How to Prevent the Continuous Damage of Noises to Model Training?
—— *Supplementary Materials* ——

This supplementary material provides more details of the Gradient Switching Strategy (GSS), including the theoretical deductions of gradient analysis, effectiveness analysis of GSS, details about GSS combination, details about dataset settings, ablation study, limitation of GSS, and source codes, which are given in the 'code.zip' file.

A. Theoretical Analysis of Gradient

To investigate how existing methods work, we analyze their training gradients, including robust loss functions (Symmetric Learning Loss [13], Generalized Cross-Entropy Loss [16], Early-Learning Regularization [8], Peer Loss [10]), sample reweighting (EG Reweighting [11], CIW [4]), and sample cleaning (Co-teaching [2], DivideMix [5]).

A.1. Gradient Analysis of Cross-Entropy

As mentioned in the original paper, the update of model weights is related to two terms: gradient weight and feature direction, which are denoted as $\frac{\partial \mathcal{L}}{\partial z_k}$ and $\frac{\partial z_k}{\partial m^l}$. According to the methodology, almost all existing methods optimize the gradient weight term to reduce the influence of uncertain samples. So we analyze this term of various existing methods in this section. First, the gradient weight of Cross-Entropy (CE) loss can be derived as:

$$\frac{\partial \mathcal{L}_{ce}(\widetilde{y})}{\partial z_k} = \sum_{k'}^{K} \frac{\partial \mathcal{L}_{ce}(\widetilde{y})}{\partial p_{k'}} \frac{\partial p_{k'}}{\partial z_k} = \frac{\partial \mathcal{L}_{ce}(\widetilde{y})}{\partial p_{\widetilde{y}}} \frac{\partial p_{\widetilde{y}}}{\partial z_k}, \tag{1}$$

where $\widetilde{y}$ denotes the annotated label, K denotes the amount of categories, and $p_{k'}$ denotes the prediction on k'-th category. Since $\mathcal{L}_{ce}(\widetilde{y}) = -\log(p_{\widetilde{y}})$, the first term in Eqn. 1 can be derived as:

$$\frac{\partial \mathcal{L}_{ce}(\widetilde{y})}{\partial p_{\widetilde{y}}} = \frac{\partial -\log(p_{\widetilde{y}})}{\partial p_{\widetilde{y}}} = \frac{-1}{p_{\widetilde{y}}}. \tag{2}$$

For the second term, the prediction p is obtained by the Softmax function. Consequently, the gradient can be derived as:

$$\frac{\partial p_{\widetilde{y}}}{\partial z_k} = \begin{cases} -p_k(1-p_k), & p_{\widetilde{y}} = k \\ -p_k p_{\widetilde{y}}, & p_{\widetilde{y}} \neq k \end{cases}. \tag{3}$$

In summary, the gradient weight of CE can be derived based on Eqn. 1-3:

$$\frac{\partial \mathcal{L}_{ce}(\widetilde{y})}{\partial z_k} = p_k - q_k, \tag{4}$$

where $q_k = \mathbb{1}[\widetilde{y} = k]$. Since $p_k \in (0, 1)$, the gradient weight is negative on the category of the annotated label and positive on the others. If annotated labels are noisy, misleading gradient directions will damage the model training.

A.2. Gradient Analysis of Existing Methods

Compared to the CE, existing methods optimize the gradient weight term to reduce the influence of noise samples. We illustrate the gradient weight of various methods in Table S1. Sample cleaning is a special case of sample reweighting with binary weights, which simply adds an additional multiplier for the gradient weight of each sample. For robust loss functions, the formulas of gradient weight are analyzed in the following.

For Symmetric Learning (SL) loss [13], the function and its gradient weight are given as follows:

$$\begin{aligned} \mathcal{L}_{sl} &= \alpha \left(\sum_k -q_k \log p_k \right) + \beta \left(\sum_k -p_k \log q_k \right) \\ &= \alpha - \log p_{\widetilde{y}} + \beta \left(-A(1 - p_{\widetilde{y}}) \right), \end{aligned} \tag{5}$$

$$\frac{\partial \mathcal{L}_{sl}}{\partial z_k} = \left(\alpha + \beta |A| p_{\widetilde{y}} \right) \left(p_k - y_k \right), \tag{6}$$

where $q_k = \mathbb{1}[\widetilde{y} = k]$, z_k denotes the logit on k-th category, α and β are two positive hyper-parameters, A is set to a negative constant to replace $-\log 0$. It can be seen that compared with CE, the gradient weight of SL has an additional non-negative term $\alpha + \beta |A| p_{\widetilde{y}}$, which is linearly correlated with $p_{\widetilde{y}}$. For the samples with low confidence, the small gradient weight will suppress the influence of noise.

Generalized Cross Entropy (GCE) [16] loss is another example of robust learning. The function and its gradient weight are given as follows:

$$\mathcal{L}_{gce} = \frac{1 - (p_{\widetilde{y}})^{\gamma}}{\gamma}, \tag{7}$$

Method	Formula of gradient weight $\partial\mathcal{L}(\widetilde{y})/\partial z_k$
Cross Entropy	$p_k - q_k$
GCE [16]	$p_{\widetilde{y}}^{\gamma}(p_k - q_k)$
SL [13]	$(\alpha + \beta\lvert A\rvert p_y)(p_k - q_k)$
ELR [8]	$(p_k - q_k) + \frac{\sum_i p_i\hat{p}_i - \hat{p}_k}{1-\sum_i p_i\hat{p}_i}\theta p_k$
Peer Loss [10]	$(p_k^{(n)} - q_k^{(n)}) - (p_k^{(n1)} - q_k^{(n2)})$
EG Reweighting [11]	$w_{EG}\,(p_k - q_k)$
CIW [4]	$w_{CIW}\,(p_k - q_k)$
Co-teaching [2]	$\mathbb{1}\left[\mathcal{L}(p^*)_y < \tau'\right](p_k - q_k)$
DivideMix [5]	$\mathbb{1}\left[GMM\left(\mathcal{L}(p^*)_y\right) > \tau''\right](p_k - q_k)$

Table S1. The summarized gradient weight of existing methods. Here $q_k = \mathbb{1}\left[\widetilde{y} = k\right]$.

$$\frac{\partial\mathcal{L}_{gce}}{\partial z_k} = p_{\widetilde{y}}^{\gamma}(p_k - y_k), \tag{8}$$

where γ is the hyper-parameter which satisfies $\gamma \in (0, 1]$. The additional term $p_{\widetilde{y}}(1 - p_{\widetilde{y}})^{\gamma-1}$ is also non-negative, which will be minimized if $p_{\widetilde{y}}$ is low.

