

A IMPLEMENTATION DETAILS

Single-model experiment on CIFAR-10 For CIFAR-10, we follow previous works (Tanaka et al., 2018; Li et al., 2019) and split the 50K training samples into 45K training and 5K validation subsets. We then introduce symmetric or asymmetric label noise to the training subset as described in Sec. 4.1. All models are trained on the noisy training subset, with hyperparameters being chosen based on the model performance on the clean validation subset. After hyperparameter tuning, we then introduce label noises to all 50K training samples and rerun all experiments for comparison.

We develop our SLSSL algorithm based on the implementation of MLNT¹ (Li et al., 2019), which adopts a Pre-Act ResNet-32 network. We follow most of the hyperparameters from MLNT, and train the model for 300 epochs using SGD with an initial learning rate 0.02 (divided by 2 after every 50 epochs), a momentum of 0.9, a weight decay of 0.0005, and a batch size of 128. The learning rate η used for obtaining the augmented models is set to 0.1 in equation 2. For our SLSSL framework with both E and M steps, we set $\eta = 0.001$ in equation 8, and conduct the M-step for 5 iterations (3 M-step epochs per iteration) at every 50 E-step model training epochs: {50, 100, 150, 200, 250}.

As for other single-model methods, we simply train the model for 300 epochs using standard cross-entropy classification loss as the Baseline, and add the ground-truth noise transition matrix for F-correction (Patrini et al., 2017). The MLNT results are reproduced by directly using their implementation.

Dual-Model co-training for CIFAR-10 and CIFAR-10N We follow the same principle from the above single-model experiment on CIFAR-10 for hyperparameter tuning, and directly apply the best set of hyperparameters for the 40% asymmetric noisy dataset to CIFAR-10N.

We integrate our SLSSL algorithm into the co-training phase of DivideMix (Li et al., 2020) based their implementation², and also follow most of their hyperparameters. We first train two Pre-Act ResNet-18 networks separately for 10 epochs in the warm-up phase, and then enter the co-training phase for the remaining 290 epochs. At the 150 epoch, we replace DivideMix’s GMM-based reweighting procedure on the 2nd network by our SLSSL-based sample reweighting algorithm, and conduct the M-step for 60 iterations (2 M-step epochs per iteration) at every 5 model training epochs: {150, 155, ..., 295}. Based on the derived sample weights, we then follow standard DivideMix and apply *co-dividing*, *co-refinement*, and *co-guessing* techniques to train the two networks.

Dual-Model co-training for Clothing1M For Clothing1M, we also follow DivideMix (Li et al., 2020) and train our model on the 1M (noisy) training dataset, with hyperparameters being chosen based on the 14K (clean) validation subset. Similar to CIFAR-10, we integrate our SLSSL algorithm into the co-training phase of DivideMix. We start our SLSSL-based sample reweighting (i.e., the M-step in Sect. 3.4) at epoch 70. Since only 32K samples are randomly selected per training epoch, we conduct M-step for every epoch, and set the total epoch number in the co-training phase as 80. The learning rate is set to 0.002, and decays with a factor of 10 after the epoch 40. All the rest hyperparameters follow DivideMix.

Algorithm We provide the pseudo codes for our SLSSL in Algorithm 1 for model training (3.3; E-step) and Algorithm 2 for sample reweighting (3.4; M-step). The two algorithms can be combined as an EM-like iterative training strategy as described in Sect. 3.5.

B VISUALIZATION ON CIFAR-10

We provide additional training statistics to further validate the proposed methods. In Fig. 7(a), we show the Area Under the Curve (AUC) for clean/noisy sample classification on CIFAR-10 with 40% asymmetric label noise in the single-model experiment. As can be seen, our M-step is able to effectively identify clean/noisy samples across successive M-steps, even for *bird* samples with 40% samples being mislabeled as *airplane*, and for *cat* samples which could be heavily confused with *dog* samples with 40% samples in each class being mislabeled as one another. In Fig. 7(b), we

