2023 17th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS) | 979-8-3503-7091-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/SITIS61268.2023.00024

iNLC: Iterative Noisy Label Correction

Seungyeon Koo
Department of Electronic Engineering
Hanyang University
Seoul, Korea
rrxloyeon@hanyang.ac.kr

Sangki Park
Department of Electronic Engineering
Hanyang University
Seoul, Korea
skpark1101@hanyang.ac.kr

Si-Dong Roh
Department of Electronic Engineering
Hanyang University
Seoul, Korea
sdroh1027@hanyang.ac.kr

Ki-Seok Chung*
Department of Electronic Engineering
Hanyang University
Seoul, Korea
kchung@hanyang.ac.kr

***Abstract*—Convolutional neural networks have achieved remarkable success in image classification, but the presence of noisy labels in the training dataset can significantly hinder their achievement. Creating clean labeled datasets is time-consuming; therefore, learning with noisy datasets is a practical approach to solving real-world problems. In this paper, we propose a novel method called iterative Noisy Label Correction (iNLC) that employs gradual data refining to mitigate the impact of incorrect labels in practice. iNLC consists of three main components: robust prior learning, ensemble strategy, and gradual data refining. The robust prior learning trains the prior model via semi-supervised learning to make it more robust against noise and facilitate subsequent gradual refinement. The ensemble strategy improves performance by combining different augmentation strategies, and the gradual data refining process progressively incorporates additional data into the training using fine-grained learning schedules based on data volume in order to prevent overfitting and underfitting. Our method achieved the classification accuracy of 93.98%, 92.96%, and 75.32% for noise ratios of 0.2, 0.4, and 0.8, respectively, on PreAct-ResNet-18 on CIFAR-10.**

***Index Terms*—learning with noisy label, semi-supervised learning, ensemble**

I. Introduction

Convolutional neural networks (CNNs) have achieved remarkable success in various tasks such as image classification, object detection, and semantic segmentation. In training these models, however, the quality of the dataset plays a crucial role in achieving high accuracy. In general, datasets are human-annotated; therefore, if care is not taken when creating the dataset, incorrectly annotated labels (i.e., noisy labels) may be included, and this noise harms the performance of network [1]–[3]. Furthermore, labeling without noisy data is time-consuming and practically infeasible. To get a massive amount of data easily, some datasets are created by crawling images and texts from the web, such as WebVision [4], Clothing 1M [5], and social media images with hash tags [6]. Unfortunately, in these cases, a significant amount of noisy data may be mixed in. To solve these noisy data problems, several types of research have been conducted to filter out noisy labels to reduce the harmfulness of incorrect labels.

According to Arpit et al. [7], neural networks tend to learn mostly on easy samples in the training dataset. In general, data without noisy labels (i.e., clean data) turn out to be easier samples than those with noisy labels (i.e., noisy data): the loss converges faster for clean data than for noisy data. Based on this observation, [8] and [9] suggested methods to exclude noisy labels from training so that a model is trained only with clean data samples. However, the classification accuracy of these methods is limited because of the information loss due to excluding a significant portion of data. Therefore, a method called learning with noisy labels (LNL) that employs semi-supervised learning (SSL) has been studied as an alternative. SSL-based LNL trains models by picking out noisy data, removing their labels, and converting them to clean data in order to include them in the training process. In this paper, we propose a new method called iterative Noisy Label Correction (iNLC) that can be used as an add-on to the SSL-based LNL method. The proposed iNLC method consists of the following three parts:

- **Robust Prior Learning**: To train the prior model, we combine a noisy label detector and an SSL method to make the model more robust to noisy labels. We use the second stage of O2U-Net [9] for noisy label detection and FixMatch [10] for SSL. The details are described in Section III-A.
- **Ensemble**: The accuracy of the network depends on the augmentation strategy. To mitigate the influence of augmentation strategies and ensure stable learning, we apply ensemble techniques to models trained using various types of augmentation methods. In each generation, the network is trained with three different augmentation strategies. The augmentation strategies are listed in Table I.
- **Gradual Data Refining**: Gradual data refining alleviates the deterioration of deep features caused by noisy data.

979-8-3503-7091-1/23/$31.00 ©2023 IEEE
DOI 10.1109/SITIS61268.2023.00024

We iteratively refine the data to increase the amount of training data progressively. In addition, we also apply epoch scheduling to prevent overfitting or underfitting due to changes in data size. The details are described in Section III-B3.

