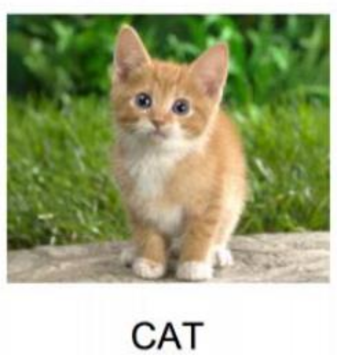
Learning to Reweight Examples for Robust Deep Learning

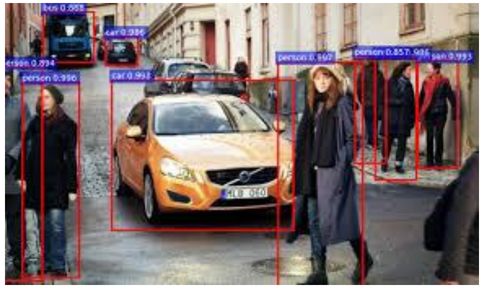
Mengye Ren [1,2] Wenyuan Zeng [1,2] Bin Yang [1,2] Raquel Urtasun [1,2]

ICML 2018

Introduction

DNNs have been shown to be very powerful modeling tools in many supervised tasks…





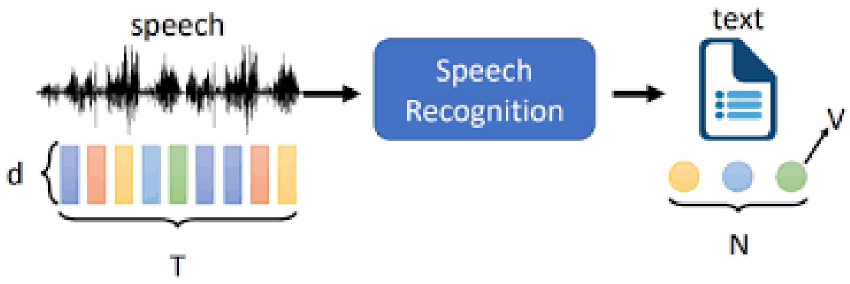


Image classification

Object detection

Speech recognition

Unfortunately, DNNs can easily be overfitting to training set bias

- Label noise
- Class imbalance

Motivation

How to deal with training set bias ?

Data resampling:

choosing the correct proportion of labels to train a network

assigning a weight to each example and minimizing a weighted training loss

Representative methods

- ✓ AdaBoost
- ✓ Self-pace Learning

} Based on training loss

Motivation

However, there existing two contradicting ideas in these training loss based methods.

	Label noise problems	Class imbalance problems
Preferable	examples with smaller training losses	examples with larger training losses
Reason	being likely to be clean images	being likely to be minority class

Claim: in order to learn general forms of training set biases, it is necessary to have a small unbiased validation set to guide training

Assumption: the best example weighting should minimize the loss of a set of unbiased clean validation examples

Method

Notations

Training set: $\{(x_i, y_i), 1 \leq i \leq N\}$

Validation set: $\{(x_i^v, y_i^v), 1 \leq i \leq M\}$ $M \ll N$

Neural network: $\Phi(x, \theta)$

Loss function: $C(\hat{y}, y)$ where $\hat{y} = \Phi(x, \theta)$

In standard training, we aim to minimize the expected loss for the training set where each input example is weighted equally

$$\frac{1}{N} \sum_{i=1}^{N} C(\hat{y}_i, y_i) = \frac{1}{N} \sum_{i=1}^{N} f_i(\theta)$$

where $f_i(\theta)$ calculates the loss for example x_i

Method: basic idea

To deal with training bias, we aim to learn a reweighting loss of the training examples

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Note that $\{\omega_i\}_{i=1}^{N}$ can be regarded as training hyperparameters

The optimal selection of ω is based on its validation performance

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

However, note that calculating ω requires two nested loops of optimization…

Method: approximation

At every step t of training, we update the parameters with a mini-batch of training examples $\{(x_i, y_i), 1 \leq i \leq n\}$ by using SGD

$$\theta_{t+1} = \theta_t - \alpha \nabla \left(\frac{1}{n} \sum_{i=1}^{n} f_i(\theta_t) \right)$$

