Supplementary material for
PLS: Robustness to label noise with two stage detection

1. Interpolated contrastive label

Figure 1 illustrates the process for computing the contrastive label L using the probability for a pseudo-label to be correct w, the guessed pseudo-labels and the positional encoding vector. Instead of simply ignoring the samples whose pseudo-label we can't reliably guess, we learn unsupervised features by enforcing the network to learn similar features between augmented views of a same image. For samples that are clean or whose pseudo-label we can rely on, we enforce the network to learn inter-image semantics between images of a same class.

2. Hyper-parameter study

Table 1 reports a study for values of μ (consistency regularization temperature) and γ (contrastive loss temperature). We report the top-1 accuracy for different on CIFAR-100 corrupted with 40% of synthetic in-distribution noise.

3. Pseudo-loss selection vs confidence threshold

A common strategy which has been used in the semi-supervised litterature [3, 4] and to some extent in the label nosie litterature [2] is to only keep pseudo-label whose confidence is superior to a threshold. We compare this approach against our pseudo-loss selection in Table 2 where we find that our pseudo-loss selection produces a better final validation accuracy. This is probably due to the fact that our selection computes a new threshold every epoch using an unsupervised Gaussian mixture. We believe that this allows to dynamically adapt to the learning state of the network as training progresses.

4. PLS algorithm

Algorithm 1 presents pseudocode for the PLS algorithm where we train a neural network ϕ robustly on a label noise dataset $\mathcal{D}$

Table 1. Ablation study on CIFAR-100 corrupted with $r_{in} = 0.4$.

μ	0.1	0.15	0.2	0.25	0.3	0.5
Top-1 val accuracy	77.86	78.65	78.45	78.78	78.45	77.74
γ	0.5	1	1.5	2	2.5	3
Top-1 val accuracy	76.39	78.54	78.43	78.33	78.45	77.99

Table 2. Thresholds on pseudo-label confidence vs our selection. CIFAR-100 corrupted with $r_{in} = 0.4$.

thresh	0.8	0.9	0.95	0.98	0.99	ours
Top-1 val accuracy	77.40	77.38	77.18	77.19	77.05	78.45

References

[1] Paul Albert, Eric Arazo, Noel O'Connor, and Kevin McGuinness. Embedding contrastive unsupervised features to cluster in-and out-of-distribution noise in corrupted image datasets. In *European Conference on Computer Vision (ECCV)*, 2022.

[2] Junnan Li, Caiming Xiong, and Steven CH Hoi. Learning from noisy data with robust representation learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[3] K. Sohn, D. Berthelot, C.-L. L, Z. Zhang, N. Carlini, E. Cubuk, A Kurakin, H. Zhang, and C. Raffel. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *arXiv: 2001.07685*, 2020.

[4] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

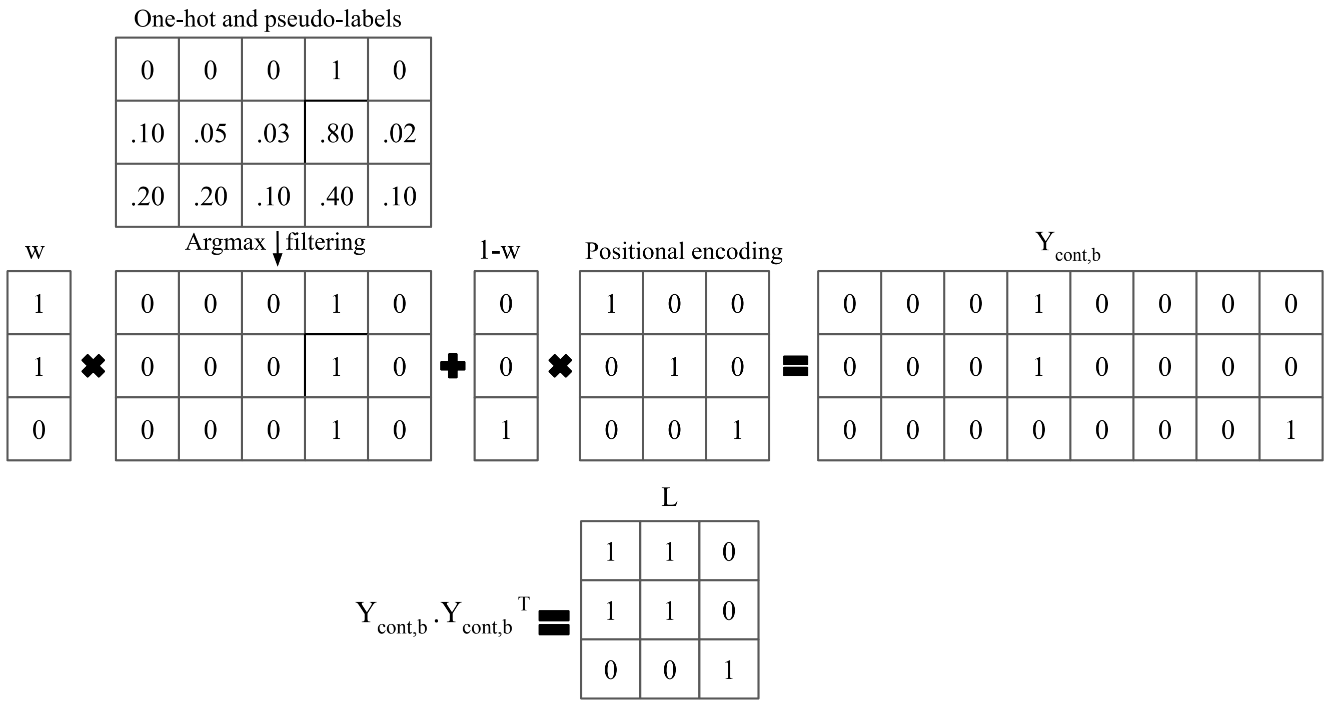

Figure 1. Toy example for the interpolated contrastive label between labeled and unlabeled with a mini-batch size b=3 and 5 classes. Multiplications are computed row-wise and the + sign indicates a concatenation. In this example, the sample in the first row is detected clean, the second is noisy but reliably corrected and the third is noisy but unreliably corrected.

Table 3. Mitigating ID noise and OOD noise on CIFAR-100 corrupted with ImageNet32 or Places365 images. Accuracy numbers from [1]. We run PLS for 100 epochs. We bold the highest best accuracy and report standard deviation over 3 random noisy corruptions and network initialization.

Corruption	r_{out}	r_{in}	CE	M	DB	JoSRC	ELR	EDM	DSOS	RRL	SNCF	PLS
INet32	0.2	0.2	63.68	66.71	65.61	67.37	68.71	71.03	70.54	72.64	72.95	**74.12** $\pm$ 0.28
	0.4	0.2	58.94	59.54	54.79	61.70	63.21	61.89	62.49	66.04	67.62	**69.31** $\pm$ 0.19
	0.6	0.2	46.02	42.87	42.50	37.95	44.79	21.88	49.98	26.76	53.26	**52.25** $\pm$ 0.26
	0.4	0.4	41.39	38.37	35.90	41.53	34.82	24.15	43.69	31.29	54.04	**53.72** $\pm$ 0.49
Places365	0.2	0.2	59.88	66.31	65.85	67.06	68.58	70.46	69.72	72.62	71.25	**74.69** $\pm$ 0.05
	0.4	0.2	53.46	59.75	55.81	60.83	62.66	61.80	59.47	65.82	64.03	**69.01** $\pm$ 0.61
	0.6	0.2	39.55	39.17	40.75	39.83	37.10	23.67	35.48	49.27	49.83	**54.34** $\pm$ 0.31
	0.4	0.4	32.06	34.36	35.05	33.23	34.71	20.33	29.54	26.67	50.95	**51.41** $\pm$ 0.55

Algorithm 1 PLS

Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ a web noise dataset. ϕ a randomly initialized CNN and g a linear projection to the contrastive space.
Parameters: $\alpha, \gamma, \mu, e_{\text{warmup}}, e_{\max}, gmm_{\text{thresh}}$
Output: Trained neural network ϕ

```
1:  for $e = 1, \dots e_{\text{warmup}}$ do                                          ▷ Warmup
2:      for $t = 1, \dots batches$ do
3:          Sample the next mini-batch $(x, y)$ from $\mathcal{D}$
4:          $x_{\text{mixed}}, y_{\text{mixed}}$ = mixup(x, y, $\alpha$)
5:          $l = \text{crossEntropy}(\phi(x_{\text{mixed}}), y_{\text{mixed}})$
6:          $h = \text{updateNetworkWeights}(l)$
7:      end for
8:  end for
9:
10: for $e = e_{\text{warmup}} + 1, \dots e_{\max}$ do                                ▷ PLS correction
11:     isNoisy = detectNoise($\mathcal{D}$, $\phi$, $gmm_{\text{thresh}}$)      ▷ Detect the noise using the small loss and a GMM
12:     for $t = 1, \dots batches$ do
13:         Two weakly augmented views $(x_1, y)$ and $(x_2, y)$ from $\mathcal{D}$
14:         $pseudoLab = \text{constReg}(\phi(x_1), \phi(x_2), \gamma)$                ▷ Pseudo-label guess from Eq. 1
15:         $y[\text{isNoisy}] = pseudoLab[\text{isNoisy}]$      ▷ Replace the labels of detected noise with the pseudo-labels
16:         Unaugmented view $(x, y)$ from $\mathcal{D}$
17:         $pseudoLoss = \text{crossEntropy}(\phi(x), pseudoLab)$        ▷ In practice, done in the detectNoise function
18:         $w = \text{GMM}(pseudoLoss)$      ▷ Compute the probability to belong to the low loss mode of the pseudoLoss
19:         $x_{\text{mixed}}, y_{\text{mixed}}$ = mixup$(x_1, y, \alpha)$                  ▷ Mixup with corrected labels
20:         $l_{\text{classif}} = w \times \text{crossEntropy}(\phi(x_{\text{mixed}}), y_{\text{mixed}})$            ▷ Weighed cross-entropy
21:
22:         $y = \text{oneHot}(y)$                                              ▷ One-hot filtering
23:         $L = \text{contLabel}(y, w)$                ▷ Compute the interpolated contrastive label Sec. 3.2.4
24:         Strong aug $X'$ from $\mathcal{D}$
25:         $sims = (g(\phi(x_{\text{mixed}})).g(\phi(X'))^T)/\mu$            ▷ Contrastive feats through projection
26:         $l_{\text{cont}}$ = contrastiveLoss(sims, L)                                ▷ From Eq. 5
27:
28:         $h = \text{updateNetworkWeights}(l_{\text{classif}} + l_{\text{cont}})$
29:     end for
30: end for
31: return $\phi$                                                   ▷ Robustly trained network
```