Early Learning Regularization (ELR) [8] sets the previous model predictions as targets, which is defined as:

$$\mathcal{L}_{elr} = -\log p_{\widetilde{y}} + \theta\log\left(1 - \langle p, q\rangle\right), \tag{9}$$

where θ denotes the weight of early learning regularization, q denotes the output of the model in the last epoch, $\langle p, q\rangle$ denotes the inner product of prediction p and the prediction of the model in last epoch. The gradient of the latter term w.r.t. the prediction p_i is given as follows:

$$\frac{\partial\log\left(1 - \langle p, q\rangle\right)}{\partial p_j} = \frac{-q_j}{1 - \sum_i p_i q_i}, \tag{10}$$

$$\begin{aligned}\frac{\partial\log\left(1 - \langle p, q\rangle\right)}{\partial z_k} &= \sum_j \frac{\partial\log\left(1 - \langle p, q\rangle\right)}{\partial p_j}\frac{\partial p_j}{\partial z_k} \\ &= \frac{-q_k(-p_k(p_k - 1))}{1 - \sum_i p_i q_i} \\ &\quad + \sum_{j\neq k}\frac{-q_j(-p_k p_j)}{1 - \sum_i p_i q_i} \\ &= \frac{p_k}{1 - \sum_i p_i q_i}\left(\sum_i p_i q_i - q_k\right).\end{aligned} \tag{11}$$

In summary, the gradient weight of ELR+ is derived as:

$$\frac{\partial\mathcal{L}_{elr}}{\partial z_k} = (p_k - y_k) + \frac{\theta\left(\sum_i p_i q_i - q_k\right)}{1 - \sum_i p_i q_i}p_k. \tag{12}$$

Compared with SL loss and GCE loss, ELR+ has the different gradient weight, which does not employ a non-negative multiplier to $(p_k - y_k)$. According to the default setting of ELR, $\theta = 3$ is applied for CIFAR-10 [3], and $\theta = 7$ is applied for CIFAR-100. The additional term is negative if the prediction of the model in the last epoch q_k is highly confident, rectifying the gradient weight to the right direction for the true category. Conversely, the additional term is positive for the category with low confidence in the last epoch. So that the gradient weight is positive even with the noisy label $y_k = 1$.

As mentioned in the original paper, these methods are essentially enhancing or inhibiting the gradient weight term. Though these methods avoid the negative effect of the noise, the positive effect of hard samples on the model is also suppressed. Also, the misidentified samples cause continuous damage to the training. That is why there is a big gap between these methods and the model trained by clean data.

B. Effectiveness Analysis of GSS

In Section 5.3 of the original paper, the effectiveness of gradient switching is analyzed. The theoretical analysis assumes the activation feature map a and gradient direction d of the same sample is basically constant in a small number of iterations. This section conducts experimental analyses to compare the activation feature map and gradient direction of the same sample during training.

The original paper gives the gradient bias caused by each sample within a small number of iterations $\mathcal{E}$, derived as:

$$\Delta g = \sum_e^{\mathcal{E}} \mu a^e \otimes \left|d_{\widetilde{y}}^e - d_y^e\right|, \tag{13}$$

where Δg denotes the gradient bias, $\otimes$ denotes the convolution operation, y denotes the true label, $\widetilde{y}$ denotes the annotated noisy label, $\mathcal{E}$ denotes a small amount of epochs, d_y^e denotes the shorthand for the gradient direction $\partial z_y/\partial m^l$ on e-th epoch, and μ denotes the learning rate. After assuming the activation feature map a and gradient direction d of the same sample is basically constant in a small number of iterations, the total bias is simplified as:

$$\Delta g' = \mu a \otimes \left|\mathcal{E}\left(d_{\widetilde{y}} - d_y\right)\right|, \tag{14}$$

where d_y and $d_{\widetilde{y}}$ denote the gradient direction of y and $\widetilde{y}$ category on current epoch.

To demonstrate this assumption, we conduct experiments on the activation feature map and gradient direction of various epochs during training. Ten samples from each category are randomly selected from the CIFAR-10 dataset. For each sample, the activation feature map and gradient direction of the last convolution layer is selected for visualization. Since deep convolution layers extract more complex features, the difference during training will be larger than that of shallow layers. Consequently, demonstrating

