Meta-Learning Dynamic Center Distance: Hard Sample Mining for Learning with Noisy Labels

Supplementary Material

1. Overview

Section 2 provides a detailed description of our clean set construction strategy. Section 3 explains the SSL-Training process, offering a comprehensive overview of the training methodology.Section 4 explains the bilevel optimization algorithm involved in the meta-learning process, offering a detailed overview of the methodology. Section 5 presents additional experimental results, including experiments conducted under severe label noise conditions. Section 6 showcases more visualizations to further demonstrate the effectiveness of our approach. Finally, Section 7 provides a discussion and explanation of MEDAL in this paper.

2. Clean Set Construction

To identify D_{cle}, we adopt a novel sample selection strategy leveraging the Jensen-Shannon divergence (JSD) as a disagreement measure between the predicted probabilities and the ground-truth labels:

$$d_i = \text{JSD}(\boldsymbol{y}_i, \boldsymbol{p}_i), \tag{1}$$

where $\boldsymbol{y}_i$ is the true (or pseudo) label for the i-th sample. The JSD is defined as:

$$d_i = \frac{1}{2}\text{KLD}(\boldsymbol{y}_i \| \boldsymbol{m}_i) + \frac{1}{2}\text{KLD}(\boldsymbol{p}_i \| \boldsymbol{m}_i), \tag{2}$$

where $\boldsymbol{m}_i = \frac{\boldsymbol{y}_i + \boldsymbol{p}_i}{2}$ and KLD being the Kullback-Leibler divergence. To stabilize predictions, we employ an Exponential Moving Average (EMA) [5] to refine $\boldsymbol{p}_i$, yielding smoother and more robust divergence values.

To ensure class balance in D_{cle}, we implement a uniform selection [3] mechanism. First, the cutoff divergence value d_{cutoff} is determined using:

$$d_{\text{cutoff}} = \begin{cases} d_{\text{avg}} - \frac{d_{\text{avg}} - d_{\min}}{\tau}, & \text{if } d_{\text{avg}} \geq d_\mu \\ d_{\text{avg}}, & \text{otherwise} \end{cases} \tag{3}$$

where d_{avg} and $d_{\min}$ are the average and minimum divergence values, τ is a filter coefficient, and d_μ is an adjustment threshold. This approach dynamically adjusts d_{cutoff} based on network predictions, avoiding excessive noisy sample selection early in training.

Next, we divide samples into class-specific subsets and apply uniform filtering:

$$\mathbf{d}^{(j)}_{\text{filtered}} = \text{Lowest } R \text{ portion of } \mathbf{d}^{(j)}, \tag{4}$$

where R is the filter rate determined by the percentage of samples below d_{cutoff}, and $\mathbf{d}^{(j)}$ represents the divergence values for class j. The final clean subset is obtained by combining selected samples across all classes:

$$D_{\text{cle}} = \bigcup_{j=1}^{C} D^{(j)}_{\text{cle}}, \tag{5}$$

where C is the total number of classes.

In high noise rate scenarios, small-loss selection often yields too few clean samples, limiting training effectiveness. To address this, we apply a label correction strategy by identifying high-confidence samples and correcting their labels, even when they differ from the noisy labels.

Let p_i be the maximum predicted probability for sample i, and $\hat{y}_i = \arg\max_k p_{i,k}$ be its predicted label, where $p_{i,k}$ is the predicted probability of class k for sample i. We fit a Gaussian Mixture Model (GMM) [7] with two components to the distribution of p_i and define a threshold T based on the mean μ_h and variance σ_h^2 of the high-confidence component:

$$T = \mu_h + \beta\sigma_h. \tag{6}$$

We select samples satisfying $p_i > T$ to form a high-confidence set. For these samples, if the predicted label $\hat{y}_i$ differs from the noisy label y_i, we correct the label:

$$y_i = \begin{cases} \hat{y}_i, & \text{if } p_i > T \text{ and } \hat{y}_i \neq y_i \\ y_i. & \text{otherwise} \end{cases} \tag{7}$$

By combining these corrected samples with the initial clean samples from small-loss selection, we enhance the training dataset's quality and improve the robustness of the learned model. Based on our empirical observations, we set $\beta = 0$ to achieve optimal performance.