We want to understand the impact of training example towards the performance of the validation set at training step t.

$$f_{i,\epsilon}(\theta) = \epsilon_i f_i(\theta),$$

$$\hat{\theta}_{t+1}(\epsilon) = \theta_t - \alpha \nabla \sum_{i=1}^{n} f_{i,\epsilon}(\theta) \Big|_{\theta=\theta_t}$$

Obtain the optimal ϵ^* by minimizing the validation loss

$$\epsilon_t^* = \arg\min_{\epsilon} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta_{t+1}(\epsilon))$$

get a cheap estimate by taking a single gradient descent step

$$u_{i,t} = -\eta \frac{\partial}{\partial \epsilon_{i,t}} \frac{1}{m} \sum_{j=1}^{m} f_j^v(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0}$$

$$\tilde{w}_{i,t} = \max(u_{i,t}, 0).$$

Method: approximation

Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

Require: θ_0, $\mathcal{D}_f$, $\mathcal{D}_g$, n, m

Ensure: θ_T

for $t = 0 \ldots T-1$ **do**

$\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f$, n)

$\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g$, m)

$\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$

$\epsilon \leftarrow 0$; $l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$

$\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$

$\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$

$l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$

$\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$

$\tilde{w} \leftarrow \max(-\nabla\epsilon, 0)$; $w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$

$\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$

$\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$

end for

$$f_{i,\epsilon}(\theta) = \epsilon_i f_i(\theta),$$

$$\hat{\theta}_{t+1}(\epsilon) = \theta_t - \alpha\nabla\sum_{i=1}^{n} f_{i,\epsilon}(\theta)\Big|_{\theta=\theta_t}$$

$$u_{i,t} = -\eta\frac{\partial}{\partial\epsilon_{i,t}}\frac{1}{m}\sum_{j=1}^{m} f_j^v(\theta_{t+1}(\epsilon))\Big|_{\epsilon_{i,t}=0}$$

$$\tilde{w}_{i,t} = \max(u_{i,t}, 0).$$

Example: learning to reweight in a MLP

Consider parameters for each layer $\theta = \{\theta_l\}_{l=1}^{L}$ we have the outputs of each layer

$$z_l = \theta_l^\top \tilde{z}_{l-1}$$
$$\tilde{z}_l = \sigma(z_l).$$

Then the gradient towards ϵ can be expressed by a sum of local dot products

$$\frac{\partial}{\partial \epsilon_{i,t}} \mathbb{E}\left[f^v(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0} \right]$$
$$\propto -\frac{1}{m} \sum_{j=1}^{m} \frac{\partial f_j^v(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}^\top \frac{\partial f_i(\theta)}{\partial \theta} \Big|_{\theta=\theta_t}$$
$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} ({\tilde{z}_{j,l-1}^v}^\top \tilde{z}_{i,l-1})({g_{j,l}^v}^\top g_{i,l})$$