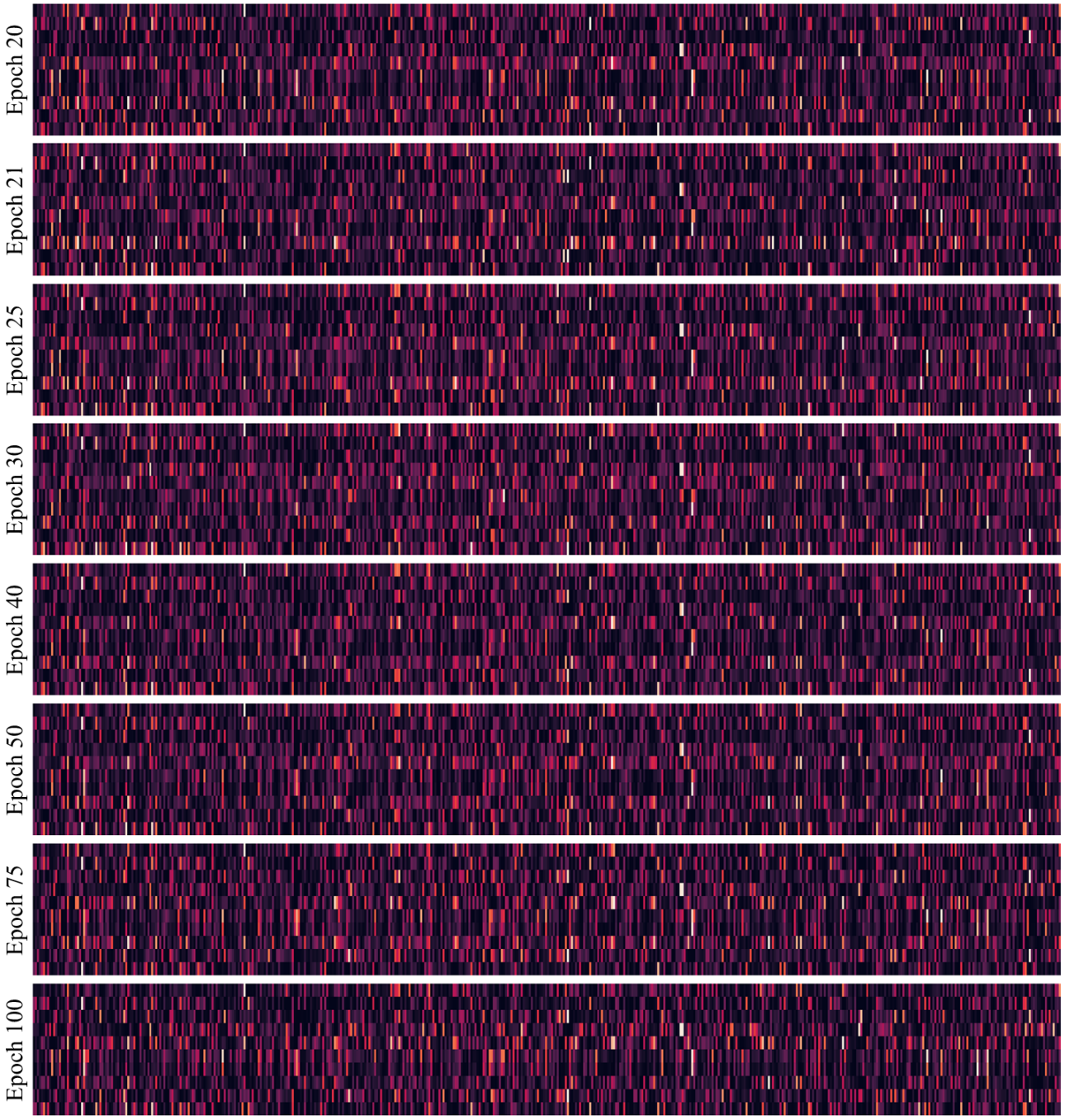

Figure S1. The visualization of the activation feature map a during training. Ten samples from each category are randomly selected from the CIFAR-10 dataset. The activation feature maps are obtained from the last convolution layer and normalized in each dimension. Each sub-figure denotes the normalized feature map distribution during training, and each line in these sub-figures represents a sample. The visualization results during epoch $20 \sim 100$ are given to show the difference of activation feature maps.

the biases in deep layers are neglected also means the biases in shallow layers are neglected. The distributions are normalized in each dimension of feature maps and gradient directions. The visualization results are shown in Fig. S1 (activation feature map) and Fig. S2 (gradient direction).

From the visualization results, it can be seen that both activation feature map a and gradient direction d have small differences during training. To be noted, the calculation of