II. Related Work

A. Learning with Noisy Labels

Zhang et al. [2] reported that when CNNs were trained with noisy data, their performance would decrease significantly because they tended to overfit to incorrectly labeled data. Therefore, studies for LNL have suggested various ways to design the loss function to exclude noisy data or to make the model more robust against noisy labels [8], [9], [11], [12]. Reed et al. [11] proposed a perceptual consistency objective to ensure that similar percepts should make the same prediction. Their objective is to replace the target of cross entropy with a convex combination of the noisy training label and the current prediction of the model, so that correction can be made even if the noise distribution is hidden. However, the impact of the noisy label cannot be ignored, as the noisy label is required to calculate the bootstrapping loss. The bootstrapping loss is divided into soft loss and hard loss. The soft loss uses the current prediction of the convex combination with softmax, and the hard loss uses the result of argmax. Patrini et al. [12] proposed a loss correction approach that predicted a noise transition matrix that contained information about the probability that a given label was contaminated with other classes. This method learns by weighting the loss with the noise transition matrix. Huang et al. [9] proposed O2U-Net, a noisy label detection method that can be easily implemented using learning rate scheduling with a few iterations instead of complex losses. O2U-Net proceeds in three stages. In the first stage, a model is trained until the validation accuracy converges. In the second stage, it performs short-period (e.g., ten epochs) cyclical training to avoid overfitting and learns robustly to noisy labels. In the last stage, it calculates the average loss based on the cross entropy loss in the second stage to exclude the top-k% data and pass the remaining clean data to the final training network. However, the pretraining process of O2U-Net is not robust as it learns with noisy data, which limits the performance of noisy label detection in the subsequent cyclic training phase. Therefore, we replaces the pre-training phase of O2U-Net in the iNLC with a semi-supervised learning approach.

B. Semi-Supervised Learning

Semi-supervised learning is a method of training a network using both labeled and unlabeled data to leverage the advantages of both supervised and unsupervised learning. It generally updates the network by defining a new loss function for unlabeled data and combining it with the loss function for labeled data [10], [13], [14]. Berthelot et al. [13] proposed a consistency regularization method called Mixmatch. The labeled data loss of MixMatch is the widely used cross-entropy loss, and the unlabeled loss is designed to ensure the consistency of predictions for transformed inputs. Evolving from the previous approach, ReMixMatch [14] targets logits estimated with weakly augmented input and learns them to maintain prediction consistency of weakly augmented inputs and strongly augmented inputs. FixMatch [10] selectively trains the consistency of augmentations that have logits above a certain threshold in order to boost unsupervised learning. Schick et al. [15] used iterative ensembles to perform semi-supervised learning on a cloze question task. Inspired by [15], we verified that iterative ensembles used in the natural language process domain are also effective for LNL tasks in the image domain.

III. Proposed Method

In this section, we illustrate the process of iNLC and how to apply it to existing semi-supervised learning-based LNL methods. Steps are described in two parts: robust prior model learning as in a semi-supervised learning approach, which serves as a base model of iNLC, and the iNLC algorithm itself.

A. Robust Prior Learning

Training a robust prior model is crucial as it serves as the initial model during the subsequent gradual data refinement. The initial data refining process employs a noisy label detection approach to distinguish clean data from the initial noisy dataset $\mathcal{D}$. It utilizes clean data as labeled data for semi-supervised learning to perform the prior learning. Robust prior learning proceeds as follows:

1) Initial data refining: We divide $\mathcal{D}$ into a labeled dataset $\mathcal{X}$ and an unlabeled dataset $\mathcal{U}$. We assume the dataset $\mathcal{D}$ contains symmetric noise with a noise ratio r and contains rN noisy labels out of $n(D) = N$ data samples. We define remain ratio p as the proportion of data that will remain labeled and forget ratio $f = 1 - p$ as the proportion of data that will not be labeled. Because the noise ratio is unknown in the real world, we set the remain ratio p to be small. We denote $\mathcal{X} = ((x_n, y_n); n \in (1, ..., pN))$ as the labeled dataset, and $\mathcal{U} = ((u_n); n \in (1, ..., fN))$ as the unlabeled dataset when x_n and u_n are data, and y_n is label. To divide $\mathcal{D}$ into $\mathcal{X}$ and $\mathcal{U}$, we use the noisy label detection approach proposed in [9]. According to [9], noisy data can be excluded when cyclical training between overfitting and underfitting by tuning the learning rate is repeatedly carried out. To facilitate the iterative process of overfitting and underfitting, the learning rate lr at epoch t is changed according to the following equation:

$$\begin{aligned} \text{when } v(t) &= \frac{(1 + ((t-1) \bmod T))}{T}, \\ lr(t) &= (1 - v(t)) \times lr(0) + v(t) \times \frac{lr(0)}{10} \end{aligned} \tag{1}$$

where $lr(0)$ is the initial learning rate and $v(t)$ is the intermediate term that causes the learning rate to oscillate with a period T.

