ROBUST TRAINING WITH ENSEMBLE CONSENSUS

Anonymous authors
Paper under double-blind review

ABSTRACT

Since deep neural networks are over-parametrized, they may memorize noisy examples. We address such memorizing issue under the existence of annotation noise. From the fact that deep neural networks cannot generalize neighborhoods of the features acquired via memorization, we find that noisy examples do not consistently incur small losses on the network in the presence of perturbation. Based on this, we propose a novel training method called *Learning with Ensemble Consensus* (LEC) whose goal is to prevent overfitting noisy examples by eliminating them identified via consensus of an ensemble of perturbed networks. One of the proposed LECs, LTEC outperforms the current state-of-the-art methods on MNIST, CIFAR-10, and CIFAR-100 despite its efficient memory usage.

1 INTRODUCTION

Deep neural networks (DNNs) have shown excellent performance (Krizhevsky et al., 2012; He et al., 2016) on visual recognition datasets (Deng et al., 2009). However, it is difficult to obtain annotated datasets of such high quality in practice (Wang et al., 2018a). Even worse, DNNs may not generalize training data in the presence of noisy examples (Zhang et al., 2016). Therefore, there is an increasing demand for robust training methods. In general, DNNs trained on noisy datasets first generalize clean examples (Arpit et al., 2017). Based on this, recent studies consider examples that incur small training losses in the early stage of training as being clean (Han et al., 2018; Yu et al., 2019). However, small-loss examples may be highly corrupted especially when the training set contains a high level of annotation noise.

Therefore, choosing safe examples from the noisy training set with small-loss criteria can be impractical. To address this, we attempt to find a way to discriminate clean and noisy examples within small-loss examples. Since mislabeling reduces the correlation with other examples, it is likely that noisy examples are learned via memorization rather than generalization. In general, under the existence of a small perturbation, network predictions for memorized features easily fluctuate, while those for generalized features do not. Based on this, we hypothesize that out of small-loss examples, training losses of noisy examples would change dynamically by injecting perturbation, while those of clean examples would not. This suggests that clean examples can be found out of small-loss examples by selecting examples consistently incurring small losses on an ensemble of perturbed networks, i.e., by selecting via ensemble consensus on small-loss examples. This idea comes from the difference between generalization and memorization, thus it can be utilized for any architecture optimized with the gradient-based method.

In this work, we introduce simple perturbation methods to generate the ensemble for discriminating clean and noisy examples. By embedding those perturbation methods into training, we propose a new robust training scheme termed *learning with ensemble consensus* (LEC). In LEC, the network is first trained on the entire training set for a while and then trained on examples selected based on whether training losses are consistently small across the ensemble of perturbed networks. We present three LECs with different perturbations: LNEC, LSEC, and LTEC, and empirically show that three LECs are effective at identifying noisy examples out of small-loss examples. In particular, one of LECs, LTEC outperforms existing robust training methods on three benchmark datasets with random label noise (Goldberger & Ben-Reuven, 2016; Ma et al., 2018) and open-set noise (Wang et al., 2018b).

2 Related Work

Generalization of DNNs. Although DNNs are over-parametrized, they have impressive generalization ability (Krizhevsky et al., 2012; He et al., 2016) . To explain this, some studies argue that gradient-based optimization plays an important role in regularizing DNNs (Neyshabur et al., 2014; Zhang et al., 2016). As a result, DNNs optimized with SGD first generalize and then memorize the training set (Krueger et al., 2017; Arpit et al., 2017). Due to the difference of correlation with other examples within the training set, in general, noisy examples are learned via memorization, while some clean examples are learned via generalization. Therefore, in order to discriminate clean and noisy examples, we analyze the difference between generalized features and memorized features.

Training DNNs with Noisy datasets. Annotation issues can be addressed by reducing the impact of noisy examples. One direction is to train with a modified loss function based on the noise distribution. Most studies of this direction estimate the noise distribution prior to training as it is not accessible in general (Sukhbaatar et al., 2014; Goldberger & Ben-Reuven, 2016; Patrini et al., 2017; Hendrycks et al., 2018). Without going through the estimation step, there is another direction to train with modified labels by using the current model prediction (Reed et al., 2014; Ma et al., 2018). Aside from both modification methods, recent work suggests a method of exploiting small-loss examples (Jiang et al., 2017; Han et al., 2018; Yu et al., 2019). The effectiveness of these studies comes from the property of DNNs optimized with SGD where a considerable portion of small-loss examples during generalization is well-annotated (Han et al., 2018). However, it is still challenging to determine examples to be learned by using only training losses. In this study, we propose a simple method to overcome such problem of small-loss criteria to find clean examples.

3 Robust training with Ensemble Consensus

3.1 Problem statement

Assume that a network trained on a dataset containing ϵ% annotation noise learns all clean examples without fitting noisy examples. Then (100-ϵ)% small-loss examples of the network are all clean. However, it is generally hard to learn all clean examples at the beginning particularly on the highly corrupted training set. Therefore, it may be problematic to consider (100-ϵ)% small-loss examples as being clean. To mitigate this, we suggest a simple idea: to find noisy examples among (100-ϵ)% small-loss examples.