the gradients of loss wrt. z_l

the similarity between the training and validation inputs t o the layer

the similarity between the training and validation gradient direction

Experiments: data imbalance

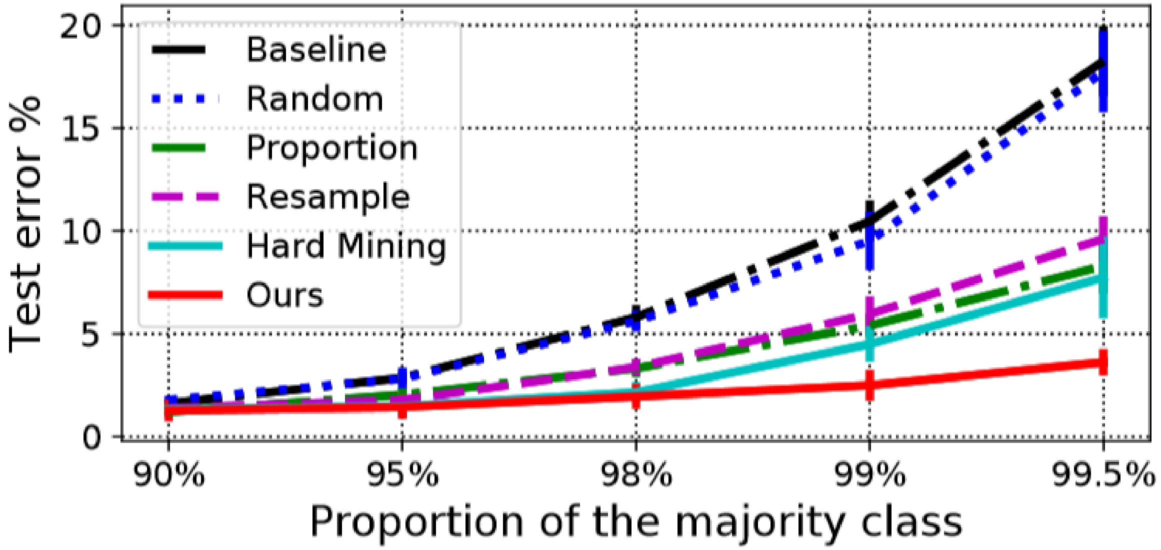


Figure 2. MNIST 4-9 binary classification error using a LeNet on imbalanced classes. Our method uses a small balanced validation split of 10 examples.

1) PROPORTION weights each example by the inverse frequency 2) RESAMPLE samples a class-balanced mini-batch for each iteration 3) HARD MINING selects the highest loss examples from the majority class and 4) RANDOM is a random example weight baseline that assigns weights based on a rectified Gaussian distribution:

$$w_i^{\text{rnd}} = \frac{\max(z_i, 0)}{\sum_i \max(z_i, 0)}, \quad \text{where } z_i \sim \mathcal{N}(0, 1). \qquad (16)$$

Experiments: label noise

Table 1. CIFAR UNIFORMFLIP under 40% noise ratio using a WideResNet-28-10 model. Test accuracy shown in percentage. Top rows use only noisy data, and bottom uses additional 1000 clean images. "FT" denotes fine-tuning on clean data.

MODEL	CIFAR-10	CIFAR-100
BASELINE	67.97 ± 0.62	50.66 ± 0.24
REED-HARD	69.66 ± 1.21	51.34 ± 0.17
S-MODEL	70.64 ± 3.09	49.10 ± 0.58
MENTORNET	76.6	56.9
RANDOM	$86.06 \pm 0.32.$	58.01 ± 0.37
USING 1,000 CLEAN IMAGES		
CLEAN ONLY	46.64 ± 3.90	9.94 ± 0.82
BASELINE +FT	78.66 ± 0.44	54.52 ± 0.40
MENTORNET +FT	78	59
RANDOM +FT	86.55 ± 0.24	58.54 ± 0.52
OURS	$\mathbf{86.92 \pm 0.19}$	$\mathbf{61.34 \pm 2.06}$

- UNIFORMFLIP: All label classes can uniformly flip to any other label classes, which is the most studied in the literature.

- REED, proposed by Reed et al. (2014), is a bootstrapping technique where the training target is a convex combination of the model prediction and the label.

- S-MODEL, proposed by Goldberger & Ben-Reuven (2017), adds a fully connected softmax layer after the regular classification output layer to model the noise transition matrix.

- MENTORNET, proposed by Jiang et al. (2017), is an RNN-based meta-learning model that takes in a sequence of loss values and outputs the example weights. We compare numbers reported in their paper with a base model that achieves similar test accuracy under 0% noise.

Experiments: label noise

Table 2. CIFAR BACKGROUNDFLIP under 40% noise ratio using a ResNet-32 model. Test accuracy shown in percentage. Top rows use only noisy data, and bottom rows use additional 10 clean images per class. “+ES” denotes early stopping; “FT” denotes fine-tuning.