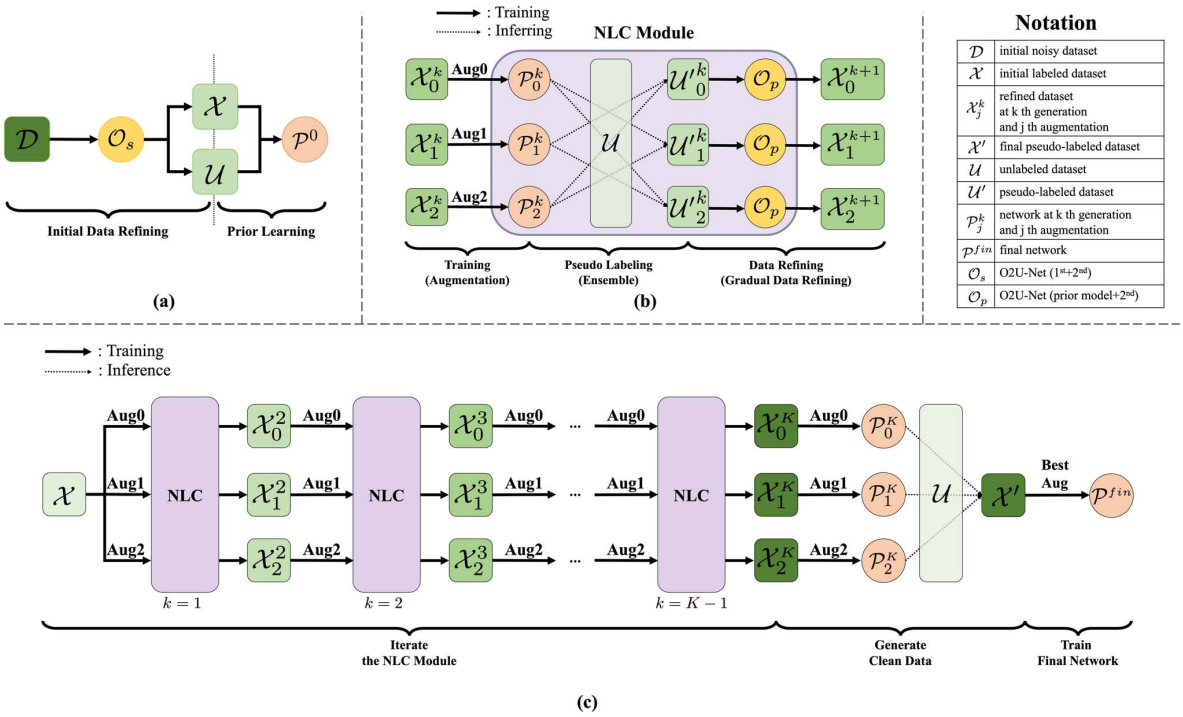

Fig. 1. Overview of iNLC. (a) Robust prior learning process. The O2U-Net noise detector divides the noisy dataset $\mathcal{D}$ into labeled dataset $\mathcal{X}$ and unlabeled dataset $\mathcal{U}$. (b) The NLC module. The proposed iNLC method repeats this process $K-1$ times. (c) The overall iNLC process after the prior learning. After all the iterations are finished, $\mathcal{P}^K$ determines a soft label for every unlabeled datum. In the final step, we use this cleanly labeled dataset to train the final model.

The network trained with Eq. 1 prioritizes learning using clean data, leading to an increased loss of noisy data [9]. Assuming dataset $\mathcal{D}$ has C classes, we compute the loss for the samples predicted as the l^{th} class and select $c_0(l) = pN/C$ samples of clean data where p is the remain ratio. Subsequently, we combine $c_0(l)$ samples for all classes to get the clean dataset $\mathcal{X}$.

2) Prior learning: We train a prior model $\mathcal{P}^0$ with $\mathcal{X}$ and $\mathcal{U}$ as in a semi-supervised learning (SSL) approach. Compared to other SSL methods [13], [14], FixMatch [10] presented a simpler yet more accurate way to learn unlabeled data. Therefore, we used FixMatch as the SSL method for the prior learning process.

