Adaptive Sample Selection for Robust Learning under Label Noise - Supplementary Material

Network architectures & Optimizers

We use one MLP and one CNN architecture. For MNIST we train a 1-hidden layer fully-connected network with Adam (learning rate = 2×10^{-4} and a learning rate scheduler: ReduceLROnPlateau). (This is same as the network used for Co-Teaching and Co-Teaching+ [11, 49]). For CIFAR-10 we train a 4-layer CNN with Adam [17] (learning rate = 2×10^{-3} and a learning rate scheduler: ReduceLROnPlateau). All networks are trained for 200 epochs. For MR, SGD optimizer with momentum 0.9 and learning rate of 1×10^{-3} is used as the meta-optimizer. For MN, SGD optimizer with learning rate of 2×10^{-3} is used as meta-optimizer. For CL, soft hinge loss is used as suggested in [27] instead of cross-entropy loss. Rest of the algorithms use cross-entropy loss. All the simulations are run for 5 trials. A pre-trained ResNet-50 is used for training on Clothing-1M with SGD (learning rate of 1×10^{-3} that is halved at epochs 6 and 11) with a weight decay of 1×10^{-3} and momentum 0.9 for 14 epochs. All experiments use PyTorch [34], NumPy [12], scikit-learn [36], and NVIDIA Titan X Pascal GPU with CUDA 10.0.

These settings of optimizer, learning rate, and learning rate scheduler were found to work the best for our experimental and hardware setup.

With our work for robust learning under label noise, our objective has been to show the degree of robustness offered by BARE for any given architecture. Keeping this in mind along with maintaining some commonality with the architectures used for baseline schemes (i.e., their corresponding papers), we have shown simulation results for the aforementioned MLPs and 4-layer CNNs. However, it should be noted that BARE continues to demonstrate its effectiveness on bigger architectures such as ResNet-50 as seen in Clothing-1M performance results (Table 3).

Test accuracies on MNIST & CIFAR-10

We tabulate the test accuracies of all the algorithms on MNIST and CIFAR-10 in Tables 4 – 9. The best two results are in bold. These are accuracies achieved at the end of training. For CoT [11] and CoT+ [49], we show accuracies only of that network which performs the best out of the two that are trained. In the main paper we showed the plots of accuracies.

It may be noted that in a couple of cases the standard deviation in the accuracy for MN is high. As we mentioned in the main paper, we noticed that MN is very sensitive to the tuning of hyper parameters. While we tried our best to tune all the hyper parameters, maybe the final ones we found for these two cases are still not the best and that is why the standard deviation is high.

Performance on Clothing-1M dataset

Table 3 shows how the proposed algorithm fares against several baselines in terms of test accuracy on Clothing-1M dataset. For the baselines, we report the accuracy values as reported in the corresponding papers.

Table 3: Test accuracies on Clothing-1M dataset

Algorithm	Test Accuracy (%)
CCE	68.94
D2L [29]	69.47
GCE [53]	69.75
Forward [35]	69.84
CoT [11] (as reported in [7])	70.15
JoCoR [44]	70.30
SEAL [7]	70.63
DY [2]	71.00
SCE [42]	71.02
LRT [55]	71.74
PTD-R-V [45]	71.67
Joint Opt. [41]	72.23
BARE (Ours)	**72.28**
C2D [54] (ELR+ [26] with SimCLR [8])	74.58
DivideMix [24]	74.76

Table 4: Test Accuracy (%) for MNIST - $\eta = 0.5$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [11]	90.80 ± 0.18
CoT+ [49]	$\mathbf{93.17 \pm 0.3}$
MR [38]	90.39 ± 0.07
MN [39]	74.94 ± 9.56
CL [27]	92.00 ± 0.26
CCE	74.30 ± 0.55
BARE (OURS)	$\mathbf{94.38 \pm 0.13}$

Table 5: Test Accuracy (%) for MNIST - $\eta = 0.7$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [11]	87.17 ± 0.45
CoT+ [49]	87.26 ± 0.67
MR [38]	85.10 ± 0.28
MN [39]	65.52 ± 21.35
CL [27]	$\mathbf{88.28 \pm 0.45}$
CCE	61.19 ± 1.29
BARE (OURS)	$\mathbf{91.61 \pm 0.60}$

Table 6: Test Accuracy (%) for MNIST - $\eta = 0.45$ (class-conditional)

ALGORITHM	TEST ACCURACY
CoT [11]	$\mathbf{95.20 \pm 0.22}$
CoT+ [49]	91.10 ± 1.51
MR [38]	$\mathbf{95.40 \pm 0.31}$
MN [39]	75.03 ± 0.59
CL [27]	81.52 ± 3.27
CCE	74.96 ± 0.21
BARE (OURS)	94.11 ± 0.77