¹<https://github.com/LiJunnan1992/MLNT>

²<https://github.com/LiJunnan1992/DivideMix>

Algorithm 1 SLSSL-E (model training in Sect. 3.3)

```

1: Input: Noisy training dataset  $\mathcal{D} = \{(x_n, \tilde{y}_n, w_n)\}$  (fixed  $\{w_n\}$ ), learning model  $\theta$ , EMA model  $\theta^*$ , number of E-step epochs  $P$ , number of label corruption for each mini-batch  $M$ , estimated noise transition matrix at the  $e^{th}$  epoch  $\hat{T}_e$ , estimated noise transition matrix  $\hat{T}$ .
2: if first E-step then
3:   Random initialize  $\theta$ 
4:   Initialize EMA model  $\theta^* = \theta$ 
5:   Initialize  $\hat{T}$  as identity
6:   Initialize all sample weights  $w_n = 1$ 
7: end if
8: for  $e = 1$  to  $P$  do
9:   while not done do
10:    Sample mini-batch  $S = \{(x_n, \tilde{y}_n, w_n)\}_{n=1}^N$  from  $\mathcal{D}$ 
11:    for  $m = 1$  to  $M$  do
12:      Randomly sample a class pair  $(i, j)$ 
13:      Obtain augmented set  $S' = \{(x_n, \tilde{y}'_n, w_n)\}_{n=1}^N$  by corrupting  $(i, j)$  in  $S$  (Eq. 1)
14:      Derive augmented model  $\theta'_m$  from  $\theta$  by single-step gradient descent on  $S'$  (Eq. 2)
15:      Compute the KL divergence  $D_{KL}$  between  $\theta'_m$  and  $\theta^*$  (Eq. 4)
16:    end for
17:    Compute set-level self-supervised loss  $L_{SSL}$  (Eq. 3)
18:    Update  $\theta$  by minimizing  $L_{SSL}$ 
19:    Compute classification loss  $L_{CE}$  and then correct it using  $\hat{T}$  (Eq. 7)
20:    Update  $\theta$  by minimizing  $L_{CE}$ 
21:    Aggregate  $D_{KL}$ 's derived by corrupting  $(i, j)$  to estimate the  $(i, j)^{th}$  entry of  $\hat{T}_e$  (Eq. 6)
22:  end while
23:  Update  $\hat{T}$  using EMA of  $\hat{T}_e$ 
24:  Update  $\theta^*$  using EMA of  $\theta$ 
25: end for

```

Algorithm 2 SLSSL-M (sample reweighting in Sect. 3.4)

```

1: Input: Noisy training dataset  $\mathcal{D} = \{(x_n, \tilde{y}_n, w_n)\}$ , learning model  $\theta$  (fixed), number of M-step epochs  $Q$ , number of label corruption for each mini-batch  $M$ .
2: for  $e = 1$  to  $Q$  do
3:   while not done do
4:    Sample mini-batch  $S = \{(x_n, \tilde{y}_n, w_n)\}_{n=1}^N$  from  $\mathcal{D}$ 
5:    for  $m = 1$  to  $M$  do
6:      Randomly sample a class pair  $(i, j)$ 
7:      Obtain augmented set  $S' = \{(x_n, \tilde{y}'_n, w_n)\}_{n=1}^N$  by corrupting  $(i, j)$  in  $S$  (Eq. 1)
8:      Derive  $\theta'_m(\mathbf{w})$  from  $\theta$  by weighted single-step gradient descent on  $S'$  (Eq. 8)
9:      Derive  $\theta'_0(\mathbf{w})$  from  $\theta$  by weighted single-step gradient descent on  $S$  (Eq. 8)
10:    end for
11:    Compute the sample reweighting loss  $L_{RW}$  (Eq. 9)
12:    Update  $\{w_i\}_{i=1}^N$  by minimizing  $L_{RW}$ 
13:  end while
14: end for

```

show the training and testing accuracy values. As can be seen, the accuracy is improved after each iteration of M-step at epochs 50 and above, further validating the effectiveness of our SLSSL-based sample reweighting procedure.

C ADDITIONAL EXPERIMENT RESULTS

CIFAR-10 and CIFAR-10N In Tables 2 and 4, results of DivideMix (Li et al., 2020) and DM-AugDesc (Nishi et al., 2021) were directly copied from their papers (i.e., no standard deviation were reported by neither works). Here, we further conduct additional runs with different random seeds for our SLSSL and DivideMix on selected noisy settings from CIFAR-10 and CIFAR-10N. The results

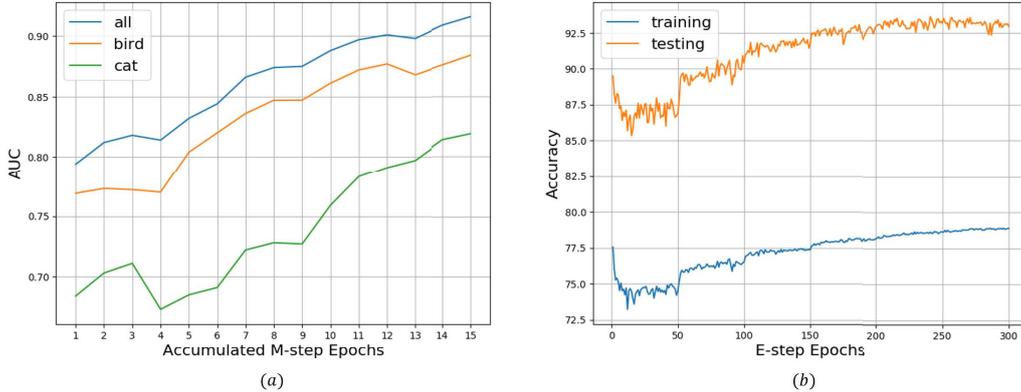


Figure 7: Evaluation on CIFAR-10 with 40% asymmetric label noise. (a) Area Under the Curve for clean/noisy sample classification based on the sample weights $\{w_n\}$ obtained by our M-steps. The X-axis indicates the *accumulated* M-step epochs over 5 iterations (3 M-step epochs per iteration) at E-step epochs from 50 up to 250. Note that our M-step is shown to identify clean/noisy samples across successive M-steps by assigning proper weights, improving both Net1 and Net2 in the co-training process. (b) Training and testing accuracy values on the first network over all 300 training epochs. The accuracy is improved after each iteration of M-step at epochs 50 and above.

