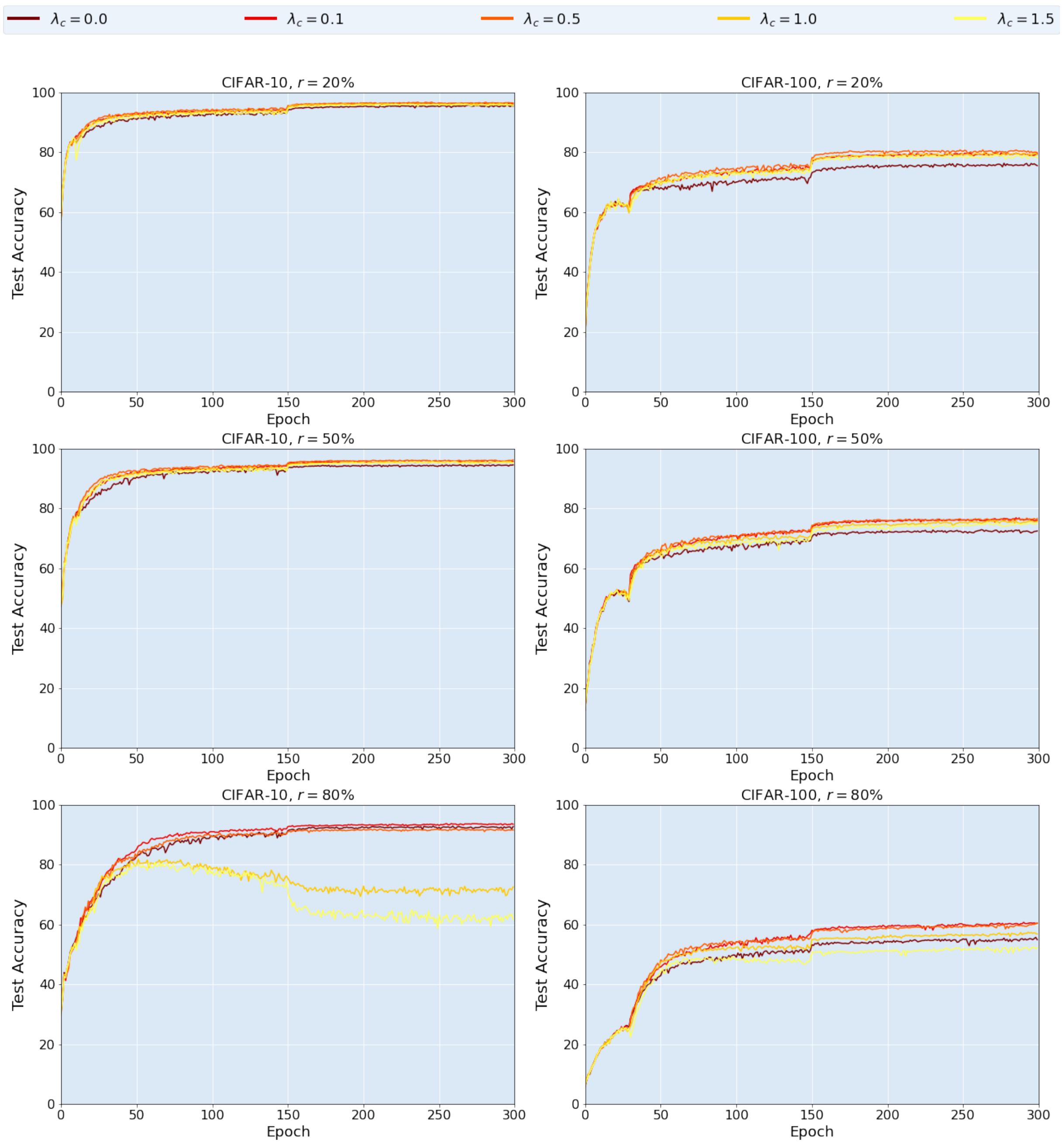


Figure 4.2: Test Accuracy on CIFAR-10 and CIFAR-100 dataset with different values of lambda. Each row shows different noise rate $(20\%, 50\%, 80\%)$.

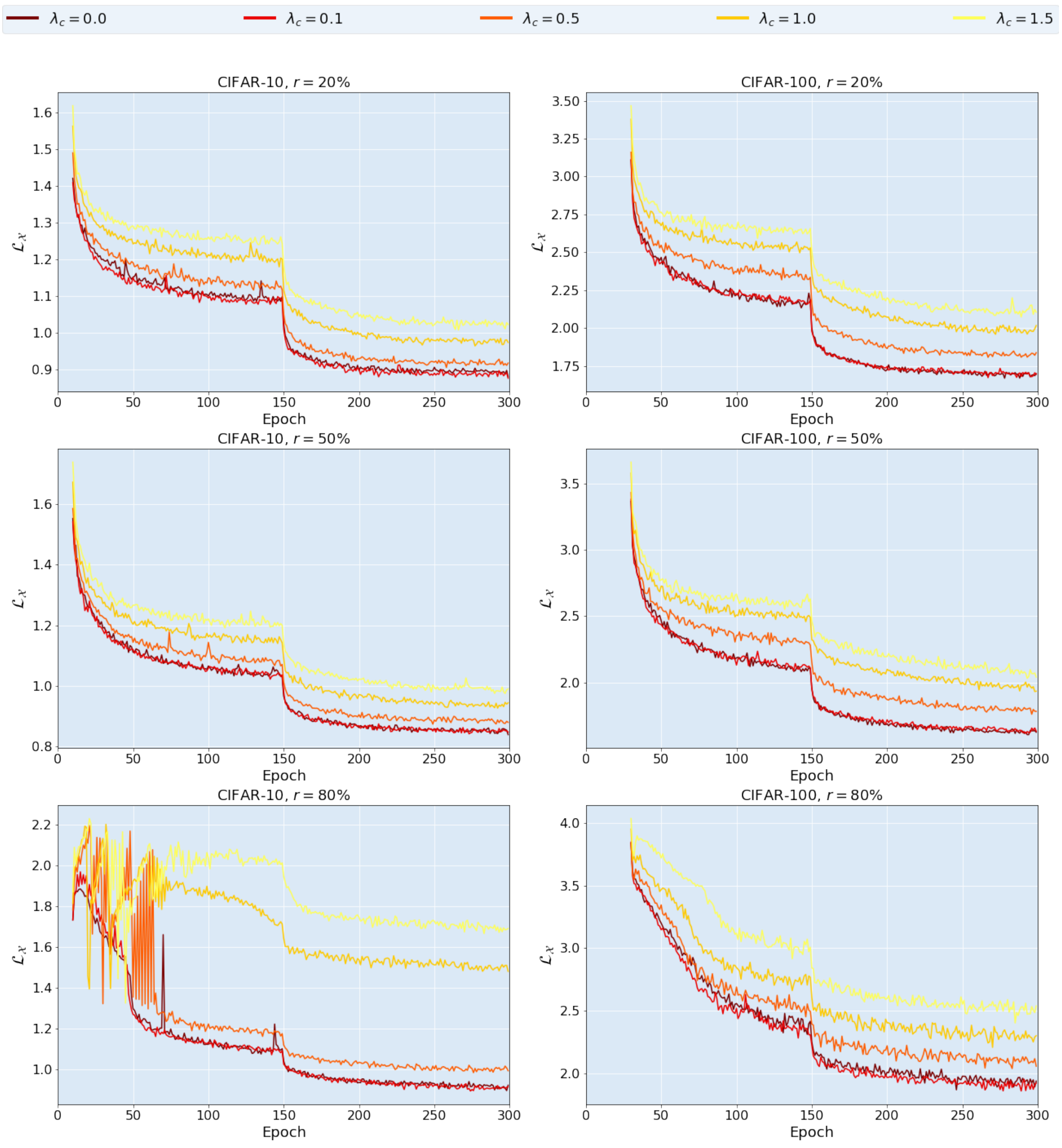


Figure 4.3: Results of $\mathcal{L}_\mathcal{X}$ on CIFAR-10 and CIFAR-100 dataset with different values of lambda. Each row shows different noise rate $(20\%, 50\%, 80\%)$. Right column: $\mathcal{L}_\mathcal{X}$ vs number of epochs on CIFAR-10; Left column: $\mathcal{L}_\mathcal{X}$ vs number of epochs on CIFAR-100.

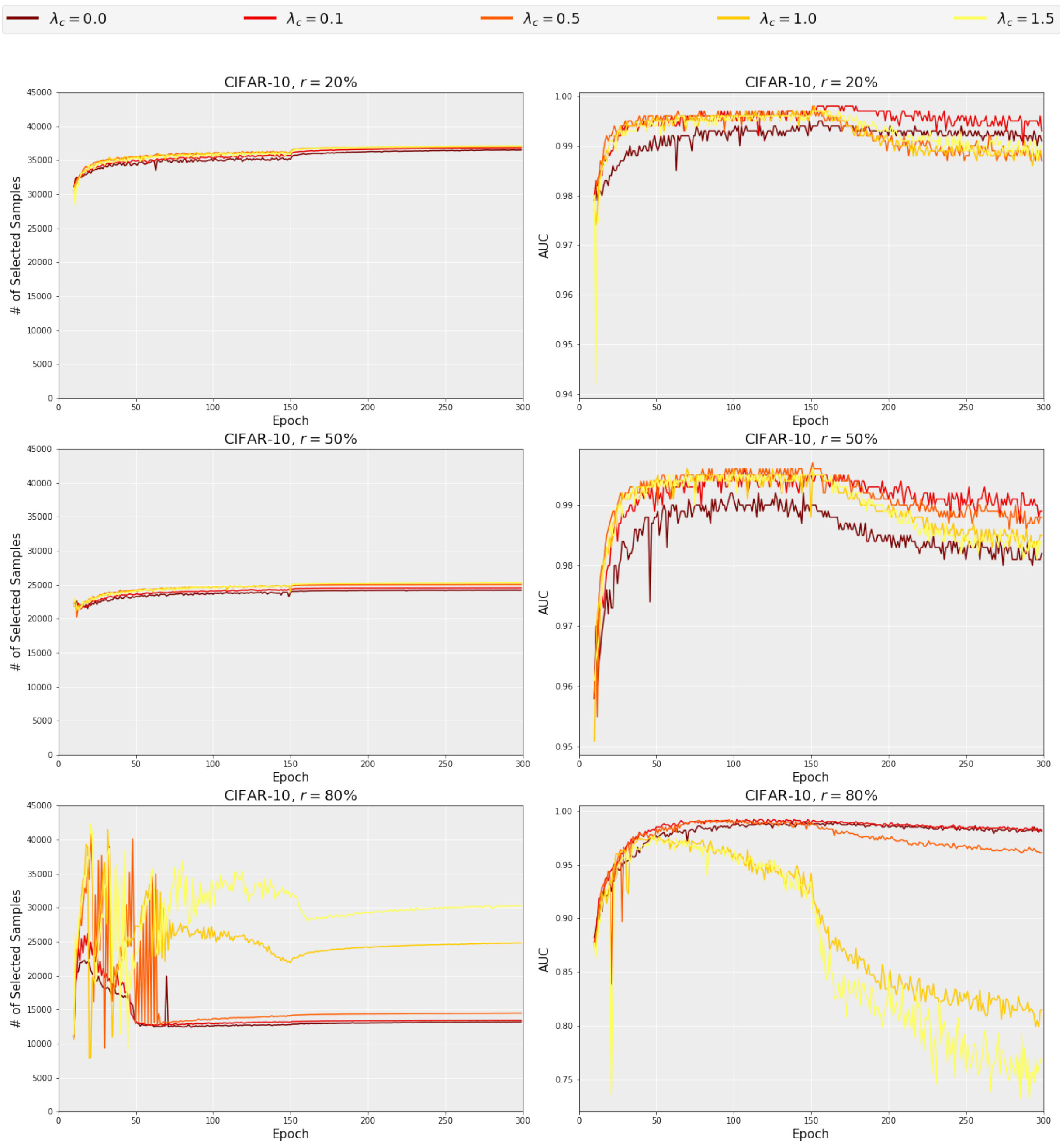


Figure 4.4: Results of the number of selected samples by GMM and AUC of classification accuracy for noisy/clean labels at each epoch on CIFAR-10. Left column: Each plot shows number of samples selected as clean vs. Number of epochs. Right column: Each plot shows Area Under the Curve (AUC) for clean/noisy classification vs number of epochs.

