PAPER

Consistency Regularization on Clean Samples for Learning with Noisy Labels

Yuichiro NOMURA[†a)], ***Nonmember and*** **Takio KURITA**[†], ***Fellow***

SUMMARY In the recent years, deep learning has achieved significant results in various areas of machine learning. Deep learning requires a huge amount of data to train a model, and data collection techniques such as web crawling have been developed. However, there is a risk that these data collection techniques may generate incorrect labels. If a deep learning model for image classification is trained on a dataset with noisy labels, the generalization performance significantly decreases. This problem is called Learning with Noisy Labels (LNL). One of the recent researches on LNL, called DivideMix [1], has successfully divided the dataset into samples with clean labels and ones with noisy labels by modeling loss distribution of all training samples with a two-component Mixture Gaussian model (GMM). Then it treats the divided dataset as labeled and unlabeled samples and trains the classification model in a semi-supervised manner. Since the selected samples have lower loss values and are easy to classify, training models are in a risk of overfitting to the simple pattern during training. To train the classification model without overfitting to the simple patterns, we propose to introduce *consistency regularization* on the selected samples by GMM. The consistency regularization perturbs input images and encourages model to outputs the same value to the perturbed images and the original images. The classification model simultaneously receives the samples selected as clean and their perturbed ones, and it achieves higher generalization performance with less overfitting to the selected samples. We evaluated our method with synthetically generated noisy labels on CIFAR-10 and CIFAR-100 and obtained results that are comparable or better than the state-of-the-art method.

key words: *deep learning, noisy labels, image classification, consistency regularization*

1. Introduction

In recent years, the development of research on deep learning, a machine learning method, has led to the solution of more advanced problems in many fields such as computer vision, natural language processing, and recommendation systems [2]. One drawback of methods using deep learning is that training model requires a large amount of data. Labeling samples by experts is accurate, but very expensive and time consuming. To solve this problem, some data collecting techniques such as web crawling [3], [4] and crowdsourcing [5] have been developed. However, the labels on websites and those given by human-annotators tend to be inaccurate [3]–[5] and the performance of machine learning model deteriorates due to over-fitting to these noisy labels. In particular, the deep learning model for image classification easily causes overfitting to the noisy labels due to its large number of parameters, and this phenomenon is called the memorization effect [6], [7]. The problem of inaccurate labeling of collected data is important in various fields, and research on learning deep learning models to be robust against the label noise is attracting a lot of attention.

Analysis of the memorization effects revealed that deep learning model trained on dataset with noisy labels first learns simple patterns then gradually overfits to the training samples with noisy labels, resulting in good generalization performance in the early learning phase [6]. Taking advantage of this property, several approaches have been proposed to tackle the problem of learning with noisy labels (LNL) [8]–[10]. In particular, DivideMix [1] achieves excellent results on baseline datasets with noisy labels by modeling the loss distribution to select clean samples and training a classification model in a semi-supervised manner.

Following the results of DivideMix's successful selection of clean samples, we propose to adopt a consistency regularization on the selected samples during training for preventing a classification model from overfitting to the simple patterns of the selected samples. The consistency regularization is one of the machine learning techniques developed in the field of semi-supervised learning. In our problem setting, the consistency regularization encourages model to make consistent predictions on the perturbed images that match the predictions to the original images. Since the selected samples tend to be easy to classify, the sample selection may overlook samples that have the correct label but are difficult to classify. The consistency regularization on clean samples mitigates this drawback by adding noise to samples with simple patterns to reduce overfitting. In recent years, several approaches for data augmentation have been proposed, and we use one of them, RandAugment [11], to perturb the selected samples. An overview of our method is shown in Fig. 1.

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

Manuscript received June 6, 2021.
Manuscript revised September 26, 2021.
Manuscript publicized October 28, 2021.

[†]The authors are with Graduate School of Advanced Science and Engineering, Hiroshima University, Higashihiroshima-shi, 739–8511 Japan.
a) E-mail: d202757@hiroshima-u.ac.jp

DOI: 10.1587/transinf.2021EDP7127

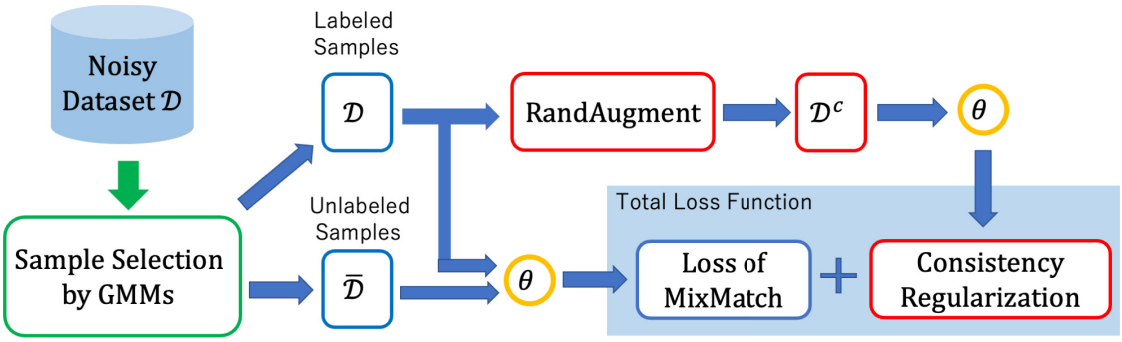

Fig. 1 Diagram of DivideMix with consistency regularization. After the samples are divided into a set of labeled samples $\mathcal{D}_\theta$ and a set of unlabeled samples $\bar{\mathcal{D}}_\theta$ by GMM (green box), each set is fed into the model θ (bottom) and compute the loss function of MixMatch following DivideMix [1]. At the same time, RandAugment is applied only to the labeled samples $\mathcal{D}_\theta$ (top) to obtain modified samples $\mathcal{D}^c_\theta$ and is fed into θ to compute the cross entropy loss as a consistency regularization. The model θ is trained on the total loss consists of the loss of MixMatch and the consistency regularization.