B. Iterative Noisy Label Correction

We propose a gradual data refining technique inspired by Schick et al. [15] to carefully assign labels to data for which correction is needed using strong augmentation strategies. This section discusses the details of the module for noisy label correction (NLC module). The proposed NLC method proceeds in three steps: 1) augmentation, 2) ensemble, and 3) data refining. In the initial augmentation step, models are trained differently using three types of strong augmentations. Then, in the ensemble step, two of the three trained models are combined to assign pseudo-labels to the unlabeled data. This results in a total of three pseudo-labeled datasets $\mathcal{U}'^k_j$. Lastly, the noisy label detection approach is employed to filter and select a part of pseudo-labeled datasets as correctly assigned labels, which are then utilized as the training data $\mathcal{X}^{k+1}$ for the next generation. By repeating this process, iNLC gradually corrects noisy labels, as shown in Fig. 1 and Algorithm 1.

TABLE I
AUGMENTATION STRATEGIES

i	strategies
0	RandomCrop, RandomHorizontalFlip, **RandAugment**
1	RandomCrop, RandomRotation, **RandCutout**
2	RandomCrop, RandomHorizontalFlip, **RandomGrayScale**

1) Augmentation: Fig. 1 (b) shows the structure of the NLC module. The NLC module uses an ensemble with models that are trained with different types of strong augmentations. The performance of a CNN network is heavily dependent on the types of augmentation strategies. Therefore, we use strong transformations as an augmentation strategy as shown in Table I; RandAugment [16], RandCutOut [17], RandGrayScale. RandGrayScale can be used as a strong strategy because CNNs tend to learn with a bias towards the texture of an image [18].

2) Ensemble: Because the performance of CNN depends on the types of augmentation strategies, inappropriate augmentations could lead to a high noise ratio in pseudo-labeling. To alleviate this problem, we select the top c_k samples with the smallest loss and use them as the refined dataset for the next generation. Two models form an ensemble weighted by respective scores of the two models. The score s_j^k of a model $\mathcal{P}_j^k$ at the k^{th} generation with the j^{th} augmentation function $Augment_j$ is defined as the accuracy of the prior model evaluated with dataset $\mathcal{X}_j^k$:

$$s_j^k = accuracy(y, \mathcal{P}_j^k(x)), (x, y) \in Augment_j(\mathcal{X}_j^k). \quad (2)$$

The final model $\mathcal{P}^{fin}$ uses a weighted ensemble of the three models to generate a soft label for the unlabeled dataset and then updates the parameter with distillation loss [19] using only the clean dataset.

3) Gradual Data Refining: As training with noisy data in the dataset would be worse than training with an insufficient amount of data, the clean data should be distinguished from the noisy data. However, in practice, the quantity of clean data in D is unknown. Thus, after the initial amount of clean data is set conservatively, a certain amount of noisy labels is corrected. Then, the expanded clean data is utilized to create a better classification model gradually. To increase the data involved in training while reducing the noise ratio in $\mathcal{X}^k$, the size of the refined dataset $\mathcal{X}^k$ is progressively increased by a scale factor d per generation. In each generation, $\mathcal{O}_p$ samples c_k data from the $\mathcal{U}'^k$, then sampled data are merged with the $\mathcal{X}$. The number of sampled data of l^{th} class ($c_k(l)$) and the size of the refined dataset ($n(\mathcal{X}^k)$) are determined by the following equations, respectively:

$$\begin{aligned} c_k(l) &= (d^{k-1} - 1)c_0(l), \\ n(\mathcal{X}^k) &= \sum_{l=1}^{C} \{c_k(l) + c_0(l)\}. \end{aligned} \quad (3)$$

The total count of generations (K) is determined by a scale factor d and the conservatively determined remain ratio p as follows:

$$K = \lceil \frac{-log(p)}{log(d)} \rceil. \quad (4)$$

The epochs are scheduled in proportion to the epochs of the final model (e_{fin}) since a fixed epoch scheduling in all generations may lead to overfitting in earlier generations or underfitting in later generations. By employing a variable epoch scheduling, we can not only achieve performance improvements but also reduce a significant amount of training time by eliminating redundant learning processes in the early generations. The size of the epoch for the k^{th} generation (e_k) is determined as follows:

$$e_k = e_{fin} \cdot \frac{\sum_{l=0}^{C} \{c_0(l) + c_k(l)\}}{N}. \quad (5)$$

The sampling process for data is as follows. A pseudo-labeled dataset $\mathcal{U}'^k_j$ is created by ensembling the predictions of the two models on the unlabeled dataset. Next, the dataset is sorted in ascending order of loss values. Finally, the top-c_k samples are selected. Although the O2U-Net noise detector selects the cleanest examples, some bad examples may still be mixed in. So, we use soft-labeled loss [20]. We create $\mathcal{X}'$ by adding a clean label to the unlabeled dataset to train the final model. The clean label is the weighted ensemble of the three trained models $\mathcal{P}_j^K (j = 0, 1, 2)$ with the score s_j^K, and the best augmentation strategy is used to train $\mathcal{P}^{fin}$.