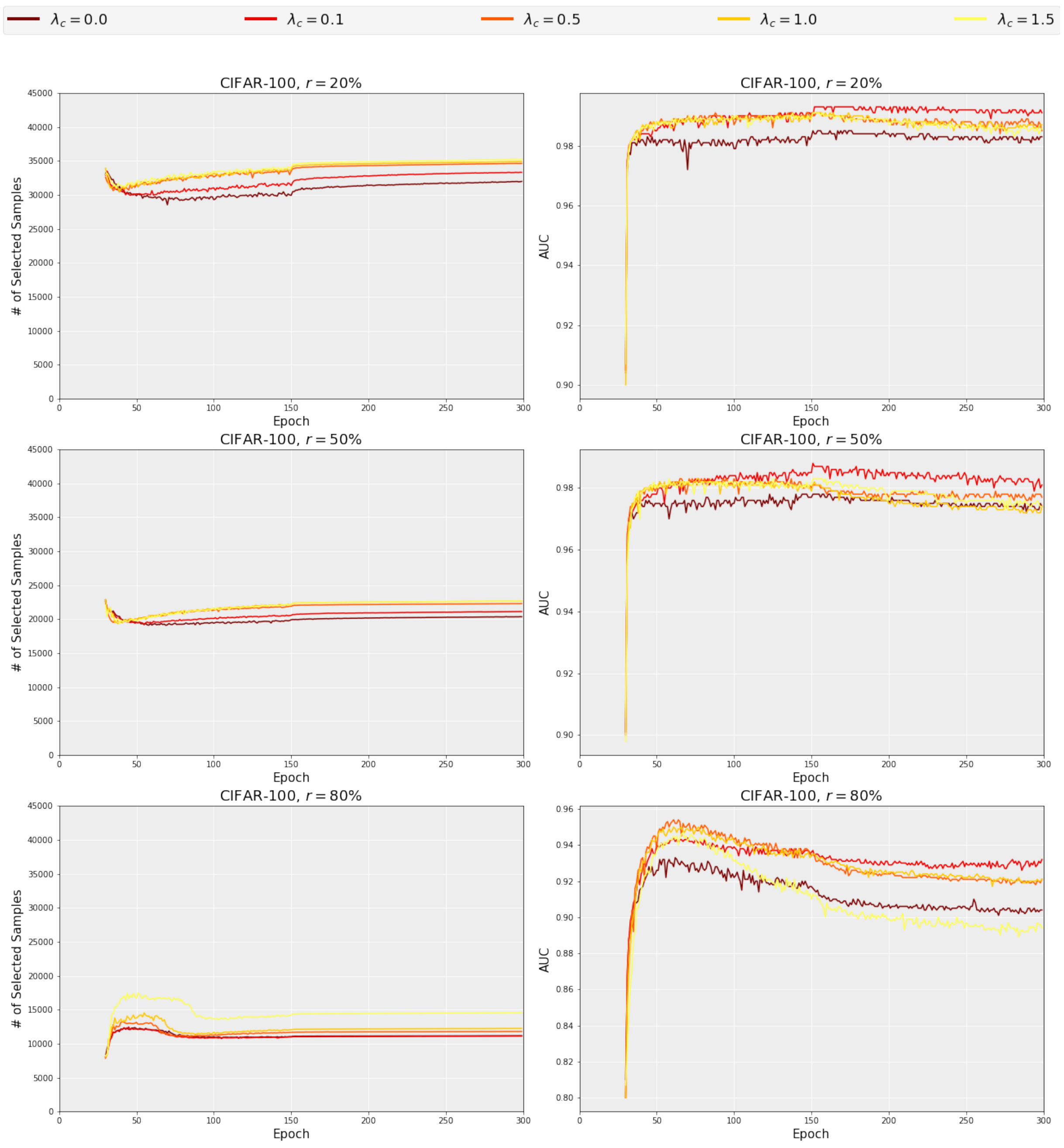


Figure 4.5: Results of the number of selected samples by GMM and AUC of classification accuracy for noisy/clean labels at each epoch on CIFAR-100. Left column: Each plot shows number of samples selected as clean vs. Number of epochs. Right column: Each plot shows Area Under the Curve (AUC) for clean/noisy classification vs number of epochs.

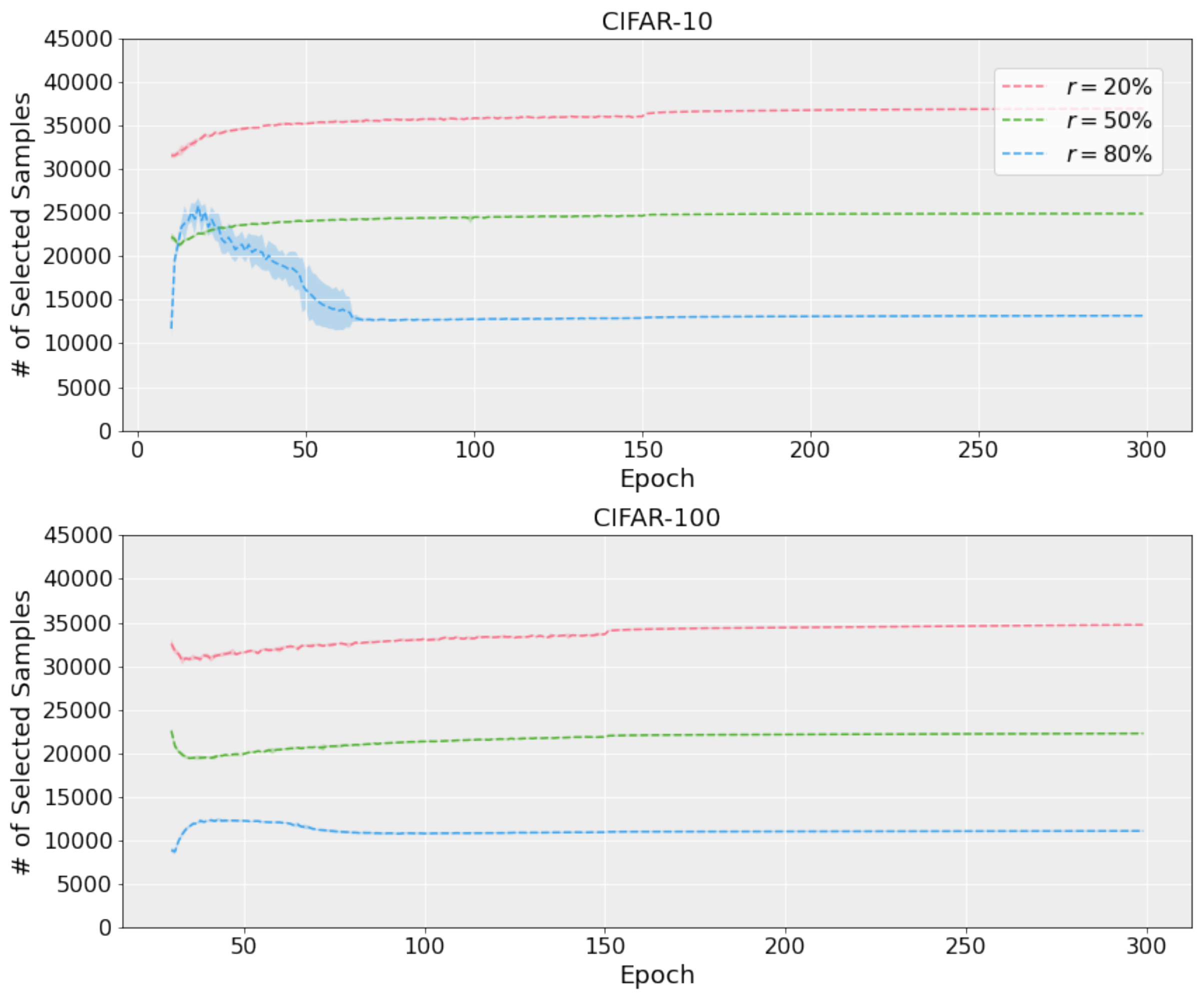


Figure 4.6: Top and bottom plot show the number of selected samples vs number of epochs on CIFAR-10 and CIFAR-100 with different label noise ratio over five trials.

Chapter 5

Sample selection approach with number of false predictions

5.1 Introduction

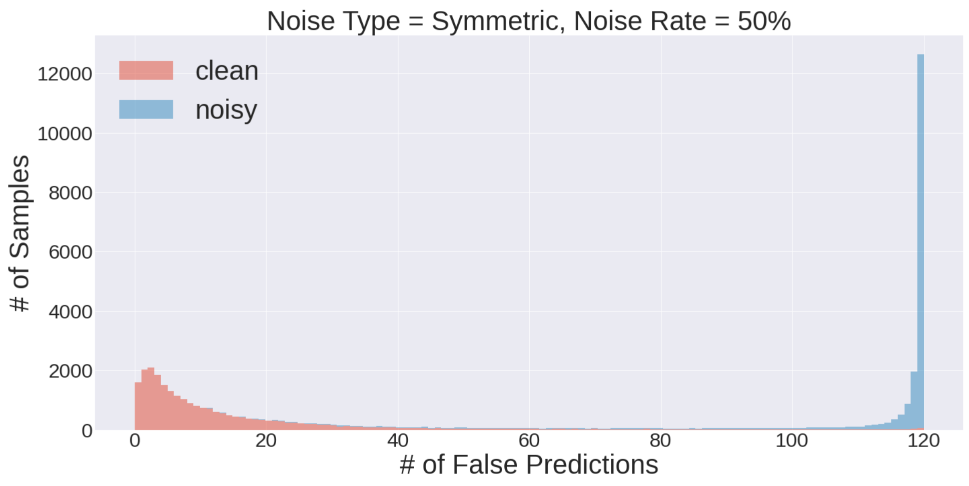


Figure 5.1: Visualization of total number of false predictions during training for each sample of CIFAR-10 dataset with 50% symmetric label noise. The model is trained for 120 epochs, and count the false prediction at each epoch. Red and Blue show the samples with clean labels and ones with noisy labels, respectively.

Analysis of the memorization effect revealed that DNNs are robust against the noisy labels in the early stage of learning before overfitting to the noisy labels [2]. During train-

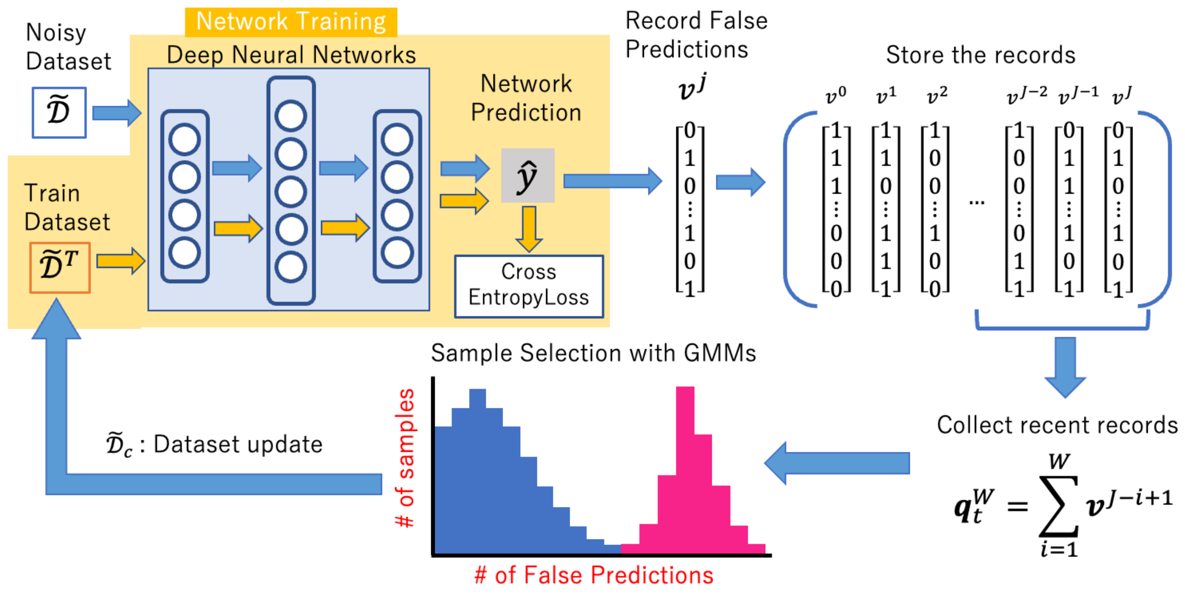


Figure 5.2: At first, we prepare a copy of original noisy dataset $\tilde{\mathcal{D}}$ as $\tilde{\mathcal{D}}^T$. SSFP trains DNNs with training dataset $\tilde{\mathcal{D}}^T$ and performs sample selection with $\tilde{\mathcal{D}}$. At epoch t during training, we record the false predictions $\boldsymbol{v}^j$ based on the model output $\hat{\boldsymbol{y}}$ with $\tilde{\mathcal{D}}$. For sample selection, SSFP collect only W recent records of false predictions to summarize recent records as $\boldsymbol{q}_t^W$. Then SSFP fits a two-components Gaussian Mixture Models to $\boldsymbol{q}_t^W$ and perform clean sample selection to update the training dataset $\tilde{\mathcal{D}}^T$ to $\tilde{\mathcal{D}}_c$. We resume training the DNNs with the updated dataset $\tilde{\mathcal{D}}^T$.