2. Related Works

2.1 Learning with Noisy Labels

Robust-loss approaches train the classification model with a new loss function that mitigates the effect of label noise. Mean Absolute Error (MAE) is a more noise tolerant loss than widely used cross-entropy loss [12] and Generalized Cross Entropy [13] is a method leverages the advantages of both the MAE and cross entropy. *Loss-correction* approaches [8], [14] estimate the noise transition matrix to correct loss function and mitigate the effect of noisy labels while training. *Robust-loss* and *loss-correction* techniques do not utilize the robustness of deep learning in the early learning phase [6].

Sample selection [9] exploits the robustness in the early learning phase to determine if the samples are labeled correctly or not. The value of loss function on the mislabeled data tend to be higher than ones with correctly labeled. Then samples with lower loss value are selected as clean samples. One drawback of the sample selection approach is that samples with low loss values and easy classification are always selected over samples that are correctly labeled but difficult to classify [15].

Label correction approaches also utilizes the robustness of deep learning in the early learning stage to update the original labels with the prediction output from the softmax layer of the training model [10], [16], [17]. Some other approaches use both label correction and iterative sample selection [1], [18], [19]. Most notably, DivideMix [1] uses two networks for sample selection with a two-component mixture model, and applies the semi-supervised learning technique called MixMatch [20]. Recently, AugDesc [21] has been proposed which models the loss distribution with weakly augmented samples for sample selection and trains the model with strongly augmented samples.

More recently, two studies similar to our method have been published by R. Yi et al. [22] and K. Nishi et al. [21] In the former work, the baseline model is a self-ensemble network, and the sample selection is based on the total classification loss of the model for the original images and transformed images by *scaling rotation*, and *flipping*. Our method differs from that method in that it selects images based on the loss distribution of the original image and transforms the selected images with simple patterns by RandAugment. Furthermore, the baseline model of our method is DivideMix. In the latter work, K. Nishi et al. [21] proposed AugDesc which utilizes weak augmentation and strong augmentation to images for sample selection and parameter update. The images transformed by the weak augmentation are fed to the model and calculate the loss values for each image. The GMM is fitted to these values, and divide the training image data into clean samples and noisy samples. Based on these binary classification, the dataset with strong augmentation is divided into labeled samples and unlabeled samples, and the model is trained using a semi-supervised learning method.

2.2 Dividemix

Dividemix [1] selects training samples with lower loss value as a set of labeled data and the rest of the samples are treated as a set of unlabeled data, and train the model in a semi-supervised learning manner. Dividemix fits a two-components Gaussian Mixture Model (GMM) [23] to the loss distribution of all training samples to find the clean probability of each sample then the samples, and divides the samples based on that probability.

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

$$\ell(\theta) = \{\ell_i\}_{i=1}^N = \{-\sum_{l=1}^{L} y_i^l \log(\mathrm{p}^l_{\mathrm{model}}(x_i, \theta))\}_{i=1}^N \quad (1)$$

where $\mathrm{p}^l_{\mathrm{model}}$ is a softmax output from the model for class

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

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

After dataset is divided into two sets, and a mini-batch of labeled set $\{(x_b, y_b, w_b); b \in (1, \cdots, B)\}$ is given, label refinement is performed as follows:

$$\bar{y}_b = w_b y_b + (1 - w_b) p_b \tag{2}$$

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

Given $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$, MixUp [24] is applied to them where each sample is interpolated with another sample randomly chosen from the combined mini-batch of $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$. The transformed sets are denoted as $\mathcal{X}'$ and $\mathcal{U}'$. Finally, semi-supervised method called MixMatch [20] is applied to the augmented dataset $\mathcal{X}'$ and $\mathcal{U}'$.

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

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \sum_{l} p_l \log(\mathrm{p}^{l}_{\mathrm{model}}(x;\theta)) \tag{3}$$

$$\mathcal{L}_{\mathcal{U}} = -\frac{1}{|\mathcal{U}'|} \sum_{x,p \in \mathcal{U}'} \|p - \mathrm{p}_{\mathrm{model}}(x;\theta)\|_2^2 \tag{4}$$

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

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\mathrm{reg}} \tag{5}$$

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

2.3 RandAugment

Data augmentation is a widely used technique to increase the number of training samples to enhance the generalization performance of image classification model. Typical data augmentation methods for images include rotation, flipping, cropping, etc. In general, data augmentation methods require expertise in each domain to apply plausible transformations to each sample. Recently, several approaches [25], [26] have been developed that learn optimal policies on a small proxy task for automatically designing augmentation strategies without prior knowledge of each domain. However, these approaches require a huge computational cost to find the optimal hyperparameters in their search space.

Instead of searching for hyperparameters in a proxy task, RandAugment [11] performs a grid search on the validation set to determine the best hyperparameters with drastically reduced computational cost. RandAugment is a simple data augmentation method using hyperparameters n and m, where n controls the number of transformations to be applied to a single sample and m controls the magnitude of each transformation. RandAugment selects n transformation from the following transformations with uniform probability for every image in every minibatch.