Algorithm 1: Iterative Noise Label Correction

Input: prior learned network $\mathcal{P}^0$, initial labeled dataset $\mathcal{X} = ((x_n, y_n);\ n \in (1, ..., pN))$, unlabeled dataset $\mathcal{U} = ((u_n);\ n \in (1, ..., fN))$, noise detector $\mathcal{O}_p$, augmentation strategies ($Augment_j;\ j \in (0, 1, 2)$), scale factor d, total generations K, the number of pseudo-labeled data c_k

Output: refined noisy dataset $\mathcal{X}'$

/ Iterate the generation */*

$\mathcal{X}^1 = \mathcal{X}$

for $k = 1$ ***to*** K **do**

  / Train refined dataset */*

  for *augment id* $j \in \{0, 1, 2\}$ **do**

    Initialize $\mathcal{P}_j^k$ with $\mathcal{P}^0$

    $\hat{\mathcal{X}}_j^k = Augment_j(\mathcal{X}_j^k)$

    Train $\mathcal{P}_j^k(\hat{\mathcal{X}}_j^k)$

    Get Score s_j^k

  end

  / Refine $\mathcal{U}$ by above trained model */*

  for *augment id* $j \in \{0, 1, 2\}$ **do**

    Assign $a, b \neq j$

    $y_{u,j} = argmax\{(s_a^k \mathcal{P}_a^k(\mathcal{U}) + s_b^k \mathcal{P}_b^k(\mathcal{U}))/(s_a^k + s_b^k)\}$

    $\mathcal{U}'_j = ((u_n, y_{u,j,n});\ n \in (1, ..., fN))$

    $mask = \mathcal{O}_p(Concat(\mathcal{X}, \mathcal{U}'_j), c_{k+1})$

    $\mathcal{X}_j^{k+1} = \mathcal{X} + mask(\mathcal{U})$

  end

end

$y_u = \frac{\sum_{j=0}^{2} s_j^K \mathcal{P}_j^K(\mathcal{U})}{\sum_{j=0}^{2} s_j^K}$

$\mathcal{X}' = ((u_n, y_{u,n});\ n \in (1, ..., fN))$

return $\mathcal{X}'$

IV. Experimental Results

A. Implementation Details

We injected symmetric noisy labels into the CIFAR-10 training set and used PreAct ResNet-18 as the CNN architecture. In augmentations, we used $n = 2$ for RandAug; the probability of image transformation is 0.1 for RandGrayScale; and 0.5 for

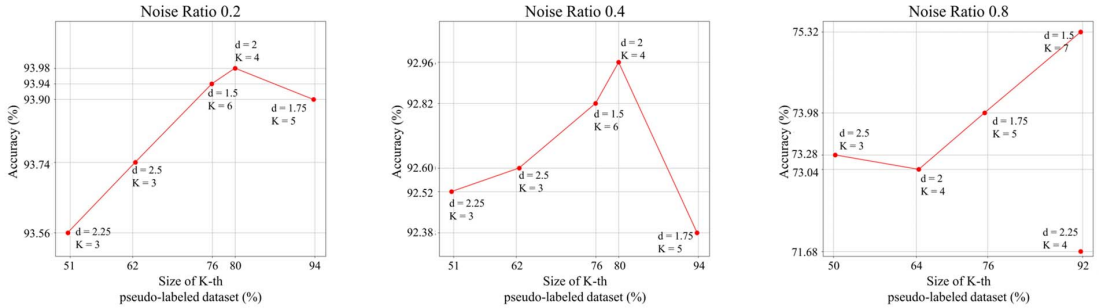

Fig. 2. Sensitivity analysis of $n(\mathcal{X}^K)$. To find an appropriate scale factor, we varied d from 1.5 to 2.5 in increments of 0.25 and measured the test accuracy. The results revealed that accuracy is influenced more by $n(\mathcal{X}^K)$ than by d. To illustrate this, we plotted a graph with the x-axis representing the percentage of $n(\mathcal{X}^K)$ upon $n(\mathcal{D})$ and the y-axis representing test accuracy. Since $n(\mathcal{X}^K)$ is dependent on d, we indicated the corresponding d for each point.