3. SSL-Training Details

3.1. Training Pipeline

In this semi-supervised learning (SSL) framework, we begin by splitting the training data into two subsets, D_{clean} and D_{noisy}, through a uniform selection process. Here, D_{clean} refers to the labeled set, while D_{noisy} represents the unlabeled set. At the outset of the SSL procedure, we generate several augmented data sets to facilitate both label refinement and model parameter updates. Specifically, we create weakly-augmented (WA) data for both the labeled and unlabeled data:

$$\hat{D}_{\text{WA}}^{\text{labeled}} = \left\{ \hat{\mathbf{x}}_{i,1}^{\text{weak}}, \hat{\mathbf{x}}_{i,2}^{\text{weak}} \mid i = 1, \ldots, N \right\}. \tag{8}$$

$$\hat{D}_{\text{WA}}^{\text{unlabeled}} = \left\{ \hat{\mathbf{u}}_{i,1}^{\text{weak}}, \hat{\mathbf{u}}_{i,2}^{\text{weak}} \mid i = 1, \ldots, N \right\}. \tag{9}$$

Additionally, we generate strongly-augmented (SA) data for both labeled and unlabeled samples:

$$\hat{D}_{\text{SA}}^{\text{labeled}} = \left\{ \hat{\mathbf{x}}_{i,1}^{\text{strong}}, \hat{\mathbf{x}}_{i,2}^{\text{strong}} \mid i = 1, \ldots, N \right\}. \tag{10}$$

$$\hat{D}_{\text{SA}}^{\text{unlabeled}} = \left\{ \hat{\mathbf{u}}_{i,1}^{\text{strong}}, \hat{\mathbf{u}}_{i,2}^{\text{strong}} \mid i = 1, \ldots, N \right\}. \tag{11}$$

Weak augmentations, $\hat{\mathbf{x}}_{i,m}^{\text{weak}}$, are primarily used for label-refinement and pseudo-label generation. For each weakly-augmented labeled sample $\hat{\mathbf{x}}_{i,m}^{\text{weak}}$, we calculate the predicted probability using the current model's output. Denote the output probabilities of the network for the weakly-augmented samples as follows:

$$\mathbf{p}_i = \frac{1}{2} \sum_{m=1}^{2} \mathbf{h} \left(\mathbf{f} \left(\hat{\mathbf{x}}_{i,m}^{\text{weak}}; \theta^{(k)} \right); \phi^{(k)} \right), \tag{12}$$

where $\mathbf{h} \left(\mathbf{f} \left(\hat{\mathbf{x}}_{i,m}^{\text{weak}}; \theta^{(k)} \right); \phi^{(k)} \right)$ denotes the softmax output of network k ($k = 1, 2$) for the weakly-augmented input $\hat{\mathbf{x}}_{i,m}^{\text{weak}}$. This output is then used to refine the label $\mathbf{y}_i$ through a weighted combination of the original label and the network's predicted label:

$$\bar{\mathbf{y}}_i = w_i \mathbf{y}_i + (1 - w_i) \mathbf{p}_i, \tag{13}$$

Here, w_i is the label refinement coefficient, which is computed based on the Jensen-Shannon Divergence (JSD) between the softmax predictions of the two networks:

$$w_i = \begin{cases} 1 - d_i, & \text{if } d_i \geq d_\omega \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

where d_i is the JSD value for the weakly-augmented sample $\hat{\mathbf{x}}_i$, and d_ω is a threshold used to control the weight assigned to the original label versus the predicted label. After obtaining the refined labels, we apply temperature sharpening to adjust the confidence of the predicted labels, as described in [6], to generate $\hat{\mathbf{y}}_i$.

Similarly, for unlabeled data, we generate pseudo-labels by averaging the predictions of both networks on the weakly-augmented unlabeled data:

$$\begin{aligned} \bar{\mathbf{q}}_b = \frac{1}{4} \sum_{m=1}^{2} & \mathbf{h} \left(\mathbf{f} \left(\hat{\mathbf{u}}_{b,m}^{\text{weak}}; \theta^{(1)} \right); \phi^{(1)} \right) \\ & + \mathbf{h} \left(\mathbf{f} \left(\hat{\mathbf{u}}_{b,m}^{\text{weak}}; \theta^{(2)} \right); \phi^{(2)} \right). \end{aligned} \tag{15}$$

The resulting pseudo-labels $\mathbf{q}_b$ are then refined by applying temperature sharpening.

We now aggregate the labeled and unlabeled data together, with their respective refined labels and pseudo-labels. The sets of labeled and unlabeled data are as follows:

$$\hat{\mathcal{X}} = \left\{ \left(\hat{\mathbf{x}}_{i,m}^{\text{strong}}, \hat{\mathbf{y}}_i \right) \mid i = 1, \ldots, N, m = 1, 2 \right\}. \tag{16}$$

$$\hat{\mathcal{U}} = \left\{ \left(\hat{\mathbf{u}}_{i,m}^{\text{strong}}, \mathbf{q}_i \right) \mid i = 1, \ldots, N, m = 1, 2 \right\}. \tag{17}$$