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

3. Consistency Regularization on Selected Samples

Our proposed method is based on DivideMix [1] which is an excellent approach for learning with noisy labels by modeling the loss distribution and selecting clean samples. Since recent studies show that a deep learning model in the early learning phase is robust against the noisy labels [6], [15], DivideMix selects samples with small loss values that are easy to classify as clean samples. In other words, DivideMix may overfit to the samples with simple patterns and fail to select samples with true labels but hard to classify.

Our method prevents DivideMix from overfitting to the samples with simple patterns by introducing a consistency regularization, which is widely used in semi-supervised learning (SSL). In SSL, consistency regularization encourages the training model to output the same values to the perturbed version of the unlabeled sample as to the original sample. In our method, consistency regularization is applied to selected samples by GMM and it encourages predictions on perturbed selected samples to be consistent with predictions on the original ones. This prevents training model from overfitting to the simple patterns, and encourages the model to learn samples with true labels but hard to classify by transforming easier samples by perturbation. While DivideMix uses only random cropping and horizontal flipping as data augmentation methods, our method uses RandAugment [11] as a method for adding perturbation to training images. The abstract of our method is summarized in Fig. 1.

At first, we train two networks θ_1, θ_2 on the original noisy dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(x_i, y_i)\}_{i=1}^{N}$ for a few epochs. This training period comes from the belief about robustness of deep learning in the early learning phase. After the

warmup period, all training samples are fed into θ_1 and θ_2 and a two-components GMM is fit to these loss distributions. Then select a set of clean samples as a set of labeled samples $\mathcal{D}_{\theta_1}, \mathcal{D}_{\theta_2}$ and define a complementary set as a set of unlabeled samples $\bar{\mathcal{D}}_{\theta_1}, \bar{\mathcal{D}}_{\theta_2}$ based on the output from a two-components GMM. If a clean probability w_i of each sample i exceeds τ, i is selected as clean. As shown in DivideMix, compute the refined labels $\bar{y}_b$ and $\bar{q}_b$ of each sample from $\mathcal{D}_k$ and $\bar{\mathcal{D}}_k$ for $k \in \{\theta_1, \theta_2\}$. After transforming each label, MixUp and MixMatch transforms each dataset into $\mathcal{D}'_{\theta_k}$ and $\bar{\mathcal{D}}'_{\theta_k}$. The loss on $\mathcal{D}'_{\theta_k}$ is the cross entropy loss $\mathcal{L}_{\mathcal{X}}$ and the loss on $\bar{\mathcal{D}}'_{\theta_k}$ is the mean squared error $\mathcal{L}_{\mathcal{U}}$ as follows:

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{D}'_{\theta_k}|} \sum_{x,p \in \mathcal{D}'_{\theta_k}} \sum_{l} p_l \log(\mathrm{p}^l_{\mathrm{model}}(x;\theta)) \tag{6}$$

$$\mathcal{L}_{\mathcal{U}} = -\frac{1}{|\bar{\mathcal{D}}'_{\theta_k}|} \sum_{x,p \in \bar{\mathcal{D}}'_{\theta_k}} \|p - \mathrm{p}_{\mathrm{model}}(x;\theta)\|_2^2 \tag{7}$$

where p is a mixed label for input x. In addition to these two terms, the loss for MixMatch consists of a regularization term as in Eq. (5).

Given a set of selected samples $\mathcal{D}_{\theta_k} = \{\mathcal{X}_{\theta_k}, \mathcal{Y}_{\theta_k}\}$ where $k = \{1, 2\}$ be a set of clean samples for each network (θ_1, θ_2) at each epoch, each label y_b is refined to $\hat{y}_b$ by label refinement and sharpening of DivideMix. Then for each $(x_b, \hat{y}_b) \in \mathcal{D}_{\theta_k}$ for $b \in (1, \cdots, B)$, RandAugment is applied to x_b to convert the easy to classify samples into the hard to classify samples and a modified sample and modified sets of samples are denoted as x^c_b and $\mathcal{D}^c_{\theta_k} = \{\mathcal{X}^c_{\theta_k}, \mathcal{Y}^c_{\theta_k}\} = \{\mathcal{X}^c_{\theta_k}, \hat{\mathcal{Y}}_{\theta_k}\}$ where $y^c_b \in \mathcal{Y}^c_{\theta_k}$ of each sample is equal to the $\hat{y}_b$ of original samples $\mathcal{D}_{\theta_k}$. In terms of the consistency regularization, a sample $(x_b, \hat{y}_b) \in \mathcal{D}_{\theta_k}$ corresponds to original sample and $(x^c_b, \hat{y}_b) \in \mathcal{D}^c_{\theta_k}$ corresponds to the perturbed sample. Then the consistency on selected samples for k-th network is defined as follows:

$$\mathcal{L}_c = -\frac{1}{|\mathcal{D}^c_{\theta_k}|} \sum_{x^c, \hat{y} \in \mathcal{D}^c_{\theta_k}} \sum_{l} \hat{y}_l \cdot \log(\mathrm{p}^l_{\mathrm{model}}(x^c;\theta)) \tag{8}$$

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

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}}\mathcal{L}_{\mathcal{U}} + \lambda_r\mathcal{L}_{\mathrm{reg}} + \lambda_c\mathcal{L}_c \tag{9}$$

where λ_c is a hyperparameter.