Table 5: Classification accuracy (%) on CIFAR-10 across different noisy labeling schemes with NLL methods adopting dual models (i.e., co-training based approaches). We report results over 3 independent runs.

Method		CIFAR-10	CIFAR-10N
		Asym 40%	Aggregate
DivideMix (Li et al., 2020) (reproduced)	Best	93.20±0.20	95.37±0.12
	Last	93.25±0.15	95.10±0.08
Our SLSSL	Best	94.20±0.10	95.73±0.12
	Last	93.75±0.15	95.57±0.12

are listed in Table 5. Following the setting of DivideMix (Li et al., 2020), we report the best test accuracy (Best) and the averaged test accuracy over the last 10 epochs (Last). From this table, it can be seen that the improvements of our SLSSL over DivideMix were statistically significant.

CIFAR-100 To provide additional performance evaluation, we also add experiments on CIFAR-100. We select two noise types (20% and 50% symmetric noise) and report results from a single run using the same random seed to ensure the same noise setting between DivideMix and our SLSSL. As can be seen from Table 6, our SLSSL again outperformed DivideMix and also MD-DYR-SH (Arazo et al., 2019) on CIFAR-100.

D ADDITIONAL COMPARISONS

Compare to instance-level SSL approaches As discussed in Sect. 1, it is not clear whether existing instance-level SSL techniques would result in robust representations when tackling the NLL tasks. To make our discussions and comparisons more complete, we compare our method to MOIT (Ortego et al., 2021), a recent instance-based SSL approach to NLL, on CIFAR-10. For fair comparisons, we follow MOIT to adapt a single PreAct ResNet-18 network, and report the test accuracy in the last training epoch. As can be seen from Table 7, our SLSSL reached comparable performance for asymmetric noise at 40%, while outperformed MOIT with significant margins for different symmetric noise levels.

Compare to recent related works SOP (Liu et al., 2022) proposes to model the label noise and learn to separate it from the data by enforcing its sparsity, and serves as one of the most recent works of NLL. However, as can be seen from Table 8, our SLSSL still performs favorably against SOP, especially on the more challenging Clothing1M dataset.

Table 6: Classification accuracy (%) on CIFAR-100 across different noisy labeling schemes with NLL methods adopting dual models (i.e., co-training based approaches). For DivideMix(Li et al., 2020) and our SLSSL, we report result from a single run using the same random seed.

Method		Sym 20%	Sym 50%
MD-DYR-SH (Arazo et al., 2019) (reported)	Best	73.9	66.1
	Last	73.4	65.4
DivideMix (Li et al., 2020) (reproduced)	Best	77.50	74.20
	Last	77.00	73.80
Our SLSSL	Best	78.08	74.34
	Last	77.83	73.95

Table 7: Classification accuracy (%) on CIFAR-10 across different noisy labeling schemes from MOIT (Ortego et al., 2021) and our SLSSL.

	Sym 20%	Sym 80%	Asym 40%	Mean
MOIT (Ortego et al., 2021) (reported)	94.08	75.83	93.27	87.73
Our SLSSL	95.18	92.82	92.23	93.41

E LIMITATIONS

Since our proposed SLSSL can be viewed as a unique meta-learning scheme on existing NLL methods like DivideMix (Li et al., 2020), we expect longer computation time during training. However, take CIFAR-10 for example, DivideMix took 15 hours to train using a single Nvidia TITAN-V GPU, while the full version of our SLSSL (i.e., implemented as an EM algorithm) required 25 hours. It can be seen that, the overall computation time of our SLSSL is still in the same order of that of SOTAs like DivideMix.

Also, as noted in (Patrini et al., 2017; Hendrycks et al., 2018; Xia et al., 2019; Wang et al., 2020; Yao et al., 2020), NLL methods based on class-wise noise transition matrix estimation share the limitation that the number of classes for NLL would be reasonable (e.g., 100 in CIFAR-100). This is to avoid the potential problem of estimating a large noise transition matrix. Sharing the concern of the above works, this would also among the current limitation of our work.

Table 8: Classification accuracy (%) on CIFAR-10N (Aggregate) and Clothing1M from SOP (Liu et al., 2022) and our SLSSL.

	CIFAR-10N (Aggregate)	Clothing1M
SOP (Liu et al., 2022) (reported)	95.61	73.5
Our SLSSL	95.73	74.51