ing with the stochastic gradient descent, DNNs tend to predict true labels of samples with noisy labels in the early phase due to its robustness, and then gradually predict the assigned noisy labels, whereas the model frequently predicts the true label of a sample with a clean label throughout the training. From these observations, we call an incorrect prediction for a given label a false prediction, and assume that the number of false predictions for each sample during training reflects whether the sample has a noisy label or not.

To intuitively understand how the number of false predictions are different for samples with noisy labels and samples with clean labels, we trained ResNet-34 with the cross entropy loss on a noisy dataset for 120 epochs and recorded false prediction for each sample at each epoch. Here we prepared CIFAR-10 dataset and flip the labels of randomly chosen 50% training samples to one of the 10 classes. Fig.5.1 visualizes the total number of false predictions during training of each samples, where x-axis and y-axis show the total number of false predictions and the number of samples, respectively. As shown in Fig.5.1, samples with clean labels tend to have lower number of false predictions whereas samples with noisy labels have higher number of false predictions. From these observations, we propose a new sample selection approach for LNL utilizing the number of false predictions during training [46]. Our method iteratively performs sample selection based on the some past records of false predictions and updating model parameters with the selected samples. An overview of our method is shown in Fig.5.2.

The contribution of this study are summarized as follows: (1) We propose SSFP, a new **S**ample **S**election approach using the number of **F**alse **P**redictions for each sample during training. (2) Extensive evaluation on CIFAR-10 and CIFAR-100 with synthetically generated noisy labels is performed to validate that SSFP is better or comparative to the current state-of-the-art approaches. (3) We performed ablation study on hyper-parameters of SSFP to understand how we should set them for various label noise settings.

5.2 Related works

5.2.1 FINE and F-coteaching

FINE [24] is a novel noise detector for selecting clean samples from noisy dataset. FINE utilizes the principal components of latent feature representations of DNNs produced by the eigen decomposition for splitting dataset into clean set and noisy set. FINE first creates a gram matrix of feature vectors of training samples for each class, then conduct the eigen decomposition on the matrix to find the first principal components with the largest eigen value. Then FINE measures the similarity between the feature vector of each sample and the eigenvector with the largest eigen value to give the alignment score. This eigenvector is a class-representative vector and FINE treats the samples with higher alignment score as clean data, and samples with lower score as noisy data. After scores of all samples are obtained, FINE fits a two-component Gaussian Mixture Model to the distribution of scores to divide the samples into clean set and noisy set. The samples belong to a cluster having a larger mean value treated as a clean set. A sample selection method with FINE trains the DNNs with the selected samples as clean and periodically update the clean set during training. F-coteaching is an improved version of Co-teaching, where sample selection step is substituted to FINE.

5.3 Proposed method

5.3.1 Preliminary knowledge

Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i^N$ denotes a set of N training samples, where $\boldsymbol{x} \in \mathcal{X}$ is an image and $\boldsymbol{y} \in \mathcal{Y}$ is an one-hot vector of ground truth label for k classes of $\boldsymbol{x}$. Suppose we have a DNNs model for image classification task with model parameters $\boldsymbol{\theta}$, we formulate the loss function for image classification, known as cross entropy loss $\ell(\boldsymbol{\theta})$, as follows:

$$\ell(\theta) = -\sum_{i=1}^{N}\sum_{c=1}^{k} y_i^c \log(\mathrm{p}^c(f(\boldsymbol{x}_i, \boldsymbol{\theta}))) \tag{5.1}$$

where $f(\boldsymbol{x}_i, \boldsymbol{\theta})$ is a logit outputs from the last fully connected layer of DNNs with θ, and $\mathrm{p}^c(f(\boldsymbol{x}_i, \boldsymbol{\theta}))$ is c-th element of output from the softmax function. In our problem setting, training dataset with noisy labels is denoted as $\tilde{\mathcal{D}}$ and the observed label for i-th sample is denoted as $\tilde{\boldsymbol{y}}_i$. Training model θ on $\tilde{\mathcal{D}} = \{\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i\}_i^N$ by minimizing Eq.(5.1) significantly deteriorates the generalization performance. Our proposed method trains the model $\boldsymbol{\theta}$ robustly with a novel sample selection approach and achieves higher generalization performance.

Algorithm 3 Training with false predictions

```
Require: Dataset with noisy labels $\tilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^N$, Network parameter $\boldsymbol{\theta}$, Record frequency $F$, Sample selection frequency $K$, Window size $W$, Total epochs $T$, set of records $\mathcal{V}$, clean probability threshold $\gamma$, mini-batch size $B$
 1: // Initialization
 2: $\mathcal{V} \leftarrow \{\}$ # empty set for records
 3: $m = 0$ # batch count
 4: $j = 0$ # record count
 5: $\tilde{\mathcal{D}}^T = \tilde{\mathcal{D}}$ # training dataset
 6: // Training stage
 7: $\theta = \text{WarmUp}(\tilde{\mathcal{D}}, \theta)$
 8: while $t < T$ do
 9:     if ($t$ mod $K == 0$) and ($t >$ warmup) then
10:         Collect recent records : $\boldsymbol{q}_t^W = \sum_{j=J-W}^{J} \boldsymbol{v}^j$
11:         Fit a GMM to $\boldsymbol{q}_t^W$
12:         Clean probability : $\boldsymbol{o}_t = p(g_1|\boldsymbol{q}_t^W)$
13:         Update training dataset : $\tilde{\mathcal{D}}^T = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)|o_i \geq \gamma\}_{i=1}^N$ where $\tilde{\mathcal{D}}^T \in \tilde{\mathcal{D}}$
14:     end if
15:     for iter = 1 to $\left\lceil |\tilde{\mathcal{D}}^T|/B \right\rceil$ do
16:         $m = m + 1$
17:         Draw a mini-batch $\tilde{\mathcal{D}}_B^T$ from $\tilde{\mathcal{D}}^T$
18:         $\mathcal{L} = \text{CrossEntropy}(\tilde{\mathcal{D}}_B^T, \boldsymbol{\theta})$
19:         $\boldsymbol{\theta} = \text{SGD}(\tilde{\mathcal{D}}_B^T, \boldsymbol{\theta})$
20:         if $m$ mod $F == 0$ then
21:             Obtain false prediction $\boldsymbol{v}^j$ by evaluating $\tilde{\mathcal{D}}$
22:             $\mathcal{V} \leftarrow \boldsymbol{v}^j$
23:             $j = j + 1$
24:         end if
25:     end for
26: end while
```

5.3.2 Number of False Predictions

Suppose a DNNs model is trained for T epochs by the Stochastic Gradient Descent (SGD) with the mini-batch size of B. During training, we record whether the model's prediction matches the given label for each sample at every some constant mini-batches using the original dataset $\tilde{\mathcal{D}}$. This constant value is defined as *Frequency* of mini-batches for recording, and is denoted as F. Suppose that the predicted label of the i-th sample by the model is formulated as follows:

$$\hat{y}_i = \arg\max_c \mathbf{p}(f(\boldsymbol{x}_i, \boldsymbol{\theta}))) \tag{5.2}$$

Let $\boldsymbol{v}^j \in \mathbb{Z}^N$ store the information whether model's prediction is correct or not for each training sample at j-th record and formulated as follows:

$$v_i^j := \begin{cases} 1 & \text{if } \hat{y}_i \neq \arg\max_c \tilde{\boldsymbol{y}}_i, \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

This record for all training samples $\boldsymbol{v}^j$ is defined as *False Predictions* and the set of these past records represents the long term confidence of model's prediction with the given labels. The learning characteristics of DNNs in the early stages shows that the training model tends to predicts the true labels of training samples rather than the noisy labels and takes a while to over-fit to the noisy labels. Therefore, we assume that the model produces more false predictions for samples with noisy labels than for those with clean labels. Based on this assumption, we utilize the number of false predictions for each sample, which reflects the long term confidence of model's prediction, to classify the training samples into samples with noisy and clean labels. In order to save the computational cost, we record $\boldsymbol{v}^j$ at every F mini-batches instead of every mini-batch.

5.3.3 Clean sample probability and adaptive updates

SSFP, our proposed method trains a DNNs model on a training dataset $\tilde{\mathcal{D}}^T$, which is a copy of the original dataset $\tilde{\mathcal{D}}$, for a few epochs as before start splitting the dataset. This step is necessary in order to tune the model as a feature extractor. After the warm-up phase, SSFP periodically divides the samples $\tilde{\mathcal{D}}$ into clean set $\tilde{\mathcal{D}}_c$ and noisy set $\tilde{\mathcal{D}}_n$ at every K epoch and updates the training dataset as $\tilde{\mathcal{D}}^T = \tilde{\mathcal{D}}_c$ which only contains the selected clean samples.

Here we introduce a new term *Window*, denoted as W. This is a constant that indicates how many past records of false predictions are used for sample selection. Despite that the total number of false predictions after training is also useful for sample selection as shown in Fig.5.1, SSFP adopts an adaptive approach which only uses the recent records for sample selection. Since SSFP periodically updates the training dataset by removing samples with noisy labels, it prevents performance degradation of the model due to noisy labels. Therefore false predictions given by recent model are more reliable and SSFP only considers recent W records.

Suppose at some epoch point t for sample selection, J records are collected by evaluating the original dataset $\tilde{\mathcal{D}}$: $\mathcal{V} = \{\boldsymbol{v}^1, \cdots, \boldsymbol{v}^J\}$. SSFP only uses recent W records of false predictions: $\mathcal{V}_W = \{\boldsymbol{v}^{J-W}, \boldsymbol{v}^{J-W+1} \cdots, \boldsymbol{v}^{J-1}, \boldsymbol{v}^J\}$. Then the sum of number of false prediction for i-th sample with recent W records at t-th epoch is calculated as follows:

$$q_{t,i}^W = \sum_{j=J-W}^{J} v_i^j \tag{5.4}$$

$\boldsymbol{q}_t^W$ represents the sum of W records at t-th epoch for all training samples. Then a two-components GMM [51] is fit to $\boldsymbol{q}_t^W$ using the Expectation-Maximization algorithm. Let g_1 and g_2 be means of GMM and $g_1 \leq g_2$, then clean probability that a sample has clean label is defined as o_i and equals to the posterior probability $p(g_1|q_{t,i}^W)$ from the Gaussian component g_1. Then i-th sample is included in the clean set $\tilde{\mathcal{D}}_c$ if o_i exceeds a threshold γ ($o_i \geq \gamma$). Then the training dataset is updated with the clean set: $\tilde{\mathcal{D}}^T = \tilde{\mathcal{D}}_c$. After the sample selection, SSFP resumes training DNNs with the new dataset. SSFP iteratively performs sample selection and parameters update until training is completed. Algorithm.3 summarizes our proposed method. In the experiment, we evaluated various combination of record frequency F and window size W.