In Algorithm 1, we summarized the entire computational procedure. In later chapters, in addition to the comparison between the proposed method and existing methods, we will discuss the change in the value of the error function for the selected samples and the change in the accuracy of sample selection by consistency regularization.

Algorithm 1 Training with Consistency Regularization

Input: Dataset with noisy labels $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(x_i, y_i)\}_{i=1}^N$, Two networks θ_1, θ_2, sharpening temperature T, unsupervised loss weight λ_U, regularization term weight λ_c, $\lambda_r = 1$, Beta distribution α for MixMatch, Number of augmentations J

// Training stage
θ_1, θ_2 = WarmUp($\mathcal{D}, \theta_1, \theta_2$)
while $e <$ MaxEpoch **do**
  $\mathcal{W}_2$ = GMM($\mathcal{D}, \theta_1$)
  $\mathcal{W}_1$ = GMM($\mathcal{D}, \theta_2$)
  for $k = 1, 2$ **do**
    $\mathcal{D}^{(k)}_{\theta_k,e} = \{(x_i, y_i, w_i) | w_i \geq \tau, \forall (x_i, y_i, w_i) \in (\mathcal{D}, \mathcal{W}_k)\}$
    $\bar{\mathcal{D}}^{(k)}_{\theta_k,e} = \{x_i | w_i < \tau, \forall (x_i, w_i) \in (\mathcal{D}, \mathcal{W}_k)\}$
    for iter = 1 to num_iter **do**
      Draw a mini-batch $\{(x_b, y_b, w_b)\}_{b=1}^B$ from $\mathcal{D}_{\theta_k,e}$
      Draw a mini-batch $\{(u_b)\}_{b=1}^B$ from $\bar{\mathcal{D}}_{\theta_k,e}$
      for $b = 1$ to B **do**
        x^c_b = RandAugment($x_b | N, M$)
        for $j = 1$ to J **do**
          $\hat{x}_{b,j}$ = Augment(x_b)
          $\hat{u}_{b,j}$ = Augment(u_b)
        end for
        $p_b = \frac{1}{J}\sum_j \mathrm{p}_{\mathrm{model}}(\hat{x}_{b,j};\theta_k)$
        $\hat{y}_b = w_b y_b + (1 - w_b) p_b$
        $\bar{y}_b$ = Sharpen($\bar{y}_b, T$)
        $\bar{q}_b = \frac{1}{2J}\sum_j (\mathrm{p}_{\mathrm{model}}(\hat{u}_{b,j};\theta_1) + \mathrm{p}_{\mathrm{model}}(\hat{u}_{b,j};\theta_2))$
        $\hat{q}_b$ = Sharpen($\bar{q}_b, T$)
      end for
      $\hat{\mathcal{X}} = \{(\hat{x}_{b,j}, \hat{y}_b); b \in (1, \cdots, B), j \in (1, \cdots, J)\}$
      $\hat{\mathcal{U}} = \{(\hat{u}_{b,j}, \hat{q}_b); b \in (1, \cdots, B), j \in (1, \cdots, J)\}$
      $\mathcal{L}_{\mathcal{X}}, \mathcal{L}_{\mathcal{U}}$ = MixMatch($\hat{\mathcal{X}}, \hat{\mathcal{U}}$)
      $\mathcal{D}^{(k),c}_{\theta_k,e} = \{(x, y) | x \in x^c, y \in \hat{y}\}$
      $\mathcal{L}_c$ = ConsistencyRegularization($\mathcal{D}^{(k),c}_{\theta_k,e}$)
      $\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_U\mathcal{L}_{\mathcal{U}} + \lambda_r\mathcal{L}_{\mathrm{reg}} + \lambda_c\mathcal{L}_c$
      θ_k = SGD($\mathcal{L}, \theta_k$)
    end for
  end for
end while

3.1 Extension of Consistency Regularization

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

$$\mathcal{L}_c = -\frac{1}{I \cdot |\mathcal{D}^c_{\theta_k}|} \sum_{i=1}^{I} \sum_{x^c, \hat{y} \in \mathcal{D}^{c,(i)}_{\theta_k}} \sum_{l} \hat{y}_l \log(\mathrm{p}^l_{\mathrm{model}}(x^c;\theta)) \tag{10}$$

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

4. Experiments

First, we performed an evaluation on a dataset containing synthetically generated label noise, and then conducted a comparison experiment with state-of-the-art approaches. In later subsections, we evaluate the sensitivity of the model to

Table 1 Comparison with baseline methods and current state-of-the-art approaches on CIFAR-10 and CIFAR-100 with symmetric label noise in test accuracy (%). DivideMix [1] and AugDesc-(SAW/WAW) [21] was reimplemented using public code. The mean accuracy and its standard deviation are computed over five noise realizations.