Table 7: Test Accuracy (%) for CIFAR-10 - $\eta = 0.3$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [11]	$\mathbf{71.72 \pm 0.30}$
CoT+ [49]	60.14 ± 0.35
MR [38]	62.96 ± 0.70
MN [39]	51.65 ± 1.49
CL [27]	66.124 ± 0.45
CCE	54.83 ± 0.28
BARE (OURS)	$\mathbf{75.85 \pm 0.41}$

Table 8: Test Accuracy (%) for CIFAR-10 - $\eta = 0.7$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [11]	$\mathbf{58.95 \pm 1.31}$
CoT+ [49]	37.69 ± 0.70
MR [38]	45.14 ± 1.04
MN [39]	23.23 ± 0.65
CL [27]	44.82 ± 2.42
CCE	23.46 ± 0.37
BARE (OURS)	$\mathbf{59.53 \pm 1.12}$

Table 9: Test Accuracy (%) for CIFAR-10 - $\eta = 0.4$ (class-conditional)

ALGORITHM	TEST ACCURACY
CoT [11]	65.26 ± 0.78
CoT+ [49]	63.05 ± 0.39
MR [38]	$\mathbf{70.27 \pm 0.77}$
MN [39]	63.84 ± 0.41
CL [27]	64.48 ± 2.02
CCE	64.06 ± 0.32
BARE (OURS)	$\mathbf{70.63 \pm 0.46}$

Plots for sample fraction v/s epochs

We show some more plots for fraction of samples chosen by the baselines along with the proposed scheme, BARE, as epochs go by for some more dataset and label noise combinations in Figures 7 & 8. As noted during the discussion in the paper, and as is evident from these figures, BARE is able to identify the clean samples effectively even without the knowledge of noise rates.

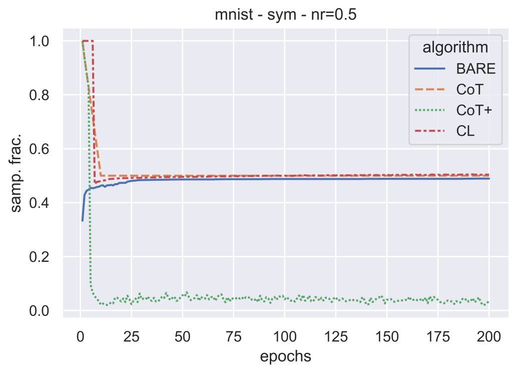


Figure 7: Sample fraction values for $\eta = 0.5$ (symmetric noise) on MNIST

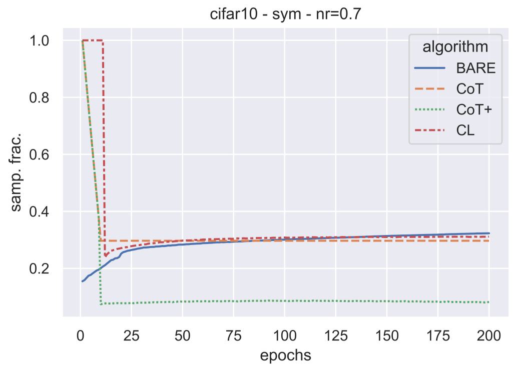


Figure 8: Sample fraction values for $\eta = 0.7$ (symmetric noise) on CIFAR-10

Results on Arbitrary Noise Transition Matrix

In the main paper, we showed results for special class-conditional noise cases taken from literature. Here we provide results for an arbitrary, digonally-dominant noise transition matrix in Tables 10–13. $\eta = 0.45$ and $\eta = 0.4$ are supplied as the estimated noise rates to CoT, CoT+, and CL baselines for MNIST and CIFAR-10 respectively. The best two results are in bold. It can be seen that the proposed algorithm continues to perform well. The noise transition matrices are arbitrary but for the sake of completeness we show them at the end of this supplementary material.