5.4 Experiments

In this section, we explain experimental setting for validating SSFP and show the results of comparison between SSFP with the state-of-the-art approaches. In addition, we evaluate the sensitivity of SSFP to the values of window size W and frequency F for clean probability calculation on classification accuracy. We also visualize the distribution of false predictions about for sample selection at some epoch point and monitor how the F-score for sample selection changes during training with various window sizes.

5.4.1 Datasets and Implementation Details

SSFP is validated using CIFAR-10 and CIFAR-100, which are baseline datasets for image classification. Both datasets consist of 50k training images and 10k test images. For hyper-parameters tuning with validation set, we split the original training images into 45k training set and 5k validation set. In our problem setting, training dataset contains noisy labels whereas validation and test set do not. In order to validate our method, artificially synthesized noisy labels with two different noise types (Symmetric and Asymmetric) and different noise rates r are added to CIFAR-10 and CIFAR-100. Symmetric label noise flips the labels of $r\%$ training samples randomly to one of the classes with a specific noise rate r, while Asymmetric label noise only flips the labels of $r\%$ training samples to a specific class. For CIFAR-10, Asymmetric noise are applied according to the following mapping rules: TRUCK$\rightarrow$AUTOMOBILE, BIRD$\rightarrow$AIRPLANE, DEER$\rightarrow$Horse, CAT$\leftrightarrow$DOG. In the case of CIFAR-100, 20 hyper-classes are prepared and label of leaf classes in each hyper-class is flipped to one of the leaf classes. We tested 20%, 50%, and 80% noise rates for symmetric noise and 40% noise rate for asymmetric noise.

We used the model architecture and hyper-parameter settings following the setup of [24]. The baseline model architecture is 34-layers ResNet for all experiments. We train each model using SGD with a moment of 0.9, a weight decay of $1.0e-3$, and a batch size of 128. Total number of training epochs for CIFAR-10 and CIFAR-100 are 120 and

150, respectively. The initial learning rate is 0.02, which is multiplied by 0.1 after 40-th and 80-th epochs for CIFAR-10, and 80-th and 120-th epochs for CIFAR-100. Threshold on clean probability for sample selection is $\gamma = 0.5$. Initial training period of 40-epochs is warm-up phase, and sample selection for dateset update is performed every 10 epochs after the warm-up. The window size W is set to one of $\{1, 5, 10, 50, 100, 200\}$ and the value of frequency F is set to one of $\{50, 100, 150\}$. The best values of W and F for each noise setting are selected based on the validation accuracy.

5.4.2 Comparison with State-of-the-art Methods

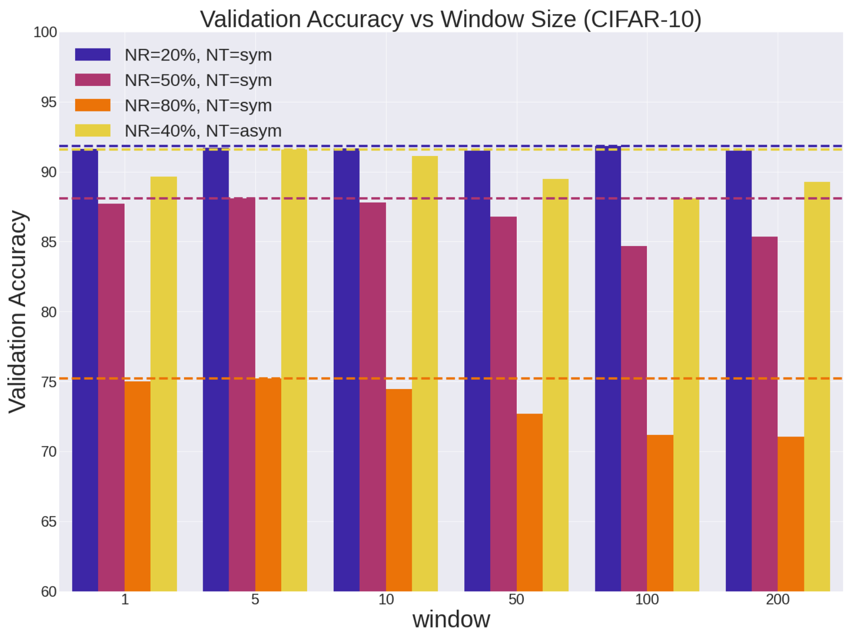
Table.5.1 shows the performance of our method, denoted as *SSFP*, on CIFAR-10 and CIFAR-100 dataset with different levels of symmetric and asymmetric label noise. In Table.5.1, we compare our method with the following conventional approaches: Bootstrap [54], Forward [50], Co-teaching [17], Co-teaching+ [73], TopoFIlter [67], CRUST [41]. FINE is the current state-of-the-art for learning with noisy labels through sample selection. F-coteaching is an improved version of Co-teaching by substituting its sample selection method with FINE algorithm. F-coteaching prepares two networks and each network teaches each other the selected sample as clean by FINE algorithm. Inspired by F-coteaching, SSFP is also adopted to Co-teaching, denoted as 'SSFP+co' in Table.5.1. 'SSFP+co' also prepares two networks and each network teaches clean samples each other by our adaptive sample selection approach. The two hyper-parameters of 'SSFP' and 'SSFP+co', F and W are summarized in Table.5.2 and 5.3. Table.5.1 summarizes the average test accuracy over three trials of different sample selection approaches on various label noise settings, and the results of FINE and F-coteaching are reproduced by their official implementation while the other results are taken from [24]. For the most part, our methods yields better results than the other state-of-the-art approaches, or comparative results in several noise settings. In the case of CIFAR-10, our methods yields comparable accuracy to F-coteaching in 20% and 50% symmetric label noise. And our methods achieves better results than the other methods in 80% symmetric noise and 40% asymmetric noise. In the case of CIFAR-100, our methods yields comparable accuracy to the other methods in 50% symmetric noise and 40% asymmetric noise. And our methods outperform the other methods in 20% and 80% symmetric noise.

5.4.3 Sensitivity of our methods with Window size and Frequency

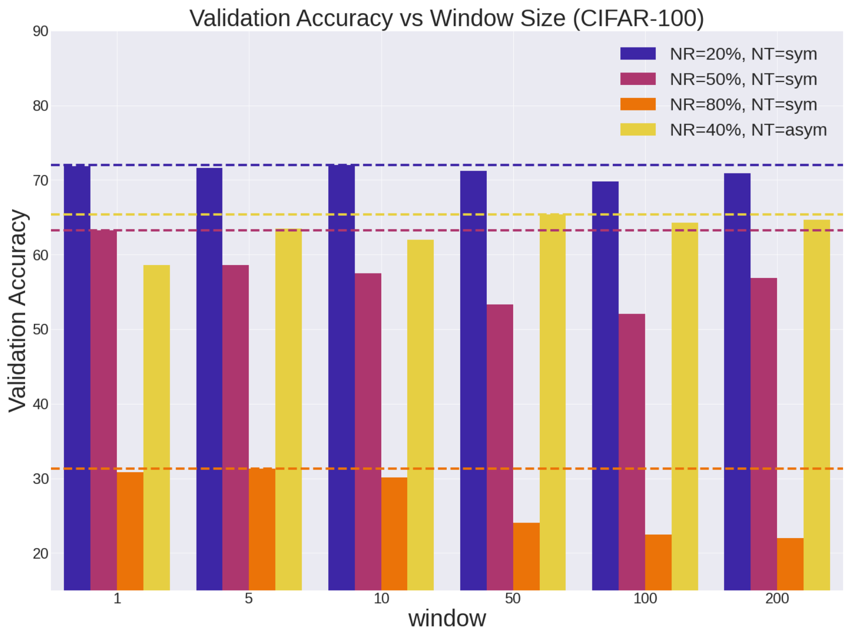
In our experiments, size window W is set to one of $\{1, 5, 10, 50, 100, 200\}$ and the value of Frequency F is set to one of $\{50, 100, 150\}$. Fig.5.3 and 5.4 show the sensitivity of SSFP+co to W and F by visualizing validation accuracy in various noise settings. As shown in Fig.5.3, SSFP+co is less sensitive to the size of window when the noise rate is low, while is highly sensitive when the noise rate is high, such as 50% and 80% symmetric label noise on the both datasets. The value of F in each noise setting in in Fig.5.3 is fixed to the setup in Table.5.2 and 5.3. When the label noise is symmetric 20%, validation accuracy remains unchanged for different values of W. In the case of 50% and 80% symmetric noise, smaller value of W achieves higher validation accuracy and higher W deteriorates the classification performance on the both datasets. In the case of 40% asymmetric label noise, lower value of W is better for CIFAR-10 and higher value of W is better for CIFAR-100. While our method is sensitive to W, less sensitive to the value of F as shown in Fig.5.4. The validation accuracy does not change by changing the value of F. In Fig.5.4, the value of W is also fixed as shown in Table.5.2 and 5.3 in each noise setting.

5.4.4 Visualizing the number of false predictions during training

Fig.5.5 and 5.6 show the distribution of total number of false prediction for each training sample at 40-th training epoch with various value of W. Fig.5.5 shows the results on CIFAR-10 dataset with 20%, 50%, 80% symmetric label noise and Fig.5.6 shows the the results on CIFAR-100 with 20%, 50%, 80% symmetric label noise. The number of false predictions is collected after 40-th epoch. The color of each bar represents whether sample is noisy or clean. As shown in each plot, histograms show that two clusters are formed,

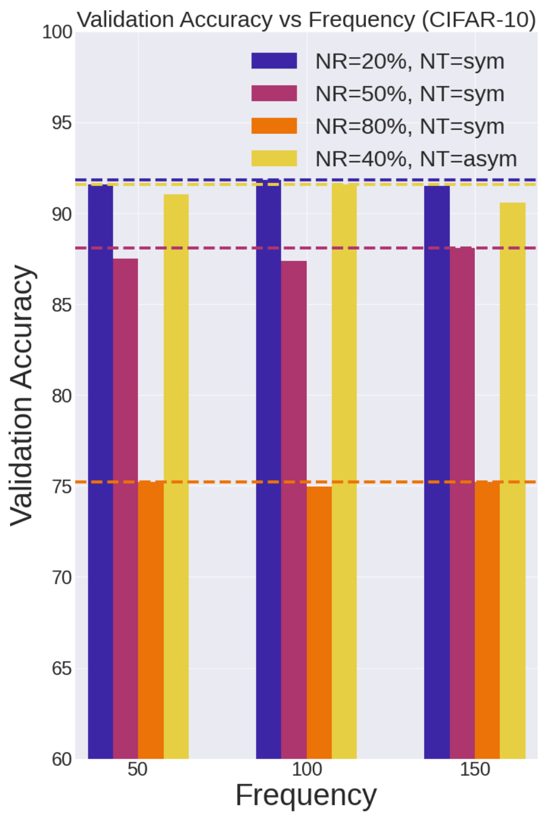


(a)

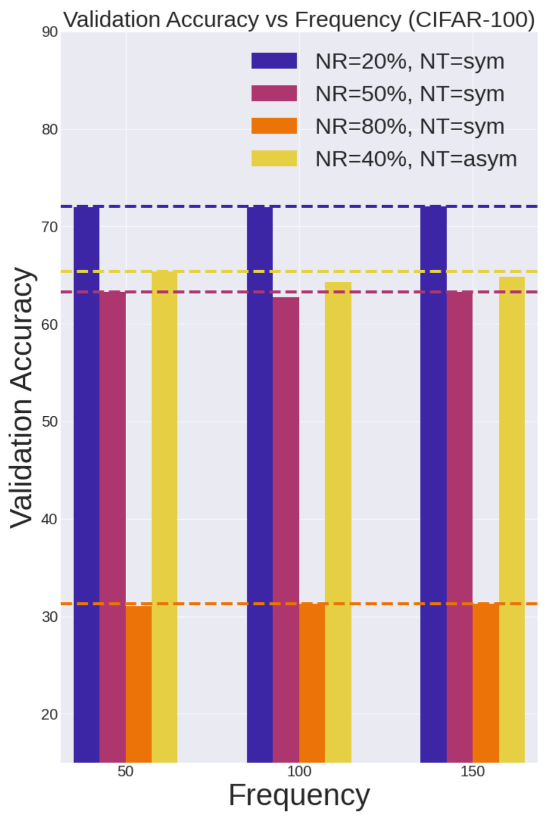


(b)

Figure 5.3: Validation accuracy of SSFP+co with different size of Window W. (a) and (b) show the results on CIFAR-10, and CIFAR-100, respectively. Each dashed line corresponds to the noise setting with same color and shows the best validation accuracy.