		CIFAR-10			CIFAR-100		
Model	Noise	20%	50%	80%	20%	50%	80%
Cross Entropy	Best	86.39 ± 0.51	79.49 ± 0.48	59.48 ± 1.19	61.99 ± 0.39	47.26 ± 0.59	22.35 ± 0.80
	Last	83.63 ± 0.17	58.49 ± 0.31	26.44 ± 0.53	61.73 ± 0.37	38.58 ± 0.47	10.72 ± 0.20
DivideMix [1]	Best	96.07 ± 0.07	94.72 ± 0.06	92.72 ± 0.27	76.38 ± 0.16	72.98 ± 0.13	56.02 ± 0.62
	Last	95.79 ± 0.07	94.44 ± 0.11	92.47 ± 0.31	75.93 ± 0.14	72.46 ± 0.07	55.65 ± 0.64
AugDesc-SAW [21]	Best	96.13 ± 0.07	94.55 ± 0.43	93.50 ± 0.40	78.86 ± 0.04	76.72 ± 0.17	55.44 ± 3.66
	Last	95.96 ± 0.09	94.39 ± 0.44	93.33 ± 0.38	78.61 ± 0.04	76.44 ± 0.22	55.20 ± 3.65
AugDesc-WAW [21]	Best	96.17 ± 0.08	95.24 ± 0.13	93.56 ± 0.48	78.86 ± 0.09	76.67 ± 0.20	**64.53 ± 0.50**
	Last	95.99 ± 0.09	95.04 ± 0.16	93.38 ± 0.49	78.61 ± 0.12	76.38 ± 0.20	**64.30 ± 0.50**
Ours (I=1)	Best	**96.73 ± 0.08**	96.20 ± 0.09	**93.99 ± 0.13**	80.83 ± 0.09	**77.17 ± 0.27**	61.19 ± 0.58
	Last	**96.47 ± 0.12**	95.96 ± 0.09	**93.78 ± 0.14**	80.28 ± 0.14	**76.73 ± 0.41**	60.81 ± 0.68
Ours (I=2)	Best	96.67 ± 0.05	**96.37 ± 0.07**	93.83 ± 0.11	**81.11 ± 0.17**	76.50 ± 0.38	61.05 ± 0.45
	Last	96.39 ± 0.04	**96.11 ± 0.06**	93.64 ± 0.09	**80.51 ± 0.12**	76.20 ± 0.35	60.68 ± 0.51

Table 2 The value of RandAugment hyperparameters N and M used in Table 1 with CIFAR-10 and CIFAR-100 at different levels of noise rate.

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

the hyperparameter λ_c in test accuracy and check the effect of consistency regularization on the value of the loss function for the selected sample.

4.1 Datasets and Implementation Details

We used CIFAR-10 and CIFAR-100 [27] and dataset for validating our proposed method. Both dataset contains 50K training images and 10K test images of size 32×32. We extract 5K samples from training samples as a validation set. Following the prior works [1], [9], we add synthetically generated label noise with various noise rate to each dataset for evaluating our method. The type of label noise is symmetric noise, which randomly flips the labels of the training samples to one of the categories with a certain probability.

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

4.2 Comparison with State-of-the-Art Methods

We compared our method with DivideMix [1] and Augment Descent (AugDesc) [21], which is the current state-of-the-art for learning with noisy labels through sample selection and data augmentation, using the same network architecture. AugDesc defines the common random flip and crop image augmentation as weak data augmentation, and AutoAugment [25] as strong data augmentation. AugDesc models loss distribution on weakly or strongly augmented training samples by a two-component GMM to divide the dataset into a labeled set and an unlabeled set. Then AugDesc trains a classification model on strongly augmented training samples in a semi-supervised manner following training procedures of DivideMix. AugDesc-SAW (AugDesc-WAW) trains model with strong (weak) data augmentation during warmup period. The proposed method and AugDesc are quite similar in terms of sample selection and data augmentation strategy, but AugDesc is a method that learns only with perturbed samples for parameter updates, whereas our proposed method imposes constraints on classification model to ensure that the outputs for perturbed samples are similar to ones for original samples. This means that AugDesc does not obtain information of original samples and our proposed method receives samples selected as clean more frequently than AugDesc. In Table 1, Cross-Entropy denotes baseline model trained with Cross-Entropy loss using the same network architecture. We report the best test accuracy obtained during training and mean test accuracy averaged across the last 10 training epochs. The results with different levels of symmetric label noise on CIFAR-10 and CIFAR-100 are summarized in the Table 1 and the results given by our method (Ours ($I = 1$)) is compared with the above state-of-the-art approaches, where I is the number of augmented samples used in the consistency regularization. We also report the results from a variants of our method: consistency regularization with 2 trials of data augmentation (Ours ($I = 2$)). As the number of I increases, the model receives more augmented samples for stronger effect on pre-