3.2 Learning with Ensemble Consensus (LEC)

DNNs learn features either via generalization or via memorization (Krueger et al., 2017). However, the networks cannot generalize neighborhoods of the memorized features. In general, noisy examples are learned via memorization. Therefore, even if a network trained on a noisy dataset $D_{clean} \cup D_{noisy}$ fits a certain noisy example, i.e., $\arg\max f(x;\theta) = y$ where $(x, y) \in D_{noisy}$, it is difficult to fit that noisy example even in the presence of perturbation δ. This can be expressed as follows:

$$d(f(x;\theta), y) < \zeta \Rightarrow d(f(x;\theta+\delta), y) > \zeta \text{ for } (x, y) \in D_{noisy}$$

where d indicates the cross-entropy loss for a classification task. Unlike noisy examples, some clean examples can be generalized in the early stage of training. This suggests that network predictions for the generalized clean examples do not fluctuate by injecting perturbation δ as follows:

$$d(f(x;\theta), y) < \zeta \Rightarrow d(f(x;\theta+\delta), y) < \zeta \text{ for } (x, y) \in D_{clean}$$

On a dataset with noise ratio of ϵ%, our goal is to train network $f(;\theta)$ with only clean examples by removing noisy examples from (100-ϵ)% small-loss examples. To this end, we discriminate clean and noisy examples by exploiting the consistency of training losses in the presence of perturbation δ. More precisely, the network is trained on examples consistently incurring small losses across an ensemble of perturbed networks. We call it ***ensemble consensus filtering*** because examples to be trained are selected via ensemble consensus on small-loss examples.

Algorithm 1 Learning with Ensemble Consensus (LEC)

```
Require: noisy dataset $\mathcal{D}$, noise ratio $\epsilon\%$, duration of Warming-up $T_w$, # of predictions $M$, perturbation r.v. $\delta$
 1: Initialize $\theta$ randomly
 2: for $t = 1 : T_w$ do                                              ▶ Warming-up process
 3:     for mini-batch index $b = 1 : B$ do
 4:         $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{|\mathcal{B}_b|} \sum_{(x,y)\in\mathcal{B}_b} CE(f_\theta(x), y)$
 5:     end for
 6: end for
 7: for $t = T_w + 1 : T_{end}$ do                                    ▶ Filtering process
 8:     for mini-batch index $b = 1 : B$ do
 9:         for $m = 1 : M$ do
10:             $\theta_m = \theta + \delta_m$ where $\delta_m \sim \delta$               ▷ Perturbation
11:             Find $\mathcal{S}_{m,b}$ := $(100 - \epsilon)\%$ small-loss examples of $f_{\theta_m}$ within $\mathcal{B}_b$
12:         end for
13:         $\mathcal{B}_b{}' = \mathcal{S}_{1,b} \cap \mathcal{S}_{2,b} \cap ... \cap \mathcal{S}_{M,b}$          ▷ Ensemble consensus filtering
14:         $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{|\mathcal{B}_b{}'|} \sum_{(x,y)\in\mathcal{B}_b{}'} CE(f_\theta(x), y)$
15:     end for
16: end for
```

The pseudocode for our proposed LEC is described in Algorithm 1. We assume that noise ratio $\epsilon\%$ is accessible. We remark that it is easier to estimate the noise ratio than the noise distribution. Our LEC algorithm consists of Warming-up and Filtering processes. During T_w epochs of Warming-up process, the network is trained on all training examples. The goal of this process is to obtain θ which generalizes clean examples as many as possible. Therefore, other techniques such as pre-training (Hendrycks et al., 2019) may be adopted. During Filtering process, for each batch update, an ensemble is generated by adding perturbation δ to the network M times. Then the network is trained on the intersection of (100-ϵ)% small-loss examples of networks in the ensemble within a mini-batch.

3.3 Perturbation to identify noisy examples

Depending on the way of perturbing the network δ, various LECs can be created. In the following, we present three LECs with different perturbations to make the ensemble. The pseudocodes for the following LECs can be found in Section A.1.

- **Network-Ensemble Consensus (LNEC)** : Inspired by the observation that an ensemble of networks is correlated during generalization and is decorrelated during memorization (Morcos et al., 2018), the perturbation of prediction δ comes from the difference between the ensemble of multiple networks with the same architecture. During Warming-up process, M networks with the same architecture are trained independently. During Filtering process, for each batch update, the ensemble of M networks is trained on the intersection of (100-ϵ)% small-loss examples of M networks within a mini-batch.
- **Self-Ensemble Consensus (LSEC)** : We obtain insights from Morcos et al. (2018) and Lakshminarayanan et al. (2017): network predictions for memorized features are uncertain and those for generalized features are certain. By noting that the uncertainty of predictions also can be captured by multiple stochastic predictions (Gal & Ghahramani, 2016), the perturbation of prediction δ comes from the difference between multiple stochastic predictions. Here, the randomness is caused by stochastic operations as in Laine & Aila (2016). During Filtering process, for each batch update, the network is trained on the intersection of (100-ϵ)% small-loss examples evaluated with M stochastic predictions within a mini-batch.
- **Temporal-Ensemble Consensus (LTEC)** : Inspired by the observation that during training, atypical features are more easily forgetful compared to typical features (Toneva et al., 2018), the perturbation of prediction δ comes from the difference between predictions at the current and the preceding epochs. During Filtering process, for each batch update, the network is trained on the intersection of (100-ϵ)% small-loss examples evaluated with the current prediction within a mini-batch and those evaluated at the preceding $M - 1$ epochs. In order to reduce memory usage, we collect (100-ϵ)% small-loss examples obtained for each batch update at the preceding $M - 1$ epochs, rather than network parameters.