To improve model generalization, we apply the MixMatch strategy [1] for data augmentation. First, we shuffle the concatenated labeled and unlabeled sets:

$$\mathcal{W} = \text{Shuffle} \left(\text{Concat} \left(\hat{\mathcal{X}}, \hat{\mathcal{U}} \right) \right). \tag{18}$$

Then, we perform MixUp on both labeled and unlabeled sets:

$$\hat{\mathcal{X}} = \left(\text{MixUp} \left(\hat{\mathcal{X}}_i, \mathcal{W}_i \right); i \in (1, \ldots, |\hat{\mathcal{X}}|) \right). \tag{19}$$

$$\hat{\mathcal{U}} = \left(\text{MixUp} \left(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|} \right); i \in (1, \ldots, |\hat{\mathcal{U}}|) \right). \tag{20}$$

Here, MixUp [8] generates convex combinations of two samples (labeled and unlabeled) along with their corresponding labels (ground-truth or pseudo). This strategy encourages the model to learn more robust decision boundaries by interpolating between samples in both the feature and label spaces.

3.2. Loss Functions

After completing the weight calculation, we first select a clean subset with normal sample selection and label correction techniques [3]. Then we employ a weighted cross-entropy to focus more on the hard-clean examples. The loss function for hard-labeled samples is defined as:

$$\mathcal{L}_{\mathcal{HF}} = - \sum_{i=1}^{N} \Gamma_i \cdot H \left(y_i, h \left(f(x_i; \theta_f); \theta_h \right) \right), \tag{21}$$

where $H(y, \hat{y})$ is the cross-entropy loss between the true label y and the predicted probability distribution $\hat{y}$.

To integrate both clean and noisy samples in a semi-supervised learning (SSL) framework [1], we combine the supervised loss with additional loss terms to handle label noise and improve robustness. The overall training objective is then defined as:

$$\mathcal{L}_{\text{MEDAL}} = \mathcal{L}_{\mathcal{HF}} + \lambda_u \mathcal{L}_u + \lambda_r \mathcal{L}_r + \lambda_{\mathcal{C}} \mathcal{L}_{\mathcal{C}}, \tag{22}$$

where $\mathcal{C}$ is the contrastive loss, $\mathcal{L}_u$ and $\mathcal{L}_r$ handle unlabeled data and regularization, respectively. $\lambda_{\mathcal{C}}, \lambda_u, \lambda_r$ are the corresponding hyper-parameters.

Additionally, we introduce a regularization term based on a prior uniform distribution ($\pi_c = 1/C$), as follows:

$$\mathcal{L}_r = \sum_c \pi_c \log \left(\frac{\pi_c}{\frac{1}{|\hat{\mathcal{X}}+\hat{\mathcal{U}}|} \sum_{\mathbf{x} \in \hat{\mathcal{X}}+\hat{\mathcal{U}}} \mathbf{h}\left(\mathbf{f}(\mathbf{x};\theta);\phi\right)} \right). \tag{23}$$

This regularization term helps in controlling the distribution of network outputs across all samples in the mini-batch.

In addition to these components, we include a contrastive loss that applies only to noisy samples. Let the projection head outputs for the strong augmentations $\hat{\mathbf{u}}_{i,1}^{\text{strong}}$ and $\hat{\mathbf{u}}_{i,2}^{\text{strong}}$ be $\mathbf{z}_i$ and $\mathbf{z}_j$, respectively. The contrastive loss function [4] is defined as:

$$\ell_{i,j} = -\log \left(\frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\kappa\right)}{\sum_{b=1}^{2B} \mathbb{1}_{b \neq i} \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_b)/\kappa\right)} \right), \tag{24}$$

where $\mathbb{1}_{b \neq i}$ is an indicator function that takes a value of 1 if $b \neq i$, κ is the temperature constant, and B is the mini-batch size. The similarity $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$ is the cosine similarity between the projections $\mathbf{z}_i$ and $\mathbf{z}_j$.

For each mini-batch, we have $2B$ augmented samples, and the pairs (i, j) form positive pairs. The remaining $2B-2$ samples are considered as negative examples. The final contrastive loss $\mathcal{L}_{\mathcal{C}}$ is computed as:

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{2B} \sum_{b=1}^{2B} \left[\ell_{2b-1,2b} + \ell_{2b,2b-1}\right]. \tag{25}$$

This formulation does not require any ground-truth labels or pseudo-labels. Since contrastive loss does not rely on labels, it helps mitigate the impact of noisy label memorization.

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

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MEDAL}} + \lambda_{\mathcal{C}} \mathcal{L}_{\mathcal{C}}, \tag{26}$$

where $\lambda_{\mathcal{C}}$ is the contrastive loss coefficient, and $\mathcal{L}_{\mathcal{C}}$ is the contrastive loss defined in Eq. (25).

4. