Joint Optimization Framework for Learning with Noisy Labels

Daiki Tanaka   Daiki Ikami   Toshihiko Yamasaki   Kiyoharu Aizawa
The University of Tokyo, Japan
`{tanaka, ikami, yamasaki, aizawa}@hal.t.u-tokyo.ac.jp`

Abstract

Deep neural networks (DNNs) trained on large-scale datasets have exhibited significant performance in image classification. Many large-scale datasets are collected from websites, however they tend to contain inaccurate labels that are termed as noisy labels. Training on such noisy labeled datasets causes performance degradation because DNNs easily overfit to noisy labels. To overcome this problem, we propose a joint optimization framework of learning DNN parameters and estimating true labels. Our framework can correct labels during training by alternating update of network parameters and labels. We conduct experiments on the noisy CIFAR-10 datasets and the Clothing1M dataset. The results indicate that our approach significantly outperforms other state-of-the-art methods.

1. Introduction

DNNs trained on large-scale datasets have achieved impressive results on many classification problems. Generally, accurate labels are necessary to effectively train DNNs. However, many datasets are constructed by crawling images and labels from websites and often contain incorrect noisy labels (*e.g.*, YFCC100M [17], Clothing1M [21]). This study addresses the following question: how can we effectively train DNNs on noisy labeled datasets without manually cleaning the data?

The prominent issue in training DNNs on noisy labeled datasets is that *DNNs can learn or memorize, any training dataset*, and this implies that DNNs are subject to total overfitting on noisy data.

To address this problem, commonly used regularization techniques including dropout and early stopping are helpful. However, these methods do not guarantee optimization because they prevent the networks from reducing the training loss. Another method involves using prior knowledge, such as the confusion matrix between clean and noisy labels, which typically cannot be used in real settings.
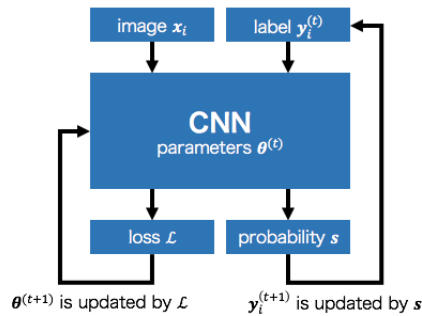

Figure 1. The concept of our joint optimization framework. Noisy labels are reassigned to the probability output by CNNs. Network parameters and labels are alternatively updated for each epoch.

Consequently, we need a new framework of optimization. In this study, we propose an optimization framework for learning on a noisy labeled dataset. We propose optimizing the labels themselves as opposed to treating the noisy labels as fixed. The joint optimization of network parameters and the noisy labels corrects inaccurate labels and simultaneously improves the performance of the classifier. Fig. 1 shows the concept of our proposal. The main contributions are as follows.

1. We propose a joint optimization framework for learning on noisy labeled datasets. Our optimization problem has two optimization network parameters and class labels that are optimized by an alternating strategy.

2. We observe that a DNN trained on noisy labeled datasets does not memorize noisy labels and maintains high performance for clean data under a high learning rate. This reinforces the findings of Arpit *et al.* [1] that suggest that DNNs first learns simple patterns and subsequently memorize noisy data.

3. We evaluate the performance on synthetic and real noisy datasets. We demonstrate state-of-the-art performance on the noisy CIFAR-10 dataset and a comparable performance on the Clothing1M dataset [21].

2. Related Works

2.1. Generalization abilities of DNNs

Recently, generalization and memorization abilities of neural networks have attracted increasing attention. Specifically, we focus on the ability of learning labels. Zhang *et al*. showed that DNNs can learn any training dataset even if the training labels are completely random [22]. This leads to two problems. First, the performance of a DNN decreases when it is trained on a noisy dataset and completely learns noisy labels. Second, it is difficult to learn which label is noisy given the perfect learning ability. To the best of our knowledge, most studies on deep learning with respect to noisy labels do not focus on the aforementioned problems that are caused by the memorization ability of DNNs. This study involves addressing these two problems to improve the classification accuracy by preventing completely learning for noisy labels.

2.2. Learning on noisy labeled datasets

We briefly review existing studies on learning on noisy labeled datasets.

Regularization: Regularization is an efficient method to deal with the issue of DNNs easily fitting noisy labels, as described in Section 2.1. Arpit *et al*. showed the performances of DNNs trained on noisy labeled datasets with several regularizations [1] including weight decay, dropout, and adversarial training [6]. Zhang *et al*. used a mixup [23] involving the utilization of a linear combination of images and labels for training.

These techniques improve performance on clean labels. However, these methods do not explicitly deal with noisy labels, and therefore long-time training leads to performance degradation as follows: the performance of the last epoch is generally worse than that of the best epoch [23]. Furthermore, it is not possible to use the training loss on the noisy labeled dataset as a measure of performance on clean labels. Therefore, training-loss based early stopping does not work well.

Noise transition matrix: Let l and l^{GT} be the noisy and true labels. We define the noise transition matrix T by $t_{ij} = p(l = j | l^{GT} = i)$. Then, we can use T to modify the cross entropy loss as follows:

$$\mathcal{L}(\boldsymbol{\theta}, Y, X) = -\frac{1}{n} \sum_{i=1}^{n} \log\left(\boldsymbol{y}_i^{\mathrm{T}} T \boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i)\right). \quad (1)$$

This formulation was used in many studies [16, 11, 14]. In deep learning, some studies presuppose the ground-truth noise transition matrix T [14, 19] and achieve the state-of-the-art performance in the noisy CIFAR-10 dataset. Other studies estimate T from noisy data. Specifically, T is modeled by a fully connected layer and is trained in an end-to-end manner [16, 11]. However, these methods do not carefully focus on the memorization ability of DNNs. Patrini *et al*. proposed an estimation method for T [14]; however, its performance is slightly worse than that obtained with the true T.

Robust loss function: A few studies achieve noise-robust classification by using a noise-tolerant loss functions, such as ramp loss [3] and unhinged loss [20]. For further details please refer to [5]. In deep learning, Ghosh *et al*. used mean square error and mean absolute error [4] for noise-tolerant loss functions. It should be noted that they do not consider the problem that DNNs can learn arbitrary labels.

Other approaches using deep learning: Reed *et al*. used a bootstrapping scheme to handle noisy labels [15]. Our method is similar to this study. Xiao *et al*. constructed a noise model with multiple noise types and trained two networks: an image classifier and a noise type classifier [21]. It should be noted that this method requires a low amount of accurately labeled datasets.

2.3. Self-training and pseudo-labeling

Pseudo-labeling [24, 7, 13] is a type of self-training that is generally used in semi-supervised learning with few labeled data and many unlabeled data. In this technique, pseudo-labels are initially assigned to unlabeled data by predictions of a model trained on a clean dataset. Subsequently, the algorithm repeats retraining the model on both labeled and unlabeled data and updating pseudo-labels.

In semi-supervised learning, we know which data is labeled or not and only need to assign pseudo-labels to only unlabeled data. However, with respect to learning on noisy labeled data, it is necessary to treat all data equally because we do not know which data is noisy. Reed *et al*. proposed a self-training scheme [15] for training a DNN on noisy labeled data. Their approach is similar to that proposed in this study. However, they use original noisy labels for learning until the end of training, and thus the performance is degraded by the remaining effects of noisy labels for a high noise rate [15, 11]. Conversely, we completely replace all labels by pseudo-labels and use the same for training.

3. Notation and Problem Statements

In this study, column vectors and matrices are denoted in bold (*e.g.* $\boldsymbol{x}$) and capitals (*e.g.* X), respec-

tively. Specifically, $\mathbf{1}$ is a vector of all-ones. We define hard-label spaces $\mathcal{H} = \{\boldsymbol{y} : \boldsymbol{y} \in \{0,1\}^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$ and soft-label spaces $\mathcal{S} = \{\boldsymbol{y} : \boldsymbol{y} \in [0,1]^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$.

In supervised c-class classification problem setting, we have a set of n training images $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ with ground-truth labels $Y^{GT} = [\boldsymbol{y}_1^{GT}, \ldots, \boldsymbol{y}_n^{GT}] = \mathbb{R}^{c \times n}$, where $\boldsymbol{y}_i^{GT} \in \mathcal{H}$ is a one-hot vector representation of the true class label. The objective function is an empirical risk, such as the cross entropy, as follows:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c y_{ij}^{GT} \log s_j(\boldsymbol{\theta}, \boldsymbol{x}_i), \qquad (2)$$

where $\boldsymbol{\theta}$ denotes the set of network parameters and $\boldsymbol{s}$ denotes the output of the final layer, namely c-class softmax layer, of the network.

If a clean training dataset is present, then the network parameters $\boldsymbol{\theta}$ are learned by optimizing Eq. (2) by using a gradient descent method. However, in this study, we consider the classification problem with noisy labels as follows: Let $\boldsymbol{y}_i$ be the noisy label, and only the noisy training label set $Y = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n]$ is given. The task involves training CNNs to predict true labels. In the next section, we describe the proposed method for training on noisy labels.

4. Classification with Label Optimization

In this section, we present our proposed training method with noisy labels. Generally, with respect to supervised learning on clean labels, the optimization problem is formulated as follows:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|X, Y), \qquad (3)$$

where $\mathcal{L}$ denotes a loss function such as the cross entropy loss Eq. (2). Eq. (3) works well on clean labels. However, if we train the network by Eq. (3) on noisy labels, its performance decreases.

As we will describe in Section 5.3, we experimentally found that a high learning rate suppresses the memorization ability of a DNN and prevents it from completely fitting to labels. Thus, we assume that a network trained with a high learning rate will have more difficulty fitting to noisy labels. In other words, the loss Eq. (3) is high for noisy labels and low for clean labels. Given this assumption, we obtain clean labels by updating labels in the direction to decrease Eq. (3). Therefore, we formulate the problem as the joint optimization of network parameters and labels as follows:

$$\min_{\boldsymbol{\theta}, Y} \mathcal{L}(\boldsymbol{\theta}, Y|X). \qquad (4)$$

The concept of our proposal is shown in Fig. 1.

Algorithm 1 Alternating Optimization

for $t \leftarrow 1$ to num_epochs **do**
    update $\boldsymbol{\theta}^{(t+1)}$ by SGD on $\mathcal{L}(\boldsymbol{\theta}^{(t)}, Y^{(t)}|X)$
    update $Y^{(t+1)}$ by Eq. (8) (hard-label)
            or Eq. (9) (soft-label)
end for

Our proposed loss function $\mathcal{L}(\boldsymbol{\theta}, Y|X)$ is constructed by three terms as follows:

$$\mathcal{L}(\boldsymbol{\theta}, Y|X) = \mathcal{L}_c(\boldsymbol{\theta}, Y|X) + \alpha \mathcal{L}_p(\boldsymbol{\theta}|X) + \beta \mathcal{L}_e(\boldsymbol{\theta}|X), \quad (5)$$

where $\mathcal{L}_c(\boldsymbol{\theta}, Y|X)$, $\mathcal{L}_p(\boldsymbol{\theta}|X)$, $\mathcal{L}_e(\boldsymbol{\theta}|X)$ denote the classification loss and two regularization losses, respectively, and α and β denote hyper parameters. In this study, we use the Kullback-Leibler (KL)-divergence for $\mathcal{L}_c(\boldsymbol{\theta}, Y|X)$ as follows:

$$\mathcal{L}_c(\boldsymbol{\theta}, Y|X) = \frac{1}{n} \sum_{i=1}^n D_{KL}(\boldsymbol{y}_i || \boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i)), \qquad (6)$$

$$D_{KL}(\boldsymbol{y}_i || \boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i)) = \sum_{j=1}^c y_{ij} \log\left(\frac{y_{ij}}{s_j}(\boldsymbol{\theta}, \boldsymbol{x}_i)\right). \quad (7)$$

In the following subsections, we first describe an alternating optimization method to solve this problem, and we then describe the definition of $\mathcal{L}_p$, $\mathcal{L}_e$.

4.1. Alternating Optimization

In our proposed learning framework, network parameters $\boldsymbol{\theta}$ and class labels Y are alternatively updated as shown in Algorithm 1. We will describe the update rules of $\boldsymbol{\theta}$ and Y.

Updating $\boldsymbol{\theta}$ with fixed Y: All terms in the optimization problem Eq. (5) are sub-differentiable with respect to $\boldsymbol{\theta}$. Therefore, we update $\boldsymbol{\theta}$ by the stochastic gradient descent (SGD) on the loss function Eq. (5).

Updating Y with fixed $\boldsymbol{\theta}$: In contrast to other methods, we update and optimize the labels that we perform the training on. With respect to updating Y, it is only necessary to consider the classification loss $\mathcal{L}_c(\boldsymbol{\theta}, Y|X)$ from Eq. (5) with fixed $\boldsymbol{\theta}$. The optimization problem Eq. (6) on Y is separated for each $\boldsymbol{y}_i$.

As a method of optimizing labels, two methods can be considered: the hard-label method and the soft-label method. In the case of the hard-label method, $Y \in \mathcal{H}$ is updated as follows:

$$y_{ij} = \begin{cases} 1 & \text{if} \quad j = \underset{j'}{\text{argmax}}\, s_{j'}(\boldsymbol{\theta}, \boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases}. \qquad (8)$$

In the case of the soft-label method, the KL-divergence from $\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i)$ to $\boldsymbol{y}_i$ is minimized when $\boldsymbol{y}_i = \boldsymbol{s}$, and thus

the update rule for $Y \in \mathcal{S}$ is as follows:

$$\boldsymbol{y}_i = \boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i). \tag{9}$$

As we will describe in Section 5.4, we experimentally determined that the performance of the soft-label method exceeded that of the hard-label method. Thus, we applied soft-labels to all experiments if not otherwise specified.

4.2. Regularization Terms

We describe definitions and roles of two regularization losses of $\mathcal{L}_p(\boldsymbol{\theta}|X)$ and $\mathcal{L}_e(\boldsymbol{\theta}|X)$.

Regularization loss $\mathcal{L}_p$: The regularization loss $\mathcal{L}_p(\boldsymbol{\theta}|X)$ is required to prevent the assignment of all labels to a single class: In the case of minimizing only Eq. (6), we obtain a trivial global optimal solution with a network that always predicts constant one-hot $\hat{\boldsymbol{y}} \in \mathcal{H}$ and each label $\boldsymbol{y}_i = \hat{\boldsymbol{y}}$ for any image $\boldsymbol{x}_i$. To overcome this problem, we introduce a prior probability distribution $\boldsymbol{p}$, which is a distribution of classes among all training data. If the prior distribution of classes is known, then the updated labels should follow the same. Therefore, we introduce the KL-divergence from $\bar{\boldsymbol{s}}(\boldsymbol{\theta}, X)$ to $\boldsymbol{p}$ as a cost function as follows:

$$\mathcal{L}_p = \sum_{j=1}^{c} p_j \log \frac{p_j}{\bar{s}_j(\boldsymbol{\theta}, X)} \tag{10}$$

This approach follows [10]. The mean probability $\bar{\boldsymbol{s}}(\boldsymbol{\theta}, X)$ in the training data is approximated by performing a calculation for each mini-batch $\mathcal{B}$ as Eq. (11).

$$\bar{\boldsymbol{s}}(\boldsymbol{\theta}, X) = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}_i) \approx \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}}\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}) \tag{11}$$

This approximation cannot treat a large number of classes and extreme imbalanced classes, however it works well in the experiments on the noisy CIFAR-10 dataset and the Clothing1M dataset.

Regularization loss $\mathcal{L}_e$: The term $\mathcal{L}_e$ is required for the training loss when we use the soft-label. We consider the case of Eq. (5) with $\alpha = \beta = 0$. In this case, when Y is updated by Eq. (9), both $\boldsymbol{\theta}$ and Y are stuck in local optima and the learning process does not proceed. To overcome this problem, we introduce an entropy term to concentrate the probability distribution of each soft-label to a single class as follows:

$$\mathcal{L}_e = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} s_j(\boldsymbol{\theta}, \boldsymbol{x}_i)\log s_j(\boldsymbol{\theta}, \boldsymbol{x}_i). \tag{12}$$

4.3. Additional Details

Our method has two steps for training on noisy labels. In the first step, we obtain clean labels by updating labels as described in Section 4.1. In the second step, we initialize the network parameters and train the network by usual supervised learning with the labels obtained in the first step.

5. Experiments

5.1. Datasets

CIFAR-10: We use the CIFAR-10 dataset [12] and retain 10% of the training data for validation. Subsequently, we define three types of the training data, namely Symmetric Noise CIFAR-10 (SN-CIFAR), Asymmetric Noise CIFAR-10 (AN-CIFAR), and Pseudo Label CIFAR-10 (PL-CIFAR).

In SN-CIFAR, we inject the symmetric label noise. Symmetric label noise is as follows:

$$\boldsymbol{y}_i = \begin{cases} \boldsymbol{y}_i^{GT} \text{ with the probability of } 1-r \\ \text{random one-hot vector with the probability of } r \end{cases}. \tag{13}$$

In AN-CIFAR, we inject the asymmetric label noise. The asymmetric label noise is discussed in [14]. The rationale involves mimicking a part of the structure of real mistakes for similar classes: TRUCK → AUTOMOBILE, BIRD → AIRPLANE, DEER → HORSE, CAT ↔ DOG. Transitions are parameterized by $r \in [0, 1]$ such that the probabilities of ground-truth and inaccurate class correspond to $1-r$ and r, respectively.

In PL-CIFAR, pseudo labels are assigned to unlabeled training data. Pseudo labels are generated by applying k-means++ [2] to features that are outputs of pool5 layer of ResNet-50 [8] pre-trained on ImageNet. This setting is motivated by transfer learning. The overall accuracy of the pseudo labels is 62.50%.

Clothing1M: We use the Clothing1M dataset [21] to examine the performance of our method in a real setting. The Clothing1M dataset contains 1 million images of clothing obtained from several online shopping websites that are classified into the following 14 classes: T-shirt, Shirt, Knitwear, Chiffon, Sweater, Hoodie, Windbreaker, Jacket, Down Coat, Suit, Shawl, Dress, Vest, and Underwear. The labels are generated by using surrounding texts of the images that are provided by the sellers, and therefore contain many errors. In [21], it is reported that the overall accuracy of the noisy labels is 61.54%, and some pairs of classes are often confused with each other (*e.g.*, Knitwear and Sweater). The Clothing1M dataset also contains 50k, 14k and 10k of clean data for training, validation, and

testing, respectively although we do not use the 50k clean training data.

5.2. Implementation

We implemented all the models with the deep learning framework Chainer v2.1.0 [18].

CIFAR-10: Training on SN-CIFAR, AN-CIFAR and PL-CIFAR, we used the network based on PreAct ResNet-32 [9] as detailed in the supplementary material. With respect to preprocessing, we performed mean subtraction and data augmentation by horizontal random flip and 32×32 random crops after padding with 4 pixels on each side. We used SGD with a momentum of 0.9, a weight decay of 10^{-4}, and batch size of 128.

In the first step of our method, we trained the network for 200 epochs and began updating labels from the 70th epoch. We determined the values of a learning rate and the hyper parameters (α, β in Eq. (5)) for SN-CIFAR, AN-CIFAR, and PL-CIFAR respectively based on the validation accuracy. The details are described in each experimental section. As we will describe in Section 5.4, soft-labels performed better than hard-labels, and thus we applied soft-labels to all the experiments in Section 5.5, Section 5.6, and Section 5.7. In this case, the prior distribution p is uniform distribution because each class has the same number of images in the CIFAR-10 dataset. While updating the noisy label $\boldsymbol{y}_i$ by the probability $\boldsymbol{s}$, we used the average output probability of the network of the past 10 epochs as $\boldsymbol{s}$. We experimentally determined that this averaging technique is useful in preventing inaccurate updates since it has a similar effect to ensemble.

In the second step of our method, we trained the network for 120 epochs on the labels obtained in the first step. We began training with a learning rate of 0.2 and divided it by 10 after 40 and 80 epochs. We used only $\mathcal{L}_c$ for the training loss in this step.

Clothing1M: Training on the Clothing1M dataset, we used ResNet-50 pre-trained on ImageNet to align experimental condition with [14]. For preprocessing, we resized the images to 256×256, performed mean subtraction, and cropped the middle 224×224. We used SGD with a momentum of 0.9, a weight decay of 10^{-3}, and batch size of 32.

In the first step of our method, we trained the network for 10 epochs and began updating labels from the 1st epoch. We used a learning rate of 8×10^{-4}, and used 2.4 for α and 0.8 for β. While updating the noisy label $\boldsymbol{y}_i$ by the probability $\boldsymbol{s}$, we used the average output probability of the network of all the past epochs as $\boldsymbol{s}$. We applied soft-labels to the experiment
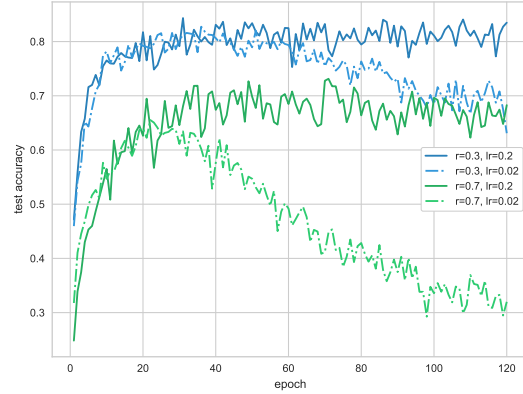

Figure 2. The test accuracy curve with different learning rates. The test accuracy gradually decreases when the learning rate is low (lr=0.02). Conversely, the test accuracy remains high at the end of training when the learning rate is high (lr=0.2).
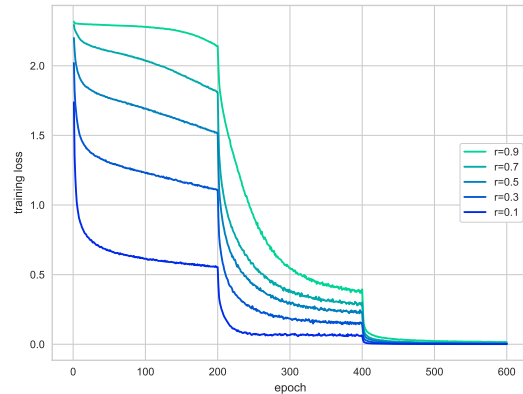

Figure 3. The training loss curve with different noise rates. At the end of training with a low learning rate, the value of the training loss is close to 0 even if the error rate is 0.9. In contrast, in the early phase of training with a high learning rate, an increase in the noise rate increases the training loss.

in Section 5.8.

In the second step of our method, we trained the network for 10 epochs on the labels obtained in the first step. We began training with a learning rate of 5×10^{-4} and divided it by 10 after 5 epochs.

5.3. Generalization and Memorization

To examine the effect of the learning rate (lr) and the noise rate (r) on the training loss and the test accuracy, we trained the network on SN-CIFAR with only the cross entropy loss.

Fig. 2 shows the test accuracy curve with different learning rates. We trained the network for 120 epochs with a learning rates of 0.2 or 0.02. In the case of the low learning rate (lr=0.02), the test accuracy was
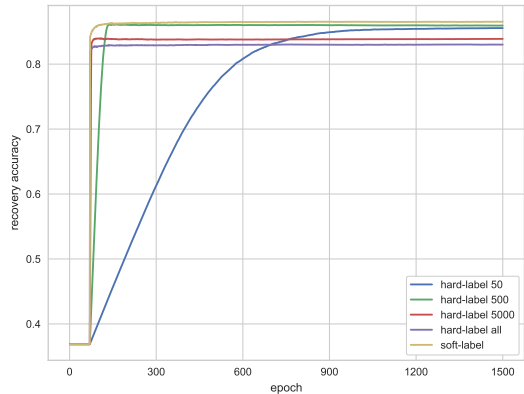
Figure 4. Comparison between the soft-label and the hard-label methods, showing the recovery accuracy. The soft-label method achieves faster convergence than the hard-label methods, and performs the best recovery accuracy.

high at the early phase of training and then gradually decreased because the network fitted the noisy labels. This is the same result reported in [1]. Conversely, in the case of the high learning rate (lr=0.2), the network exhibited high test accuracies during training. This means that a high learning rate prevents the network from memorizing and fitting the noisy labels.

Fig. 3 shows how the manner in which training loss declines during training. We trained the network for 600 epochs. We commenced training with a learning rate of 0.2 and divided it by 10 after 200 and 400 epochs. At the end of training, our model fit the noisy labels even if the noise rate was high (for *e.g.*, $r = 0.9$). However, with respect to training with a high learning rate, the training loss clearly increases when the noise rate increases. This indicates that it is possible to optimize the labels towards lowering the training loss when the learning rate is high.

5.4. Hard-Label vs. Soft-Label

To prove the effectiveness of the soft-label, we trained the network on SN-CIFAR (noise rate $r = 0.7$) for 1500 epochs by using the first step of our method. We compare the hard-label methods and the soft-label method. For the hard-labels methods, we update top 50, 500, 5000, or all labels whose current labels are most different from the predicted classes to the predicted hard-labels every epoch. For the soft-label method, we update all labels to the predicted soft-labels every epoch. In Fig. 4, we show the recovery accuracy, which is defined as the accuracy of the re-assigned labels, in the first step of our method. The soft-label method achieves faster convergence and better recovery accuracy than any hard-label methods.

Subsequently, by using the second step of our method, we performed training on the labels obtained

in the first step. In the hard-label methods, updating 500 labels every epoch is optimal and the test accuracy is 85.7%. Conversely, the test accuracy of the soft-label method is 86.0%. It shows that though the recovery accuracy of the soft-label method obtained in the first step is 86.0%, which is approximately equal to 85.9% of the hard-label method (updating 500 labels every epoch), the test accuracy is improved by 0.3%. The reason why the soft-label method performed better is considered as that soft-labels contain the probabilities of each class in themselves. Soft-labels reflect confidences of the trained network unlike hard-labels, which are assigned by ignoring confidences. Our results indicate that confidences are important in the case of training on noisy labels.

5.5. Experiment on SN-CIFAR

To evaluate the performance of our method on synthesized noisy labels, we trained the network on SN-CIFAR (the noise rate $r = 0.0, 0.1, 0.3, 0.5, 0.7, 0.9$) by using our method. In the first step of our method, we used the optimal learning rate, α and β for each noise rate based on the validation accuracy as detailed in the supplementary material. As a comparison, we also trained on initial noisy labels in the same manner as the second step of our method.

The results are reported in Table 1. As shown in Table 1, *best* denotes the scores of the epoch where the validation accuracy is optimal, and *last* denotes the scores at the end of training. The recovery accuracy for our method is defined as the accuracy of the reassigned labels. Conversely, other methods do not reassign the noisy labels, and thus the recovery accuracy is reported as the prediction accuracy on the ground-truth labels of the noisy training data.

Our method achieves overall better test accuracy and recovery accuracy on SN-CIFAR. When training was performed on initial noisy labels, the test accuracy decreases after approximately the 40th epoch (when we divided the learning rate by 10). This indicates that lowering the learning rate assists the network in fitting the noisy labels as described in Section 5.3. Conversely, when we trained on the labels optimized by our method, the test accuracy was high until the end of training. This is the important effects of our joint optimization.

5.6. Experiment on AN-CIFAR

To evaluate the performance of our method in the settings in [14], we trained the network on AN-CIFAR (the noise rate $r = 0.1, 0.2, 0.3, 0.4, 0.5$) by using our method. In the first step of our method, we used a learning rate of 0.03 and used 0.8 for α and 0.4 for β,

Table 1. Test and recovery accuracy of different baselines on the CIFAR-10 dataset with symmetric noise. We report the average score of 5 trials.

#	method		Test Accuracy (%)						Recovery Accuracy (%)					
	noise rate (%)		0	10	30	50	70	90	0	10	30	50	70	90
1	Cross Entropy Loss	best	93.5	91.0	88.4	85.0	78.4	41.1	**100.0**	96.4	92.7	88.2	80.1	41.4
		last	93.4	87.0	72.2	55.3	36.6	20.4	**100.0**	91.1	74.6	57.6	39.6	21.7
2	Our Method	best	93.4	92.7	91.4	89.6	85.9	58.0	**100.0**	97.9	**95.1**	91.7	86.3	58.2
		last	**93.6**	**92.9**	**91.5**	**89.8**	**86.0**	**58.3**	99.9	**98.1**	**95.1**	**91.8**	**86.4**	**58.3**

Table 2. Test and recovery accuracy of different baselines on the CIFAR-10 dataset with asymmetric noise. We report the average score of 5 trials. #2, #3 are the results by our implementation.

#	method		Test Accuracy (%)					Recovery Accuracy (%)				
	noise rate (%)		10	20	30	40	50	10	20	30	40	50
1	Cross Entropy Loss	best	91.8	90.8	90.0	87.1	77.3	97.2	95.8	94.3	91.0	80.5
		last	89.8	85.4	81.0	75.7	70.5	95.0	90.2	85.3	80.2	75.2
2	Forward [14]	best	92.4	91.4	91.0	90.3	83.8	97.7	96.7	95.9	94.7	88.0
		last	91.7	89.7	88.0	86.4	80.9	97.9	95.8	93.6	91.5	85.5
3	CNN-CRF [19]	best	92.0	91.5	90.7	89.5	84.0	97.4	96.5	95.3	93.7	88.1
		last	90.3	86.6	83.6	79.7	76.4	95.1	90.5	86.4	82.1	78.7
4	Our Method	best	**93.2**	92.7	**92.4**	91.5	84.6	**98.3**	**97.2**	**96.3**	**95.2**	**88.3**
		last	**93.2**	**92.8**	**92.4**	**91.7**	**84.7**	98.1	97.1	**96.3**	**95.2**	88.1

respectively for all the noise rates. As a comparison, we also performed training on initial noisy labels in the same manner as the second step of our method with the cross entropy loss or the forward corrected loss [14].

The results of our experiments are shown in Table 2. The forward corrected loss [14] and the CNN-CRF model [19] require the ground-truth noise transition matrix. Conversely, we need only the prior distribution p, and thus our condition is more general than that of [14, 19].

Our method achieves significantly better test accuracy and recovery accuracy on AN-CIFAR. However, only when the noise rate is 50%, there is no significant improvement in accuracy when compared with other noise rates. Since we generated label noise to exchange CAT and DOG classes, it is impossible to accurately determine the class for CAT and DOG when the noise rate is 50%.

In a manner similar to Section 5.5, when training is performed on initial noisy labels, the test accuracy decreases due to the network fitting noisy labels with a low learning rate. This trend is also observed if we use the forward corrected loss [14], while the test accuracy does not decrease and remains high in our method.

5.7. Experiment on PL-CIFAR

To evaluate the performance of our method in the settings of transfer learning, we trained the network on PL-CIFAR by using our method. In the first step of our method, we used a learning rate of 0.04 and used 1.2 for α and 0.8 for β. As a comparison, we also trained on initial pseudo-labels in the same manner as the second step of our method.

Fig. 5 shows the test accuracy curve with different labels, and Fig. 6 shows the decline in the training loss during training. In both figures, we show the results of training on SN-CIFAR (the noise rate $r = 0.3, 0.5$) because the noise rate of the pseudo labels is between 0.3 and 0.5. Additionally, we show the results of training on the ground-truth labels because the training loss curve of training on optimized labels is near the curve for the same.

Although the number of inaccurate labels in the pseudo labels exceeds that of the symmetric noise labels ($r = 0.3$), the value of the training loss of the pseudo labels is lower than that of the symmetric noise labels. This fact seems to conflict with extant knowledge that states that "the training loss increases when the noise rate increases", as described in Section 5.3. However, we can explain the reason of this conflict as follows: the difference in the training loss depends on the noise rate as well as the type of the noise. The pseudo labels are generated from the outputs of ResNet-50 pre-trained on ImageNet, and thus they are already considered as "the optimized labels" by the network. Thus, the pseudo labels were not updated adequately. The test accuracy of training on the labels recovered from the noisy labels is worse than that of training on the ground-truth labels, and this indicates that the optimized labels do not necessarily denote optimal labels. This is a limitation of the proposed method.

5.8. Experiment on the Clothing1M dataset

Finally, we trained the network on the Clothing1M dataset [21] by using our method to evaluate the per-
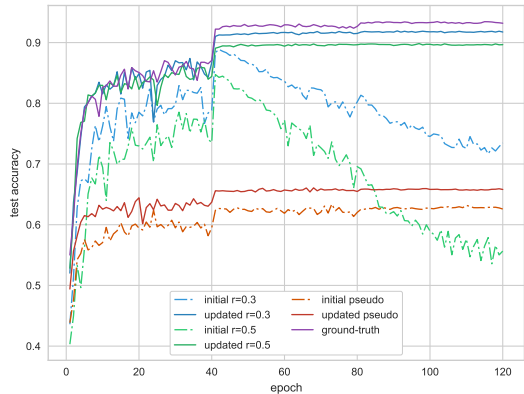
Figure 5. The test accuracy curve with different labels. The test accuracy on the pseudo labels is lower than that on the symmetric noise labels even if the number of inaccurate labels is lower. This trend remains if the labels are updated.
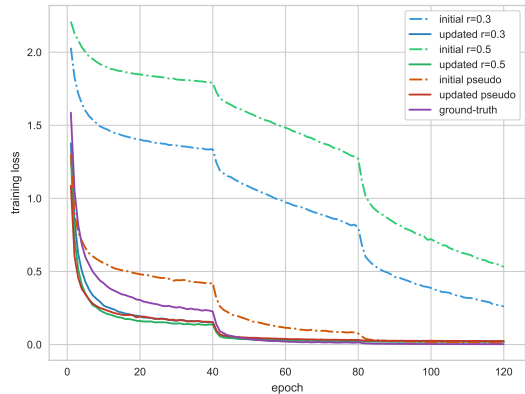

Figure 6. The training loss curve with different labels. The training loss on the pseudo labels is lower than that on the symmetric noise labels even if the number of inaccurate labels is higher. Each of the training losses on the labels updated from different noisy labels follows the training loss on the ground-truth labels. This implies that the updated labels are completely optimized for the network.

formance of our method in a real setting. As a comparison, we also trained on initial noisy labels in the same manner as the second step of our method. The results of our experiments are shown in Table 3. Additionally, we also show the scores (#1, #2) reported in [14].

In #2, Patrini *et al.* exploited the curated labels of 50k clean data and their noisy versions in 1M noisy data to obtain the ground-truth noise transition matrix, which is not often used in real-world settings. Conversely, we only used the distribution of the noisy labels, which can be always used, for the prior distribution p, and therefore our condition is more general than #2. Nevertheless, our method achieves better test accuracy than #2 on the Clothing1M dataset.

In Fig. 7, Fig. 8, we show the examples of the images whose labels were reassigned to classes different from the original ones by our method. Additionally, we show the probability of the class that the label is reassigned to. When the probability is high, the label seems to be updated correctly. Conversely, when the probability is low, the label seems to be updated incorrectly. As opposed to the hard-labels, the soft-labels contain the probabilities of each class in themselves, and thus the network can consider the incorrectly updated labels as not important. Specifically, this effect contributes to improving the test accuracy.

6. Conclusion

We proposed a joint optimization framework for learning on noisy labeled datasets, which alternatively updates network parameters and class labels. The performance of the framework is guaranteed by our finding that training under a high learning rate prevents the network from memorizing noisy labels. We showed that our framework performed remarkably well on the

Table 3. Test accuracy of different baselines on the Clothing1M dataset. #1 and #2 are quoted from [14], and #3 is the reproduced result by our reimplementation.

#	method		accuracy
1	Cross Entropy Loss		68.94
2	Forward [14]		69.84
3	Cross Entropy Loss	*best*	69.15
	(reproduced)	*last*	66.76
4	Our Method	*best*	72.16
		last	**72.23**

Figure 7. The images with the top-2 and the bottom-2 probabilities of T-shirt whose labels are reassigned from Hoodie to T-shirt.

Figure 8. The images with the top-2 and the bottom-2 probabilities of Hoodie whose labels are reassigned from T-shirt to Hoodie.

noisy CIFAR-10 dataset and the Clothing1M dataset, outperforming the state-of-the-art methods [14, 19].

References

[1] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al. A closer look at memorization in deep networks. In *ICML*, 2017.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM*, 2007.

[3] J. P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations research*, 2011.

[4] A. Ghosh, H. Kumar, and P. Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, 2017.

[5] A. Ghosh, N. Manwani, and P. Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 2015.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[7] G. R. Haffari and A. Sarkar. Analysis of semi-supervised learning with the yarowsky algorithm. In *UAI*, 2007.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[10] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning discrete representations via information maximizing self augmented training. In *PMLR*, 2017.

[11] I. Jindal, M. Nokleby, and X. Chen. Learning deep networks from noisy labels with dropout regularization. In *ICDM*, 2016.

[12] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Master's thesis, Technical report, University of Tronto, 2009.

[13] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML*, 2013.

[14] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu. Making neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017.

[15] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*, 2015.

[16] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. In *ICLR*, 2015.

[17] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *Communications of the ACM*, 2016.

[18] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *NIPS*, 2015.

[19] A. Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, 2017.

[20] B. Van Rooyen, A. Menon, and R. C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In *NIPS*, 2015.

[21] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[22] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[23] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[24] X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Science, University of Wisconsin-Madison, 2006.