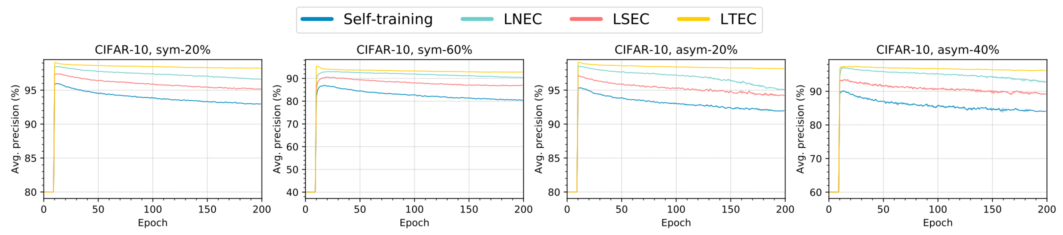


Figure 1: **Label precision** (%) of Self-training and three LECs on CIFAR-10 with random label noise. We plot the average as a solid line and the standard deviation as a shadow around the line over 4 runs.

4 Experiments

In this section, we will empirically verify (i) whether three proposed perturbation methods are effective for screening noisy examples in small-loss examples and (ii) whether training on examples selected via ensemble consensus filtering improves test accuracy.

4.1 Experimental setup

Annotation noise. We study two representative types of annotation noise: random label noise (Goldberger & Ben-Reuven, 2016; Ma et al., 2018) and open-set noise (Wang et al., 2018b). To simulate these types of noise, we corrupt three benchmark datasets: MNIST (LeCun et al., 1998), CIFAR-10/100 (Krizhevsky et al., 2009) that are commonly used to assess the robustness. For each benchmark dataset, we only corrupt its training set, while leaving its test set intact for testing.

- **Random label noise.** Annotation issues can occur in easy images as well as hard images in practice (Wang et al., 2018a). We simulate this by mislabeling some of the training images randomly chosen in two ways: **sym-ϵ%** and **asym-ϵ%**. For sym-ϵ%, ϵ% of examples are randomly mislabeled to one of the other labels and for asym-ϵ%, each label of ϵ% of examples i is changed to $i+1$. Specifically, in order to mimic a low and a high level of label noise, we choose sym-20% and asym-20%, and sym-60% and asym-40%, respectively.
- **Open-set noise.** In reality, annotated datasets may contain out-of-distribution data. To mimic such out-of-distribution data in training set as in Wang et al. (2018b), we replace ϵ% of training images that are randomly chosen with images sampled from other source datasets. We conduct the experiments on CIFAR with 20% and 40% open-noise.

Architecture and optimization. Unless otherwise specified, we use a variant of 9-convolutional layer architecture (Laine & Aila, 2016; Han et al., 2018). All parameters are trained for 200 epochs with Adam (Kingma & Ba, 2014) with a batchsize of 128. The details can be found in Section A.2.2.

Evaluation. To evaluate the robustness against annotation noise, we use two metrics: test accuracy and label precision (Han et al., 2018). At the end of each epoch, test accuracy is measured as the ratio of correctly predicted test examples to all test examples and label precision is measured as the ratio of clean examples to examples used for training. Thus, for both metrics, higher is better. For methods with multiple networks, the averaged precision and the averaged accuracy are reported. We report peak accuracy in the course of training as well as final accuracy because test accuracy can be measured with a small validation set during training in practice.

Hyperparameter. Our proposed LEC involves three hyperparameters: duration of Warming-up T_w, noise ratio ϵ%, and the number of predictions M. Unless otherwise specified, T_w and M are set to 10 and 5, respectively, and we assume that noise ratio ϵ% is given.

4.2 Effectiveness of LECs at identifying noisy examples

Comparison with Self-training. In Section 3.1, we argue that on a dataset with ϵ% noise, it is inappropriate to regard (100-ϵ)% small-loss as clean examples. To show this, we train 9-conv architecture on the entire training set for 10 epochs and then on (100-ϵ)% small-loss examples for

Table 1: Average of **final/peak test accuracy** (%) over 4 runs of Self-training and three LECs on CIFAR-10 with random label noise. The best is highlighted in **bold**.

Dataset	Noise type	Self-training	LNEC	LSEC	LTEC
CIFAR-10	sym-20%	84.96/85.02	86.72/86.78	85.42/85.63	88.18/**88.28**
	sym-60%	73.99/74.35	79.61/79.64	76.73/76.92	80.38/**80.52**
	asym-20%	85.02/85.24	86.90/87.11	85.44/85.64	88.86/**88.93**
	asym-40%	78.84/79.66	84.01/84.48	80.74/81.49	86.36/**86.50**

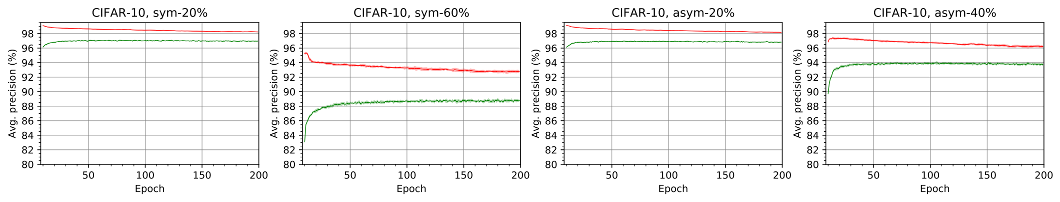


Figure 2: **Label precision** (%) of after (in red) and before (in green) consensus filtering in LTEC on CIFAR-10 with random label noise. We plot the average as a solid line and the standard deviation as a shadow around the line over 4 runs.