MODEL	CIFAR-10	CIFAR-100
BASELINE	59.54 ± 2.16	37.82 ± 0.69
BASELINE +ES	64.96 ± 1.19	39.08 ± 0.65
RANDOM	69.51 ± 1.36	36.56 ± 0.44
WEIGHTED	79.17 ± 1.36	36.56 ± 0.44
REED SOFT +ES	63.47 ± 1.05	38.44 ± 0.90
REED HARD +ES	65.22 ± 1.06	39.03 ± 0.55
S-MODEL	58.60 ± 2.33	37.02 ± 0.34
S-MODEL +CONF	68.93 ± 1.09	46.72 ± 1.87
S-MODEL +CONF +ES	79.24 ± 0.56	54.50 ± 2.51
USING 10 CLEAN IMAGES PER CLASS		
CLEAN ONLY	15.90 ± 3.32	8.06 ± 0.76
BASELINE +FT	82.82 ± 0.93	54.23 ± 1.75
BASELINE +ES +FT	85.19 ± 0.46	55.22 ± 1.40
WEIGHTED +FT	85.98 ± 0.47	53.99 ± 1.62
S-MODEL +CONF +FT	81.90 ± 0.85	53.11 ± 1.33
S-MODEL +CONF +ES +FT	85.86 ± 0.63	55.75 ± 1.26
OURS	$\mathbf{86.73} \pm \mathbf{0.48}$	$\mathbf{59.30} \pm \mathbf{0.60}$

- BACKGROUNDFLIP: All label classes can flip to a single background class. This noise setting is very realistic. For instance, human annotators may not have recognized all the positive instances, while the rest remain in the background class. This is also a combination of label imbalance and label noise since the background class usually dominates the label distribution.

Experiments: understanding the reweighting

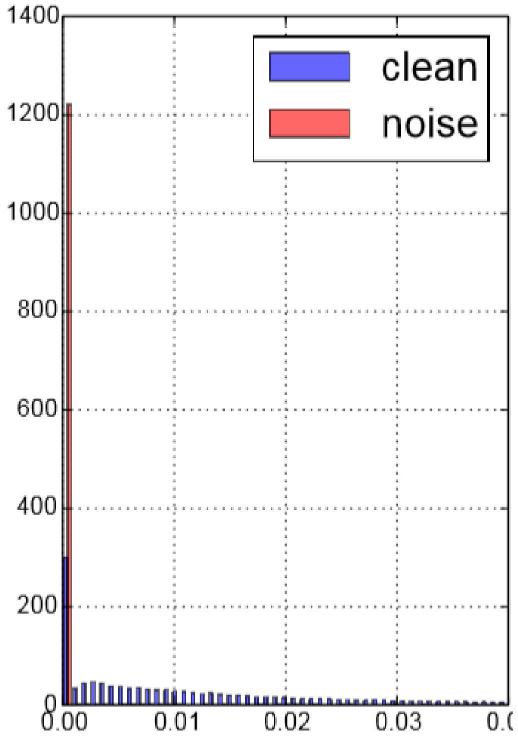


Figure 3. Example weights d
Left: a hyper-validation batch
noises. Right: a hyper-validat
label class, with flipped bacl
non-background classes.

Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$

Ensure: θ_T

for $t = 0 \ldots T-1$ **do**

$\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$

$\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$

$\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$

$\epsilon \leftarrow 0;\ l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$

$\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$

$\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$

$l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$

$\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$

$\tilde{w} \leftarrow \max(-\nabla\epsilon, 0);\ w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$

