NLNL: Negative Learning for Noisy Labels

Youngdong Kim Junho Yim Juseung Yun Junmo Kim
School of Electrical Engineering, KAIST, South Korea
{ydkim1293, junho.yim, st24hour, junmo.kim}@kaist.ac.kr

Abstract

Convolutional Neural Networks (CNNs) provide excellent performance when used for image classification. The classical method of training CNNs is by labeling images in a supervised manner as in "input image belongs to this label" (Positive Learning; PL), which is a fast and accurate method if the labels are assigned correctly to all images. However, if inaccurate labels, or noisy labels, exist, training with PL will provide wrong information, thus severely degrading performance. To address this issue, we start with an indirect learning method called Negative Learning (NL), in which the CNNs are trained using a complementary label as in "input image does not belong to this complementary label." Because the chances of selecting a true label as a complementary label are low, NL decreases the risk of providing incorrect information. Furthermore, to improve convergence, we extend our method by adopting PL selectively, termed as Selective Negative Learning and Positive Learning (SelNLPL). PL is used selectively to train upon expected-to-be-clean data, whose choices become possible as NL progresses, thus resulting in superior performance of filtering out noisy data. With simple semi-supervised training technique, our method achieves state-of-the-art accuracy for noisy data classification, proving the superiority of SelNLPL's noisy data filtering ability.

1. Introduction

Convolutional Neural Networks (CNNs) have improved the performance of image classification significantly [17, 8, 29, 11, 7, 38]. For this supervised task, huge dataset composed of images and their corresponding labels is required for training CNNs. CNNs are powerful tools for classifying images if the corresponding labels are correct. However, accurately labeling a large number of images is daunting and time-consuming, occasionally yielding mismatched labeling. When the CNNs are trained with noisy data, it can overfit to such a dataset, resulting in poor classification performance. Therefore, training CNNs properly with noisy data is of great practical importance. Many approaches address this problem by applying a number of techniques and regularization terms along with *Positive Learning* (PL), a typical supervised learning method for training CNNs that "input image belongs to this label" [6, 2, 34, 20, 3, 39, 26, 30, 22, 33, 21]. However, when the CNN is trained with images and mismatched labels, wrong information is being provided to the CNN.

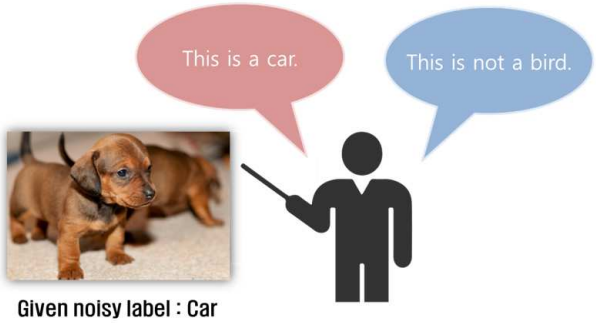


Figure 1: Conceptual comparison between *Positive Learning* (PL) and *Negative Learning* (NL). Regarding noisy data, while PL provides CNN the wrong information (red balloon), with a higher chance, NL can provide CNN the correct information (blue balloon) because a dog is clearly not a bird.

To overcome this issue, we suggest *Negative Learning* (NL), an indirect learning method for training CNN that "input image does not belong to this complementary label." NL does not provide wrong information as frequently as PL (Figure 1). For example, when training CNN with noisy CIFAR10 using PL, if the CNN receives an image of a dog and the label "car", the CNN will be trained to acknowledge that this image is a car. In this case, the CNN is trained with wrong information. However, with NL, the CNN will be randomly provided with a complementary label other than "car," for example, "bird." Training CNN to acknowledge that this image is not a bird is in some way an act of providing CNN the right information because a dog is clearly not a bird. In this manner, noisy data can contribute to training CNN by providing the "right" information with a high chance of not selecting a true label as a complementary label, whereas zero chance is provided in

PL. Our study demonstrates the effectiveness of NL as it prevents CNN from overfitting to noisy data.

Furthermore, utilizing NL training method, we propose *Selective Negative Learning and Positive Learning* (SelNLPL), which combines PL and NL to take full advantage of both methods for better training with noisy data. Although PL is unsuitable for noisy data, it is still a fast and accurate method for clean data. Therefore, after training CNN with NL, PL begins to train CNN selectively using only training data of high classification confidence. Through this process, SelNLPL widens the gap between the confidences of clean data and noisy data, resulting in excellent performance for filtering noisy data from training data.

Subsequently, by discarding labels of filtered noisy data and treating them as unlabeled data, we utilize semi-supervised learning for noisy data classification. Based on the superior filtering ability of SelNLPL, we demonstrate that state-of-the-art performance on noisy data classification can be achieved with a simple semi-supervised learning method. Although this is not the first time that noisy data classification has been addressed by filtering noisy data [2, 6, 24], the filtering results have not been promising owing to the use of PL for noisy data.

The main contributions of this paper are as follows:

- We apply the concept of *Negative Learning* to the problem of noisy data classification. We prove its applicability by demonstrating that it prevents the CNN from overfitting to noisy data.
- Utilizing the proposed NL, we introduce a new framework, called SelNLPL, for filtering out noisy data from training data. Following NL, by selectively applying PL only to training data of high confidence, we can achieve accurate filtering of noisy data.
- We achieved state-of-the-art noisy data classification results with relatively simple semi-supervised learning based on the superior noisy data filtering achieved by SelNLPL.
- Our method does not require any prior knowledge of the type or number of noisy data points. It does not require any tuning of hyper-parameters that depend on prior knowledge, making our method applicable in real life.

