## APPENDIX A CO-TRAINING AND MODEL ENSEMBLE

Co-training and model ensemble have been shown to be useful in combating label noise (Han et al., 2018; Li et al., 2020a). Therefore, we incorporate these two techniques by (1) simultaneously train two models that are randomly initialized and average their soft label $\boldsymbol{q}_i^t$ to produce a new soft label, (2) use their ensemble prediction during test. The results on CIFAR datasets are shown in Table 7.

| Dataset | CIFAR-10 | | | | | CIFAR-100 | | | |
|---|---|---|---|---|---|---|---|---|---|
| Noise type | Sym. | | | | Asym. | Sym. | | | |
| Noise ratio | 20% | 50% | 80% | 90% | 40% | 20% | 50% | 80% | 90% |
| DivideMix w/o ensemble | 95.0 | 93.7 | 92.4 | 74.2 | 91.4 | 74.8 | 72.1 | 57.6 | 29.2 |
| DivideMix w/ ensemble | 95.7 | 94.4 | 92.9 | 75.4 | 92.1 | 76.9 | 74.2 | 59.6 | 31.0 |
| Ours | 95.8 | 94.3 | 92.4 | 75.0 | 91.9 | 79.1 | 74.8 | 57.7 | 29.3 |
| Ours w/ co-training | 96.1 | 94.8 | 92.8 | 76.3 | 92.4 | 79.8 | 75.3 | 58.9 | 31.5 |
| Ours w/ co-training & ensemble | **96.4** | **95.3** | **93.3** | **77.4** | **92.6** | **80.3** | **76.0** | **61.1** | **33.1** |

Table 7: Results of our proposed method with co-training and model ensemble.

## APPENDIX B PSEUDO-CODE FOR THE PROPOSED METHOD

**Algorithm 1:** Noise-Robust Contrastive Learning.

```
1  Input: noisy training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, model parameters $\theta$.
2  for $t \leftarrow 0$ to $t_0 - 1$ do        // learn from noisy labels for $t_0$ epochs (warm-up)
3      $\{\hat{\boldsymbol{z}}_i\}_{i=1}^n = \{f_\theta(\boldsymbol{x}_i)\}_{i=1}^n$
         // get normalized low-dimentional embeddings for all center-cropped images
4      $\{\hat{\boldsymbol{z}}^c\}_{c=1}^C = $ Calculate-Prototype($\{\hat{\boldsymbol{z}}_i, y_i\}_{i=1}^n$)
         // calculate class prototypes as normalized mean embeddings
5      for $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{2b}$ in $\mathcal{D}$ do        // load a minibatch with weak and strong augmentations
6          $\hat{\boldsymbol{z}}_i = f_\theta(\boldsymbol{x}_i)$        // obtain normalized low-dimensional embeddings
7          $\lambda \sim \text{Beta}(\alpha, \alpha)$        // sample a mixup weight from a beta distribution
8          $\boldsymbol{x}_i^m = \lambda \boldsymbol{x}_i + (1 - \lambda)\boldsymbol{x}_{m(i)}$        // generate virual training samples
9          $\hat{\boldsymbol{z}}_i^m = f_\theta(\boldsymbol{x}_i^m)$        // obtain embeddings for virtual samples
10         $\mathcal{L} = \sum_{i=1}^{b} \mathcal{L}_{\text{ce}}(\boldsymbol{x}_i, y_i) + \sum_{i=1}^{2b} \left(\omega_{\text{cc}}\mathcal{L}_{\text{cc}}(\hat{\boldsymbol{z}}_i) + \omega_{\text{pc}}\mathcal{L}_{\text{pc\_mix}}(\hat{\boldsymbol{z}}_i^m, y_i, \lambda) + \omega_{\text{recon}}\mathcal{L}_{\text{recon}}(\boldsymbol{x}_i, \hat{\boldsymbol{z}}_i)\right)$
11         $\theta = \text{SGD}(\mathcal{L}, \theta)$        // compute loss and update model parameters
12     end
13 end
14 for $t \leftarrow t_0$ to MaxEpoch do        // learn from psuedo-labels
15     $\{\hat{\boldsymbol{z}}_i^t, \boldsymbol{p}_i^t\}_{i=1}^n = \{f_\theta(\boldsymbol{x}_i)\}_{i=1}^n$
         // get embeddings and softmax predictions for all center-cropped images
16     $\boldsymbol{q}_i^t = \frac{1}{2}\boldsymbol{p}_i^t + \frac{1}{2}\sum_{j=1}^k w_{ij}^t \boldsymbol{q}_j^{t-1}, \quad \boldsymbol{q}_i^{t_0-1} = \boldsymbol{p}_i^{t_0}$
         // aggregate information from top-k neighbors to generate soft labels
17     $\mathcal{D}_{\text{sup}}^t = \{\boldsymbol{x}_i, y_i \mid q_i^t(y_i) > \eta_0\} \cup \{\boldsymbol{x}_i, \hat{y}_i^t = \arg\max_c q_i^t(c) \mid \forall \max_c q_i^t(c) > \eta_1, c \in \{1, .., C\}\}$
         // construct a subset with clean samples and pseudo-labeled samples
18     Repeat line 4-12, but only use samples from $\mathcal{D}_{\text{sup}}^t$ to compute $\hat{\boldsymbol{z}}^c$, $\mathcal{L}_{\text{ce}}$, $\mathcal{L}_{\text{pc\_mix}}$.
19 end
```

