UNICON: Combating Label Noise Through Uniform Selection and Contrastive Learning (Supplementary Material)

Nazmul Karim[†] Mamshad Nayeem Rizve[‡] Nazanin Rahnavard[†] Ajmal Mian[§] Mubarak Shah[‡]
[†]Department of Electrical and Engineering, UCF, USA
[‡]Center for Research in Computer Vision, UCF, USA
[§]Department of Computer Science and Software Engineering, UWA, Australia
{nazmul.karim18, nayeemrizve}@knights.ucf.edu, nazanin.rahnavard@ucf.edu
ajmal.mian@uwa.edu.au, shah@crcv.ucf.edu

1. Overview

Section 2 describes our SSL-Training method in detail. Section 3 discusses the findings of our ablation studies. Section 5 has some details about hyperparameter settings and experimental results.

2. SSL-Training Details

Before semi-supervised learning (SSL), we separate the training set into $\mathbb{D}_{clean}$ and $\mathbb{D}_{noisy}$ by applying uniform selection. A sample training set with 60% noise level is shown in Figure 1. We consider $\mathbb{D}_{clean}$ and $\mathbb{D}_{noisy}$ to be labeled and unlabeled data, respectively. At the beginning of SSL, we create four sets of weakly-augmented (WA) data:

- Two sets of weakly-augmented labeled data $\{\hat{\mathbf{x}}_{i,1}^{weak}, \hat{\mathbf{x}}_{i,2}^{weak} : i \in (1, ..., N)\}$.
- Two sets for weakly-augmented unlabeled data $\{\hat{\mathbf{u}}_{i,1}^{weak}, \hat{\mathbf{u}}_{i,2}^{weak} : i \in (1, ..., N)\}$.

In addition, we also generate four sets of strongly-augmented (SA) data:

- Two sets of strongly-augmented labeled data $\{\hat{\mathbf{x}}_{i,1}^{strong}, \hat{\mathbf{x}}_{i,2}^{strong} : i \in (1, ..., N)\}$.
- Two sets of strongly-augmented unlabeled $\{\hat{\mathbf{u}}_{i,1}^{strong}, \hat{\mathbf{u}}_{i,2}^{strong} : i \in (1, ..., N)\}$.

Here, weak augmentations are used for label updating (label-refinement and pseudo-label guessing). We employ strong augmentations for updating the network parameters using backpropagation. For label-refinement [7], we use the networks' prediction to a weakly-augmented sample $\mathbf{x}_i$ for refining the given-label $\mathbf{y}_i$. For $\{\hat{\mathbf{x}}_{i,1}^{weak}, \hat{\mathbf{x}}_{i,2}^{weak}\}$, the output probabilities can be written as,

$$\mathbf{p}_i = \frac{1}{2}\sum_{m=1}^{2} \mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_{i,m}^{weak}; \theta^{(k)}); \phi^{(k)}), \tag{1}$$

where N is the number of data points in the training set and $\mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_{i,m}^{weak}; \theta^{(k)}); \phi^{(k)})$ is the Softmax probabilities of network-k (k=1,2) corresponding to $\hat{\mathbf{x}}_{i,m}^{weak}$.

After getting $\mathbf{p}_i$, we refine the label as follows:

$$\bar{\mathbf{y}}_i = w_i \mathbf{y}_i + (1 - w_i)\mathbf{p}_i, \tag{2}$$

where w_i is the label refinement coefficient. However, w_i can be calculated from the JSD values as,

$$w_i = \begin{cases} 1 - d_i, & \text{if } d_i \geq d_\omega \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

where d_ω is the label-refinement threshold that adjusts w_i based on the JSD of sample $\mathbf{x}_i$. Next, we follow the temperature sharpening [7] step given that gives us $\hat{\mathbf{y}}_i$.

Similarly, we calculate pseudo-label by averaging the predictions of both networks [7], i.e.

$$\bar{\mathbf{q}}_b = \frac{1}{4}\sum_{m=1}^{2} \big(\mathbf{h}(\mathbf{f}(\hat{\mathbf{u}}_{b,m}^{weak}; \theta^{(1)}); \phi^{(1)}) + \mathbf{h}(\mathbf{f}(\hat{\mathbf{u}}_{b,m}^{weak}; \theta^{(2)}); \phi^{(1)})\big) \tag{4}$$

and apply temperature sharpening on it to get $\mathbf{q}_b$.

We aggregate the labeled and unlabeled images with their ground-truth labels and pseudo-labels, respectively. That is, $\hat{\mathcal{X}} = \{(\hat{\mathbf{x}}_{i,m}^{strong}, \hat{\mathbf{y}}_i); i \in (1, ..., N), m = (1, 2)\}$, and $\hat{\mathcal{U}} = \{(\hat{\mathbf{u}}_{i,m}^{strong}, \mathbf{q}_i); i \in (1, ..., N), m = (1, 2)\}$ are the labeled and unlabeled sets. We use MixMatch [2] to have

$$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big), \tag{5}$$

$$\hat{\mathcal{X}} = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \ldots, |\hat{\mathcal{X}}|)\big), \tag{6}$$

$$\hat{\mathcal{U}} = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \ldots, |\hat{\mathcal{U}}|)\big). \tag{7}$$

Figure 1. Sample images from Clothing1M [16] dataset. We show the given label (bottom) and indicate label noise (top) for each image. Noisy samples are marked as positive (red) while clean samples contain negative marks (green). Here, the noise rate is 60% (3/5). Note that these images are taken for demonstration purpose only and corresponding labels are not their original given labels.

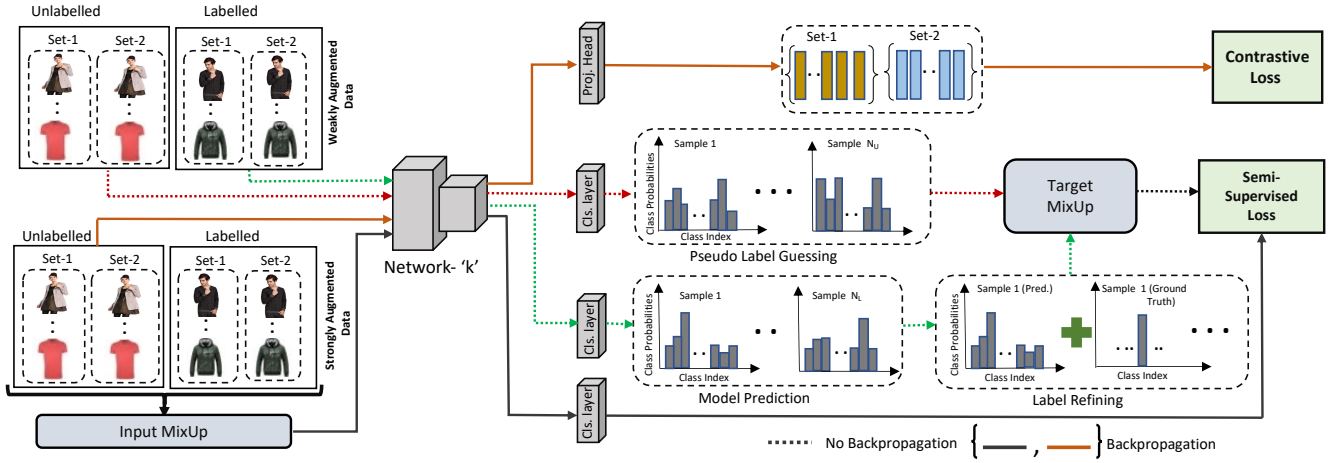


Figure 2. SSL-Training of network-'k' (k=1,2). After separating the samples, we create total 8 sets of weak and strong augmented data. While weakly augmented data helps with target label generation, strongly augmented data are used for updating the parameters through backpropagation. There are two types of label generation here: pseudo label guessing (4) (represented by green color) and label-refinement (2) (represented by red color). We have semi-supervised (eq. 11) and contrastive (eq. 13) losses that are minimized during training. Note that for pseudo-label guessing we take the average of both network-(1,2) predictions which is not shown here (eq. 4).

MixUp [21] proposed a strategy for generating convex combination of two inputs: in this case, samples from labeled and unlabeled sets and their corresponding ground-truth labels and pseudo-labels.

2.1. Loss Functions

After applying MixMatch, the semi-supervised losses are calculated as follows [2],

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\hat{\mathcal{X}}|} \sum_{\mathbf{x},\mathbf{p} \in \hat{\mathcal{X}}} \mathrm{H}(\mathbf{p}, \mathbf{h}(\mathbf{f}(\mathbf{y} \mid \mathbf{x}; \theta); \phi)), \tag{8}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{|\hat{\mathcal{U}}|} \sum_{\mathbf{u},\mathbf{q} \in \hat{\mathcal{U}}} \|\mathbf{q} - \mathbf{h}(\mathbf{f}(\mathbf{y} \mid \mathbf{u}; \theta); \phi)\|_2^2, \tag{9}$$

where $\mathrm{H}(\mathbf{p}, \mathbf{q})$ is the cross-entropy between distributions $\mathbf{p}$ and $\mathbf{q}$ with $\mathbf{y}$ as the given label.

Additionally, to prevent single-class assignment of all samples, we use a regularization term based on a prior uniform distribution ($\pi_c = 1/C$) to regularize the network's output across all samples in the mini-batch similar to Tanaka et al. [13] ,

$$\mathcal{L}_{reg} = \sum_{c} \pi_c log\Big(\frac{\pi_c}{\frac{1}{|\hat{\mathcal{X}}+\hat{\mathcal{U}}|}\sum_{\mathbf{x} \in |\hat{\mathcal{X}}+\hat{\mathcal{U}}|} \mathbf{h}(\mathbf{f}(\mathbf{x}; \theta); \phi)}\Big) \tag{10}$$

This gives us our semi-supervised loss function as shown in Figure 2,

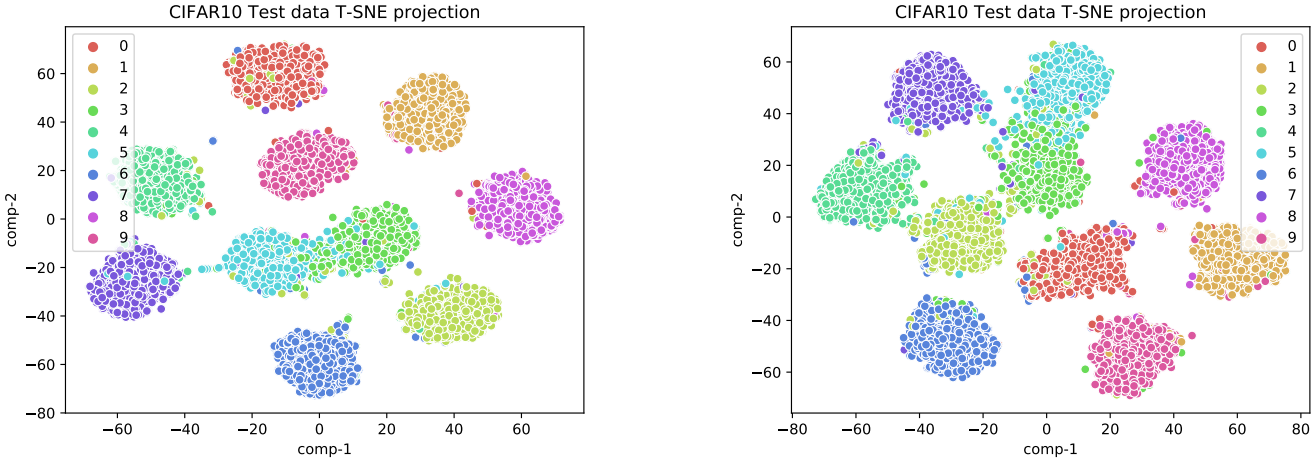
$$\mathcal{L}_{semi} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\mathrm{reg}}. \tag{11}$$

Here, $\lambda_{\mathcal{U}}$ and λ_r are unsupervised loss coefficient and regularization coefficient, respectively.