The remainder of this paper is organized as follows: Section 3 describes the overall process of our method with detailed explanations of each step. Section 4 demonstrates the superior filtering ability of SelNLPL. Section 5 describes the experiments for evaluating our method, and Section 6 describes the experiments to further analyze our method. Finally, we conclude the paper in Section 7.

2. Related works

Noisy label learning Recently, numerous methods have been proposed for learning with noisy labels. Herein, we briefly review the relevant studies.

Some methods attempted to create noise-robust losses [1, 32, 23, 4, 3, 39]. Ghosh *et al*. [4, 3] demonstrated theoretically that the mean absolute error (MAE) is robust to noisy labels; however, the MAE can degrade the accuracy when adopted in neural networks. Zhang *et al*. [39] proposed a generalized cross entropy loss that not only shows robustness to label noise but also performs well on deep neural networks.

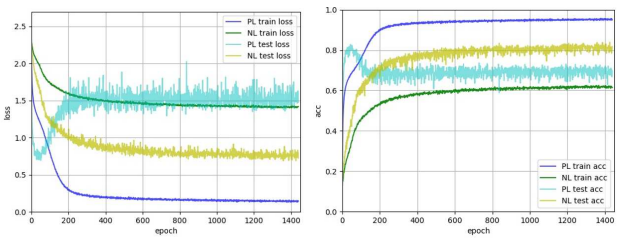
In other studies, each training sample is re-weighted differently depending on the reliability of the given label [13, 27, 20]. [13, 27] used a meta-learning algorithm that learns the optimal weights for each sample. However, both these methods require a certain amount of clean data, which is difficult to obtain in many cases. CleanNet [20] is also limited because it requires a label that is verified as correct.

Some approaches use correction methods. Loss correction [25, 31, 36, 10] methods assume that the noise transition matrix is already known or that some clean data are obtainable to calculate the noise transition matrix. [28, 14, 5] modeled the noise transition matrix by adding an additional layer. Several other approaches attempted to correct the label directly [33, 21]. However, in these methods, clean data is required to train the label cleaning network and teacher network. Additional label-cleaning methods exist, that gradually change the data label to the prediction value of the network [26, 30, 22].

Other approaches include jointly modeling labels and worker quality [15], creating a robust method to learn in open-set noisy label situations [34] and attempting to prune the correct samples [6, 2, 24]. Ding *et al*. [2] suggested pruning the correct samples based on softmax outputs. Samples deemed unreliable are trained in a semi-supervised manner rather than by using label information.

Our method utilizes both pruning of the correct sample and the label correction method. The existing pruning and label cleaning methods [6, 2, 26, 30, 22] use a network trained directly with a given noisy label; thus overfitting to a noisy label can occur even if the pruning or cleaning process is performed. Meanwhile, we use NL method, which indirectly uses noisy labels, thereby avoiding the problem of memorizing the noisy label and exhibiting remarkable performance in filtering only noisy samples.

Using complementary labels This is not the first time that complementary labels have been used. Previous studies [12, 37] focused on a classification task in which complementary labels are given. However, unlike the complementary label classification task, we generate complementary labels from given noisy labels and use them for NL.



(a) (b)

Figure 2: Comparison between PL and NL. (a): Loss graph of PL and NL. (b): Accuracy graph of PL and NL.

3. Method

This section describes our overall method for noisy data classification. Section 3.1 describes the concept and implementation of NL, demonstrating that it is more suitable for training with noisy data compared to PL. Section 3.2, and 3.3 respectively introduce *Selective Negative Learning* (SelNL) and *Selective Positive Learning* (SelPL), which are the subsequent steps after NL to further make CNN train better with noisy training data and simultaneously prevent overfitting. The combination of all these methods is called *Selective Negative Learning and Positive Learning* (SelNLPL), which demonstrates excellent performance for filtering noisy data from training data (Section 3.4). Finally, semi-supervised learning is performed for noisy data classification, utilizing the filtering ability of SelNLPL (Section 3.5).

3.1. Negative Learning

As mentioned in Section 1, typical method of training CNNs for image classification with given image data and the corresponding labels is PL. It is a method of training CNNs "input image belongs to this label." In contrast, with NL, the CNNs are trained that "input image does not belong to this complementary label."