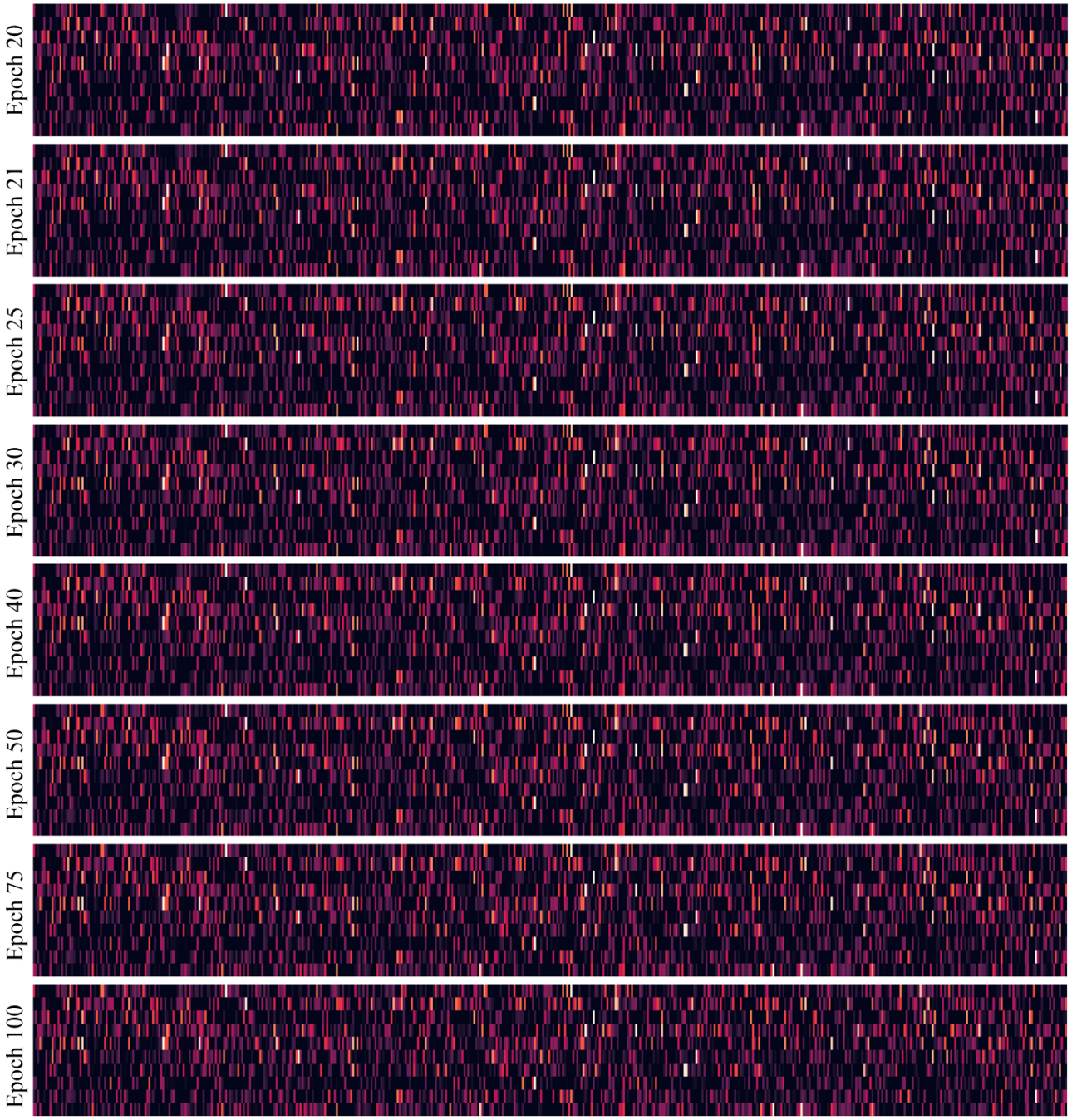

Figure S2. The visualization of the gradient direction d during training. Ten samples from each category are randomly selected from the CIFAR-10 dataset. The activation feature maps are obtained from the last convolution layer and normalized in each dimension. Each sub-figure denotes the normalized gradient direction during training, and each line in these sub-figures represents a sample. The visualization results during epoch $20 \sim 100$ are given to show the difference of gradient directions.

gradient bias in the original paper is conducted in 10 epochs ($\mathcal{E} = 10$). Nevertheless, the results show the distributions in epoch 20 and epoch 100 are also similar. The visualization results indicate that the activation feature map or gradient direction of each sample changes slowly during the model training, so that the approximate in Eqn. 14 has little influence on the calculation of gradient bias.

To further analyze this assumption quantitatively, we cal-

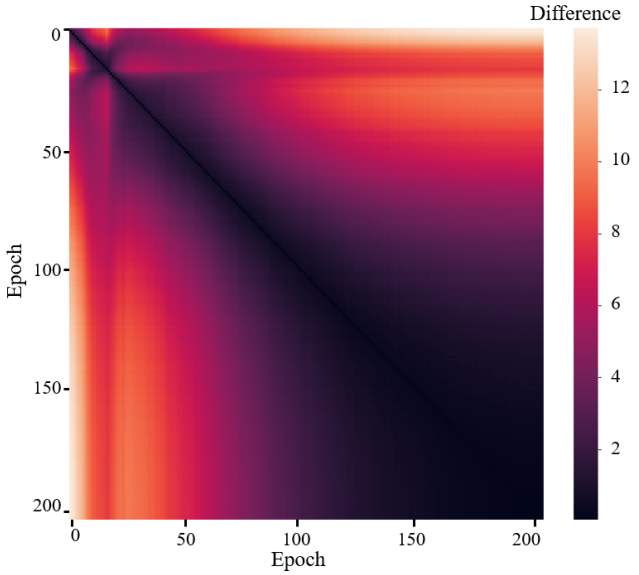

Figure S3. The heat map of the difference between gradient directions in various epochs. The darker color denotes the smaller difference of gradient directions. The experiment is conducted with GSS-SB on CIFAR-10 with 40% noise, and the gradient switching strategy is started at 15-th epoch.

Epochs		**50**	**100**	**150**
Gradient Bias	Δg_{ori}	2.15×10^2	5.65×10^2	1.76×10^3
	Δg_{gss}	1.20×10^2	2.61×10^2	6.53×10^2
Epoch Range		**50 $\sim$ 60**	**100 $\sim$ 110**	**150 $\sim$ 160**
Gradient Direction Difference		0.89	0.54	0.25

Table S2. The experimental analysis of gradient biases and gradient direction differences in different training stages. The results are calculated by adding the absolute values, and the average biases/difference are shown. The experiments are conducted on CIFAR-10 and CIFAR-100 with 40% symmetric noise.

culate the differences of gradient directions during training. In a total of 200 epochs of training, the heat map of gradient direction differences is illustrated in Fig. S3. The darker color denotes the smaller difference of gradient directions. The gradient switching is implemented from the 15-th epoch. It can be seen the difference of gradient directions is less in the late training period, which is consistent with the expected results. Even in the early stage of the training, the change of gradient directions is also small within a few epochs.

Also, we compare the difference with the gradient biases in Table S2. During training, the gradient bias becomes larger, but the gradient direction difference becomes smaller. So that the influence of the approximation also becomes smaller in the model training. Still, even in the early stage (50-th epoch), the gradient direction difference caused by the approximation (0.89) is much smaller than the gradient bias difference (95). And the difference is even smaller in the later stage (150 $\sim$ 160 epoch), while the gradient bias is larger than that in the early stage. In summary, the approximation in Eqn. 14 has little effect on gradient biases.

Algorithm 1: Gradient Switching Strategy with the Single Branch (GSS-SB).

Input: Model weights w; The noisy dataset $\mathbb{D} = \{(x^{(n)}, \widetilde{y}^{(n)})\}_{n=1}^{N}$; The learning rate μ; The training epoch E, and batch number $\mathcal{N}_{batch}$; The updating weights λ_1 and λ_2.

Initialize: Let $\mathcal{D}^{(n)}$ be the gradient direction pool of n-th sample, initialized as $\mathcal{D}^{(n)} = Onehot(\widetilde{y}^{(n)})$;
for $e = 1$ **to** E **do**
  for $batch = 1$ **to** $\mathcal{N}_{batch}$ **do**
    Fetch mini-batch data $\mathbb{D}'$;
    Select directions $\widehat{y}_e$ from $\mathcal{D}$;
    Update $w = w - \mu \nabla \mathcal{L}(x, \widehat{y}_e)$;
    $p = f(x; w), \forall x \in \mathbb{D}'$
    $v^{or} = p_{\widehat{y}}(1 - e/E)$;
    $v^{pr} = p_{\widehat{y}}(\lambda_1 e/E)$;
    $v^{rd} = \lambda_2 e/E$;
    Update gradient direction pool $\mathcal{D} = \mathcal{D} + \mathcal{Y}^{or} v^{or} + \mathcal{Y}^{pr} v^{pr} + \mathcal{Y}^{rd} v^{rd}$;
  end
end
Output: w.

C. Dataset Setting Details

This section shows more details of datasets. For synthetic noisy datasets, two widely used benchmark datasets are applied for better evaluation, including CIFAR-10 and CIFAR-100 [3]. Synthetic noisy labels consist of two types, including symmetric noise and asymmetric noise. Symmetric noise is generated by randomly flipping labels with uniform distribution. Asymmetric noise generation needs to take categories' similarity into account, which varies from datasets.

In this paper, we define some confusing category pairs in each dataset as the target of generating asymmetric noise. Specifically, there are four pairs in CIFAR-10 (truck $\leftrightarrow$ automobile, bird $\leftrightarrow$ airplane, cat $\leftrightarrow$ dog, deer $\leftrightarrow$ horse). Samples of these category pairs have very similar features. In CIFAR-100, 100 categories are divided into 20 superclasses, including aquatic mammals, fish, flowers, food, fruit and vegetables, household electrical devices, household, insects, large carnivores, large man-made outdoor things, large natural outdoor scenes, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals, trees, vehicles 1

Algorithm 2: Gradient Switching Strategy with the Dual Branches (GSS-DB).

Input: Dual branch weights w and w'; The noisy dataset $\mathbb{D} = \{(x^{(n)}, \widetilde{y}^{(n)})\}_{n=1}^{N}$; The learning rate μ; The training epoch E, and batch number $\mathcal{N}_{batch}$; The updating weights λ_1 and λ_2.

Initialize: Let $\mathcal{D}^{(n)}$ and $\mathcal{D}'^{(n)}$ be the gradient direction pools of n-th sample, initialized as $\mathcal{D}^{(n)} = \mathcal{D}'^{(n)} = Onehot(\widetilde{y}^{(n)})$;

for $e = 1$ **to** E **do**

  for $batch = 1$ **to** $\mathcal{N}_{batch}$ **do**

    Fetch mini-batch data $\mathbb{D}_{batch}$;

    Obtain small-loss subset $\bar{\mathbb{D}}$ that $|\bar{\mathbb{D}}| = R(e)|\mathbb{D}_{batch}|$

    Obtain small-loss subset $\bar{\mathbb{D}}'$ that $|\bar{\mathbb{D}}'| = R(e)|\mathbb{D}_{batch}|$

    Select directions $\widehat{y}_e$ from $\mathcal{D}$;

    Select directions $\widehat{y}'_e$ from $\mathcal{D}'$;

    Update $w = w - \mu \nabla \mathcal{L}(x, \widehat{y}'_e)$;

    Update $w' = w' - \mu \nabla \mathcal{L}(x, \widehat{y}_e)$;

    $p = f(x; w), \forall x \in \bar{\mathbb{D}}$

    $p' = f(x; w'), \forall x \in \bar{\mathbb{D}}'$

    $v^{or} = p_{\widetilde{y}}(1 - e/E)$;

    $v'^{or} = p'_{\widehat{y}}(1 - e/E)$;

    $v^{pr} = p_{\widetilde{y}}(\lambda_1 e/E)$;

    $v'^{pr} = p'_{\widehat{y}}(\lambda_2 e/E)$;

    $v^{rd} = \lambda_2 e/E$;

    Update gradient direction pool

    $\mathcal{D} = \mathcal{D} + \mathcal{Y}'^{or} v'^{or} + \mathcal{Y}'^{pr} v'^{pr} + \mathcal{Y}'^{rd} v^{rd}$;

    $\mathcal{D}' = \mathcal{D}' + \mathcal{Y}^{or} v^{or} + \mathcal{Y}^{pr} v^{pr} + \mathcal{Y}^{rd} v^{rd}$;

  end

end

Output: w.

and vehicles 2. Since the sub-classes in each superclass are especially similar, the asymmetric noise is generated within each superclass. Since not all the categories are covered, the overall asymmetric noise ratio is lower than the symmetric one.

To evaluate the comprehensive performance of each method with different severity of noise, this paper generates the symmetric noise from 20% to 80% and asymmetric noise from 20% to 40%. And the quantitative evaluation results are given in Section 6.1 of the original paper.

D. GSS Combination Details

The original paper proposes the GSS combinations with the single branch, dual branches, and dual branches with semi-supervised learning, denoted as GSS-SB, GSS-DB, and GSS-SSL, respectively. This section will supplement more details of these combinations.

D.1. GSS with Single Branch

As we introduced in Section 5.2 of the original paper, GSS-SB is the simplest version of Gradient Switching Strategy with the single branch. The flow of GSS-SB is shown in Algorithm 1. In GSS-SB, all samples have their gradient direction pool $\mathcal{D}$ updated by dynamic weights as follows:

$$v^{or} = p_{\widetilde{y}}(1 - e/E), \tag{15}$$

$$v^{pr} = p_{\widetilde{y}}(\lambda_1 e/E), \tag{16}$$

$$v^{rd} = \lambda_2 e/E, \tag{17}$$

where $p_{\widetilde{y}}$ denotes the predicted confidence on noisy labels, E denotes the total amount of epochs, e denotes the current epoch, and λ_1, λ_2 are parameters to control the tendency of the gradient switching in GSS-SB. The parameter λ_1 determines the importance of predicted labels during training, and the parameter λ_2 determines the importance of randomness. Commonly, the gradient direction pool tends to the direction of predicted labels, guaranteeing the accuracy of the unfixed directions selected in every epoch. In datasets with severe noise, the model predictions are less credible, so more randomness should be added to the gradient direction pool. The core effectiveness of GSS-SB is to prevent the continuous damage of noise through the dynamic gradient direction pool.