TABLE II
HYPERPARAMETERS

Hyperparameters	Prior Learning			iNLC	
	O2U(1st)	O2U(2nd)	FixMatch	$\mathcal{P}^k$	$\mathcal{P}^{fin}$
epochs	100	20	100	scheduling	200
batch size	64	64	64	64	64
optimizer	SGD	SGD	SGD	SGD	SGD
LRscheduler	constant	O2U	Cos. Anneal.	Cos. Anneal.	Cos. Anneal.
Initial LR	0.01	0.01	0.01	0.01	0.01
SGD momentum	0.9	0.9	0.9	0.9	0.9
SGD weight decay	5e-4	5e-4	5e-4	5e-4	5e-4

RandCutout. We used $\mu = 7$ and $\lambda_u = 1$ for FixMatch, and $T = 10$ in Eq. 1. We set the remain ratio p to 0.1. However, in the case of a noise ratio of 0.8, we set $p = 0.0805$ due to the limitation of the O2U-Net that gathering more than 5,000 clean data is infeasible. The scale factor d is 2 when the noise ratio is 0.2 or 0.4, and $d = 1.5$ when the noise ratio is 0.8. Other hyperparameters are shown in Table II. For the gradual data refining process, the O2U-Net used the same hyperparameters as those used in the second stage of the prior learning process. All experiments were conducted on a single NVIDIA RTX 3090. Under these experimental settings, the iNLC process requires 2 hours of training, and the robust prior learning step takes 9 hours.

B. The Effect of the Scale Factor on Performance

The scale factor d determines the size of the refined dataset $n(\mathcal{X}^K)$. As illustrated in Fig. 2, small values of $n(\mathcal{X}^K)$ tend to show low accuracy. $n(\mathcal{X}^K)$ in the final generation tends to increase as the value of d decreases, and K may also increase, leading to longer training times. Because the larger amount of data in the final generation does not guarantee better performance, we choose $d = 2$ for 0.2, 0.4 noise ratios, and $d = 1.5$ for 0.8 to ensure the model can utilize a suitable quantity of dataset.

C. Main Results

Table III shows the accuracy comparison results of iNLC and other well-known LNL methods. RN and PRN mean ResNet and PreAct-ResNet, respectively. Under all three noise ratio conditions, iNLC outperformed the others by achieving the best accuracy of 93.98% at a 0.2 noise ratio. Despite being a CNN network with fewer layers, the test accuracy was significantly better than O2U-Net [9]. Especially when the noise ratio is set to 0.8, iNLC achieved an accuracy of 75.32%, which is 37.56%p higher than O2U-Net. We also showed that adding iNLC to the prior model can improve the accuracy by ranging from 2.23%p to 45.17%p compared to the case where the prior model is used alone. Our proposed method outperforms all the compared LNL methods at a noise ratio of 0.8, mainly due to our excellent label correction during the gradual data refining step. The performance evaluation of gradual data refining is described in the next section. Our approach showed 37.56%p better performance than the compared noisy label removal method [9] and up to 59.73%p better performance than the compared loss correction method [11].

TABLE III
TEST ACCURACY ON CIFAR-10 (%)

Method	Architecture	CIFAR-10		
		0.2	0.4	0.8
Reed et al. (soft) [11]	RN-101	83.20	69.91	18.12
Reed et al. (hard) [11]	RN-101	84.88	68.90	15.59
Patrini et al. [12]	RN-32	87.90	-	-
Han et al. [8]	9-layer CNN	82.32	-	-
Huang et al. [9]	RN-101	92.57	90.33	37.76
Prior Model	PRN-18	91.00	90.73	30.15
iNLC (Ours)	PRN-18	**93.98**	**92.96**	**75.32**

TABLE IV
ABLATION STUDY

Method	Accuracy (%)
iNLC	**93.98**
iNLC w/o gradual data refining ($K = 1$)	92.78
iNLC w/o ensemble	93.19
iNLC w/o fixmatch	89.36
iNLC w/o epoch scheduling	93.13