Algorithm 1 Complementary label generation

Input: Training label $y \in \mathcal{Y}$

while iteration **do**

$\overline{y}$ = Randomly select from $\{1, ..., C\} \setminus \{y\}$

Output: Complementary label $\overline{y}$

We consider the problem of c-class classification. Let $\boldsymbol{x} \in \mathcal{X}$ be an input, $y, \overline{y} \in \mathcal{Y} = \{1, ..., c\}$ be its label and complementary label, respectively, and $\boldsymbol{y}, \overline{\boldsymbol{y}} \in \{0, 1\}^c$ be their one-hot vector. Suppose the CNN $f(x; \theta)$ maps the input space to the c-dimensional score space $f : \mathcal{X} \to \mathbb{R}^c$, where θ is the set of network parameters. If f passes through the softmax function, the output can be interpreted as a probability $\boldsymbol{p} \in \Delta^{c-1}$, where Δ^{c-1} denotes the c-dimensional simplex. When training with PL, the cross entropy loss function of the network f becomes:

$$\mathcal{L}(f, y) = -\sum_{k=1}^{c} \boldsymbol{y}_k \log \boldsymbol{p}_k \tag{1}$$

where $\boldsymbol{p}_k$ denotes the k^{th} element of $\boldsymbol{p}$. Eq. 1 is suitable for optimizing the probability value corresponding to the given label as 1 ($\boldsymbol{p}_y \to 1$), satisfying the purpose of PL. However, NL differs from PL as it optimizes the output probability corresponding to the complementary label to be far from 1 (to reach 0 in the end ($\boldsymbol{p}_{\overline{y}} \to 0$)). Therefore, we propose a loss function as follows:

$$\mathcal{L}(f, \overline{y}) = -\sum_{k=1}^{c} \overline{\boldsymbol{y}}_k \log(1 - \boldsymbol{p}_k) \tag{2}$$

This complementary label is completely random in that it is selected randomly from the labels of all classes except for the given label y for every iteration during training (Algorithm 1). Eq. 2 enables the probability value of the complementary label to be optimized as zero, resulting in an increase in the probability values of other classes, meeting the purpose of NL.

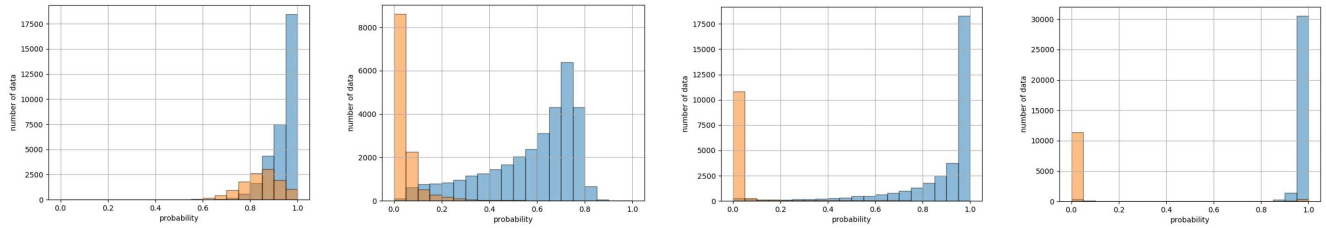
A distinct comparison between PL and NL is shown in Figure 2. The CNN was trained with either PL or NL on CIFAR10 corrupted with 30% *symm-inc* noise. The types of noise used in our paper are explained in Section 5. Note that, while the CNN is trained with either PL (Eq. 1) or NL (Eq. 2), all losses shown in the graph (Figure 2 (a)) are computed using Eq. 1. With PL, the test loss drops and the test accuracy increases in the early stage. However, it eventually causes the CNN to overfit to noisy training data, resulting in poor performance on the clean test data. In contrast, NL is shown to train the CNN without overfitting to noisy training data, as gradual decrease in test loss and an increase in test accuracy are observed. Figure 3 (a) and (b) respectively show the histogram of the training data after PL and NL. While the confidence of both clean and noisy data increased with PL, the confidence of noisy data is much lower than that of clean data with NL, again indicating the capability of NL to prevent the CNN from overfitting to noisy data.

3.2. Selective NL

As mentioned in Section 3.1, NL can prevent the CNN from overfitting to noisy data, as shown by its low confidence values (Figure 3 (b)). As the next step, we introduce SelNL to improve convergence after NL. After training with NL, SelNL trains the CNN only with the data having confidence over $\frac{1}{c}$. After thresholding, the data involved in training tends to be less noisy than before, thus improving the convergence of the CNN efficiently. Figure 3 (c) shows the result of SelNL after NL.

3.3. Selective PL

NL can be a better learning method when noisy data is involved. However, if the training data is verified to have



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

Figure 3: Histogram showing the distribution of CIFAR10 training data with 30% *symm-inc* noise, according to probability $\boldsymbol{p}_y$ (confidence). Blue indicates clean data, whereas orange indicates noisy data. (a): PL. (b): NL. (c): NL→SelNL. (d): NL→SelNL→SelPL (SelNLPL).

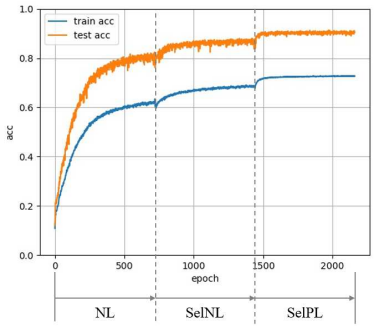


Figure 4: Accuracy graph of SelNLPL. Training is performed sequentially with NL, SelNL, and SelPL.