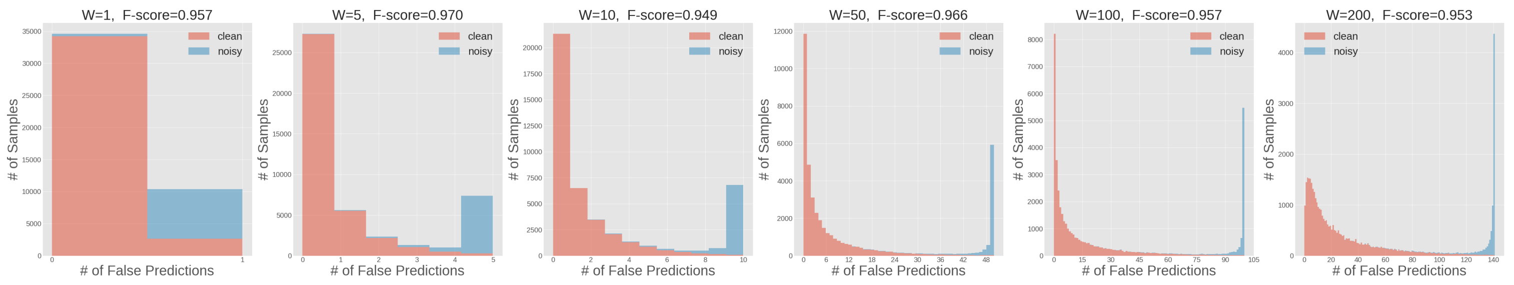


(a)

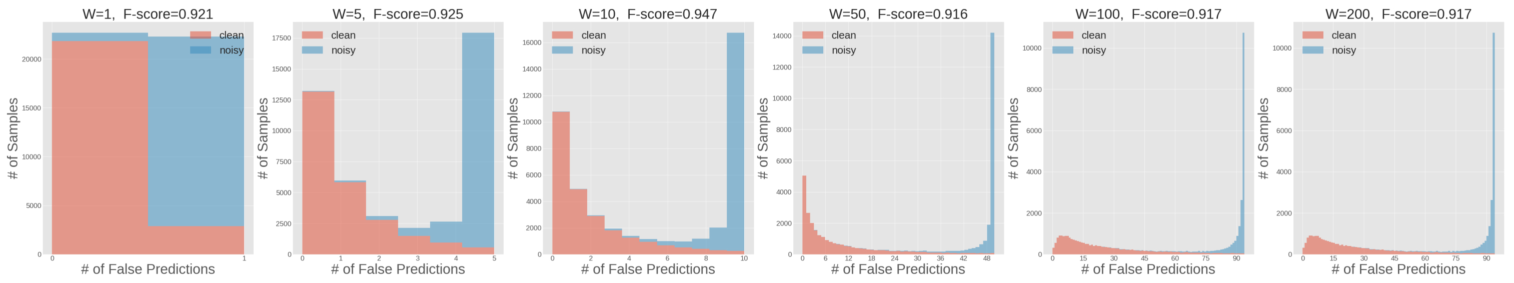


(b)

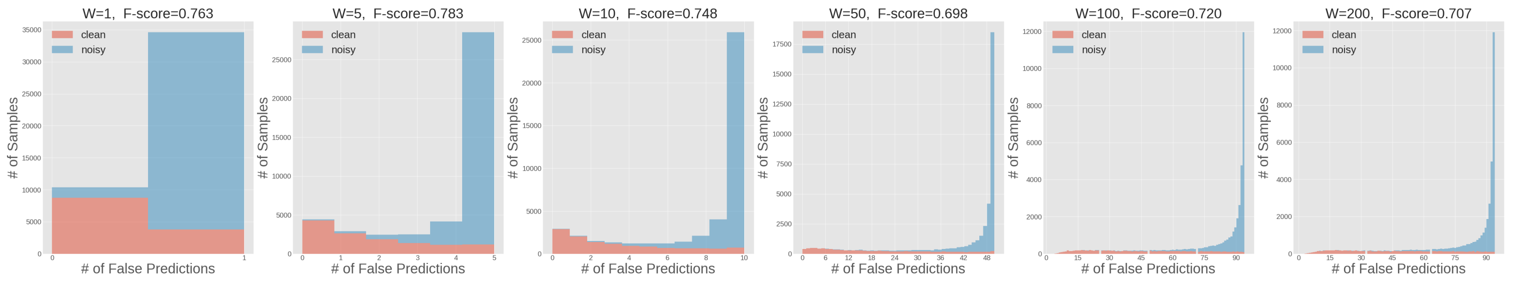
Figure 5.4: Validation accuracy of SSFP+co with different Frequency F. (a) and (b) show the results on CIFAR-10, and CIFAR-100, respectively. Each dashed line corresponds to the noise setting with same color and shows the best validation accuracy.



(a) CIFAR10, 20% symmetric noisy labels.

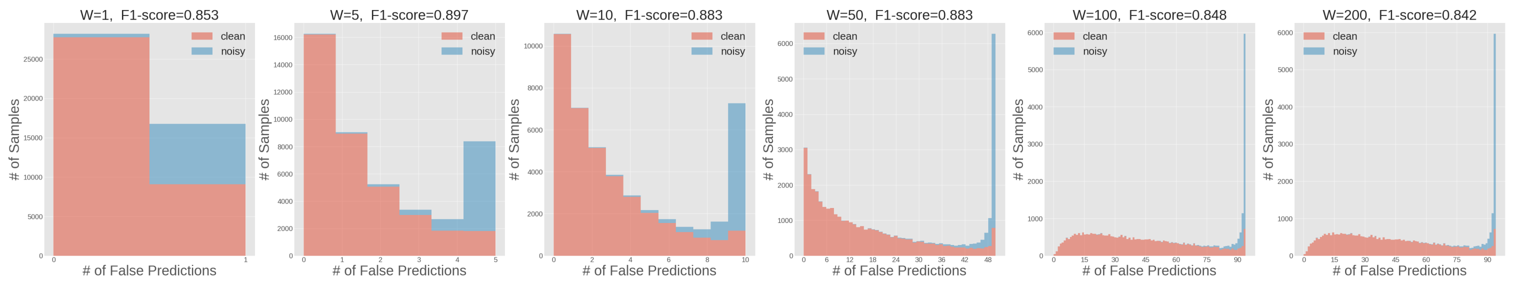


(b) CIFAR10, 50% symmetric noisy labels.

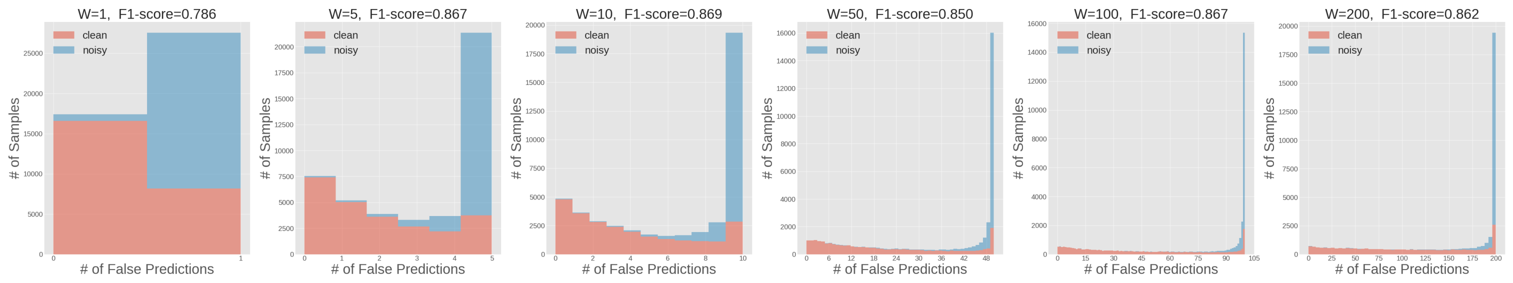


(c) CIFAR10, 80% symmetric noisy labels.

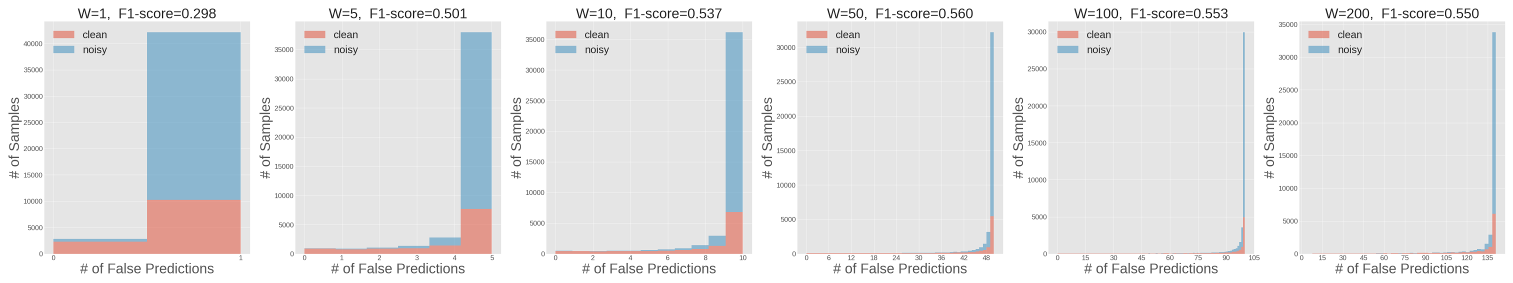
Figure 5.5: Visualization of the number of false predictions at 40-th training epoch with the various size of W. x-axis of each plots corresponds to the number of false predictions and y-axis corresponds to the number of training samples. On top of each plot, we show the value of W and F-score after fitting a two-components GMMs.



(a) CIFAR100, 20% symmetric noisy labels.



(b) CIFAR100, 50% symmetric noisy labels.



(c) CIFAR100, 80% symmetric noisy labels.

Figure 5.6: Visualization of the number of false predictions at 40-th training epoch with the various size of W. x-axis of each plots corresponds to the number of false predictions and y-axis corresponds to the number of training samples. On top of each plot, we show the value of W and F-score after fitting a two-components GMMs.

one mainly containing clean samples and the other mainly containing noisy samples while the value of W changes. And therefore we can suspect that the samples are intuitively classified into clean set and noisy set. However, after fitting a two-compoenent GMMs, we can see that F-score about binary classification for sample selection is different for each value of W. The F-score is the harmonic mean of the accuracy and recall, where accuracy is the fraction of clean samples out of all samples predicted to be clean, and recall is the fraction of clean samples correctly predicted out of all clean samples. On top of each plot in Fig.5.5 and 5.6, we show the value of W and F-score after fitting the GMMs. While each plot shows the two clusters, F-score after fitting the GMM is slightly different for each W. In these cases, training with W which yields highest F-score achieves highest validation accuracy as shown in Fig.5.3.

5.4.5 F-score of sample selection during training

Fig.5.7 shows how F-score of sample selection changes as training of SSFP+co progresses. Each plot shows different noise settings in the both datasets and each line plot represents the different size of W. In each plot, the bold line shows the W which achieves the highest validation accuracy at the end of training as shown in Fig.5.5 and 5.6. In CIFAR-10 with 20% symmetric label noise, $W = 100$ achieves the highest validation accuracy although the F-score during training is relatively low. However, in the higher noise rate setting such as 50% and 80% symmetric label noise, smaller size of window such as $W = 5$ which achieves the best validation accuracy and yields higher F-score during training. In CIFAR-100, some best W settings do not give higher F-score at the initial stage of training such as 50% symmetric noise, but F-score gradually increases during training, and W which achieves best validation accuracy also yields higher F-score at the end of training.