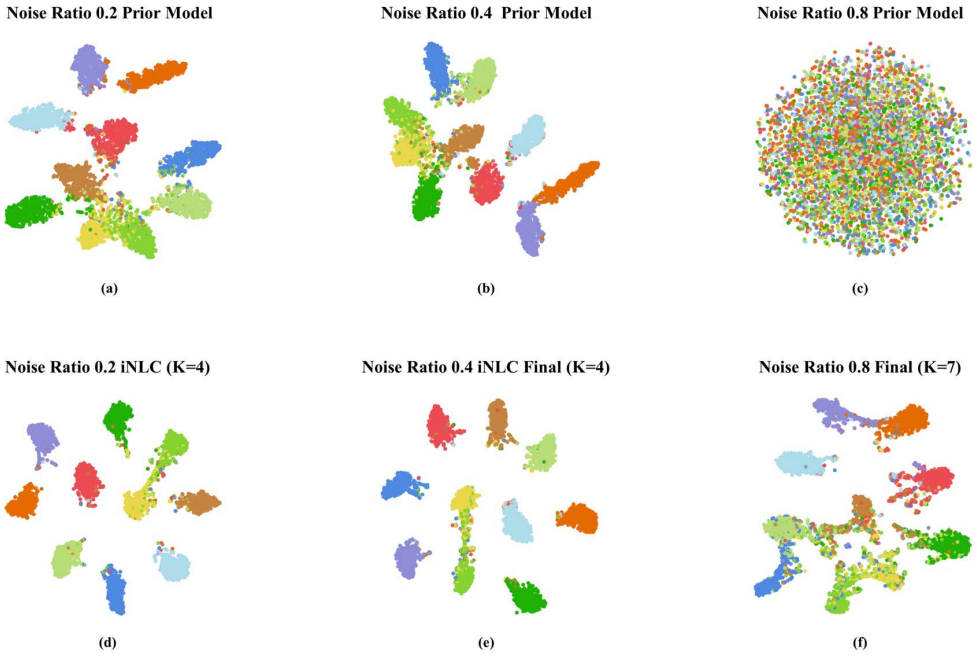

Fig. 3. t-SNE visualization of deep features. All t-SNE visualizations utilize a randomly sampled subset of 5000 data points from the test dataset. (a-c) represents the deep features of the prior models with each noise ratio setting. (d-f) portrays the $\mathcal{P}^{fin}$ with iNLC.

D. Ablation Study

Table IV shows how much each step of the iNLC method affects the performance. All the performance results were evaluated under a symmetric noise environment with a noise ratio 0.2. To observe if there was a progressive data-cleaning effect, we fixed the total number of generations to 1 and measured the accuracy. The result was 92.78% which is 1.2%p lower than the iNLC. Next, to observe the effect of the ensemble, we redesigned it to generate pseudo-labels on its own using only one augmentation strategy, and it achieved an accuracy of 93.19%, which is 0.79%p lower than the iNLC. When the FixMatch prior learning was not applied, the accuracy result was 89.36%, which is 4.62%p lower than the iNLC. However, as shown in Table III, it can be used without prior learning. Finally, we trained for 75 epochs across all generations to validate the epoch scheduling and found that a 0.85%p reduction was observed over the epoch scheduling. Thus, all the applied components of iNLC are confirmed to be positively contributing to achieving better performance.

To confirm that gradually increasing the amount of data is adequate for noise correction, we compared it to the noise ratio of $\mathcal{X}'$ when only one generation was performed (Table V). The results show that gradually increasing the amount of data reduces the noise ratio of $\mathcal{X}'$ by ranging from 0.53%p to 6.73%p. Specifically, when the noise ratio of $\mathcal{D}$ is 0.8, the decrease is 6.73%p, implying that retraining the prior model with gradual data refinement is more essential.

To confirm the validity of Eq. 4, we conducted experiments

TABLE V
NOISE RATIO OF REFINED DATASET $\mathcal{X}'$ ON REFINEMENT ITERATIONS K (%)

noise ratio in D	$K=1$	$K=4$	Increment
0.2	8.96	8.43	-0.53%p
0.4	11.78	9.78	-2.00%p
0.8	36.02	29.29	-6.73%p

TABLE VI
TEST ACCURACY (%) ACCORDING TO K FOR EACH NOISE RATIO

K	0.2 ($d=2$)	0.4 ($d=2$)
0	84.34	66.34
1	92.78	89.68
2	93.70	91.18
3	93.58	92.28
4	**93.98**	**92.96**
5	93.84	92.70

of arbitrarily selecting values for K without using Eq. 4 to set the value for K. As summarized in Table VI, if K is smaller than the value determined by Eq. 4, it decreases accuracy due to insufficient data utilization. On the other hand, if K becomes larger than the value determined by Eq. 4, it leads to data duplication, resulting in accuracy reduction due to overfitting.