Algorithm 2 Overall process of SelNLPL

Input: Training data $(\boldsymbol{x}, y) \in (\mathcal{X}, \mathcal{Y})$, network $f(\boldsymbol{x}; \boldsymbol{\theta})$, total epoch T

for $i \leftarrow 1$ to T **do** ▷ NL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$
  Update f by minimizing Eq. 2
for $i \leftarrow 1$ to T **do** ▷ SelNL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > 1/\mathrm{c}$
  Update f by minimizing Eq. 2
for $i \leftarrow 1$ to T **do** ▷ SelPL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > \gamma$
  Update f by minimizing Eq. 1

Output: Network $f(\boldsymbol{x}; \boldsymbol{\theta})$

clean labels, PL is a faster and more accurate method than NL. After training with NL and SelNL, the confidences of clean and noisy data are separated by a large margin (Figure 3 (c)). SelPL trains CNN only with data having confidence over γ, assuming that such data is clean data. In this study, we set γ to 0.5. Figure 3 (d) shows the result of SelPL after Figure 3 (c), exhibiting high confidence value near 1 for almost all clean data.

3.4. Selective NL and PL

To summarize, the combination of NL, SelNL, and SelPL is called SelNLPL. The overall process of SelNLPL is shown in Algorithm 2. Figure 4 displays the performance change for each step. It clearly indicates the enhancement in performance when each step is applied, thus demonstrating the significance of each step in SelNLPL. It is proven in Figure 4 that each step of SelNLPL contributes to convergence, while simultaneously preventing overfitting to noisy data, resulting in a higher test accuracy than training accuracy throughout the training process.

As shown in Figure 3 (d), the overall confidence of clean data and noisy data is separated with a large margin. This implies that SelNLPL can be used for filtering noisy data from training data. This area is further analyzed in Section 4.

3.5. Semi-supervised learning

With the filtering ability of SelNLPL, the semi-supervised learning method can be applied to clean data and filtered noisy data, discarding the labels of filtered noisy data. For semi-supervised learning, we apply the pseudo labeling method [19]. Figure 5 shows the overall process of pseudo labeling. Firstly, the training data is divided into clean data and noisy data by using the CNN that is trained with SelNLPL (Figure 5 (a)). Next, in Figure 5 (b), the initialized CNN is trained with clean data obtained from SelNLPL. Then, the noisy data's label is updated with the output of the CNN in Figure 5 (b). Here, we used the soft label as the updated label, similar to [30]. The typical label for image classification is in the form of a one-hot vector, whereas the soft label is simply the output of trained CNN. It was shown that soft labels are better when updating labels [30]. Finally, clean data and label-updated noisy data are used to train the initialized CNN (Figure 5 (c)). This resulted in state-of-the-accuracy, proving the high filtering ability of SelNLPL. The results are shown in Section 5.

4. Filtering ability

It is mentioned in Section 3.4 that SelNLPL is effective for filtering noisy data from training data. In this section, we explain the filtering process of SelNLPL further.

When training CNN with SelNLPL, data with confidence exceeding γ are assumed to be clean. Following

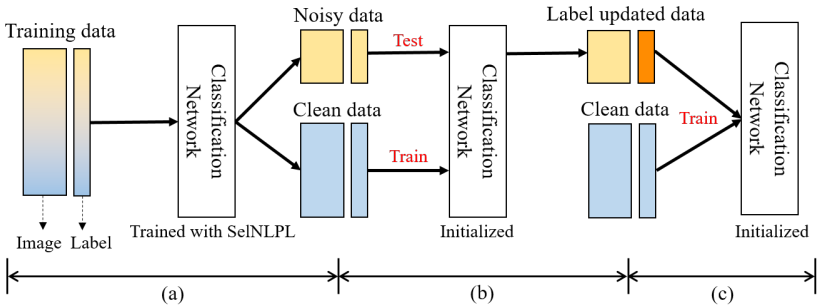


Figure 5: Pseudo labeling for semi-supervised learning. (a): Division of training data into either clean or noisy data with CNN trained with SelNLPL. (b): Training initialized CNN with clean data from (a), then noisy data's label is updated following the output of CNN trained with clean data. (c): Clean data and label-updated noisy data are both used for training initialized CNN in the final step.

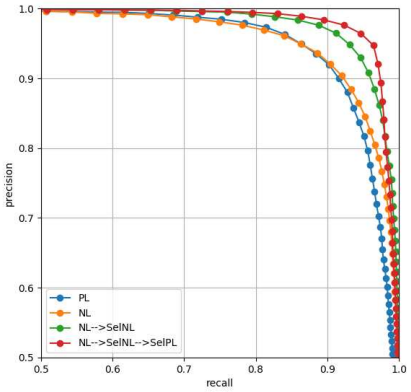


Figure 6: Precision-Recall curve when filtering noisy data. Each curve represents the filtering performance of PL, NL, NL→SelNL, and NL→SelNL→SelPL (SelNLPL).

		Estimated noise (%)	Recall	Precision
Noise (%)	9	10.25	92.87	85.20
	27	27.60	96.28	94.01
	45	45.15	95.80	95.38

Table 1: Results for filtering noisy data from CIFAR10 with 10%, 30%, and 50% *symm-inc* noise. Owing to the characteristics of *symm-inc* noise (Section 5.1), the actual noise in each case is 9%, 27%, and 45%, respectively.

this approach, we filter out data that were not trained with PL as noisy data. Table 1 summarizes the filtering results for SelNLPL on CIFAR10 with various noise ratios (*symm-inc*). The estimated noise ratio refers to the amount of data that was not trained with PL. Recall and Precision are measures for the quality of noisy data filtering. It shows that the estimated noise ratio almost matches the actual noise ratio by 88% to 99%. Furthermore, Table 1 shows our method of filtering noisy data resulted in high recall and precision values, indicating that our method filtered out most of the pure noisy data from the training data. This implies that even if the amount of noise mixed in the training data is unknown, which is normal in practical situations, the amount of noise can be estimated with SelNLPL, which is a huge advantage because it can be used as an indicator of the training data quality.