$\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$

$\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$

end for

ıet(trained at half of

ple weight distribution
d batch of validation

Experiments: Robustness to overfitting noise

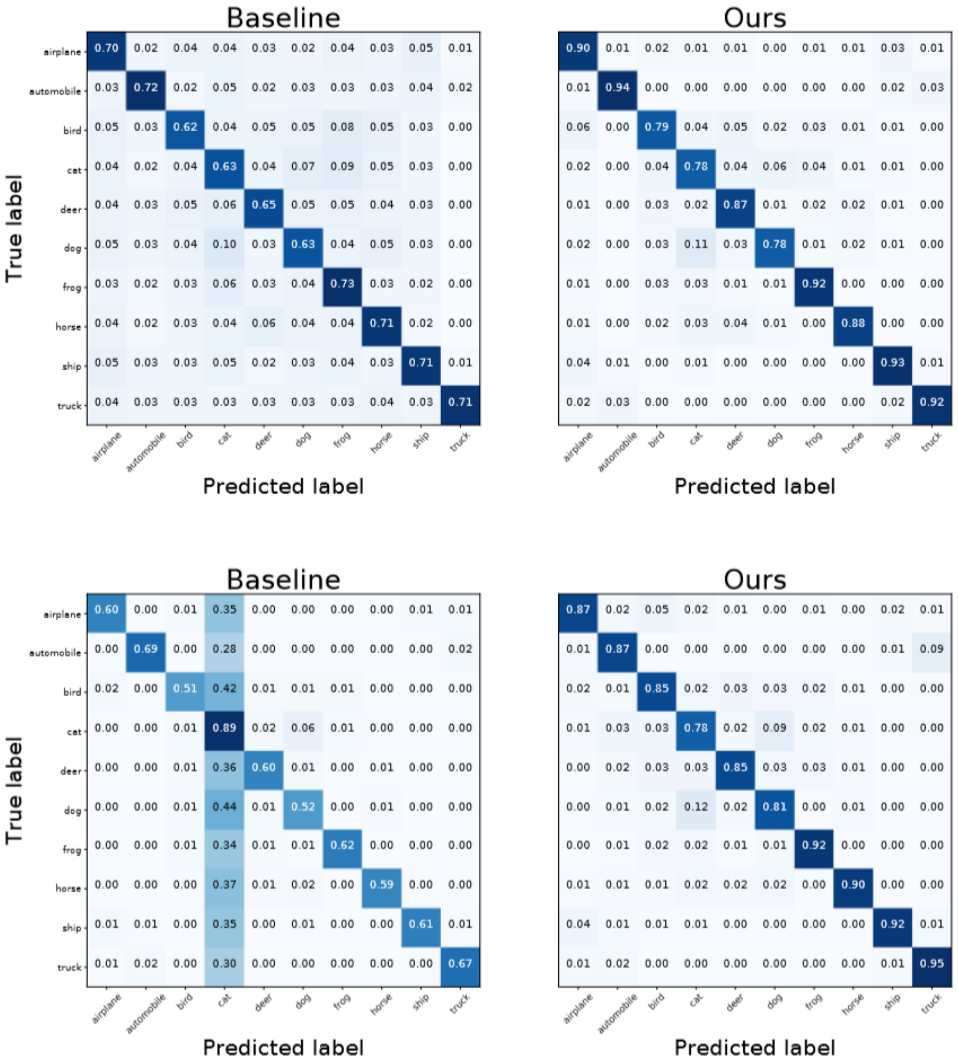
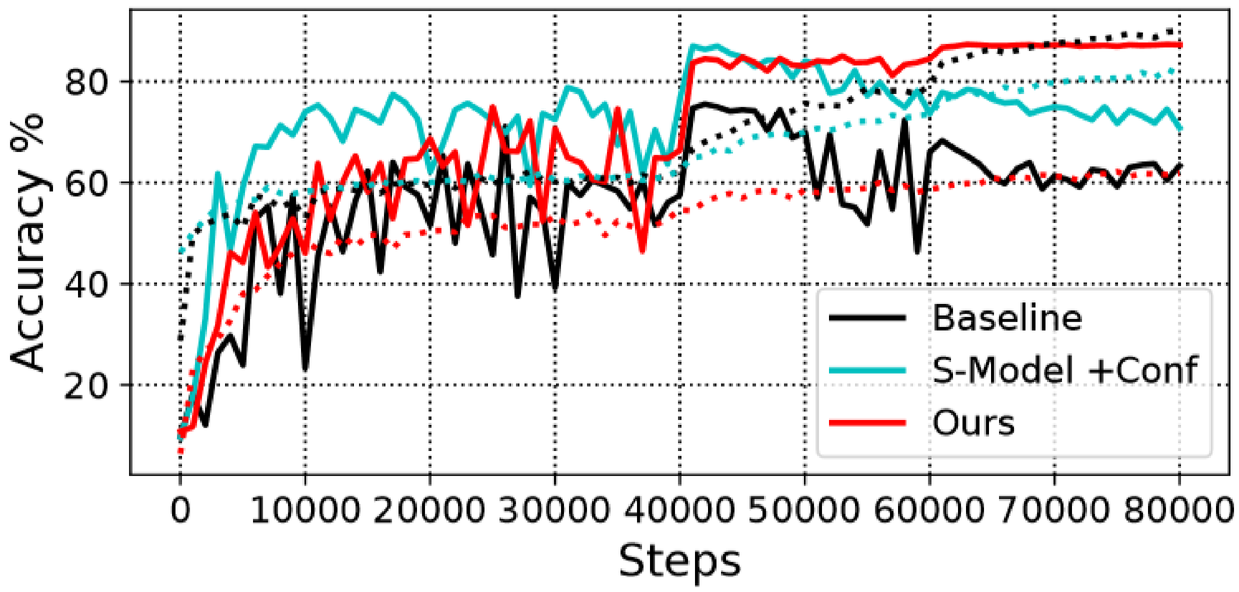