the remaining epochs. This methodology is similar to the idea of Jiang et al. (2017). For simplicity, we call it ***Self-training***. As shown in Figure 1, on CIFAR-10 with random label noise, the label precision of Self-training is quite high. This suggests that a substantial portion of (100-ϵ)% small-loss examples is well-annotated. However, the label precision of (100-ϵ)% small-loss examples decreases as the noise ratio ϵ% increases. Indeed, this shows the unreliability of small-loss criteria to identify clean examples. Compared to Self-training, our proposed LECs are trained on examples of higher label precision, resulting in higher test accuracy as seen in Table 1. Out of three LECs, LTEC performs the best in both label precision and test accuracy. Furthermore, Figure 1 states that the label precision of (100-ϵ)% small-loss examples in LTEC does not decrease as training proceeds. This indicates that filtering via temporal-ensemble consensus is most effective in identifying noisy examples out of small-loss examples on CIFAR-10.

Noisy examples are removed via filtering. In order to examine whether noisy examples are removed by ensemble consensus filtering, we compare label precision of (100-ϵ)% small-loss examples (before) and that of remaining examples after ensemble consensus filtering on those small-loss examples (after) in the course of running LTEC. As seen in Figure 2, on CIFAR-10 with random label noise, the label precision of after filtering is always higher than that of before filtering. Specifically, the gain of filtering is larger on sym-60% and asym-40%. This implies that as learning clean examples in the early stage gets more difficult, (100-ϵ)% small-loss examples become more corrupted, resulting in higher contribution of the filtering.

4.3 Comparison with state-of-the-art methods

Under the existence of annotation noise, we compare the performance of our proposed LTEC with those of other existing methods.

Competing methods. The competing methods include a regular training method: ***Standard***, a method of training with corrected labels: ***D2L***, a method of training with modified loss function based on the noise distribution: ***Forward***, and a method of exploiting small-loss examples: ***Co-teaching***. We tune all the methods individually as described in Section A.2.3.

Results on MNIST/CIFAR with random label noise. The overall results can be found in Figure 3, Figure 4, and Table 2. Figure 3 states that the test accuracies of Standard and D2L sharply drop in the early stage of training. In the highly corrupted settings such as sym-40% and asym-60%, the test accuracy of D2L does not increase as training progresses. Since D2L puts large weights on given labels in the early stage, the performance of D2L is relevant to that of Standard. Forward shows its strength only in a few settings. Co-teaching does not work well on CIFAR-100 with asym-40%. This reveals that the cross-training scheme of Co-teaching is vulnerable to small-loss

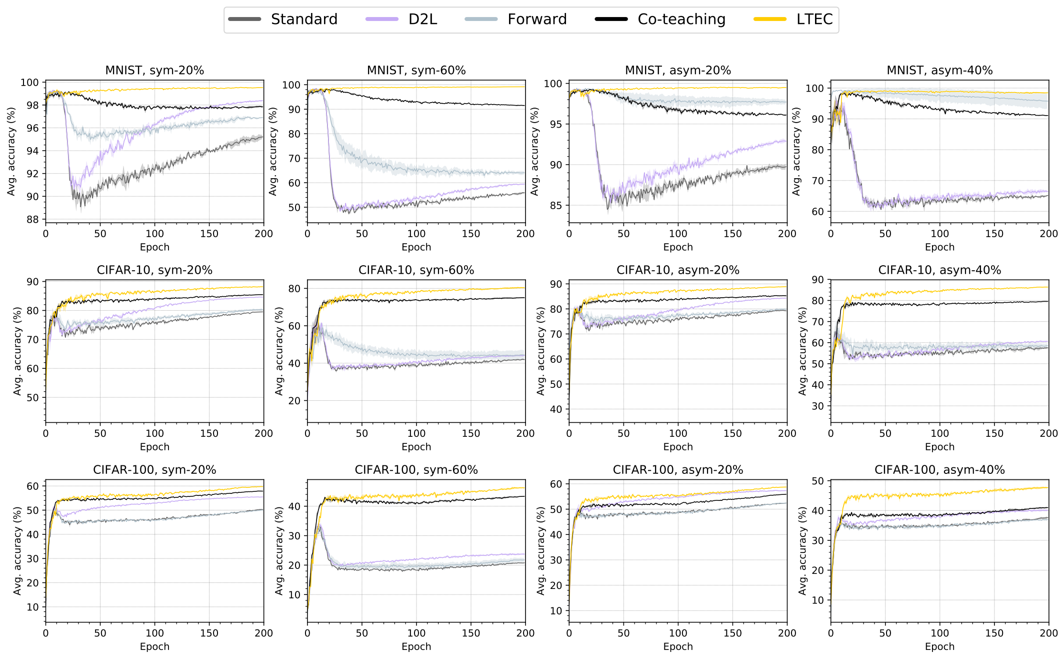


Figure 3: **Test accuracy** (%) of different algorithms on random label noise. We plot the average as a solid line and the standard deviation as a shadow around the line over 4 runs.