Figure 6 compares the overall filtering ability of the proposed methods. The curve for PL is obtained when the CNN is trained with PL before overfitting to noisy data. The curve indicates that each step of SelNLPL contributes to an increase in filtering performance, surpassing that of PL.

To summarize, SelNLPL exhibits excellent results for filtering noisy data from training data, as shown in Figure 6. Additionally, the estimated noise ratio from SelNLPL almost matches that of the actual noise ratio; hence, can be used to indicate training data quality in practical situations where the actual noise ratio is not available.

5. Experiments

In this section, we describe the experiments performed to evaluate our method. The results of pseudo labeling after SelNLPL are compared to those of other existing methods for noisy data classification. To show that our method generalizes to various environments, we follow each different experimental settings of other baseline methods, which vary in terms of CNN architecture, dataset, and type of noise in the training data. (Table 3, 4, 5, 6)

5.1. Experimental settings

We conducted our experiments following experimental settings of four different baselines. The experimental results of each baseline are reported, and we added our results to each of those result tables for comparison (Table 3, 4, 5, 6). Table 2 summarizes different experimental settings for each baseline. Datasets we used are CIFAR10, CIFAR100 [16], FashionMNIST [35], and MNIST [18].

We applied three different types of noise following the baseline methods.

Symmetric noise The basic idea of symmetric noise is to randomly select a label with equal probabilities among the classes. In this experiment, two symmetric noises were

		Table 3	Table 4	Table 5	Table 6
Validation split		✓	✓	x	✓
noise	*symm inc*	x	✓	x	x
	symm exc	✓	x	✓	✓
	asymm	✓	✓	x	✓

Table 2: Experimental settings (dataset preparation and types of noise) used for Table 3, 4, 5, 6.

used: *symm-inc* noise and *symm-exc* noise. *Symm-inc* noise is created by randomly selecting the label from all classes, including the ground truth label, whereas *symm-exc* noise flips the ground truth label to one of the other class labels, thus excluding the ground truth label. *Symm-inc* noise is used in Table 4, and *symm-exc* noise is used in Table 3, 5, 6.

Asymmetric noise As described by Patrini *et al*. [25], this noise mimics some of the structures of real errors for similar classes. For CIFAR10, asymmetric (*asymm*) noise was generated by mapping TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ PLANE, DEER $\rightarrow$ HORSE, and CAT $\leftrightarrow$ DOG. For FashionMNIST, BOOT $\rightarrow$ SNEAKER, SNEAKER $\rightarrow$ SANDALS, PULLOVER $\rightarrow$ SHIRT, and COAT $\leftrightarrow$ DRESS were mapped, following [39]. For MNIST, 2 $\rightarrow$ 7, 3 $\rightarrow$ 8, 7 $\rightarrow$ 1, and 5 $\leftrightarrow$ 6 were mapped, following [25]. For CIFAR100, the noise flipped each class into the next, circularly within super-classes, following [25].

For optimization, we used stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 10^{-4}, and batch size of 128. For NL, SelNL, and SelPL, each of them trained the CNN for 720 epochs. Except for MNIST, learning rate was unified across all datasets and CNN architectures. Learning rates for NL, SelNL, and SelPL were set to 0.02, 0.02, and 0.1, respectively. For pseudo labeling, for each step (Figure 5 (b), (c)), learning rate was scheduled to start from 0.1 and was divided by 10 at 192, 288 epochs (480 epochs total). As an exception, learning rates for NL and SelNL were set to 0.1 when using MNIST.

5.2. Results

Table 3 shows the results from Zhang *et al*. [39], supplemented with our results. An 18-layer ResNet [8] was used for FashionMNIST, whereas a 34-layer ResNet [8] was used for CIFAR10 and CIFAR100. Our method achieved the best overall accuracy in almost all cases, regardless of CNN architecture, dataset, noise type, or noise ratio. In some cases, our method outperformed the others significantly, up to a maximum of 5%. Our method only failed to converge when the *symm-exc* noise is 80%, which can be neglected because such a scenario is unrealistic. It should be noted that Zhang *et al*. [39] referenced the accuracy of the validation data to prevent overfitting to noisy data, whereas our method is advantageous as it did not reference any validation accuracy. The results of CIFAR100 were achieved by the extended version of our method, which is detailed in Section 6.1.

Table 4 is taken from Tanaka *et al*. [30]. A 32-layer Pre-ResNet [9] was used for CIFAR10. Similar to those in Table 3, our method outperformed all other comparable methods reported in Tanaka *et al*. [30], regardless of noise types and ratios. This result is noteworthy because Tanaka *et al*. [30] conducted their experiments by varying a few hyper-parameters depending on both noise type and noise ratio. In realistic cases, this setting is not applicable because the type or ratio of noise is unknown. Our method is excellent as the hyper-parameters do not vary according to noise type and ratio. Furthermore, for fair comparison for the case of *asymm* noise, we matched the parameter settings used for *asymm* noises to those used for 10% (Joint*) and 30% (Joint**) *symm-inc* noises, because the amount of noisy data for *asymm* noise cases lies between that of 10% *symm-inc* and 30% *symm-inc*. In that case, the overall accuracy of [30] for *asymm* noise changed, making our method superior for all *asymm* noise cases.