## APPENDIX C EFFECT OF THE PROPOSED LOSSES

In order to study the effect of the proposed losses, we remove each of them and report the classifier's accuracy across four benchmarks. As shown in Table 8, the mixup prototypical contrastive loss

($\mathcal{L}_{\text{pc\_mix}}$) is most crucial to the model's performance. The consistency contrastive loss ($\mathcal{L}_{\text{cc}}$) has a stronger effect with corrupted input or larger number of classes.

| | CIFAR-10 Sym 50% | + CIFAR-100 20k | + Image Corruption | CIFAR-100 Sym 50% |
|---|---|---|---|---|
| Ours | 94.3 | 91.5 | 91.4 | 74.8 |
| without $\mathcal{L}_{\text{recon}}$ | 93.3 | 90.7 | 90.2 | 73.2 |
| without $\mathcal{L}_{\text{cc}}$ | 93.7 | 91.3 | 89.4 | 71.9 |
| without $\mathcal{L}_{\text{pc\_mix}}$ | 85.9 | 79.7 | 81.6 | 65.6 |

Table 8: Effect of the proposed losses on classifier's accuracy.

## APPENDIX D IMPLEMENTATION DETAILS FOR REAL-WORD NOISY DATASETS

Here we provide the implementation details for our experiments on WebVision and Clothing1M. For WebVision, we follow previous works (Chen et al., 2019; Li et al., 2020a) and use inception-resnet v2 (Szegedy et al., 2017) as the encoder. We train the model using SGD with a weight decay of 0.0001 and a batch size of 64. We train for 40 epochs with an initial learning rate of 0.04. The hyper-parameters are set as $d = 50, \omega_{\text{cc}} = 1, \omega_{\text{pc}} = 2, \omega_{\text{recon}} = 1, \tau = 0.3, \alpha = 0.5, \eta_0 = 0.05, \eta_1 = 0.8, t_0 = 15$. For Clothing1M, we follow previous works (Han et al., 2019; Li et al., 2020a) and use ResNet-50 with ImageNet pretrained weights. We sample 1000 mini-batches as one epoch, and train the model for 50 epochs with an initial learning rate of 0.01. The hyper-parameters are set as $d = 32, \omega_{\text{cc}} = 1, \omega_{\text{pc}} = 1, \omega_{\text{recon}} = 1, \tau = 0.3, \alpha = 0.5, \eta_0 = 0.4, \eta_1 = 0.9, t_0 = 1$.

## APPENDIX E EXAMPLES OF NOISE CLEANING

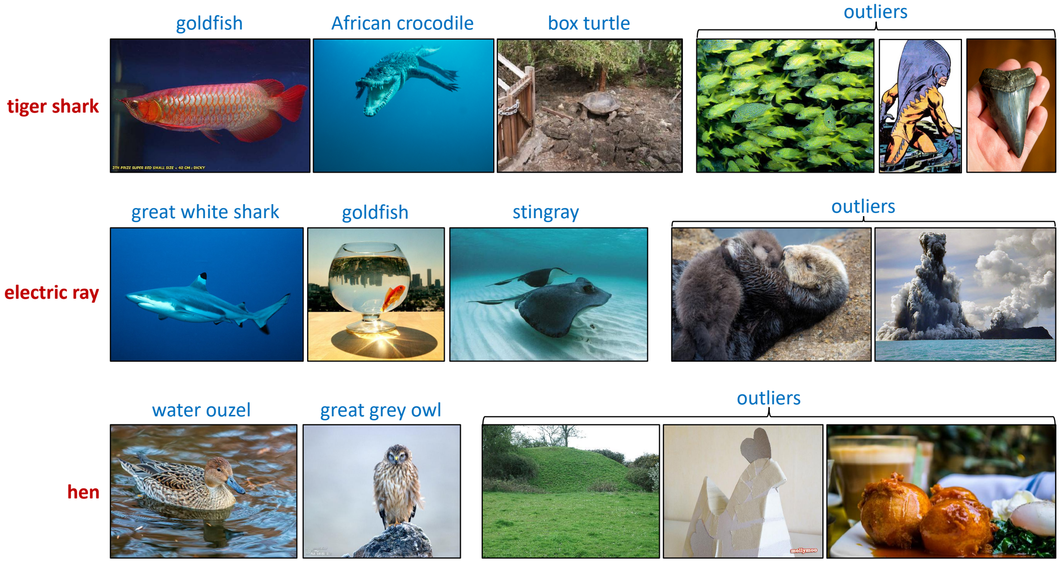

Figure 6: Examples of noisy images from WebVision that are identified by our method. The original corrupted labels are shown in red, and the hard pseudo-labels generated by model are shown in blue. Outliers are samples that did not pass the confidence threshold and thus not included in the supervised subset.