We consider another loss function, contrastive loss, which is used only for the data points in $\mathbb{D}_{noisy}$. Let the projection head output corresponding to $\hat{\mathbf{u}}_{i,1}^{strong}$ and $\hat{\mathbf{u}}_{i,2}^{strong}$ be $\mathbf{z}_i$ and $\mathbf{z}_j$, respectively. The contrastive loss function [3, 6]

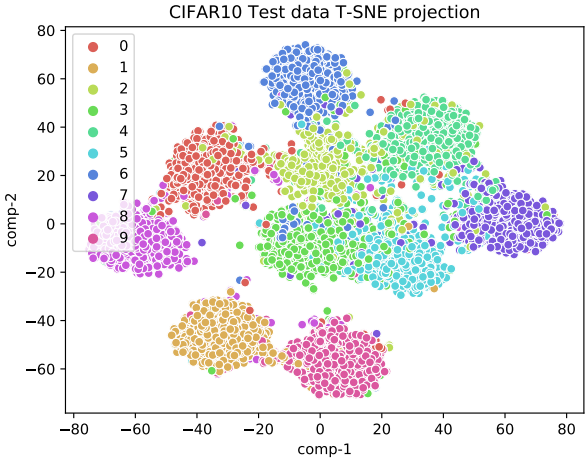
Dataset	CIFAR10				CIFAR100			
Noise Rate	50%		80%		50%		80%	
Method	Best	Last	Best	Last	Best	Last	Best	Last
UniCon w/o $\mathcal{L}_{\mathcal{U}}$	94.89	94.70	87.82	87.10	74.99	74.73	56.94	56.04
UniCon w/o $\mathcal{L}_{reg}$	95.38	95.11	93.59	93.26	76.48	75.87	61.75	60.90
UniCon	**95.61**	**95.24**	**93.97**	**93.97**	**77.63**	**76.91**	**63.98**	**63.13**

Table 1. Contribution of different loss functions on the performance of UniCon. While removing each loss term decreases the test accuracy, $\mathcal{L}_{\mathcal{U}}$ plays the most important role in obtaining SOTA performance. Test accuracies from the last epoch are also shown.

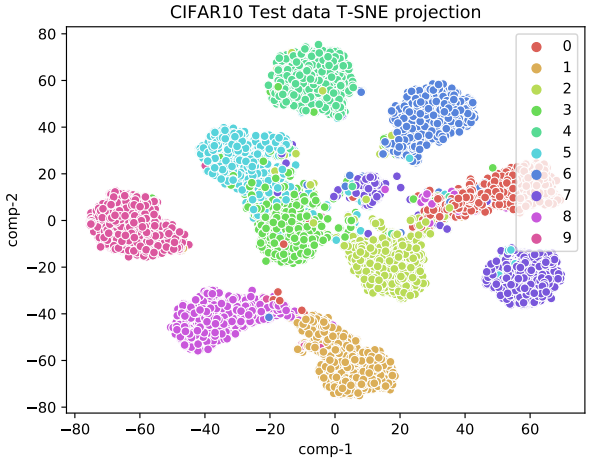


(a) Symmetric 50%

(b) Symmetric 80%



(c) Symmetric 90%



(d) Asymmetric 40%

Figure 3. T-SNE visualizations of network features of test images. The graphs show class distribution after training the network for 300 epochs on CIAFAR10 dataset with different noise types: (a) 50% symmetric, (b) 80% symmetric, (c) 90% symmetric, (d) 40% asymmetric. Even under extreme label-noise, UniCon effectively learns the true class distributions.