Table 10: Test Accuracy (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

Algorithm	Test Accuracy
CoT [11]	**95.3**
CoT+ [49]	93.07
CL [27]	88.41
BARE (Ours)	**95.02**

Table 11: Avg. Test Accuracy (last 10 epochs) (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

Algorithm	Avg. Test Accuracy (last 10 epochs)
CoT [11]	**95.22**
CoT+ [49]	93.08
CL [27]	88.56
BARE (Ours)	**95.03**

Table 12: Test Accuracy (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Test Accuracy
CoT [11]	71.92
CoT+ [49]	68.56
CL [27]	**72.12**
BARE (Ours)	**76.22**

Table 13: Avg. Test Accuracy (last 10 epochs) (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Avg. Test Accuracy (last 10 epochs)
CoT [11]	71.86
CoT+ [49]	68.99
CL [27]	**72.27**
BARE (Ours)	**75.96**

Table 14: Network Architectures used for training on MNIST and CIFAR-10 datasets

MNIST	CIFAR-10
DENSE $28\times28 \rightarrow 256$	3×3 CONV., 64 RELU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
	3×3 CONV., 128 RELU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
	3×3 CONV., 196 RELU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	3×3 CONV., 16 RELU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
DENSE $256 \rightarrow 10$	DENSE $256 \rightarrow 10$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0.1 \\
0 & 0 & 0 & 0.5 & 0 & 0.1 & 0 & 0 & 0.4 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.15 & 0.55 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.35 & 0.55 & 0.10 & 0 & 0 \\
0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Arbitrary Noise Transition Matrix for MNIST

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.2 & 0 & 0.7 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\
0.1 & 0 & 0 & 0.6 & 0 & 0.1 & 0 & 0 & 0.2 & 0 \\
0 & 0.1 & 0.1 & 0 & 0.7 & 0 & 0 & 0.1 & 0 & 0 \\
0 & 0 & 0 & 0.1 & 0 & 0.6 & 0 & 0 & 0 & 0.3 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0.8
\end{bmatrix}$$

Arbitrary Noise Transition Matrix for CIFAR-10

Performance under different noise rates

The proposed algorithm shows fairly stable robustness across noise rates. The paper reports results on three noise rates in Figures 1 & 2. Table 2 is about the effect of minibatch size and we included a different noise rate here. We report some more results on different noise rates. Table 15 shows performance of BARE on MNIST under different noise rates (averaged over 5 runs of 200 epochs each) and Table 16 shows some more empirical results on sensitivity of BARE to batch-sizes.

Table 15: Test Accuracy (%) of BARE on MNIST & CIFAR-10

Dataset	Noise (η)	Test Accuracy (in %)
MNIST	10% (sym.)	96.7 ± 0.31
	20% (sym.)	96.39 ± 0.23
	30% (sym.)	95.92 ± 0.17
	40% (sym.)	95.22 ± 0.23
	50% (sym.)	94.38 ± 0.13
	60% (sym.)	93.44 ± 0.27
	70% (sym.)	91.61 ± 0.59
	20% (cc)	96.45 ± 0.33
	45% (cc)	95.05 ± 0.23
CIFAR-10	10% (sym.)	78.76 ± 0.42
	20% (sym.)	77.13 ± 0.46
	30% (sym.)	75.85 ± 0.41
	40% (sym.)	73.86 ± 0.49
	60% (sym.)	66.87 ± 0.82
	70% (sym.)	59.87 ± 1.12
	20% (cc)	76.78 ± 0.38
	40% (cc)	70.63 ± 0.46

Rationale for choosing baselines in this work

There are now a very large number of algorithms motivated by different ideas for learning under label noise. Hence, we compared against only those methods that use sample selection as the main strategy and for these comparisons we used the same network for all methods and used the same noisy training set (with MNIST, CIFAR). For Clothing1M data, the results shown in supplementary material for different algorithms are all as reported in the respective papers and different results are with different network architectures. Here we reported results available in literature for different algorithms, including algorithms that do not rely on sample selection.

Table 16: Test Accuracy (%) of BARE on MNIST & CIFAR-10 with batch sizes $\in \{64, 128, 256\}$

Dataset	Noise (η)	Batch Size	Test Accuracy
MNIST	30% (sym.)	64	96.26 ± 0.10
		128	96.03 ± 0.20
		256	95.62 ± 0.13
MNIST	70% (sym.)	64	92.42 ± 0.40
		128	91.61 ± 0.60
		256	91.4 ± 0.38
CIFAR-10	70% (sym.)	64	59.97 ± 0.66
		128	59.53 ± 1.12
		256	56.72 ± 0.72