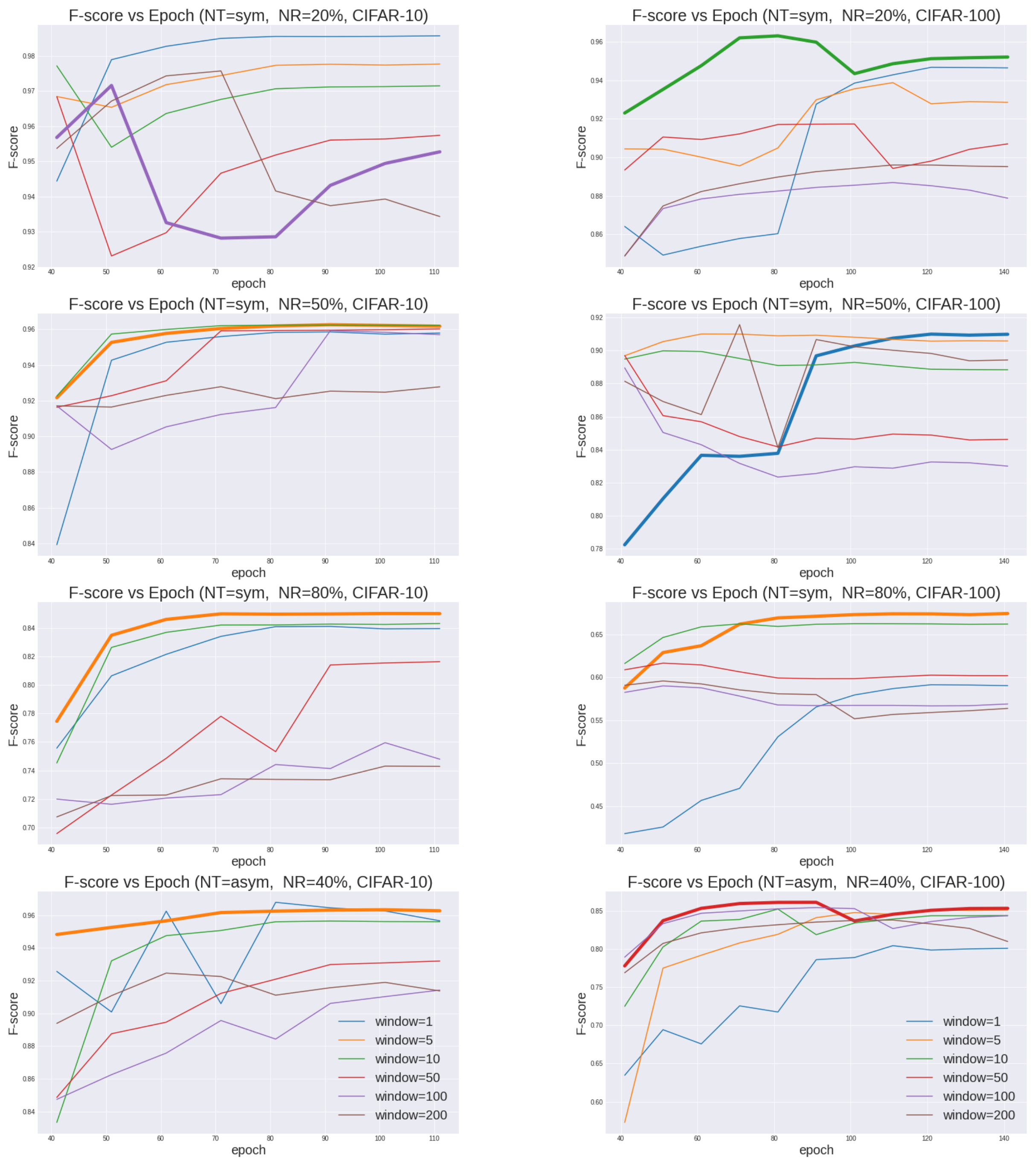


Figure 5.7: Visualization of how the F-score of clean sample selection changes during training SSFP+co with the various size of W. x-axis of each plots corresponds to the number of training epochs and y-axis corresponds F-score for sample selection. Plots on top row and plots on bottom row show the results in CIFAR-10 and CIFAR-100, respectively. On top of each plot, we show the label noise setting. Each plot corresponds to the size of Window W and bold line shows the W achieves the highest validation accuracy at the end of training.

5.4.6 Fitting a two-component Beta Mixture Model

In order to verify if there is a better distribution for sample selection, we modeled the number of false predictions by fitting a two-component Beta Mixture Model (BMM) instead of GMM. The beta distribution allows modeling the skewed distribution and we compared how the test accuracy varied with the BMM. Fig.5.8 shows the fitting results of BMM and GMM to the number of false predictions on CIFAR-10 dataset with 20% symmetric label noise at 40th epoch. The solid line shows the pdf of BMM and the dotted line shows the pdf of GMM. Both learned models capture well the distribution of the number of false predictions. Table.5.4 and 5.5 show the performance comparison of our method with GMM or BMM, denoted as *SSFP(GMM)* and *SSFP(BMM)*, on CIFAR-10 and CIFAR-100 dataset with different levels of symmetric and asymmetric label noise. The mean accuracy and its standard deviation are computed over three noise realizations. These results show that in CIFAR-10, the GMM-based method achieves better or comparable results to the BMM-based method; in CIFAR-100, the GMM-based method achieves superior results, except for the asymmetric label noise case. In summary, it can be said that GMM was superior to or comparable to BMM for sample selection in this ablation study.

5.4.7 Sensitivity of our methods with gamma

γ determines the threshold on posterior distribution of binary classification for selecting clean samples. We have tested SSFP on noisy CIFAR-10 and CIFAR-100 with various values of γ and recorded test accuracy at the end of training to check how SSFP is sensitive to γ. Table.5.6 summarizes the results. The value of γ was varied between 0.1 and 0.9, but severe performance degradation was not observed. These results show that our method is less sensitive to the value of γ.

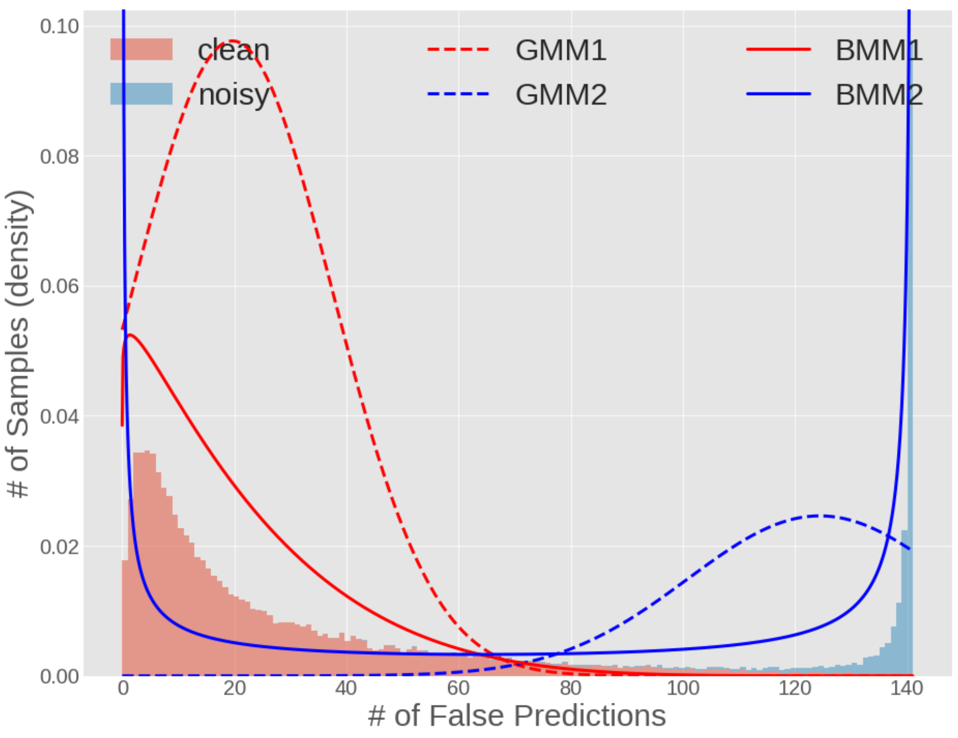


Figure 5.8: Visualization of total number of false predictions for each sample of CIFAR-10 dataset with 20% symmetric label noise at 40-th epoch, and fitted two-component GMM and BMM at 40-th. Red and Blue bars indicate the samples with clean labels and ones with noisy labels, respectively. The dotted line represents the pdf of GMM and the solid line represents the pdf of BMM.

5.4.8 Performance of SSFP on text data

To validate SSFP in text data, we evaluated SSFP in a sentence classification task using the AG News dataset. The labels of AG News dataset are “World”, “Sports”, “Business”, and “Sci/Tec”. We artificially synthesized 20% and 50% symmetric noisy labels. At first we built a vocabulary table, which identifies the index of each token in a sentence, from the training dataset. Each sentence is converted into a list of tokens, which are then converted into a list of integers.

Our classification model is composed of an embedding layer and 4 fully-connected layers. We prepare weight vectors for each token in the embedding layer. The embedding layer collects weight vectors identified by a list of integers and passes the mean vector of the collected vectors to the next fc layer.

In this experiment, the length of each weight vector is 128, and the number of weight vectors in the embedding layer is equal to the length of the vocabulary table. The number of training, validation, and test sets are 108000, 12000, and 7600, respectively. The model is trained for 30 epochs; the value of K is set to 1, the window size W is one of $\{1, 10, 100\}$, and the frequency F is one of $\{50, 100, 150\}$. SSFP starts sample selection after the 10th epoch. For 20% label noise, $W = 1$ and $F = 100$, and for 50% label noise, $W = 10$ and $F = 50$. Table.5.7 summarizes the performance of SSFP and the standard model without sample selection. The experiments show that SSFT is an effective method for learning with noisy labels in text classification.

5.4.9 Performance of SSFP with a small number of validation samples.

In the real world setting, it is not always possible to collect large amounts of clean data for validation. In this section, we reduced the number of validation samples from 5000 to 500 to see if the hyperparameters of SSFP could be adjusted with smaller number of validation samples. The tuned values of W and F on large validation set are shown in Table.5.2 and 5.3, and the tuned values of W and F on small validation set are shown Table.5.8 and 5.9. Table.5.10 shows that SSFP(small) achieves comparable test accuracy to SSFP(large). We found that tuning of hyperparameters of SSFP in CIFAR-10 and CIFAR-100 is possible with a small validation set.

5.4.10 Discussion

The results given by our methods were better than or comparative to the state-of-the-art methods in the various experimental settings. This means that proposed false predictions is one of the useful measure to detect the noisy labels. Despite the achievements of our methods, they are sensitive to hyper-parameter tuning based on validation accuracy, especially for W. As shown in Fig.5.3, when the noise rate is low such as 20% symmetric label noise in CIFAR-10, validation accuracy does not change across various value of W. However, in the higher noise setting such as 50% and 80% symmetric label noise setting, validation accuracy is high when the size of W is small, and decreases as the size of W increases. This indicates that records from recent models are more important than those form past models in severe noise setting. On the other hand, larger size of W is more effective in the case of 40% asymmetric label noise in CIFAR-100 as shown in Fig.5.3.

Fig.5.7 showed how the W changes the F-score for sample selection during training, and the relationship between F-score and the best validation accuracy. In the case of low noise rate setting such as 20% symmetric labels noise in CIFAR-10, the window size that does not results in highest F-score yields the best validation accuracy. In this case, selected samples with low F-score was accurate enough since noise rate is low and the number of selected samples was sufficient for training. In other cases such as 50% and 80% symmetric label noise, the W which achieves higher F-score also achieves the highest validation accuracy. This means that higher F-score during training is required to achieve the higher validation accuracy in the severe noise settings. Furthermore, almost plots show that each F-score gets better as learning progresses. This is because that false predictions given by the model trained with selected samples accurately identify the samples with noisy labels and iteratively improve the quality of sample selection during training.