Table 5 and 6 are taken from [22] and [2], respectively. While Table 5 adopted structure of LeNet5 for MNIST, Table 6 used a 2-layer fully connected network for MNIST and a 14-layer ResNet for CIFAR10. Both tables show our method surpassed most of the other comparable results for all CNN architectures, datasets, noise types and ratios. In some cases, the performance of our method exceeded those of other methods by up to 4$\sim$5%, demonstrating the superiority of our method. Our method only performed second best for 60% *asymm* noise in Table 6, but we believe this is unimportant because such a scenario is unrealistic.

6. Analysis

6.1. Generalization to number of classes

Our method NL is an indirect learning method using a complementary label. Owing to the nature of NL's optimization process, the amount of convergence depends on class number c in the dataset; as c increases, the training of the CNN becomes slower. Consequently, our method failed to converge on CIFAR100 when training the CNN with the same number of epochs as used when training the CNN with CIFAR10. To overcome and analyze this phenomenon, we observed the gradients resulting from NL (Eq. 2). Here, we consider the gradients associated with clean data points to gain insight into the way we extend the NL method to many class cases. Let us consider a data point with a clean label, implying given $\overline{y}$ always is not true label y. By assuming that the CNN is at its initial state, the following probability values in Eq. 2 are approximated to be uniform across all classes ($\boldsymbol{p}_i = \frac{1}{c}$). The gradients are approximated as follows:

$$\frac{\partial \mathcal{L}(f, \overline{y})}{\partial f_i} = \begin{cases} \boldsymbol{p}_i \approx \frac{1}{c} & \text{if } i = \overline{y} \\ -\frac{\boldsymbol{p}_{\overline{y}}}{1-\boldsymbol{p}_{\overline{y}}} \boldsymbol{p}_i \approx -\frac{1}{c(c-1)} & \text{if } i \neq \overline{y} \end{cases} \tag{3}$$

Datasets	Model	Methods	*Symm*				*Asymm*			
			20	40	60	80	10	20	30	40
FashionMNIST	ResNet18	CE	93.24	92.09	90.29	86.20	94.06	93.72	92.72	89.82
		MAE [3]	80.39	79.30	82.41	74.73	74.03	63.03	58.14	56.04
		Forward T [25]	93.64	92.69	91.16	87.59	94.33	94.03	93.91	93.65
		Forward $\hat{T}$ [25]	93.26	92.24	90.54	85.57	94.09	93.66	93.52	88.53
		L_q [39]	93.35	92.58	91.30	88.01	93.51	93.24	92.21	89.53
		Truncated L_q [39]	93.21	92.60	91.56	88.33	93.53	93.36	92.76	91.62
		Ours	**94.82**	**94.16**	**92.78**	-	**95.10**	**94.88**	**94.66**	**93.96**
CIFAR10	ResNet34	CE	86.98	81.88	74.14	53.82	90.69	88.59	86.14	80.11
		MAE [3]	83.72	67.00	64.21	38.63	82.61	52.93	50.36	45.52
		Forward T [25]	88.63	85.07	79.12	64.30	91.32	90.35	89.25	88.12
		Forward $\hat{T}$ [25]	87.99	83.25	74.96	54.64	90.52	89.09	86.79	83.55
		L_q [39]	89.83	87.13	82.54	64.07	90.91	89.33	85.45	76.74
		Truncated L_q [39]	89.70	87.62	82.70	67.92	90.43	89.45	87.10	82.28
		Ours	**94.23**	**92.43**	**88.32**	-	**94.57**	**93.35**	**91.80**	**89.86**
CIFAR100	ResNet34	CE	58.72	48.20	37.41	18.10	66.54	59.20	51.40	42.74
		MAE [3]	15.80	9.03	7.74	3.76	13.38	11.50	8.91	8.20
		Forward T [25]	63.16	54.65	44.62	24.83	**71.05**	**71.08**	**70.76**	**70.82**
		Forward $\hat{T}$ [25]	39.19	31.05	19.12	8.99	45.96	42.46	38.13	34.44
		L_q [39]	66.81	61.77	53.16	29.16	68.36	66.59	61.45	47.22
		Truncated L_q [39]	67.61	62.64	54.04	29.60	68.86	66.59	61.87	47.66
		Ours	**71.52**	**66.39**	**56.51**	-	70.35	63.12	54.87	45.70

Table 3: Comparison with results reported by Zhang *et al.* [39]

Datasets	Model	Methods	*Symm*				*Asymm*			
			10	30	50	70	10	20	30	40
CIFAR10	Pre-ResNet32	CE	87.0	72.2	55.3	36.6	89.8	85.4	81.0	75.7
		Forward [25]	-	-	-	-	91.7	89.7	88.0	86.4
		CNN-CRF [31]	-	-	-	-	90.3	86.6	83.6	79.7
		Joint [30]	92.9	91.5	89.8	86.0	93.2	92.8	92.4	**91.7**
		Joint* [30]	-	-	-	-	93.05	92.60	91.59	89.23
		Joint** [30]	-	-	-	-	93.26	93.06	92.04	90.65
		Ours	**94.25**	**93.42**	**91.45**	**86.13**	**94.12**	**93.44**	**92.56**	90.99