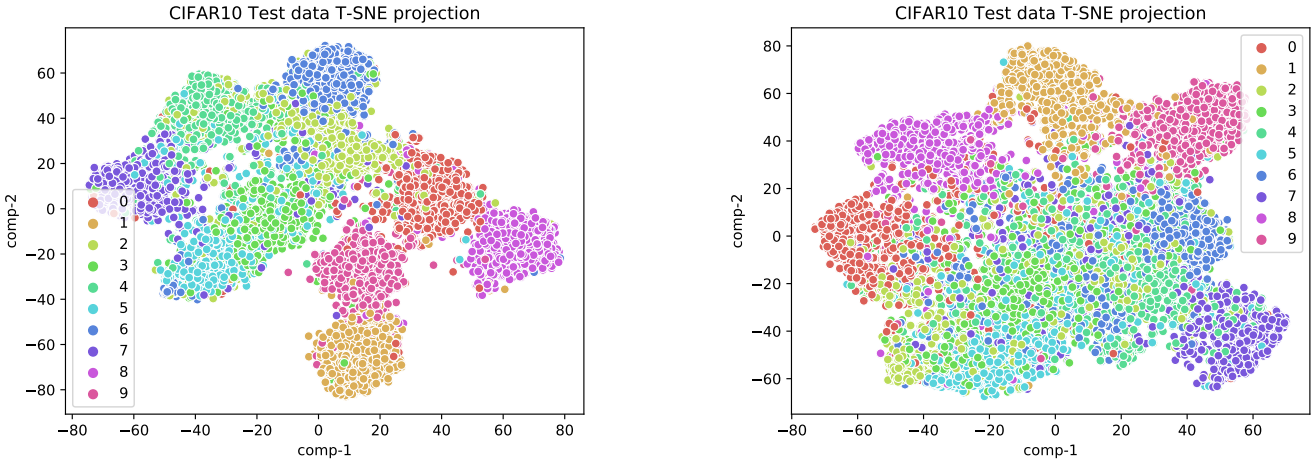
can be defined as

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\kappa)}{\sum_{b=1}^{2B} \mathbb{1}_{b \neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_b)/\kappa)}, \tag{12}$$

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{2B} \sum_{b=1}^{2B} [\ell_{2b-1,2b} + \ell_{2b,2b-1}], \tag{13}$$

where $\mathbb{1}_{b \neq i}$ is an indicator function that gives a 1 iff $b \neq i$, κ is a temperature constant, B is the number of samples in mini-batch, and $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$ can be expressed as the cosine similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$.

For each mini-batch, there are total 2B augmented sam-



(a) Class distribution learned by UNICON

(b) Class distribution learned by DMix [7]

Figure 4. Class distribution learned by (a) the proposed UNICON and (b) DMix [7] on CIFAR10 dataset with 95% symmetric noise. UNICON shows better class separation even when only 5% samples have correct labels.

Algorithm 1: One epoch of SSL Training

Input: network-1 parameters $\Theta^{(1)} = (\theta^{(1)}, \phi^{(1)}, \psi^{(1)})$ and network-2 parameters $\Theta^{(2)} = (\theta^{(2)}, \phi^{(2)}, \psi^{(2)})$, training set $\mathbb{D} = (\mathcal{X}, \mathcal{Y})$, number of samples N, number of classes C, sharpening temperature T, unsupervised loss coefficient $\lambda_{\mathcal{U}}$, contrastive loss coefficient $\lambda_{\mathcal{C}}$, and regularization coefficient λ_{r} .