One drawback of this method is that it is computationally more expensive than other conventional methods because it records the false predictions of N training samples at every F mini-batches. This method requires more time to complete training for a larger total number of training samples or for a smaller value of F.

Table 5.1: Comparison with baseline methods and current state-of-the-art approaches on CIFAR-10 and CIFAR-100 with symmetric and asymmetric label noise in test accuracy (%). FINE and F-coteaching [24] were re-implemented using their public code and the other values are taken from Kim et al. [24]. The mean accuracy and its standard deviation are computed over three trials.

Dataset	CIFAR-10				CIFAR-100			
Noise Type	Sym			Asym	Sym			Asym
Noise Ratio	20%	50%	80%	40%	20%	50%	80%	40%
Standard	87.0 ± 0.1	78.2 ± 0.8	53.8 ± 1.0	85.0 ± 0.0	58.7 ± 0.3	42.5 ± 0.3	18.1 ± 0.8	42.7 ± 0.6
Bootstrap	86.2 ± 0.2	–	54.1 ± 1.3	81.2 ± 1.5	58.3 ± 0.2	–	21.6 ± 1.0	45.1 ± 0.6
Forward	88.0 ± 0.4	–	54.6 ± 0.4	83.6 ± 0.6	39.2 ± 2.6	–	9.0 ± 0.6	34.4 ± 1.9
Co-teaching	89.3 ± 0.3	83.3 ± 0.6	66.3 ± 1.5	88.4 ± 2.8	63.4 ± 0.0	49.1 ± 0.4	20.5 ± 1.3	47.7 ± 1.2
Co-teaching+	89.1 ± 0.5	84.9 ± 0.4	63.8 ± 2.3	86.5 ± 1.2	59.2 ± 0.4	47.1 ± 0.3	20.2 ± 0.9	44.7 ± 0.6
TopoFilter	90.4 ± 0.2	86.8 ± 0.3	46.8 ± 1.0	87.5 ± 0.4	66.9 ± 0.4	53.4 ± 1.8	18.3 ± 1.7	56.6 ± 0.5
CRUST	89.4 ± 0.2	87.0 ± 0.1	64.8 ± 1.5	82.4 ± 0.0	69.3 ± 0.2	$\mathbf{62.3 \pm 0.2}$	21.7 ± 0.7	56.1 ± 0.5
FINE	90.7 ± 0.1	86.3 ± 0.4	69.6 ± 0.1	89.3 ± 0.4	68.7 ± 0.8	60.6 ± 0.5	22.8 ± 1.2	60.1 ± 1.4
F-coteaching	91.3 ± 0.1	$\mathbf{87.5 \pm 0.1}$	71.5 ± 0.8	90.5 ± 0.4	70.0 ± 0.3	61.7 ± 0.1	22.6 ± 0.7	$\mathbf{65.0 \pm 1.0}$
SSFP	$\mathbf{91.5 \pm 0.0}$	87.3 ± 0.2	72.5 ± 1.2	$\mathbf{91.1 \pm 0.3}$	71.4 ± 0.2	60.3 ± 0.6	29.6 ± 2.0	62.7 ± 0.6
SSFP+co	91.3 ± 0.0	$\mathbf{87.7 \pm 0.4}$	$\mathbf{74.1 \pm 1.0}$	90.9 ± 0.5	$\mathbf{72.0 \pm 0.2}$	$\mathbf{62.6 \pm 0.3}$	$\mathbf{30.0 \pm 1.5}$	64.7 ± 0.7

Table 5.2: The value of hyperparameters W and F used in Table.5.1 on CIFAR-10 dataset at different levels of noise rate.

Noise Type		Symmetric			Asymmetric
Noise Rate		20%	50%	80%	40%
SSFP	W	200	10	10	5
	F	100	150	100	100
SSFP+co	W	100	5	5	5
	F	100	150	150	100

Table 5.3: The value of hyperparameters W and F used in Table.5.1 on CIFAR-100 dataset at different levels of noise rate.

Noise Type		Symmetric			Asymmetric
Noise Rate		20%	50%	80%	40%
SSFP	W	10	5	5	200
	F	100	50	100	50
SSFP+co	W	10	1	5	50
	F	150	50	100	50

Table 5.4: Performance comparison of our method with GMM or BMM on CIFAR-10 dataset at different levels of noise rate in test accuracy (%).

Noise Type	Sym.			Asym.
Noise Rate	20%	50%	80%	40%
SSFP(GMM)	$\mathbf{91.5 \pm 0.0}$	$\mathbf{87.3 \pm 0.2}$	$\mathbf{72.5 \pm 1.2}$	$\mathbf{91.1 \pm 0.3}$
SSFP(BMM)	$\mathbf{91.5 \pm 0.0}$	$\mathbf{87.3 \pm 0.2}$	70.5 ± 1.4	88.9 ± 3.2

Table 5.5: Performance comparison of our method with GMM or BMM on CIFAR-100 dataset at different levels of noise rate in test accuracy (%).

Noise Type	Sym.			Asym.
Noise Rate	20%	50%	80%	40%
SSFP(GMM)	$\mathbf{71.4 \pm 0.2}$	$\mathbf{60.3 \pm 0.6}$	$\mathbf{29.6 \pm 2.0}$	62.7 ± 0.6
SSFP(BMM)	70.0 ± 0.0	58.1 ± 0.5	28.8 ± 1.5	$\mathbf{63.5 \pm 0.7}$

Table 5.6: Test accuracy of SSFP with different value of γ.

Dataset	CIFAR-10				CIFAR-100			
Noise Type	Sym			Asym	Sym			Asym
Noise Ratio	20%	50%	80%	40%	20%	50%	80%	40%
$\gamma = 0.1$	91.62	87.57	**71.41**	**91.30**	70.19	57.75	**29.34**	60.85
$\gamma = 0.3$	**91.64**	87.57	**71.41**	**91.30**	70.89	57.48	**29.34**	62.71
$\gamma = 0.4$	**91.64**	87.57	**71.41**	**91.30**	70.89	**60.03**	**29.34**	63.37
$\gamma = 0.5$	91.63	87.57	**71.41**	**91.30**	70.89	**60.03**	**29.34**	63.15
$\gamma = 0.6$	91.59	87.57	**71.41**	91.13	70.89	**60.03**	**29.34**	63.54
$\gamma = 0.7$	91.55	**87.61**	**71.41**	91.13	70.89	**60.03**	**29.34**	63.41
$\gamma = 0.9$	91.50	**87.61**	**71.41**	91.13	**71.60**	60.29	**29.34**	**64.62**

Table 5.7: Performance comparison of SSFP and standard method on AG News dataset at different levels of noise rate in test accuracy (%). The mean accuracy and its standard deviation are computed over three noise realizations.

Noise Type	Sym.		
Noise Rate	0%	20%	50%
Standard	**88.9 ± 0.2**	86.8 ± 0.4	81.2 ± 0.4
SSFP	–	**88.9 ± 0.1**	**87.1 ± 0.3**

Table 5.8: The value of hyperparameters W and F used in Table.5.10 on CIFAR-10 dataset at different levels of noise rate.

Noise Type		Symmetric			Asymmetric
Noise Rate		20%	50%	80%	40%
SSFP	W	1	5	5	1
	F	50	50	150	50

Table 5.9: The value of hyperparameters W and F used in Table.5.10 on CIFAR-100 dataset at different levels of noise rate.

Noise Type		Symmetric			Asymmetric
Noise Rate		20%	50%	80%	40%
SSFP	W	10	5	1	100
	F	50	150	100	100

Table 5.10: Comparison of SSFP with hyperparameters adjusted on a large validation set (SSFP(large)) vs. SSFP with hyperparameters adjusted on a small validation set (SSFP(small)) in test accuracy (%).

Dataset	CIFAR-10				CIFAR-100			
Noise Type	Sym			Asym	Sym			Asym
Noise Ratio	20%	50%	80%	40%	20%	50%	80%	40%
SSFP(large)	91.63	87.57	71.41	**91.30**	70.89	**61.73**	29.34	63.15
SSFP(small)	**91.72**	**87.66**	**73.20**	91.22	**71.70**	59.79	**29.60**	**63.37**

Chapter 6

Conclusion

6.1 Summary

In this study, we proposed three novel methods for robustly training deep neural networks for image classification against the noisy labels. Chapter 1 provides an overview of the development of deep neural networks in the field of machine learning, the problems caused by the noisy labels, and the importance of robust methods for addressing them. The problem of label noise is an inherent aspect of creating datasets for machine learning models, and addressing it is crucial for improving the performance of deep neural networks in various fields. We discussed the importance of the research results of this thesis as a foundation for guiding artificial intelligence to the foundations of society. Chapter 2 details the mechanism of deep neural networks, defines the type of label noise studied in this thesis, and presents a comprehensive review of previous research. In Chapter 3, we proposed a method to eliminate label noise by constructing a graph between samples based on similarity of sample features and using label propagation. By iteratively removing label noise and updating model parameters, we were able to robustly train the model while preventing overfitting to the noisy labels. In Chapter 4, we focused on the fact that sample selection methods in previous studies selected samples with small error values as clean samples, and introduced consistency regularization for those clean samples. As a result, we were able to obtain a model with high generalization performance while preventing overfitting to clean samples with simple patterns. In Chapter 5, we proposed a new sample selection method that exploits the robustness of deep neural networks against noisy labels in the early stages of learning and selects clean samples based on the number of false predictions for each sample during training. The new criterion reflects the difficulty of learning each sample. We showed that our approaches were effective in preventing deep neural networks from overfitting to noisy labels and improving generalization performance using common benchmark datasets.

6.2 Findings

The proposed method described in Chapter 3 attempted to eliminate noisy labels by exploiting the robust property of deep neural networks against noisy labels in the early stage of learning. By assuming that samples that truly belong to the same class are more similar in the features extracted from the model before overfitting to noisy labels, we constructed a similarity graph between samples. Our empirical results verified the effectiveness of the proposed method, as we observed a significant reduction in the percentage of noisy labels removed through label propagation on the similarity graph. This confirmed that the similarity of samples that truly belong to the same class is high in the feature space. Furthermore, the visualization of images in the feature space of the model trained with CIFAR-10 confirmed the clustering of samples with similar visual features, suggesting that our proposed method effectively performs label noise removal based on the visual similarity of samples in image classification tasks.

In Chapter 4, we attempted to prevent the model from overfitting to clean samples by perturbing the selected samples with image-specific data augmetation methods. By intentionally perturbing the image, we were able to promote the model's understanding of invariances and, simultaneously, enhance its ability to learn representations of clean examples that are difficult to classify. We observed an increase in the value of loss function on the selected clean sample when we applied consistency regularization to the selected samples, indicating that this method effectively prevents overfitting to the clean samples.

The sample method proposed in Chapter 5 used criteria reflecting the difficulty of each sample in learning, and learning with the selected samples achieved a superior classification accuracy compared to previous studies. In addition, performance was better when sample selection was performed by aggregating the recent number of false predictions, rather than using all the number of false predictions measured in the past. This is because the model trained with the most recent clean sample estimates the true label of each sample more accurately than the model immediately after the start of sample selection. This is because the model trained with the clean sample most recently estimates the true label of each sample more accurately than the model trained with the clean sample immediately after the start of sample selection.

6.3 Future Works

In Chapter 3, we constructed a graph based on the similarity in the feature space and performed label noise elimination by label propagation. However, it can be challenging to remove noisy labels from samples located near the classification boundary in the feature space because their ground-truth class is unclear. To construct more accurate graphs between samples, utilizing multiple types of graphs based on prior knowledge about the similarity between each sample, independent of label noise, can be considered. This prior knowledge refers to information about samples other than categories. One potential application of this proposed method to other fields is sample reduction, which identifies and removes redundant samples in the feature space for faster learning.

In Chapter 4, sample selection was performed, and perturbations were applied only to clean samples to achieve robust learning. As a direction for further research, perturbations could be applied to samples that were not selected as clean and monitor how generalization performance changes. While the proposed method employed random data augmentation as a method of perturbing samples, it would be worthwhile to investigate the effect of perturbations applied in the feature space and to determine which types of perturbations are most effective in improving generalization performance.