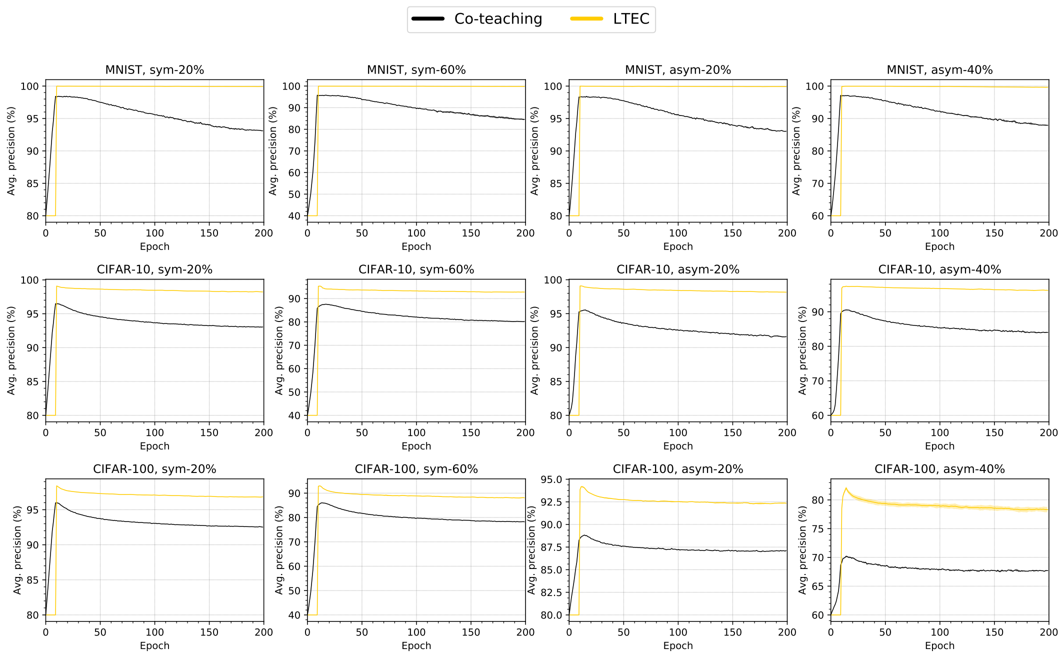


Figure 4: **Label precision** (%) of different algorithms on random label noise. We plot the average as a solid line and the standard deviation as a shadow around the line over 4 runs.

examples of a low label precision (see Figure 4). Unlike Co-teaching, LTEC attempts to prevent the network to be trained with noisy examples by eliminating them. Therefore, LTEC performs well in all the settings. Specifically, on CIFAR-100 with asym-40% noise, LTEC surpasses the second-best method by a wide margin of about 6% with less memory usage. (see Table 2).

Table 2: Average of **final/peak test accuracy** (%) over 4 runs of different algorithms on MNIST/CIFAR with random label noise. The best is highlighted in **bold**.

Dataset	Noise type	Standard	D2L	Forward	Co-teaching	LTEC (ours)
MNIST	sym-20%	95.21/99.36	98.38/99.35	96.88/99.29	97.84/99.24	99.52/**99.58**
	sym-60%	55.88/98.50	59.40/98.37	64.03/98.26	91.52/98.53	99.16/**99.25**
	asym-20%	89.74/99.32	92.88/99.41	97.71/99.52	96.11/99.40	99.49/**99.59**
	asym-40%	65.13/96.58	66.44/96.99	95.76/**99.51**	91.10/98.81	98.47/99.32
CIFAR-10	sym-20%	79.50/80.74	84.60/84.68	80.29/80.91	85.46/85.52	88.18/**88.28**
	sym-60%	41.91/65.06	44.10/65.26	44.38/61.89	75.01/75.19	80.38/**80.52**
	asym-20%	79.24/81.39	84.27/84.40	79.89/82.08	85.24/85.44	88.86/**88.93**
	asym-40%	57.50/68.77	60.63/67.46	58.53/67.19	79.53/80.19	86.36/**86.50**
CIFAR-100	sym-20%	50.28/50.89	55.47/55.58	50.01/50.58	57.87/57.94	59.73/**59.82**
	sym-60%	20.79/34.26	23.72/34.89	21.78/34.01	43.36/43.68	46.24/**46.43**
	asym-20%	52.40/52.42	57.31/57.53	52.44/52.56	55.88/55.91	58.72/**58.86**
	asym-40%	37.64/37.66	40.12/40.37	36.95/37.61	40.99/41.01	47.70/**47.82**

Table 3: Average of **final/peak test accuracy** (%) over 4 runs of different algorithms on CIFAR with open-set noise. The best is highlighted in **bold**.

Dataset + Open-set	Noise ratio	Standard	D2L	Forward	Co-teaching	LTEC (ours)
CIFAR-10 + CIFAR-100	20%	86.74/86.83	89.42/**89.49**	86.87/86.96	88.58/88.61	88.69/88.82
	40%	82.64/82.71	85.32/85.41	82.57/82.68	86.18/86.22	86.37/**86.41**
CIFAR-10 + ImageNet-32	20%	88.27/88.36	90.60/**90.64**	88.24/88.29	88.99/89.06	89.15/89.24
	40%	85.90/85.99	87.91/**87.95**	85.84/85.99	86.99/87.03	86.63/86.78
CIFAR-100 + SVHN	20%	59.08/59.19	62.89/**62.98**	58.99/59.08	60.69/60.75	61.65/61.78
	40%	53.32/53.35	56.30/56.38	53.18/53.30	56.45/56.52	56.95/**57.18**

Results on CIFAR-10/100 with open-set noise. To generate open-set noise, some of CIFAR-10 images are replaced by images of other 32×32 image datasets including SVHN (Netzer et al., 2011), CIFAR-100, and ImageNet-32 (Chrabaszcz et al., 2017). As in Yu et al. (2019), we leave labels of the training set intact. We exclude some classes of the other datasets that are similar to the original dataset. Therefore, to make open-set noise for CIFAR-10, we sample the replacing images from 75 classes of CIFAR-100 (Abbasi et al., 2018) and 748 classes of ImageNet (Oliver et al., 2018), respectively. The overall results can be found in Table 3. Considering that on average, the final accuracies of the network trained on clean CIFAR-10 and CIFAR-100 are 90.59% and 64.38%, respectively (see Table A1), it is surprising that Standard achieves such high performance in the presence of open-set noise. This indicates that there is little effect of open-set noise on generalizing in-distribution data. We speculate that this is due to little correlation between open-set noisy examples. This is also supported by the results that all the methods perform better on CIFAR-10 with ImageNet-32 noise than on CIFAR-10 with CIFAR-100 noise, as ImageNet-32 has more classes than CIFAR-100. LTEC achieves comparable or best on CIFAR-10/100 with open-set noise. As with poorly annotated examples, out-of-distribution examples are little correlated with the other examples, thus cannot be generalized in the early stage. Consequently, they can be distinguished from in-distribution examples by applying ensemble consensus filtering.

5 Discussion: Effects of pre-defined settings

In this section, we discuss the effects of pre-defined settings in LEC on the performance.

The number of predictions. In LEC, examples to be learned are determined by whether training losses of M networks are consistently small. To understand the effect of M on the performance, we run LTEC with varying the value of M on CIFAR-10 with random label noise. In particular, the range of M is set to {3, 5, 7, 9}. As shown in Table 4, the performance for M = 9 exceeds that for M = 3, but neither for M = 7 nor M = 5. Naturally, as the number of predictions M involved in the filtering increases, the number of examples used for learning decreases. Since merely increasing the number of corrupted training examples is often helpful to improve the robustness (Rolnick et al., 2017; Li et al., 2017), larger M may not lead to better performance despite higher precision.

Table 4: Average of **final/peak test accuracy** (%) over 4 runs of LTEC with varying the number of predictions M. The best is highlighted in **bold**.

Dataset	Noise type	LTEC ($M = 3$)	LTEC ($M = 5$)	LTEC ($M = 7$)	LTEC ($M = 9$)
CIFAR-10	sym-20%	87.68/87.78	88.18/88.28	88.63/88.77	88.79/**88.87**
	sym-60%	79.73/79.80	80.38/**80.52**	80.39/80.45	80.28/80.39
	asym-20%	87.85/88.15	88.86/88.93	88.96/89.07	88.99/**89.11**
	asym-40%	85.44/85.59	86.36/86.50	86.78/**86.82**	86.59/86.63

Table 5: Average of **final/peak test accuracy** (%) over 4 runs of Co-teaching and LTEC by using small-loss examples evaluated with inaccurate noise ratios. The best is highlighted in **bold**.

Dataset	Noise type	under-estimated (0.9ϵ)		correctly estimated (ϵ)		over-estimated (1.1ϵ)	
		Co-teaching	LTEC	Co-teaching	LTEC	Co-teaching	LTEC
CIFAR-10	sym-20%	84.51/84.58	87.93/88.08	85.46/85.52	88.18/88.28	86.40/86.45	88.72/**88.75**
	sym-60%	70.47/73.11	77.98/78.22	75.01/75.19	80.38/**80.52**	79.15/79.17	79.34/79.45
	asym-20%	84.61/84.73	88.15/88.39	85.24/85.44	88.86/88.93	86.41/86.57	89.04/**89.22**
	asym-40%	76.14/77.41	84.42/84.52	79.53/80.19	86.36/86.50	82.19/82.63	86.93/**86.96**

Table 6: Average of **final/peak** test accuracy (%) over 4 runs of Standard and LTEC with ResNet. The best is highlighted in **bold**.

Dataset	Noise Type	Standard (ResNet)	LTEC (ResNet)
CIFAR-10	sym-20%	81.31/85.30	89.01/**89.12**
	sym-60%	61.94/72.80	81.46/**81.66**
	asym-20%	81.93/87.32	88.90/**89.04**
	asym-40%	62.76/77.10	86.62/**86.85**

Noise ratio. In the previous sections, we assume that a noise ratio is given in advance. However, we may use a poorly estimated ratio for LEC in reality. To study the negative effects of the estimation error, we run LTEC on CIFAR-10 with random label noise by using a bit lower and higher values than the real ratio as in Han et al. (2018). For comparison, we run Co-teaching which is another small-loss based method requiring the noise ratio. The overall results can be found in Table 5. Since it is difficult to learn all the clean examples in the early stage of training, training on less number of examples by using over-estimated ratio (i.e., 1.1ϵ) is often helpful in both LTEC and Co-teaching. By contrast, using under-estimated ratio (i.e., 0.9ϵ) leads to training on more number of examples and thus may cause severe corruption of the training examples. In this case, the performance of Co-teaching is drastically low, while that of LEC is not. On CIFAR-10, LTEC shows less sensitivity to the estimation error, compared to Co-teaching. These results corroborate that LTEC can reduce the negative effect caused by inaccurate noise ratio.

Applicability to different architecture. In the previous sections, we explore the effectiveness of our proposed LEC by using only one architecture. To show that LEC does not depend on any specific architecture, we run Standard and LTEC with ResNet-20 (He et al., 2016). The network is optimized based on Chollet et al. (2015), achieving the final test accuracy of 90.67% on clean CIFAR-10. By considering learning pace of ResNet, we set T_w as 30. As shown in Table 6, LTEC (ResNet) beats Standard (ResNet) in both peak and final accuracies. This supports that the performance of LEC is agnostic to any architecture. In fact, the key idea of LEC is rooted in the difference between generalization and memorization, thus LEC can be applied to any architecture optimized with SGD.