Table 17: Test accuracies on MNIST under label noise for 500 epoch runs

Dataset	Noise (η)	Algorithm	Test Accuracy
MNIST	70% (sym.)	CoT [11]	84.65 ± 0.22
		CoT+ [49]	79.52 ± 0.91
		MR [38]	80.36 ± 0.56
		BARE (Ours)	91.52 ± 0.61
MNIST	45% (cc)	CoT [11]	95.33 ± 0.14
		CoT+ [49]	84.69 ± 1.81
		MR [38]	93.92 ± 0.20
		BARE (Ours)	94.65 ± 0.57
CIFAR-10	70% (sym.)	CoT [11]	58.08 ± 1.61
		CoT+ [49]	28.18 ± 0.57
		MR [38]	40.52 ± 1.62
		BARE (Ours)	58.03 ± 1.01
CIFAR-10	40% (cc)	CoT [11]	65.19 ± 0.61
		CoT+ [49]	62.61 ± 0.48
		MR [38]	68.84 ± 0.58
		BARE (Ours)	69.88 ± 0.44

Regarding early-stopping and long training

From Figures 1 & 2, it is easily seen that irrespective of which epoch you would stop the other algorithms, the final accuracy achieved by BARE is better. For BARE, we can run the algorithm without any such necessity of early stopping. Further, we ran other algorithms that are close to BARE for upto 500 epochs and noticed that the test accuracies either saturated at the level shown in the Figures 1 & 2 or decreased. See Table 17. Apart from this, to be able to decide some epochs after which to stop, one needs clean validation data; the proposed algorithm does not need any such validation data.

Performance with large number of classes

Food-101N [23] is a dataset for food classification that consists of 101 classes and 310k training images collected from the web. Estimate label noise rate is $\sim$ 20% and it is feature-dependent noise. The Food-101 [5] testing set is used for testing, which contains 25k cleanly-labelled images. Following the baselines, we train a ResNet-50 network (pre-trained on ImageNet) for 30 epochs with SGD optimizer. The batch size taken is 128 and the initial learning rate is 0.01, which is divided by 10 every 10 epochs. For data-augmentation, we follow same procedure as baselines – random horizontal flip, and resizing the image with a short edge of 256 and then randomly cropping a 224×224 patch from the resized image. Table 18 shows how BARE compares with the baselines. With randomly formed minibatches of size 128 (and without any regard for sample sizes of different classes in a minibatch), we get an accuracy of 84.12% (denoted as *without batch-balance*) which is better than CleanNet (83.95%) but not PLC (85.28%).

Table 18: Test Accuracy (%) on Food-101N dataset

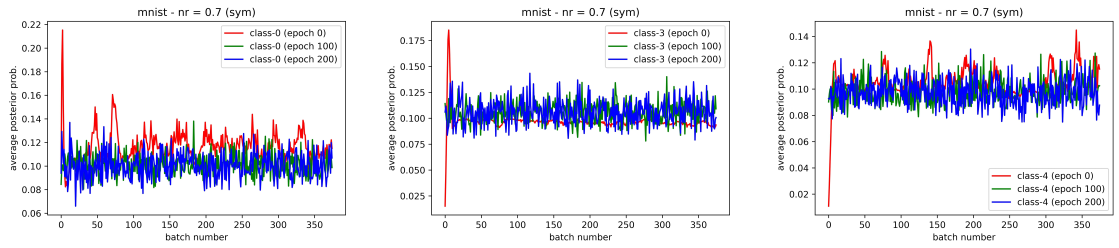
ALGORITHM	TEST ACCURACY
CCE (STANDARD)	81.67%
CLEANNET [23]	83.95%
PLC [52]	**85.28**
BARE (WITHOUT BATCH-BALANCE)	**84.12**

Rationale for choosing threshold (in BARE)

The main idea in the algorithm is to use minibatch statistics to derive the threshold on probabilities for sample selection. As is to be expected, it is effective when we choose samples from the 'top quantiles'. We now show results obtained when the threshold is taken as "mean $+ \kappa *$ std. dev." for $\kappa \in \{-1, 0, 0.5, 0.75, 1, 1.25, 1.5\}$. The method works well for all values of κ except $\kappa \in \{-1, 0\}$. These results are shown here in Table 19. We also noticed that the numerical value of the threshold varies across minibatches and is different for different clases thus justifying the motivation we started with. Figures 9 – 12 show the evolution of these threshold values (i.e. RHS of Equation 7 with $\kappa = 1$) for some of the randomly picked classes in MNIST and CIFAR-10 dataset at 0^{th}, 100^{th} and 200^{th} epoch under different noise rates and noise types.