D.2. GSS with Double Branches

The framework of dual branches is combined with GSS in GSS-DB to further prevent the gradient direction pool from being damaged. The pseudo-code is given in Algorithm 2. The GSS-DB applies two groups of gradient direction pools $\mathcal{D}$ and $\mathcal{D}'$ for all samples. After fetching mini-batch data in each iteration, the subsets of small-loss samples are selected based on the predictions of dual branches. The gradient direction pools are updated alternately between two branches, which can increase the ability to prevent damage from noisy labels.

In Co-teaching [2], the subset ratio $R(e) = 1 - min\{\mu e/\bar{E}, \mu\}$, where μ denotes the noise ratio. So that subset ratio equals $1 - \mu$ after $\bar{E}$-th epoch. Conversely, GSS-DB will gradually turn uncertain samples to clean samples with the principle gradient direction. The subset ratio will increase after preliminary training and set as $R(e) = min(1 - \mu + |e - \bar{E}|/\bar{E}, 1)$. In GSS-DB, the preliminary training stage is similar to that in Co-teaching. Afterward, the subset ratio is increased to 1 for gradually utilizing all samples in training. The strategy of gradient switching prevents the continuous damage of noise, which allows the model to learn more information from all samples. That is the advantage of our GSS-DB compared to existing methods with dual branches.

Algorithm 3: Gradient Switching Strategy with Semi-Supervised Learning (GSS-SSL).

Input: Dual branch weights w and w'; The noisy dataset $\mathbb{D} = \{(x^{(n)}, \widetilde{y}^{(n)})\}_{n=1}^{N}$; The learning rate μ; The training epoch E, and batch number $\mathcal{N}_{batch}$; The updating weights $\lambda_{\mathcal{X}}$, $\lambda_{\mathcal{U}}$, $\lambda'_{\mathcal{X}}$, $\lambda'_{\mathcal{U}}$, $\lambda^{rd}_{\mathcal{X}}$, and $\lambda^{rd}_{\mathcal{U}}$.

Initialize: Let $\mathcal{D}^{(n)}$ and $\mathcal{D}'^{(n)}$ be the gradient direction pools of n-th sample, initialized as $\mathcal{D}^{(n)} = \mathcal{D}'^{(n)} = Onehot(\widetilde{y}^{(n)})$;

for $e = 1$ **to** E **do**

  Obtain labeled set $\mathcal{X}$ and unlabeled set $\mathcal{U}$ based on $GMM(\mathbb{D}, w')$

  Obtain labeled set $\mathcal{X}'$ and unlabeled set $\mathcal{U}'$ based on $GMM(\mathbb{D}, w)$

  for $batch = 1$ **to** $\mathcal{N}_{batch}$ **do**

    Select directions $\widehat{y}_e$ from $\mathcal{D}$;

    Select directions $\widehat{y}'_e$ from $\mathcal{D}'$;

    $\mathcal{L} = \mathcal{L}(\mathcal{X}, \widehat{y}'_e) + \mathcal{L}(\mathcal{U}, \widehat{y}'_e) + \mathcal{L}_{reg}$;

    $\mathcal{L}' = \mathcal{L}(\mathcal{X}', \widehat{y}_e) + \mathcal{L}(\mathcal{U}', \widehat{y}_e) + \mathcal{L}'_{reg}$;

    Update $w = w - \mu \nabla \mathcal{L}$;

    Update $w' = w' - \mu \nabla \mathcal{L}'$;

    $v^{or} = p_{\widetilde{y}}(1 - e/E)$;

    $v'^{or} = p'_{\widehat{y}}(1 - e/E)$;

    $v^{pr} = \{p^{\mathcal{X}}_{\widetilde{y}}(\lambda_{\mathcal{X}} e/E), p^{\mathcal{U}}_{\widetilde{y}}(\lambda_{\mathcal{U}} e/E)\}$;

    $v'^{pr} = \{p^{\mathcal{X}'}_{\widehat{y}}(\lambda'_{\mathcal{X}} e/E), p^{\mathcal{U}'}_{\widehat{y}}(\lambda'_{\mathcal{U}} e/E)\}$;

    $v^{rd} = \{\lambda^{rd}_{\mathcal{X}} e/E, \lambda^{rd}_{\mathcal{U}} e/E\}$;

    Update gradient direction pool

    $\mathcal{D} = \mathcal{D} + \mathcal{Y}'^{or} v'^{or} + \mathcal{Y}'^{pr} v'^{pr} + \mathcal{Y}'^{rd} v^{rd}$;

    $\mathcal{D}' = \mathcal{D}' + \mathcal{Y}^{or} v^{or} + \mathcal{Y}^{pr} v^{pr} + \mathcal{Y}^{rd} v^{rd}$;

  end

end

Output: w.

D.3. GSS with Semi-Supervised Learning

To better train the model with uncertain samples, the Semi-Supervised Learning framework is applied in GSS-SSL. The pseudo-code given in Algorithm 3 shows the flow of GSS-SSL. Referring to the settings in DivideMix [5], GSS-SSL applies GMM [12] to distinguish confident samples and uncertain samples. But the labels of uncertain samples are not directly removed. In GSS-SSL, these samples are trained by the targets of model predictions and unfixed directions by weighting, which further prevents continuous damage on the basis of the DivideMix.