Table 4: Comparison with results reported by Tanaka *et al.* [30]

Datasets	Model	Methods	*Symm*		
			20	40	60
MNIST	LeNet	CE	88.02	68.46	45.51
		Forward [25]	96.45	94.90	82.88
		Backward [25]	90.12	70.89	52.83
		Boot-hard [26]	87.69	69.49	50.45
		Boot-soft [26]	88.50	70.19	46.04
		D2L [22]	98.84	98.49	94.73
		Ours	**99.35**	**99.27**	**98.91**

Table 5: Comparison with results reported by Ma *et al.* [22]

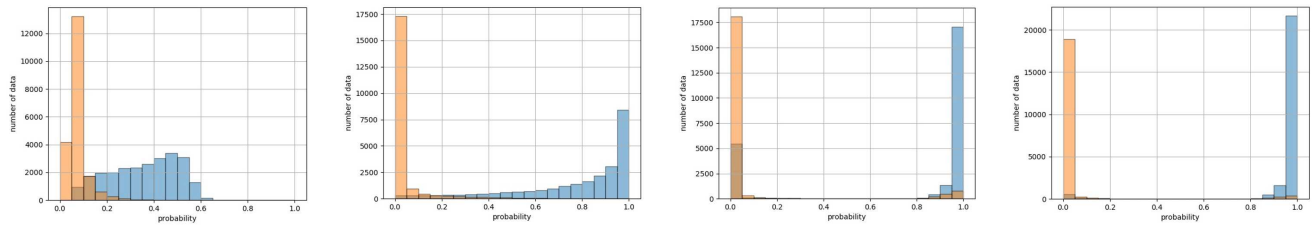
The detailed outline of Eq. 3 is given in the appendix.

Eq. 3 shows that while gradient occurs to reduce the score corresponding to the given $\overline{y}$, a gradient also occurs to enhance the scores corresponding to other remaining classes, including true label y. This implies that after training CNN with NL, the gradient received at y is $\frac{1}{c(c-1)}$.

Suppose we are training the CNN with either 10-class dataset or 100-class dataset. The gradient received at y is $\frac{1}{9*10}$ for 10-class dataset and $\frac{1}{99*100}$ for 100-class dataset. Comparing these two cases, the gradient of the 100-class dataset is 110 times smaller than that of the 10-class dataset. This analysis implies that, for NL to converge on CIFAR100, it requires more epochs than CIFAR10, approximately up to a factor of 110. However as it requires a significant amount of time to the train CNN, we extend our method to provide multiple random complementary labels for each image. We computed 110 losses on a single data with 110 random $\overline{y}$s (allowing duplication), with

Datasets	Model	Methods	*Symm*	*Asymm*	
			20	20	60
CIFAR10	ResNet14	CE	83.7	85.0	57.6
		Unhinged (BN) [32]	84.1	83.8	52.1
		Sigmoid (BN) [4]	66.6	71.8	57.0
		Savage [23]	77.4	76.0	50.5
		Bootstrap soft [26]	84.3	84.6	57.8
		Bootstrap hard [26]	83.6	84.7	58.3
		Backward [25]	80.4	83.8	66.7
		Forward [25]	83.4	87.0	74.8
		Semi [2]	84.5	85.6	**75.8**
		Ours	**89.85**	**90.1**	74.44
MNIST	FC2	CE	96.9	97.5	53.0
		Unhinged (BN) [32]	96.9	97.0	71.2
		Sigmoid (BN) [4]	93.1	96.7	71.4
		Savage [23]	96.9	97.0	51.3
		Bootstrap soft [26]	96.9	97.5	53.0
		Bootstrap hard [26]	96.8	97.4	55.0
		Backward [25]	96.9	96.7	67.4
		Forward [25]	96.9	97.7	64.9
		Semi [2]	97.7	97.8	**83.4**
		Ours	**97.96**	**97.97**	79.48

Table 6: Comparison with results reported by Ding *et al.* [2]



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

Figure 7: Histogram showing the distribution of CIFAR10 training data with 50% *symm-inc* noise, according to probability (confidence). (a): NL. (b): NL→SelNL. (c): NL→SelPL. (d): NL→SelNL→SelPL (SelNLPL).

		30% *symm-inc* noise				50% *symm-inc* noise			
		Accuracy	Estimated noise (%)	Recall	Precision	Accuracy	Estimated noise (%)	Recall	Precision
#1	NL-SelNL-SelPL	93.82	27.60	96.28	94.01	91.17	45.15	95.80	95.38
#2	NL-SelNL	92.44 (-1.38)	33.76	98.47	78.62	89.36 (-1.81)	52.75	98.56	83.99
#3	NL-SelPL	93.41 (-0.41)	28.17	97.04	92.84	72.91 (-18.26)	54.35	92.12	76.19
#4	NL	87.32 (-6.5)	52.24	99.80	51.49	72.53 (-18.64)	89.21	99.99	50.38

Table 7: Analysis of measuring significance of each step of SelNLPL. #1: SelNLPL. #2: Deleting SelPL from #1. #3: Deleting SelNL from #1. #4: Deleting SelNL and SelPL from #1.

only slight increase in time for back-propagation as the 110 losses share features computed for one image. With this simple extended method, we observed that the CNN could converge on CIFAR100 when trained with the same number of epochs as CIFAR10, and showed general improvement in noisy data classification (Table 3). For *symm-exc* noise, we achieved state-of-the-art results. For *asymm* noise, Forward T [25] shows the best performance. However, this is not a fair comparison as it relies on the prior knowledge of the confusion matrix, which summarized the probability of one class being flipped into another under noise. Therefore it can be concluded that our method achieved comparable results excluding Forward T.