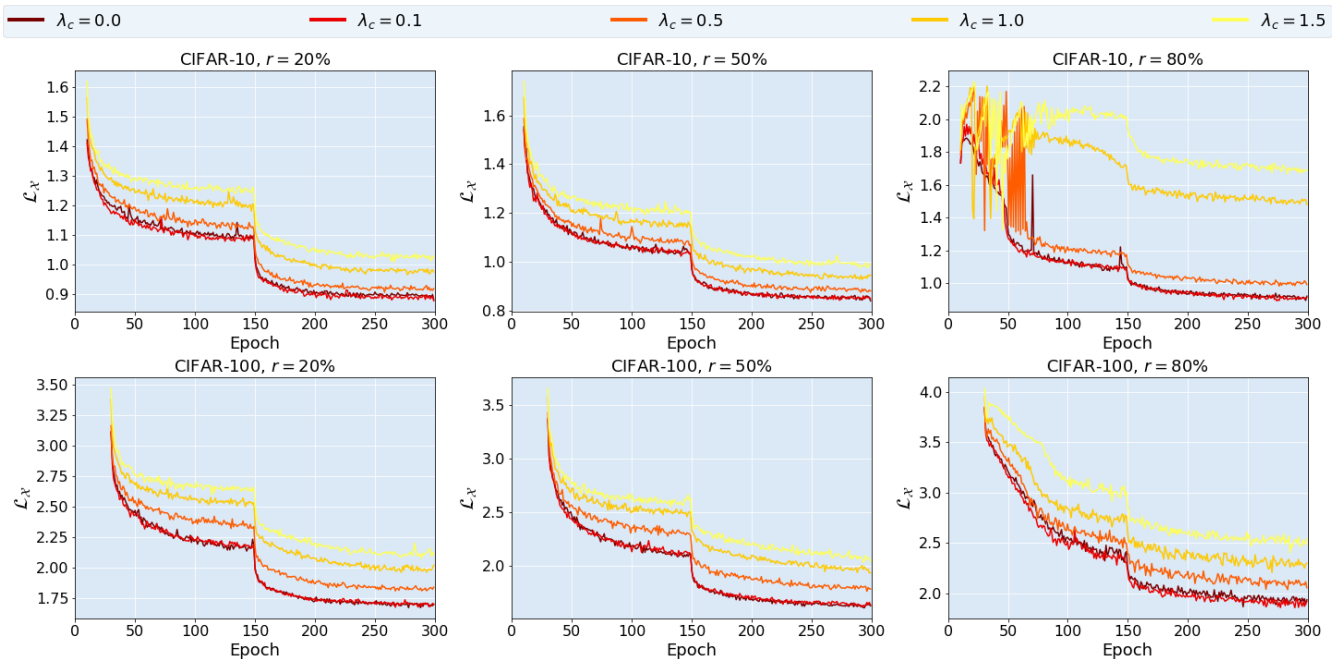

Fig. 2 Results of $\mathcal{L}_\mathcal{X}$ on CIFAR-10 and CIFAR-100 dataset with different values of lambda. Each column shows different noise rate (20%, 50%, 80%). Top Row: $\mathcal{L}_\mathcal{X}$ vs number of epochs on CIFAR-10; bottom row: $\mathcal{L}_\mathcal{X}$ vs number of epochs on CIFAR-100.

Table 3 Test accuracy on CIFAR-10 and CIFAR-100 with noisy labels with different various on Attention module. The mean accuracy and its standard deviation are computed over three noise realizations

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

venting overfitting. At the same time, learning the original samples in the classification loss ($\mathcal{L}_\mathcal{X}$) becomes more difficult. We prepared these variants to explore the appropriate number of augmented samples in the consistency regularization. Our method (Ours ($I = 1$)) achieves superior results on CIFAR-10 with 50% label noise and CIFAR-100 with 20% label noise than the state-of-the-art approaches. When the noise rate of CIFAR-100 is 80%, AugDesc-WAW achieves better results than the other methods. Our method (Ours ($I = 2$)), which performs multiple trials of data augmentation to input, yields results comparative to or lower than ours with $I = 1$.

4.3 Analysis of the Effectiveness of Consistency Regularization

Table 3 shows the results of the analysis of the effect of the hyperparameter λ_c on the performance of the model(Ours($I = 1$)) on datasets with label noise. The value of λ_c is one of $\{0.1, 0.5, 1.0, 1.5\}$ in the experiment and Table 3 shows that the performance of model is sensitive to λ_c when the noise rate is high (80%) on both datasets. For higher noise rate (80%), the model with smaller λ_c yields better results. On the contrary, the results on the different noise rate are less sensitive to the value of λ_c.

Figure 2 shows how the value of $\mathcal{L}_\mathcal{X}$ with different value of λ_c changes as learning progresses on both datasets. We observe the value of $\mathcal{L}_\mathcal{X}$ to confirm the effect of the consistency regularization by changing the value of λ_c. Y-axis of each plot shows the average value of $\mathcal{L}_\mathcal{X}$ for each epoch. The value of λ_c is one of $\{0.0, 0.1, 0.5, 1.0, 1.5\}$, and the top row is the result of CIFAR-10 and the bottom row is the result of CIFAR-100. $\mathcal{L}_\mathcal{X}$ is the loss of MixMatch on selected sample by GMM, and measures how well the model fits those samples. In all plots, the value of $\mathcal{L}_\mathcal{X}$ with larger λ_c is greater than the ones with smaller λ_c. Since the best classification accuracy of our method is given when the value of λ_c is nonzero, we can deduce that the larger value of $\mathcal{L}_\mathcal{X}$ is better for generalization performance, and the consistency regularization prevents model from overfitting to the selected samples. However, if the value of λ_c is too large, $\mathcal{L}_\mathcal{X}$ also becomes large. This causes the learning model to underfit the selected sample, resulting in poor generalization performance as shown in Table 3.

Figure 3 shows the number of selected samples as clean and Area Under the Curve (AUC) for clean/noisy sample classification at each epoch on CIFAR-10 and CIFAR-100 training data with various $\lambda_c \in \{0.0, 0.1, 0.5, 1.0, 1.5\}$ when the noise rate is 20% or 80%. In results on both CIFAR-10 and CIFAR-100, if $\lambda_c > 0$, the number of selected sam-