As mentioned in the original paper, GSS-SSL also applies the dual branches so that the main process is consistent with GSS-DB. Additionally, two updating strategies of gradient direction pools are used for labeled samples and unlabeled samples. The updating weights of the former samples are denoted as $\lambda_{\mathcal{X}}(\lambda'_{\mathcal{X}})$, $\lambda^{rd}_{\mathcal{X}}$, and the updating weights of the latter samples are $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}})$, $(\lambda^{rd}_{\mathcal{U}})$. $\lambda_{\mathcal{X}} and \lambda'_{\mathcal{X}}$ denotes the weights of labeled samples for two branches. Since most labeled samples are credible, these weights are set with small values. On the contrary, the unlabeled samples are incredible, which requires to be rectified. So that $(\lambda^{rd}_{\mathcal{U}})$ is larger to prevent the model from being damaged in the misleading gradient direction. After sufficient experiments, the optimal weights are set as $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}}) = 0.2$, $\lambda^{rd}_{\mathcal{U}} = 0.5$. More ablation study is conducted in Section E.

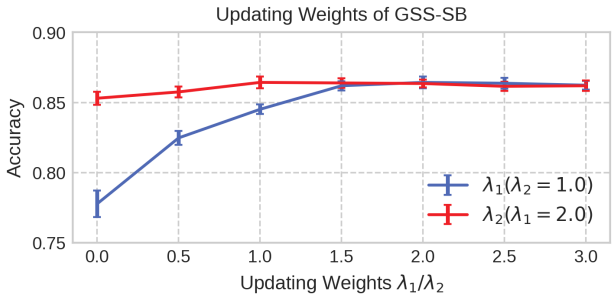

Figure S4. The ablation results with different weights of gradient direction pool updating in GSS-SB. The experiments are conducted on CIFAR-10 with 40% noisy labels with GSS-SB. All results are the average accuracy with the error bar on the test set over five experiments.

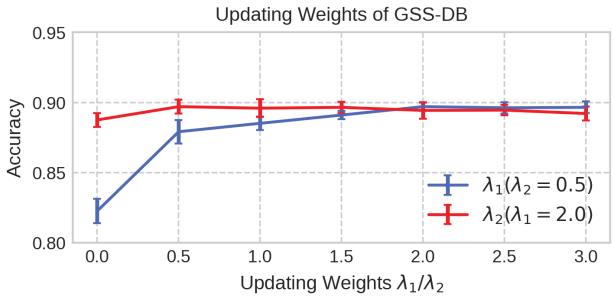

Figure S5. The ablation results with different weights of gradient direction pool updating in GSS-DB. The experiments are conducted on CIFAR-10 with 40% noisy labels with GSS-DB. All results are the average accuracy with the error bar on the test set over five experiments.

E. Ablation Study

The original paper conducts the ablation study on the effects of different updating weights and GSS combinations. Due to the page limit, the experiments of updating weights are only conducted on GSS-SB (shown in Fig. S4). So in this section, we further explore the effects of different updating weights of the gradient direction pool on GSS-DB and GSS-SSL.

As described in the original paper and Section D, the

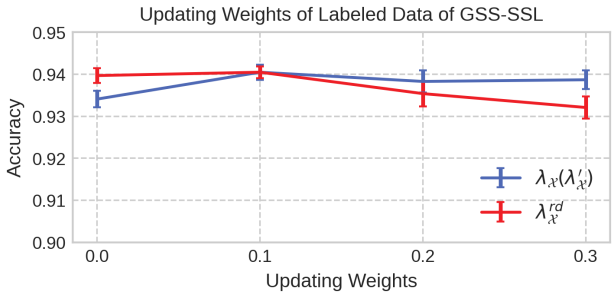

Figure S6. The ablation results with gradient direction pool updating weights on labeled samples $\lambda_{\mathcal{X}}(\lambda'_{\mathcal{X}})$ and $\lambda^{rd}_{\mathcal{X}}$ in GSS-SSL. The experiments are conducted on CIFAR-10 with 40% noisy labels with GSS-SSL. All results are the average accuracy with the error bar on the test set over five experiments.

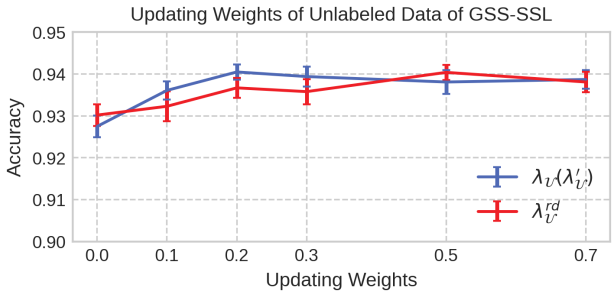

Figure S7. The ablation results with gradient direction pool updating weights on labeled samples $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}})$ and $\lambda^{rd}_{\mathcal{U}}$ in GSS-SSL. The experiments are conducted on CIFAR-10 with 40% noisy labels with GSS-SSL. All results are the average accuracy with the error bar on the test set over five experiments.

dual branches in GSS-DB apply the same weights for updating the gradient direction pools of all samples. So the ablation study is conducted with various values of λ_1 and λ_2. We use CIFAR-10 with 40% symmetric noisy labels for evaluation, and the results are shown in Fig. S5. Compared to GSS-SB, GSS-DB is less sensitive to updating weight settings. With the single branch, the performance decreases when $\lambda_1 < 1.5$. The dual branches make the model more robust that the performance only decreases when $\lambda_1 < 0.5$. Additionally, GSS-SB requires more randomness to prevent the continuous damage from misleading gradient direction, so that $\lambda_2 = 1.0$ is optimal for model training. With dual branches, GSS-DB has a more powerful ability to prevent noise damage. Consequently, small randomness is optimal that $\lambda_2 = 0.5$.

Overall, the updating weights of gradient direction pools λ_1 and λ_2 should be determined based on how much the model will be affected by noise. For the single branch, the model will be more likely to be damaged by noise, so a larger value of λ_2 is required to increase the randomness of gradient directions. In contrast, the model with dual branches is more robust to noise, and λ_2 should be set as a small value to guarantee the efficiency of training. In addition, the results seem to be insensitive to λ_1, as long as λ_1 is larger than λ_2. The model convergence will be reduced if λ_1 is too small. According to sufficient experiments, $\lambda_1 = 2.0$ is effective for GSS-SB and GSS-DB in most conditions.