Figure 6. Confusion matrices on CIFAR-10 UNIFORMFLIP (top) and BACKGROUNDFLIP (bottom)



Experiments: Robustness to overfitting noise

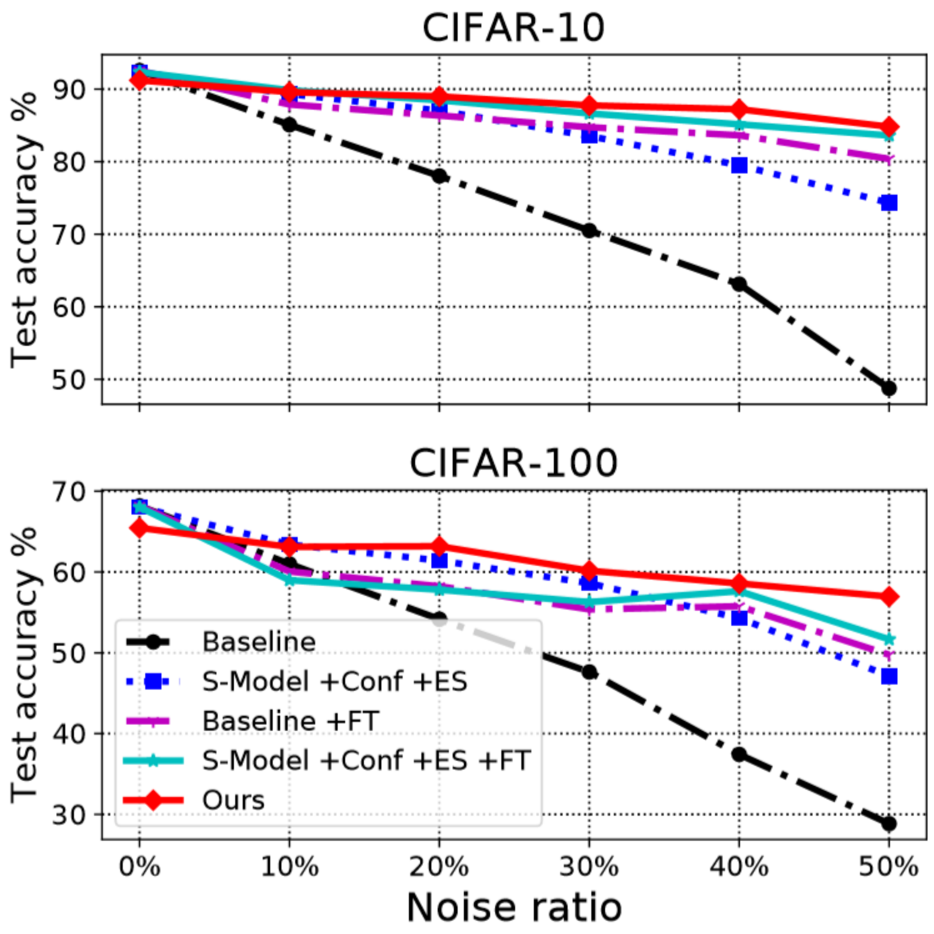


Figure 5. Model test accuracy on imbalanced noisy CIFAR experiments across various noise levels using a base ResNet-32 model. "ES" denotes early stopping, and "FT" denotes finetuning.