As shown in Fig. 3 and 4, we represented the 512-dimensional deep features of the test dataset in the 2-

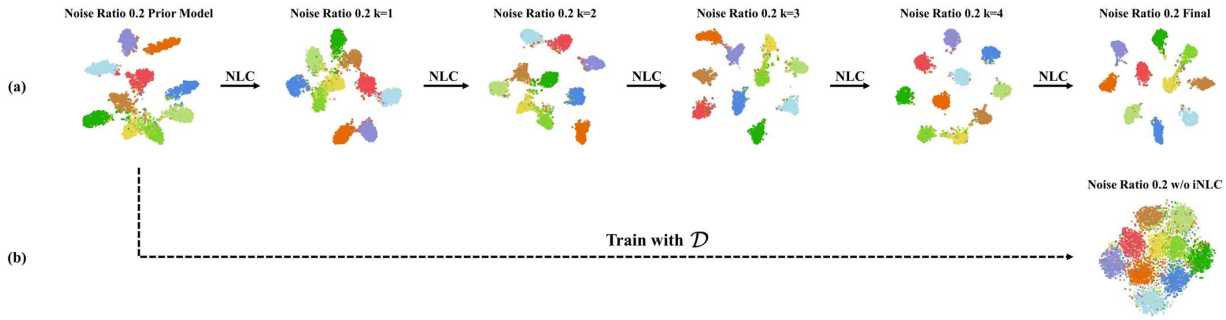

Fig. 4. Comparison between scenarios with and without gradual data refining. (a) presents the t-SNE visualization of deep features for the prior model trained using the FixMatch approach during the prior learning phase, the best-scored model among the three models trained in each generation of iNLC, and the final model trained on clean data. On the other hand, (b) illustrates the results of training directly on the noisy dataset without gradual data refining, serving as a comparison to (a).

dimensional space using t-SNE [21] to demonstrate the effectiveness of iNLC. In Fig. 3, the t-SNE results of $\mathcal{P}^{fin}$ and those of the prior model are compared. Without applying iNLC (Fig. 3 (a-c)), there are areas of overlap between the distributions of different classes. In contrast, when iNLC is applied (Fig. 3 (d-f)), deep features of different classes form more concentrated clusters.

Fig. 4 (a) illustrates the gradual change of deep features as k increases and (b) shows deep features with additional training with noisy dataset $\mathcal{D}$ (no gradual data refining). When comparing Fig. 4 (a) and (b), the data distribution of deep features tends to be more ambiguous than that of iNLC. Therefore, it is confirmed that starting model training with a small amount of the labeled dataset and gradually increasing the amount of the labeled dataset make the model more robust.

V. Conclusion

Training convolutional neural networks using real-world data is challenging mainly due to noisy labels. We propose a new learning with noisy labels (LNL) method called iterative Noise Label Correction (iNLC) that leverages gradual data refining to mitigate the negative impact of incorrect labels in practice. The iNLC method is composed of three key components: robust prior learning, ensemble strategy, and gradual data refining. We compared the accuracy of the proposed iNLC with those of several widely-used LNL methods under a symmetric noise condition. The experimental results showed that iNLC outperformed the compared LNL methods by achieving a 93.98% accuracy on CIFAR-10. We also compared the performance of iNLC with those of the LNLs with semi-supervised learning alone and confirmed that adding iNLC improves the performance by up to 45.17%p. The ablation study showed that each component of iNLC positively contributed to performance improvement.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology)

References

[1] C. Zhang, S. Bengio, M. Hardt, M. C. Mozer, and Y. Singer, "Identity crisis: Memorization and generalization under extreme overparameterization," *arXiv preprint arXiv:1902.04698*, 2019.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[3] Q. Yao, H. Yang, B. Han, G. Niu, and J. T.-Y. Kwok, "Searching to exploit memorization effect in learning with noisy labels," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 789–10 798.

[4] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[5] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[6] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 181–196.

[7] D. Arpit, S. Jastrzebski, 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 *International conference on machine learning*. PMLR, 2017, pp. 233–242.

[8] 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," *Advances in neural information processing systems*, vol. 31, 2018.

[9] J. Huang, L. Qu, R. Jia, and B. Zhao, "O2u-net: A simple noisy label detection approach for deep neural networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3326–3334.

[10] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in neural information processing systems*, vol. 33, pp. 596–608, 2020.

[11] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *arXiv preprint arXiv:1412.6596*, 2014.

[12] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1944–1952.
[13] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.
[14] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *International Conference on Learning Representations*, 2020.
[15] T. Schick and H. Schütze, "Exploiting cloze questions for few-shot text classification and natural language inference," *Computing Research Repository*, vol. arXiv:2001.07676, 2020. [Online]. Available: http://arxiv.org/abs/2001.07676
[16] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.
[17] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.
[18] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.
[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[20] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, "Does label smoothing mitigate label noise?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 6448–6458.
[21] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.