For GSS-SSL, the labeled samples and unlabeled samples apply different weights to update the gradient direction pool. The weights for labeled samples are denoted as $\lambda_{\mathcal{X}}(\lambda'_{\mathcal{X}})$, $\lambda^{rd}_{\mathcal{X}}$, and the weights for unlabeled samples are denoted as $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}})$, $(\lambda^{rd}_{\mathcal{U}})$. The ablation study on each weight is conducted with optimal values of all the other weights, $\lambda_{\mathcal{X}}(\lambda'_{\mathcal{X}}) = \lambda^{rd}_{\mathcal{X}} = 0.1$, $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}}) = 0.2$, $\lambda^{rd}_{\mathcal{U}} = 0.5$. The results are shown in Fig. S6 and Fig. S7.

The results in Fig. S6 show the effects with different updating weights of labeled data. Since the labeled data selected by GMM [12] commonly have credible labels, the gradient direction pool of these samples should be more stable. The figure verifies it since a smaller $\lambda^{rd}_{\mathcal{X}}$ can achieve better performance. Compared to the results in GSS-SB and GSS-DB, GSS-SSL is less sensitive to the updating weights of labeled data. Still, with a smaller value of $\lambda_{\mathcal{X}}(\lambda'_{\mathcal{X}})$, the accuracy of GSS-SSL is improved and achieves SOTA performance.

From the results in Fig. S7, updating weights of unlabeled data show a greater influence on model performance. The optimal settings are $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}}) = 0.2$, $\lambda^{rd}_{\mathcal{U}} = 0.5$. It is different from GSS-SB and GSS-DB that the gradient direction pools of unlabeled samples have more randomness. Considering the labels in unlabeled samples are mostly wrong, $\lambda^{rd}_{\mathcal{U}}$ should be set as a larger value than $\lambda_{\mathcal{U}}(\lambda'_{\mathcal{U}})$, encouraging the model to explore in various directions to prevent the continuous noise damage.

F. Supplemented Experiments

In this section, additional experiments are supplemented to further explore the performance of GSS, including experiments on datasets with instance-related noise, quantitative evaluation with latest works, and training time comparison.

Training on datasets with instance-related noise is a challenging task, since this type of noise is highly related with the sample characteristics. Actually, for datasets with real-world noise including Clothing1M [15], WebVision [7], and ILSVRC12, the noisy label is highly correlated with the sample itself and can be regarded as instance-related noise. We show the experimental results on these datasets in the original paper. For further evaluation, experiments on CIFAR-10 with 20% instance-related noise are also conducted and shown in Table S3.

There are also some latest works for noisy-label learning, including PSE [1], CDR [14], Sel-CL [6], and SOP+ [9]. Additional experiments are conducted for comparison,

Method	CE	Co-teaching	DivideMix	GSS-SSL (Ours)
***Acc.* (%)**	86.85	88.56	91.74	92.59

Table S3. Experiments on CIFAR-10 with 20% instance-related noise.

Method	PSE	CDR	Sel-CL	SOP+	GSS-SSL (Ours)
***Acc.* (%)**	94.08	87.33	93.91	94.75	94.20

Table S4. Quantitative comparison with latest works. Experiments are conducted on CIFAR-10 with 40% symmetric noise.

Method	Baseline	Co-teaching	DivideMix	GSS-SB	GSS-SSL
Time (s)	64.11	72.14	79.47	69.36	81.10

Table S5. Training time comparison of GSS-SB and GSS-SSL with existing methods. The average time of each epoch is shown here. Experiments are conducted on CIFAR-10 with 40% symmetric noise.

which are shown in Table S4.

Also, consider the time consuming problem, we compare GSS with existing methods by the average training time of each epoch. From the results in Table S5 it can be seen the training time of GSS-SB and GSS-SSL are not much higher than other methods. That is because GSS achieve gradient switching through flipping labels, which only leads to a small increase in time.

G. Limitations of GSS

GSS has achieved significant improvement for noisy-label learning but still has some limitations. The most crucial limitation is that the hyper-parameters can be trivial to be adjusted, especially in GSS-SSL with various weights to gradient direction pool. Aiming at this limitation, we conduct the ablation study on various selections of these hyper-parameters in Section 6.2 of the original and Section E of this supplementary material. The results show the performance of GSS is insensitive to these hyper-parameters in most cases. Additionally, a small value to updating the direction pool will also improve the performance for noisy-label learning. So a simple way to solve this limitation is to gradually increase the updating weights to prevent noise damage. In future work, we will consider adjusting the gradient direction pool adaptively.

References

[1] Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34:24392–24403, 2021. 8

[2] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018. 1, 2, 6

[3] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 2, 5

[4] Abhishek Kumar and Ehsan Amid. Constrained instance and class reweighting for robust learning under label noise. *arXiv preprint arXiv:2111.05428*, 2021. 1, 2

[5] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *International Conference on Learning Representations*, 2019. 1, 2, 7

[6] Shikun Li, Xiaobo Xia, Shiming Ge, and Tongliang Liu. Selective-supervised contrastive learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 316–325, 2022. 8

[7] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017. 8

[8] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33:20331–20342, 2020. 1, 2

[9] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In *International Conference on Machine Learning*, pages 14153–14172. PMLR, 2022. 8

[10] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *International Conference on Machine Learning*, pages 6226–6236. PMLR, 2020. 1, 2

[11] Negin Majidi, Ehsan Amid, Hossein Talebi, and Manfred K Warmuth. Exponentiated gradient reweighting for robust training under label noise and beyond. *arXiv preprint arXiv:2104.01493*, 2021. 1, 2

[12] 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*, 2006. 7, 8

[13] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019. 1, 2

[14] Xiaobo Xia, Tongliang Liu, Bo Han, Chen Gong, Nannan Wang, Zongyuan Ge, and Yi Chang. Robust early-learning: Hindering the memorization of noisy labels. In *International conference on learning representations*, 2021. 8

[15] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 8

[16] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018. 1, 2