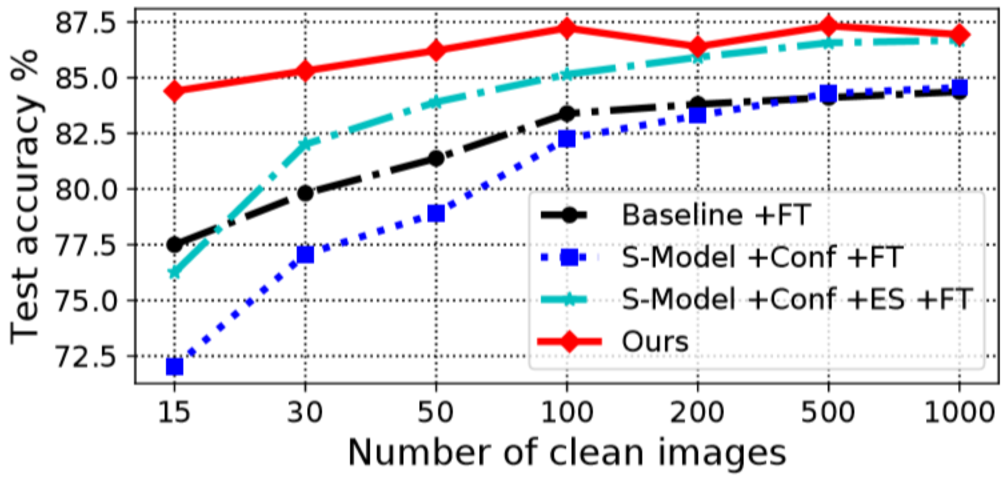


Figure 4. Effect of the number of clean imaged used, on CIFAR-10 with 40% of data flipped to label 3. "ES" denotes early stopping.

Thanks

$$\frac{\partial}{\partial \epsilon_{i,t}} \mathbb{E}\left[f^v(\theta_{t+1}(\epsilon))\right]\Big|_{\epsilon_{i,t}=0} \tag{17}$$

$$= \frac{1}{m} \sum_{j=1}^{m} \frac{\partial}{\partial \epsilon_{i,t}} f_j^v(\theta_{t+1}(\epsilon))\Big|_{\epsilon_{i,t}=0} \tag{18}$$

$$= \frac{1}{m} \sum_{j=1}^{m} \frac{\partial f_j^v(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}^{\top} \frac{\partial \theta_{t+1}(\epsilon_{i,t})}{\partial \epsilon_{i,t}}\Big|_{\epsilon_{i,t}=0} \tag{19}$$

$$\propto -\frac{1}{m} \sum_{j=1}^{m} \frac{\partial f_j^v(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}^{\top} \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t} \tag{20}$$

$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} \frac{\partial f_j^v}{\partial \theta_l}\Big|_{\theta_l=\theta_{l,t}}^{\top} \frac{\partial f_i}{\partial \theta_l}\Big|_{\theta_l=\theta_{l,t}} \tag{21}$$

$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} \operatorname{vec}\left(\tilde{z}_{j,l-1}^{v} {g_{j,l}^{v}}^{\top}\right)^{\top} \operatorname{vec}\left(\tilde{z}_{i,l-1} g_{i,l}^{\top}\right) \tag{22}$$

$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \tilde{z}_{j,l-1,p}^{v} g_{j,l,q}^{v} \tilde{z}_{i,l-1,p} g_{i,l,q} \tag{23}$$

$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} \sum_{p=1}^{D_1} \tilde{z}_{j,l-1,p}^{v} \tilde{z}_{i,l-1,p} \sum_{q=1}^{D_2} g_{j,l,q}^{v} g_{i,l,q} \tag{24}$$

$$= -\frac{1}{m} \sum_{j=1}^{m} \sum_{l=1}^{L} ({\tilde{z}_{j,l-1}^{v}}^{\top} \tilde{z}_{i,l-1})({g_{j,l}^{v}}^{\top} g_{i,l}). \tag{25}$$