for $k = 1$ **to** 2 **do**

- $\mathbb{D}_{clean}, \mathbb{D}_{noisy}, \boldsymbol{d} = \textit{Uniform–Selection}\big(\mathbb{D}, (\theta^{(1)}, \phi^{(1)}), (\theta^{(2)}, \phi^{(2)}), N, C\big)$ (see Alg. 1 of main paper) `// Separation of clean and noisy set`
- $\mathbb{W} = \textit{Weight–Estimation}\big(\boldsymbol{d}\big)$ (see eq. 3) `// Weights for label-refinement`
- **for** iter $= 1$ **to** num_iters **do**
  - From $(\mathbb{D}_{clean}, \mathbb{W})$, draw a mini-batch $\{(\mathbf{x}_b, \mathbf{y}_b, w_b); b \in (1, ..., B)\}$ `// Draw labeled data for SSL`
  - From $\mathbb{D}_{noisy}$, draw a mini-batch $\{\mathbf{u}_b; b \in (1, ..., B)\}$ `// Draw unlabeled data for SSL`
  - **for** $b = 1$ **to** B **do**
    - **for** $m = 1$ **to** 2 **do**
      - $\hat{\mathbf{x}}_{b,m}^{weak} = \textit{Weak–Augment}(\mathbf{x}_b)$ `// First weakly-augmented copy`
      - $\hat{\mathbf{u}}_{b,m}^{weak} = \textit{Weak–Augment}(\mathbf{u}_b)$ `// Second weakly-augmented copy`
      - $\hat{\mathbf{x}}_{b,m}^{strong} = \textit{Strong–Augment}(\mathbf{x}_b)$ `// First strongly-augmented copy`
      - $\hat{\mathbf{u}}_{b,m}^{strong} = \textit{Strong–Augment}(\mathbf{u}_b)$ `// Second-strongly augmented copy`
    - Get $\mathbf{p}_b$ using Eq. 1 `// Model Prediction`
    - $\bar{\mathbf{y}}_b = w_b \mathbf{y}_b + (1 - w_b)\mathbf{p}_b$ `// Label-refinement`
    - $\hat{\mathbf{y}}_b = \mathrm{Sharpen}(\bar{\mathbf{y}}_b, T)$ `// Temperature sharpening`
    - Get $\bar{\mathbf{q}}_b$ using Eq. 4 `// Pseudo-label`
    - $\mathbf{q}_b = \mathrm{Sharpen}(\bar{\mathbf{q}}_b, T)$ `// Temperature sharpening`
  - $\hat{\mathcal{X}} = \{(\hat{\mathbf{x}}_{b,m}^{strong}, \hat{\mathbf{y}}_b); b \in (1, ..., B)\}$ `// labeled Set`
  - $\hat{\mathcal{U}} = \{(\hat{\mathbf{u}}_{b,m}^{strong}, \mathbf{q}_b); b \in (1, ..., B)\}$ `// Unlabeled Set`
  - $\mathcal{L}_{\mathcal{X}}, \mathcal{L}_{\mathcal{U}} = \mathrm{MixMatch}(\hat{\mathcal{X}}, \hat{\mathcal{U}})$ `// Apply MixMatch`
  - Calculate $\mathcal{L}_{\mathcal{C}}$ using eq. 13 `// Contrastive Loss`
  - $\mathcal{L}_{tot} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}}\mathcal{L}_{\mathcal{U}} + \lambda_{\mathcal{C}}\mathcal{L}_{\mathcal{C}} + \lambda_r \mathcal{L}_{\mathrm{reg}}$ `// Total loss`
  - $\Theta^{(k)} = \mathrm{SGD}(\mathcal{L}, \Theta^{(k)})$ `// Update the Parameters`

Return: Updated $\Theta^{(1)}$, $\Theta^{(2)}$.

ples, since we are creating a pair of augmented samples out of a single sample. Let us consider i and j as a positive pair, then the rest of the data points $(2B - 2)$ are treated as negative examples. We can compute the final contrastive loss $\mathcal{L}_{\mathcal{C}}$ across all the positive pairs, both (i, j) and (j, i) in a single mini-batch. The formulation of $\ell_{i,j}$ does not require any labels (ground-truth or pseudo-labels). Since contrastive loss does not require labels, it mitigates the negative impact of

Loss Coef.		CIFAR10		CIFAR100	
$\lambda_{\mathcal{U}}$	$\lambda_{\mathcal{C}}$	Best	Last	Best	Last
20	0.025	95.38	94.80	77.12	76.89
30	0.025	**95.61**	**95.24**	**77.63**	**76.91**
40	0.025	95.42	95.26	77.34	77.18
20	0.050	95.49	94.83	77.46	76.95
30	0.050	95.17	94.56	77.28	76.12
40	0.050	95.35	94.79	77.15	76.44

Table 2. Performance analysis of UniCon with different loss coefficients (50% symmetric noise). We observe that our proposed method is stable over different values of $\lambda_{\mathcal{U}}$ and $\lambda_{\mathcal{C}}$.

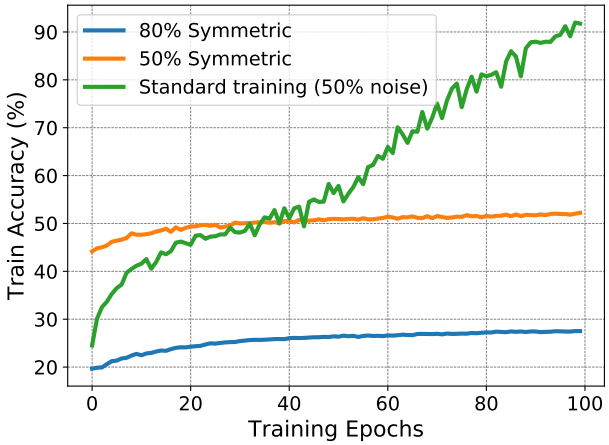


Figure 5. Training accuracy at different epochs. Low accuracy indicates that the networks do not memorize the noisy labels even after long training. Usually, standard cross-entropy based training leads to a high training accuracy (∼100%), i.e., complete memorization of noisy labels. However, our proposed joint selection and SSL scheme is shown to be effective in preventing such memorization.

noisy label memorization.