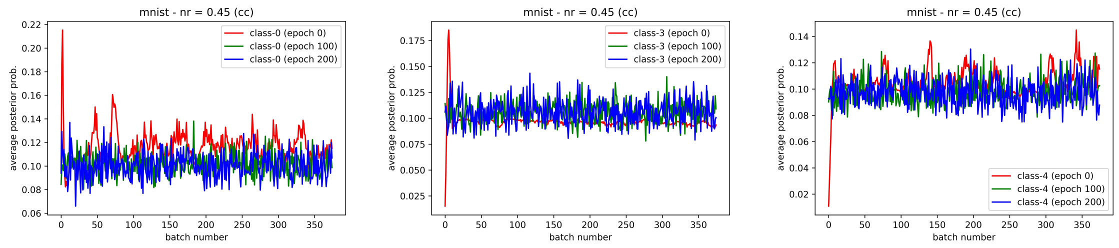
Table 19: Test accuracies of BARE under label noise for $\kappa \in \{-1, 0, 0.5, 0.75, 1, 1.25, 1.5\}$

DATASET	NOISE (η)	κ	TEST ACCURACY
MNIST	70% (SYM.)	-1	58.25 ± 0.73
		0	92.1 ± 0.76
		0.5	91.32 ± 0.82
		0.75	91.37 ± 0.37
		1	91.61 ± 0.59
		1.25	91.31 ± 0.32
		1.5	91.2 ± 0.53
MNIST	45% (CC)	-1	74.95 ± 0.39
		0	93.8 ± 1.54
		0.5	94.26 ± 1.06
		0.75	93.06 ± 1.65
		1	94.11 ± 0.77
		1.25	93.2 ± 2.84
		1.5	92.72 ± 3.71
CIFAR-10	70% (SYM.)	-1	23.3 ± 0.33
		0	42.49 ± 3.44
		0.5	58.17 ± 0.50
		0.75	59.04 ± 0.67
		1	59.93 ± 1.12
		1.25	57.14 ± 1.78
		1.5	56.66 ± 0.66
CIFAR-10	40% (CC)	-1	63.77 ± 0.11
		0	64.70 ± 0.31
		0.5	67.92 ± 0.29
		0.75	69.87 ± 0.34
		1	70.63 ± 0.46
		1.25	71.06 ± 0.56
		1.5	70.81 ± 1.22



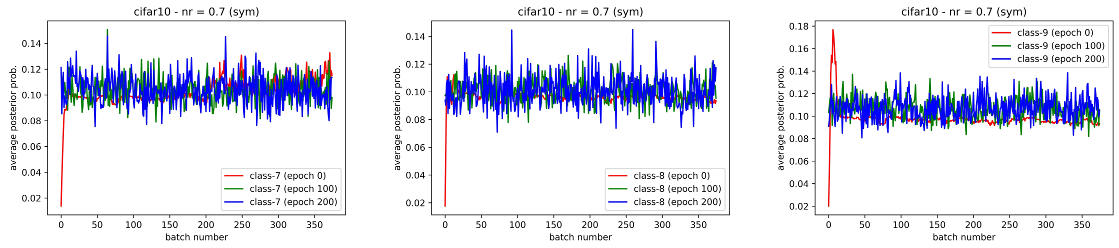
(a) (b) (c)

Figure 9: Average class-wise posterior probability across mini-batches; (a): class 0; (b): class 3; (c): class 4 – MNIST under $\eta = 0.7$ (symmetric) noise



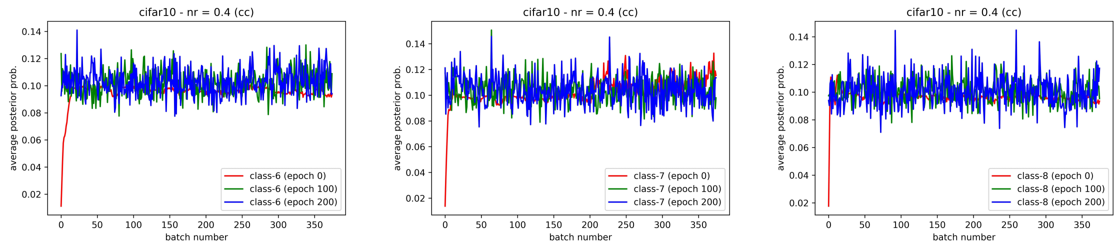
(a) (b) (c)

Figure 10: Average class-wise posterior probability across mini-batches; (a): class 0; (b): class 3; (c): class 4 – MNIST under $\eta = 0.45$ (class-conditional) noise



(a) (b) (c)

Figure 11: Average class-wise posterior probability across mini-batches; (a): class 7; (b): class 8; (c): class 9 – CIFAR-10 under $\eta = 0.7$ (symmetric) noise



(a) (b) (c)

Figure 12: Average class-wise posterior probability across mini-batches; (a): class 6; (b): class 7; (c): class 8 – CIFAR-10 under $\eta = 0.4$ (class-conditional) noise