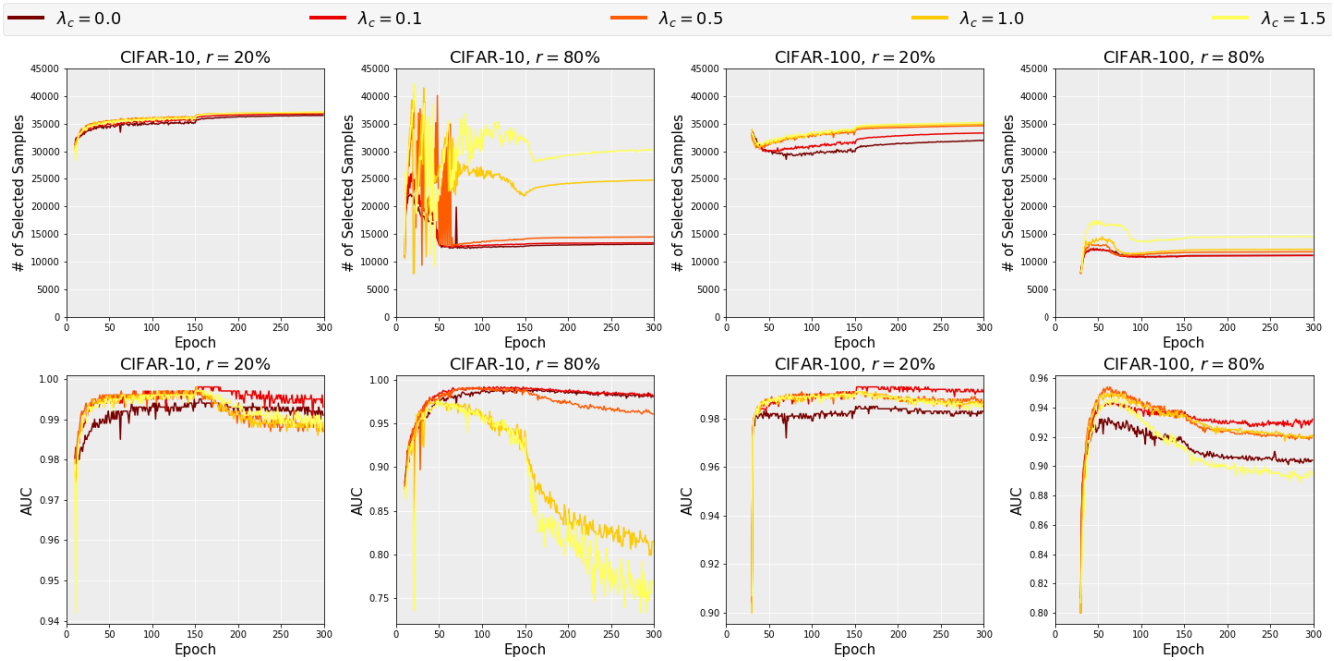

Fig. 3 Results of the number of selected samples by GMM and AUC of classification accuracy for noisy/clean labels at each epoch on CIFAR-10 and CIFAR-100. Top row: Each plot shows number of samples selected as clean vs. Number of epochs. Bottom row: Each plot shows Area Under the Curve (AUC) for clean/noisy classification vs number of epochs.

ples is greater than the case with $\lambda_c = 0$ while AUC is comaparative or better. This means that the loss distribution given by model trained with consistency regularization selects the more samples with correct labels more accurately. One possible explanation for less accurate sample selection with $\lambda_c = 0$ is that the model overfits to the samples with simple patterns and regards the samples with correct labels but hard to classify as corrupted data.

4.4 Discussion

Table 1 shows that our method achieves superior results than DivideMix [1] (baseline model) in all cases, and better than or comparative to the results given by AugDesc [21] except for 80% symmetric label noise on CIFAR-100 dataset. One possible explanation for this shortage is that the number of selected sample for each class is small by GMM for high noise rate, while AugDesc applies strong data augmentation to all training samples. Figure 4 shows that the number of selected samples by GMM at each epoch and the number of selected samples decreases as the noise rate in the dataset increases. Hyper-parameter setting used in this experiments are shown in Table 2. Therefore our method does not consider the consistency on the enormous unlabeled (not selected) samples during training when the noise rate is high. For further improvements of our methods, we should address the consistency on the unlabeled (not selected) samples during training. And we compared two variants of our method (Ours($I = 1$) and Ours($I = 2$)) in Table 1. As the value of I increases, the model receives more augmented

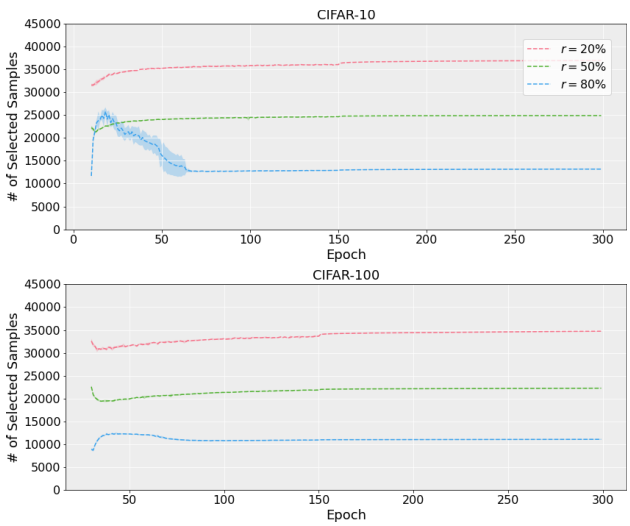

Fig. 4 Top and bottom plot show the number of selected samples vs number of epochs on CIFAR-10 and CIFAR-100 with different label noise ratio over five trials.

samples, which is expected to have a stronger effect on preventing over-fitting, but on the other hand, learning the original samples in $\mathcal{L}_X$ becomes more difficult. Experimental results show that $I = 1$ is sufficient for learning.