Finally, we accumulate all losses to get the total loss,

$$\mathcal{L}_{tot} = \mathcal{L}_{semi} + \lambda_{\mathcal{C}}\mathcal{L}_{\mathcal{C}}, \tag{14}$$

where $\lambda_{\mathcal{C}}$ is the contrastive loss coeffiecient. The summary of these steps is provided in Algorithm. 1.

3. Ablation Studies

In this section, we analyze the performance of UniCon under different scenarios.

3.1. Impact of Different Losses

We observe the contribution of each loss function on the performance of UniCon. It can be observed from Table 1 that each loss term helps in improving the performance while $\mathcal{L}_{\mathcal{U}}$ has the highest impact on performance. Training without $\mathcal{L}_{\mathcal{U}}$ indicates that we discard the selected noisy

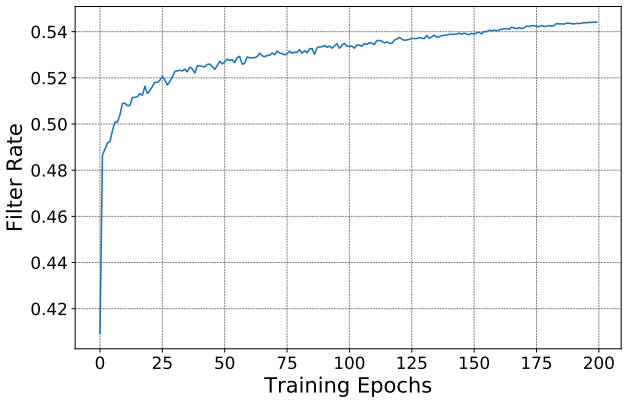


Figure 6. Our designed filtering rate R adjusts itself based on the network predictions without manual tuning at each training iteration [18]. As training progresses and the model gets confident about most of its predictions, UniCon selects more clean samples with better precision. For this graph we used CIFAR10 dataset with 50% symmetric noise.

samples completely. The drop in accuracy shows the significance of pseudo-label based feature learning. Improving the quality of these pseudo-labels is one of the primary contributions of UniCon.

3.2. Loss Coefficients

In Table 2, we show the effect of different loss coefficients. We observe that the performance of UniCon is relatively stable over a large range of coefficient values. We select a value of 30 and 0.025 for $\lambda_{\mathcal{U}}$ and $\lambda_{\mathcal{C}}$ respectively since this set of values result in optimal performance on both CIFAR10 and CIFAR100 datasets. We apply the same loss coefficient value for all datasets irrespective of the class number, number of samples, noise type, noise rate etc.

3.3. T-SNE Visualization

A t-SNE visualization [14] for features of test images is presented in Figure 3. The features are obtained from models trained under different label noise settings. We observe that class separation gets better as the noise level decreases. We further notice that UniCon obtains the best separation of test images at symmetric 50% noise. However, when the noise rate increases it becomes more challenging to learn the class distribution as shown in Figure. 3b and 3c. In addition, we compare the performance of our method with DMix [7] in the presence of 95% label noise in Figure 4. It is a difficult task to separate clean samples from noisy samples under such high noise rate. Interestingly, we observe that our simple approach effectively learns better class distribution in comparison to DMix [7]. We attribute this to the high precision of our uniform clean sample selection strategy.

Hyper Parameters	CIFAR10/100	Tiny-ImageNet200	Clothing1M	WebVision
Optimizer	SGD	SGD	SGD	SGD
Initial Learning Rate	0.02	0.01	0.002	0.01
Momentum	0.9	0.9	0.9	0.9
Weight Decay	$5e^{-4}$	$5e^{-4}$	$1e^{-3}$	$1e^{-3}$
Mini-batch Size	64	32	32	32
Total Epochs	300/350	350	8	100
T	0.5	0.5	0.5	0.5
$\lambda_{\mathcal{C}}$	0.025	0.025	0.025	0.025
$\lambda_{\mathcal{U}}$	30	30	30	30
λ_r	1	1	1	1
κ	0.05	0.05	0.05	0.05
d_ω	0.5	0.5	0.5	0.5
MixUp, α	4	2	0.5	0.5

Table 3. Hyperparameter Settings for UNICON. Most of the parameters are the same across different datasets. This shows the general applicability of the proposed UNICON method.