In this section, we demonstrated that our method can be generalized to datasets with many class numbers by providing multiple complementary labels for each image.

6.2. Ablation study

Our paper suggests a novel noisy data classification method, composed of multiple steps: SelNLPL (NL→SelNL→SelPL), followed by pseudo labeling for semi-supervised learning. To investigate the strength of each step in SelNLPL, we conducted an analysis that reveals the difference in performance when each step of SelNLPL is omitted from the entire training process. One or more steps are deleted from SelNLPL and then applied to pseudo labeling. Table 7 shows all experiments conducted for the analysis, following the experimental settings of Table 4 with 30% and 50% *symm-inc* noise. It includes SelNLPL (#1) and deleting either SelPL (#2), SelNL (#3), or both SelNL and SelPL (#4) from #1.

In Table 7, compared to #1, both #2 and #3 show deteriorated performance, while #4's performance is further decreased (Figure 7). Although the accuracy drop is quite small on *symm-inc* 30% noise case, significant degradation in performance is shown in *symm-inc* 50% noise case, especially for #3 and #4, where SelNL is deleted from the training process. Convergence of CNN when training with NL depends heavily on the amount of noise in the training data, being less convergent as the noise ratio increases (Figure 3 (b), Figure 7 (a)). Consequently, SelNL becomes crucial with the increase in noise ratio as it is responsible for ignoring the noisy data having low confidence value. SelNL enhances the overall clean data ratio involved in training, thus yielding better convergence (Figure 3 (c), Figure 7 (b)).

7. Conclusion

We proposed NL for training with noisy data, a learning method of training CNNs that "input image does not belong to this complementary label." This reduces the risk of training CNNs with wrong information as the chances of randomly choosing a complementary label that is not a ground-truth label are high. Furthermore, because PL is faster and more accurate when learning with clean data compared to NL, we developed a new method, SelNLPL, by combining PL and NL, obtaining a superior performance for filtering noisy data from training data. Our study performed successful noisy data classification by using semi-supervised learning (pseudo labeling) based on the filtering result of SelNLPL, achieving state-of-the-art results without tuning our method based on any prior knowledge.

References

[1] J Paul Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations research*, 59(2):467–479, 2011. 2

[2] Yifan Ding, Liqiang Wang, Deliang Fan, and Boqing Gong. A semi-supervised two-stage approach to learning from noisy labels. *arXiv preprint arXiv:1802.02679*, 2018. 1, 2, 6, 7

[3] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925, 2017. 1, 2, 7

[4] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015. 2, 7

[5] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016. 2

[6] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: robust training deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872*, 2018. 1, 2

[7] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6307–6315. IEEE, 2017. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 6

[10] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. *arXiv preprint arXiv:1802.05300*, 2018. 2

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 1

[12] Takashi Ishida, Gang Niu, Weihua Hu, and Masashi Sugiyama. Learning from complementary labels. In *Advances in Neural Information Processing Systems*, pages 5639–5649, 2017. 2

[13] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017. 2

[14] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. Learning deep networks from noisy labels with dropout regularization. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 967–972. IEEE, 2016. 2

[15] Ashish Khetan, Zachary C Lipton, and Anima Anandkumar. Learning from noisy singly-labeled data. *arXiv preprint arXiv:1712.04577*, 2017. 2

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[18] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. 5

[19] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013. 4

[20] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. *arXiv preprint arXiv:1711.07131*, 2017. 1, 2

[21] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, pages 1928–1936, 2017. 1, 2

[22] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. *arXiv preprint arXiv:1806.02612*, 2018. 1, 2, 6, 7

[23] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009. 2, 7

[24] Curtis G Northcutt, Tailin Wu, and Isaac L Chuang. Learning with confident examples: Rank pruning for robust classification with noisy labels. *arXiv preprint arXiv:1705.01936*, 2017. 2

[25] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2233–2241, 2017. 2, 6, 7, 8

[26] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 1, 2, 7

[27] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018. 2

[28] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 2

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[30] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. *arXiv preprint arXiv:1803.11364*, 2018. 1, 2, 4, 6, 7

[31] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017. 2, 7

[32] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The im-

portance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015. 2, 7

[33] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 6575–6583, 2017. 1, 2

[34] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. *arXiv preprint arXiv:1804.00092*, 2018. 1, 2

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 5

[36] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015. 2

[37] Xiyu Yu, Tongliang Liu, Mingming Gong, and Dacheng Tao. Learning with biased complementary labels. In *ECCV 2018*, pages 69–85, 2018. 2

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

[39] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018. 1, 2, 6, 7