Training model with consistency regularization receives the perturbed selected samples and their original ones at the same time. Then model does not overfit to the selected samples. As shown in Fig. 2 for both datasets with various noise rates, the classification loss on the selected samples

$\mathcal{L}_{\mathcal{X}}$ increases as the λ_c increases in both datasets with various noise rates, indicating that the model using consistency regularization does not overfit to the selected samples during training. Also our method with nonzero λ_c achieved the best classification accuracy, it's provably showed that preventing the model from fitting to the selected samples is benefiting for higher generalization performance. Since the samples selected by fitting GMM to the loss distribution have simple patterns and tend to be easy to classify [6], our consistency regularization contributes to preventing the model from overfitting to the simple patterns and results in higher generalization performance.

Figure 3 showed that the number of selected samples and the value of AUC of classification accuracy for noisy/clean labels given by our method is larger than the baseline ($\lambda_c = 0$) during training when noise rate is small (20%) in both datasets. This means that the model correctly outputs high loss value to the samples with noisy labels and the model was able to be trained with more correctly labeled samples during training. When the noise rate is 80% in CIFAR-100, the results are also better than the baseline. However, when the noise rate is 80% in CIFAR-10, the number of selected samples and AUC given by our method is almost equal to the baseline. As shown in Table 1, the classification accuracy is also equal to the baseline. These results also support the claim that our method is less effective when the noise rate is large, because the number of samples to which consistency regularization is applied is small. One possible improvement is to generate accurate pseudo-labels for unlabeled samples, and apply consistency regularization to them as well.

5. Conclusion

In this paper, we adopt a consistency regularization to the selected samples by fitting GMM to the loss distribution for learning with noisy labels. Our method selects clean samples and modify their representation by RandAugment, and prevents the training model from overfitting to the easy samples with simple patterns. Experimental results show that validity of our method to combat with noisy labels.

References

[1] J. Li, R. Socher, and S.C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," International Conference on Learning Representations, 2020.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol.521, no.7553, pp.436–444, 2015.

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

[4] B. Thomee, D.A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.J. Li, "Yfcc100m: The new data in multimedia research," Communications of the ACM, vol.59, no.2, pp.64–73, 2016.

[5] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy, "Learning from multiple annotators with varying expertise," Machine learning, vol.95, no.3, pp.291–327, 2014.

[6] 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," International Conference on Machine Learning, pp.233–242, PMLR, 2017.

[7] X. Ma, Y. Wang, M.E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," International Conference on Machine Learning, pp.3355–3364, 2018.

[8] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.2233–2241, 2017.

[9] 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, pp.8535–8545, 2018.

[10] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.5552–5560, 2018.

[11] E.D. Cubuk, B. Zoph, J. Shlens, and Q. Le, "Randaugment: Practical automated data augmentation with a reduced search space," Advances in Neural Information Processing Systems, pp.18613–18624, 2020.

[12] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," Proc. AAAI Conference on Artificial Intelligence, pp.1919–1925, 2017.

[13] Z. Zhang and M.R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," arXiv preprint arXiv:1805.07836, 2018.

[14] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," International Conference on Learning Representations, 2017.

[15] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, "Early-learning regularization prevents memorization of noisy labels," Advances in Neural Information Processing Systems, vol.33, 2020.

[16] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," International Conference on Learning Representations, 2015.

[17] K. Yi and J. Wu, "Probabilistic end-to-end noise correction for learning with noisy labels," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.7010–7018, 2019.

[18] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," International Conference on Machine Learning, pp.312–321, PMLR, 2019.

[19] H. Song, M. Kim, and J.G. Lee, "Selfie: Refurbishing unclean samples for robust deep learning," International Conference on Machine Learning, ICML, pp.5907–5915, PMLR, 2019.

[20] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," arXiv preprint arXiv:1905.02249, 2019.

[21] K. Nishi, Y. Ding, A. Rich, and T. Höllerer, "Augmentation Strategies for Learning with Noisy Labels," arXiv e-prints, March 2021.

[22] R. Yi and Y. Huang, "Transform consistency for learning with noisy labels," arXiv preprint arXiv:2103.13872, 2021.

[23] H. Permuter, J. Francos, and I. Jermyn, "A study of gaussian mixture models of color and texture features for image classification and segmentation," Pattern Recognition, vol.39, no.4, pp.695–706, 2006.

[24] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," arXiv preprint arXiv:1710.09412, 2017.

[25] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q.V. Le, "Autoaugment: Learning augmentation strategies from data," Proc. IEEE/CVF Conference on Computer Vision and Pattern Recogni-

tion, pp.113–123, 2019.
[26] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," Advances in Neural Information Processing Systems, pp.6665–6675, 2019.
[27] A. Krizhevsky, G. Hinton, "Learning multiple layers of features from tiny images," 2009.

Yuichiro Nomura received a master's degree in Information Science and Electrical Engineering from Kyushu University, Japan, in 2020. His research interests include machine learning, deep learning, and learning with noisy labels.

Takio Kurita received the B.Eng. degree from Nagoya Institute of Technology and the Dr.Eng. degree from the University of Tsukuba, in 1981 and in 1993, respectively. From 1981 to 2000, he was a research scientist at the Electrotechnical Laboratory. From 2001 to 2009, he was a Deputy Director of Neuroscience Research Institute, National Institute of Advanced Industrial Science and Technology. He is currently a Professor at Hiroshima University. His current research interests are statistical pattern recognition and its application to image recognition. He is a member of the IEEE, IEICE, JNNS, JSAI, and ITE.