3.4. Memorization of Noisy Labels

In case of standard training, the network memorizes the noisy labels leading to poor generalization performance. However, our proposed method UNICON demonstrates resistance to memorization of label noise. We show this phenomena in Figure 5. We observe that with standard training the accuracy improves consistently over different epochs suggesting the memorization of label noise. In sharp contrast to this, the training accuracy of UNICON saturates very quickly indicating that the network is resisting the memorization of noisy labels at later stage of training. For instance, an ideal scenario for 80% symmetric noise would be if the training accuracy is $\sim$20%, i.e. the percentage of clean samples. Furthermore, we notice that our training accuracy deteriorates as we increase the rate of label noise in the training data. This further validates our claim that UNICON is effective in combating the memorization of label noise.

3.5. Filter Rate

In Figure 6, we show that the filter rate steadily increases as the network generates more confident predictions (shown for 50% noise rate). At each epoch of training, the filter rate, R is selected based on network predictions. This design decision omits the requirement of manually tuning the selection parameter (filter rate) at each training epoch [18]. For our experiments, we set d_μ, and τ to 0.7, and 5 respectively.

4. Baseline Methods

For CIFAR10 and CIFAR100, we compare UniCon with the following state-of-the art methods: LDMI [17], M-Up [21], PCIL [19], ELR [8], DMix [7], MOIT [11]. Methods like ELR [8] focus on the importance of the early learning regularization in preventing the memorization; MOIT [11] porposes a multi-objective framework to deal with the noisy labels. For Clothing1M, we consider Joint-Optim [13], MetaCleaner [22] along with ELR [8] and DMix [7] . Furthermore, D2L [9], MentrorNet [5] Co-Teaching [4], Iterative-CV [15] are among the methods we consider for WebVision. For TinyImageNet, we compare our method with Decoupling [10], MentorNet [5], Co-teaching+ [20], M-correction [1], NCT [12] etc.

5. Training Details

5.1. Hyper-parameter Settings

We describe the hyperparameter settings in Table 3. Note that most of these hyperparameters are the same across all datasets.

5.2. WebVision and Clothing1M

For Clothing1M dataset, first, we resize the image to 256×256 and then apply random crop to those images to obtain a 224×224 image. On the other hand, each image of WebVision is resized to 320×320 and a random crop of size 299×299 is applied. For WebVision, we consider only 50 classes for training and validation. Similarly, only 50 classes are considered for ILSVRC12 validation set. The percentage of noisy labels in WebVision are estimated to be around 20%. It has been shown that our method obtains slightly lower *Top-1* accuracy than state-of-the-art. In some scenarios (low noise level), the experimental results indicate that UNICON underperforms compared to the state-of-the-art. Relatively low performance on WebVision dataset can be attributed to the presence of low label noise.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019. 6

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019. 1, 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[4] 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. *arXiv preprint arXiv:1804.06872*, 2018. 6

[5] 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, 2018. 6

[6] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020. 2

[7] Junnan Li, Richard Socher, and Steven C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning, 2020. 1, 4, 5, 6

[8] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels, 2020. 6

[9] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018. 6

[10] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". *arXiv preprint arXiv:1706.02613*, 2017. 6

[11] Diego Ortego, Eric Arazo, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Multi-objective interpolation training for robustness to label noise, 2021. 6

[12] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Noisy concurrent training for efficient learning under label noise, 2020. 6

[13] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels, 2018. 2, 6

[14] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 5

[15] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels, 2018. 6

[16] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015. 2

[17] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. l_dmi: An information-theoretic noise-robust loss function, 2019. 6

[18] Yazhou Yao, Zeren Sun, Chuanyi Zhang, Fumin Shen, Qi Wu, Jian Zhang, and Zhenmin Tang. Jo-src: A contrastive approach for combating noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5192–5201, 2021. 5, 6

[19] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels, 2019. 6

[20] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W. Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption?, 2019. 6

[21] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. 2, 6

[22] Weihe Zhang, Yali Wang, and Yu Qiao. Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2019. 6