6 Conclusion

This work presents the way of identifying noisy examples out of small-loss examples via ensemble consensus. To generate the ensemble, we explore three simple perturbation methods. Through the experiments, we verify that the network is trained without overfitting noisy examples by eliminating them found by ensemble consensus from training batches. Along with growing attention to the use of small-loss examples for robust training, we expect that discriminating clean and noisy examples by exploiting ensemble consensus will be useful for such training methods.

REFERENCES

Mahdieh Abbasi, Arezoo Rajabi, Azadeh Sadat Mozafari, Rakesh B Bobba, and Christian Gagné. Controlling over-generalization and its effect on adversarial examples generation and detection. *arXiv preprint arXiv:1808.08282*, 2018.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 233–242. JMLR. org, 2017.

François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

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*, pp. 8527–8537, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems*, pp. 10456–10465, 2018.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

David Krueger, Nicolas Ballas, Stanislaw Jastrzebski, Devansh Arpit, Maxinder S Kanwal, Tegan Maharaj, Emmanuel Bengio, Asja Fischer, and Aaron Courville. Deep nets don't learn via memorization. 2017.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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.

Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. *arXiv preprint arXiv:1806.02612*, 2018.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pp. 5727–5736, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 3235–3246, 2018.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1944–1952, 2017.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Fei Wang, Liren Chen, Cheng Li, Shiyao Huang, Yanjie Chen, Chen Qian, and Chen Change Loy. The devil of face recognition is in the noise. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 765–780, 2018a.

Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8688–8696, 2018b.

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement benefit co-teaching? *arXiv preprint arXiv:1901.04215*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A Appendix

A.1 Pseudocodes for LECs

We present three LECs with different perturbation methods to make the ensemble. The pseudocodes for LNEC, LSEC, and LTEC can be found in Algorithm A1, A2, and A3, respectively.

Algorithm A1 Learning with Network-Ensemble Consensus (LNEC)

```
Require: noisy dataset $\mathcal{D}$, noise ratio $\epsilon$%, duration of Warming-up $T_w$, # of networks $M$
 1: Initialize $\theta_1, \theta_2, ..., \theta_M$ randomly
 2: for epoch $t = 1 : T_w$ do                                              ▶ Warming-up process
 3:     for mini-batch index $b = 1 : B$ do
 4:         for network index $m = 1 : M$ do
 5:             $\theta_m \leftarrow \theta_m - \alpha \nabla_{\theta_m} \frac{1}{|\mathcal{B}_b|} \sum_{(x,y) \in \mathcal{B}_b} CE(f_{\theta_m}(x), y)$
 6:         end for
 7:     end for
 8: end for
 9: for epoch $t = T_w + 1 : T_{end}$ do                                    ▶ Filtering process
10:     for mini-batch index $b = 1 : B$ do
11:         for network index $m = 1 : M$ do
12:             Find $\mathcal{S}_{m,b} := (100 - \epsilon)\%$ small-loss examples of $f_{\theta_m}$ within mini-batch $\mathcal{B}_b$
13:         end for
14:         $\mathcal{B}_b{}' = \mathcal{S}_{1,b} \cap \mathcal{S}_{2,b} \cap ... \cap \mathcal{S}_{M,b}$      ▷ Network-ensemble consensus filtering
15:         for network index $m = 1 : M$ do
16:             $\theta_m \leftarrow \theta_m - \alpha \nabla_{\theta_m} \frac{1}{|\mathcal{B}_b{}'|} \sum_{(x,y) \in \mathcal{B}_b{}'} CE(f_{\theta_m}(x), y)$
17:         end for
18:     end for
19: end for
```

Algorithm A2 Learning with Self-Ensemble Consensus (LSEC)

```
Require: noisy dataset $\mathcal{D}$, noise ratio $\epsilon$%, duration of Warming-up $T_w$, # of predictions $M$, perturbation r.v. $\delta$
 1: Initialize $\theta$ randomly
 2: for epoch $t = 1 : T_w$ do                                              ▶ Warming-up process
 3:     for mini-batch index $b = 1 : B$ do
 4:         $\theta \leftarrow \theta - \alpha \nabla_{\theta} \frac{1}{|\mathcal{B}_b|} \sum_{(x,y) \in \mathcal{B}_b} CE(f_{\theta}(x), y)$
 5:     end for
 6: end for
 7: for epoch $t = T_w + 1 : T_{end}$ do                                    ▶ Filtering process
 8:     for mini-batch index $b = 1 : B$ do
 9:         for $m = 1 : M$ do                                              ▷ Evaluating $M$ times
10:             Obtain $\mathcal{S}_{m,b} := (100 - \epsilon)\%$ small-loss examples of $f_\theta$ within mini-batch $\mathcal{B}_b$
11:         end for
12:         $\mathcal{B}_b{}' = \mathcal{S}_{1,b} \cap \mathcal{S}_{2,b} \cap ... \cap \mathcal{S}_{M,b}$      ▷ Self-ensemble consensus filtering
13:         $\theta \leftarrow \theta - \alpha \nabla_{\theta} \frac{1}{|\mathcal{B}_b{}'|} \sum_{(x,y) \in \mathcal{B}_b{}'} CE(f_{\theta}(x), y)$
14:     end for
15: end for
```