In Chapter 5, we proposed a sample selection method that observes the model's prediction results across multiple training epochs and considers the consistency of predictions over time. However, it is worth exploring other observations, beyond predictions, that are effective for sample selection. For example, we can consider how samples in the neighborhood change over time in the feature space and how the distance from samples in other classes changes. A possible application of this method for other fields is scheduling training samples for faster learning.

Each of these proposed methods presented in this study is independent and can be utilized in combination. The findings obtained in this research are not limited to the label noise problem, but have potential applications in other fields as well. Furthermore, the investigation of noisy labels addressed in this thesis is closely related to the generalization performance of deep neural networks. By clarifying the relationship between robustness to label noise and generalization performance, we can gain a deeper understanding of the fundamental principles of learning in artificial intelligence, potentially leading to new breakthroughs in the field.

Bibliography

[1] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. Mcguinness. Unsupervised label noise modeling and loss correction. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 312–321. PMLR, 2019. URL `https://proceedings.mlr.press/v97/arazo19a.html`.

[2] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242. PMLR, 2017. URL `https://proceedings.mlr.press/v70/arpit17a.html`.

[3] P. L. Bartlett, M. I. Jordan, and J. D. Mcauliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006. URL `http://www.jstor.org/stable/30047445`.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, volume 382, page 41–48. Association for Computing Machinery, 2009. doi: 10.1145/1553374.1553380. URL `https://doi.org/10.1145/1553374.1553380`.

[5] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Proceedings of the Advances in Neural Information Processing Systems 32:*

Annual Conference on Neural Information Processing Systems 2019, pages 5050–5060, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html`.

[6] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. doi: 10.1145/792538.792543. URL `https://doi.org/10.1145/792538.792543`.

[7] F. R. Cordeiro and G. Carneiro. A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In *Proceedings of the 33rd SIBGRAPI Conference on Graphics, Patterns and Images*, pages 9–16. IEEE, 2020. doi: 10.1109/SIBGRAPI51738.2020.00010. URL `https://doi.org/10.1109/SIBGRAPI51738.2020.00010`.

[8] V. Cothey. Web-crawling reliability. *Journal of the American Society for Information Science and Technology*, 55(14):1228–1238, 2004. doi: https://doi.org/10.1002/asi.20078. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20078`.

[9] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019. doi: 10.1109/CVPR.2019.00020.

[10] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf`.

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Con-*

ference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[12] T. Durand, N. Mehrasa, and G. Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 647–657. IEEE Computer Society, 2019. doi: 10.1109/CVPR.2019.00074. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00074.

[13] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Proceedings of the Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*, pages 522–530. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/hash/1651cf0d2f737d7adeab84d339dbabd3-Abstract.html.

[14] B. Frenay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014. doi: 10.1109/TNNLS.2013.2292894.

[15] A. Ghosh, H. Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1919–1925. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14759.

[16] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H12GRgcxg.

[17] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, volume 31, pages 8536–8546. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/a19744e268754fb0148b017647355b7b-Paper.pdf`.

[18] J. Han, P. Luo, and X. Wang. Deep self-learning from noisy labels. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5137–5146. IEEE Computer Society, 2019. doi: 10.1109/ICCV.2019.00524. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00524`.

[19] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597.

[20] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1031. URL `https://aclanthology.org/P18-1031`.

[21] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5065–5074. IEEE Computer Society, 2019. doi: 10.1109/CVPR.2019.00521. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00521`.

[22] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Pro-*

ceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 2304–2313. PMLR, 2018. URL `https://proceedings.mlr.press/v80/jiang18c.html`.

[23] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65:101759, 2020. doi: https://doi.org/10.1016/j.media.2020.101759. URL `https://www.sciencedirect.com/science/article/pii/S1361841520301237`.

[24] T. Kim, J. Ko, s. Cho, J. Choi, and S.-Y. Yun. Fine samples for learning with noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*, volume 34, pages 24137–24149. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/ca91c5464e73d3066825362c3093a45f-Paper.pdf`.

[25] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *Proceedings of the Computer Vision – ECCV 2016*, pages 301–320, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9.

[26] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, volume 25, pages 1106–1114. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[28] T. Kurita and A. Hidaka. *Statistical Pattern Recognition and Discriminant Analysis*. Corona Publishing co., LTD., 2019.

[29] F. Lateef and Y. Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019. doi: 10.1016/j.neucom.2019.02.003. URL `https://doi.org/10.1016/j.neucom.2019.02.003`.

[30] Y. LeCun, C. Farabet, C. Couprie, and L. Najman. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(08):1915–1929, 2013. doi: 10.1109/TPAMI.2012.231.

[31] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL `https://doi.org/10.1038/nature14539`.

[32] K. Lee, X. He, L. Zhang, and L. Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5447–5456. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00571. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00571`.

[33] J. Li, R. Socher, and S. C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *Proceedings of the 8th International Conference on Learning Representations*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=HJgExaVtwr`.

[34] W. Li, L. Wang, W. Li, E. Agustsson, and L. V. Gool. Webvision database: Visual learning and understanding from web data. *CoRR*, abs/1708.02862, 2017. URL `http://arxiv.org/abs/1708.02862`.

[35] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim. Fast autoaugment. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference*

on Neural Information Processing Systems 2019, volume 32, pages 6662–6672. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/6add07cf50424b14fdf649da87843d01-Paper.pdf`.

[36] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 20331–20342. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/ea89621bee7c88b2c5be6681c8ef4906-Paper.pdf`.

[37] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey. Dimensionality-driven learning with noisy labels. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3355–3364. PMLR, 2018. URL `https://proceedings.mlr.press/v80/ma18d.html`.

[38] N. Manwani and P. S. Sastry. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, 43(3):1146–1151, 2013. doi: 10.1109/TSMCB.2012.2223460.

[39] W. Mason and S. Suri. Conducting behavioral research on amazon's mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

[40] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký. Strategies for training large scale neural network language models. In *Proceedings of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 196–201, 2011. doi: 10.1109/ASRU.2011.6163930.

[41] B. Mirzasoleiman, K. Cao, and J. Leskovec. Coresets for robust training of deep neural networks against noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 11465–11477. Curran Asso-

ciates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/8493eeaccb772c0878f99d60a0bd2bb3-Paper.pdf`.

[42] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, volume 26, pages 1196–1204. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/3871bd64012152bfb53fdf04b401193f-Paper.pdf`.

[43] K. Nishi, Y. Ding, A. Rich, and T. Höllerer. Augmentation strategies for learning with noisy labels. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8018–8027, 2021. doi: 10.1109/CVPR46437.2021.00793.

[44] Y. Nomura and T. Kurita. Robust training of deep neural networks with noisy labels by graph label propagation. In *Proceedings of Frontiers of Computer Vision*, pages 281–293. Springer International Publishing, 2021. ISBN 978-3-030-81638-4.

[45] Y. Nomura and T. Kurita. Consistency regularization on clean samples for learning with noisy labels. *IEICE Transactions on Information and Systems*, E105.D(2):387–395, 2022. doi: 10.1587/transinf.2021EDP7127.

[46] Y. Nomura and T. Kurita. Sample selection approach with number of false predictions for learning with noisy labels. *IEICE Transactions on Information and Systems*, E105.D(10):1759–1768, 2022. doi: 10.1587/transinf.2022EDP7033.

[47] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. Mcnamara, A. Angert, E. Banner, V. Khetan, T. Mcdonnell, A. T. Nguyen, D. Xu, B. C. Wallace, M. Rijke, and M. Lease. Neural information retrieval: At the end of the early years. *Information Retrieval Journal*, 21(2–3):111–182, jun 2018. doi: 10.1007/s10791-017-9321-y. URL `https://doi.org/10.1007/s10791-017-9321-y`.

[48] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 257–266. ACM, 2017. doi: 10.1145/3132847.3132914. URL `https://doi.org/10.1145/3132847.3132914`.

[49] G. Paolacci, J. Chandler, and P. G. Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[50] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2233–2241, 2017. doi: 10.1109/CVPR.2017.240.

[51] H. Permuter, J. Francos, and I. Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006. doi: https://doi.org/10.1016/j.patcog.2005.10.028. URL `https://www.sciencedirect.com/science/article/pii/S0031320305004334`.

[52] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Comput.*, 29(9):2352–2449, 2017. doi: 10.1162/neco_a_00990. URL `https://doi.org/10.1162/neco_a_00990`.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[54] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6596`.

[55] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[56] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for LVCSR. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618. IEEE, 2013. doi: 10.1109/ICASSP.2013.6639347. URL `https://doi.org/10.1109/ICASSP.2013.6639347`.

[57] A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 959–962. Association for Computing Machinery, 2015. ISBN 9781450336215. doi: 10.1145/2766462.2767830. URL `https://doi.org/10.1145/2766462.2767830`.

[58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1409.1556`.

[59] H. Song, M. Kim, and J.-G. Lee. SELFIE: Refurbishing unclean samples for robust deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5907–5915. PMLR, 2019. URL `https://proceedings.mlr.press/v97/song19b.html`.

[60] H. Song, M. Kim, D. Park, and J. Lee. Learning from noisy labels with deep neural networks: A survey. *CoRR*, abs/2007.08199, 2020. URL `https://arxiv.org/abs/2007.08199`.

[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[62] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018. doi: 10.1109/CVPR.2018.00582.

[63] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li. YFCC100M: the new data in multimedia research. *Commun. ACM*, 59(2): 64–73, 2016. doi: 10.1145/2812802. URL `http://doi.acm.org/10.1145/2812802`.

[64] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 1799–1807, 2014. URL `https://proceedings.neurips.cc/paper/2014/hash/e744f91c29ec99f0e662c9177946c627-Abstract.html`.

[65] J. Wang, Y. Jiang, and S. Chang. Label diagnosis through self tuning forweb image search. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1390–1397. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206729. URL `https://doi.org/10.1109/CVPR.2009.5206729`.

[66] X. Wang, E. Kodirov, Y. Hua, and N. M. Robertson. Improving MAE against CCE under label noise. *CoRR*, abs/1903.12141, 2019. URL `http://arxiv.org/abs/1903.12141`.

[67] P. Wu, S. Zheng, M. Goswami, D. N. Metaxas, and C. Chen. A topological filter for learning with label noise. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/f4e3ce3e7b581ff32e40968298ba013d-Abstract.html`.

[68] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2015. doi: 10.1109/CVPR.2015.7298885.

[69] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy. Learning from multiple annotators with varying expertise. *Machine learning*, 95(3):291–327, 2014.

[70] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang. Deep learning from noisy image labels with quality embedding. *IEEE Transactions on Image Processing*, 28(4):1909–1922, 2019. doi: 10.1109/TIP.2018.2877939.

[71] K. Yi and J. Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7010–7018, 2019. doi: 10.1109/CVPR.2019.00718.

[72] X. Yu, T. Liu, M. Gong, and D. Tao. Learning with biased complementary labels. In *Proceedings of the Computer Vision - ECCV 2018 - 15th European Conference*, volume 11205 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2018. doi: 10.1007/978-3-030-01246-5_5. URL `https://doi.org/10.1007/978-3-030-01246-5_5`.

[73] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama. How does disagreement help generalization against label corruption? In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7164–7173. PMLR, 2019. URL `https://proceedings.mlr.press/v97/yu19b.html`.

[74] S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, and S. Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *Proceedings of the*

2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2340–2350, 2021. doi: 10.1109/CVPR46437.2021.00237.

[75] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

[76] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the 6th International Conference on Learning Representations*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.

[77] W. Zhang, T. Du, and J. Wang. Deep learning over multi-field categorical data - - A case study on user response prediction. In *Proceedings of the Advances in Information Retrieval - 38th European Conference on IR Research*, volume 9626, pages 45–57. Springer, 2016. doi: 10.1007/978-3-319-30671-1_4. URL `https://doi.org/10.1007/978-3-319-30671-1_4`.

[78] Z. Zhang and M. R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 8792–8802, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html`.

[79] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 321–328. MIT Press, 2003. URL `https://proceedings.neurips.cc/paper/2003/hash/87682805257e619d49b8e0dfdc14affa-Abstract.html`.