Algorithm A3 Learning with Temporal-Ensemble Consensus (LTEC)

```
Require: noisy dataset $\mathcal{D}$, noise ratio $\epsilon$%, duration of Warming-up $T_w$, # of predictions $M$
 1: Initialize $\theta$ randomly
 2: for epoch $t = 1 : T_{end}$ do
 3:     for mini-batch index $b = 1 : B$ do
 4:         Obtain $\mathcal{S}_{t,b} := (100 - \epsilon)\%$ small-loss examples of $f_\theta$ within mini-batch $\mathcal{B}_b$
 5:         if $t < T_w + 1$ then                                           ▶ Warming-up process
 6:             $\theta \leftarrow \theta - \alpha \nabla_{\theta} \frac{1}{|\mathcal{B}_b|} \sum_{(x,y) \in \mathcal{B}_b} CE(f_{\theta}(x), y)$
 7:         else                                                            ▶ Filtering process
 8:             $\mathcal{B}_b{}' = \mathcal{P}_{t-M+1} \cap \mathcal{P}_{t-M+2} \cap ... \cap \mathcal{P}_{t-1} \cap \mathcal{S}_{t,b}$   ▷ Temporal-ensemble consensus filtering
 9:             $\theta \leftarrow \theta - \alpha \nabla_{\theta} \frac{1}{|\mathcal{B}_b{}'|} \sum_{(x,y) \in \mathcal{B}_b{}'} CE(f_{\theta}(x), y)$
10:         end if
11:     end for
12:     Obtain $\mathcal{P}_t := \cup_{b=1}^{B} \mathcal{S}_{t,b}$
13: end for
```

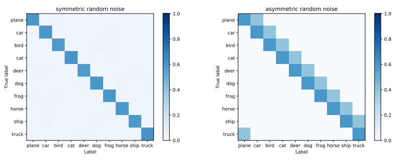
Table A1: Average ($\pm$ stddev) of **final test accuracy** over 4 runs of a regular training (Standard) on clean MNIST, CIFAR-10, and CIFAR-100.

Dataset	MNIST	CIFAR-10	CIFAR-100
Test accuracy	99.60$\pm$0.02	90.59$\pm$0.15	64.38$\pm$0.20

A.2 Implementation details

A.2.1 Corruption matrix

We study two types of random label noise: sym-ϵ% and asym-ϵ%. Figure A1 shows the corruption matrices used to generate CIFAR-10 with random label noise.



(a) Random-sym (b) Random-asym

Figure A1: Corruption matrix of CIFAR-10 with random label noise

A.2.2 Optimization details

The 9-convolutional layer architecture used in this study can be found in Table A2. The network is optimized with Adam (Kingma & Ba, 2014) with a batchsize of 128 for 200 epochs. The learning rate α is initially set to 0.1. The learning rate is linearly annealed to zero during the last 120 epochs for CIFAR-10 and during the last 100 epochs for CIFAR-100. The momentum parameters β_1 and β_2 are set to 0.9 and 0.999, respectively. β_1 is linearly annealed to 0.1 during the last 120 epochs for MNIST and CIFAR-10, and during the last 100 epochs for CIFAR-100. The images of CIFAR are divided by 255 and are whitened with ZCA. Additional regularizations such as data augmentation are not applied. The results on clean MNIST, CIFAR-10, and CIFAR-100 can be found in Table A1.

Table A2: 9-conv layer architecture

Input image
Gaussian noise (σ = 0.15)
3 $\times$ 3 conv, 128, padding = 'same' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 128, padding = 'same' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 128, padding = 'same' batch norm, LReLU (α = 0.01) 2 $\times$ 2 maxpooling, padding = 'same' dropout (drop rate = 0.25)
3 $\times$ 3 conv, 256, padding = 'same' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 256, padding = 'same' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 256, padding = 'same' batch norm, LReLU (α = 0.01) 2 $\times$ 2 maxpooling, padding = 'same' dropout (drop rate = 0.25)
3 $\times$ 3 conv, 512, padding = 'valid' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 256, padding = 'valid' batch norm, LReLU (α = 0.01) 3 $\times$ 3 conv, 128, padding = 'valid' batch norm, LReLU (α = 0.01)
global average pooling fc (128 $\rightarrow$ # of classes)

A.2.3 COMPETING METHODS

The competing methods include a regular training method: ***Standard***, a method of training with corrected labels: ***D2L***, a method of training with modified loss function based on the noise distribution: ***Forward***, and a method of exploiting small-loss examples: ***Co-teaching***. We tune all the methods individually in the following. For a fair comparison, the architecture used in LTEC (Table A2) is applied to the implementation.

- **Standard** : The network is trained using the cross-entropy loss.
- **D2L** (Ma et al., 2018) : The input vector of a fully connected layer in the architecture is used to measure the LID estimates. The parameter involved with identifying the turning point, window size W is set to 12. The network is trained using original labels until the turning point is found and then trained using the bootstrapping target with adaptively tunable mixing coefficient.
- **Forward** (Patrini et al., 2017) : Prior to training, the corruption matrix C where $C_{ji} = \mathbb{P}(y = i | y_{true} = j)$ is estimated based on the $97th$ percentile of probabilities for each class on MNIST and CIFAR-10, and the $100th$ percentile of probabilities for each class on CIFAR-100 as in Hendrycks et al. (2018). The network is then trained using the corrected labels for 200 epochs.
- **Co-teaching** (Han et al., 2018) : Two networks are employed. For every batch update, they select examples based on their training losses and then provide them to each other. The ratio of selected examples based on training losses is linearly annealed from 100% to (100-ϵ)% over the first 10 epochs.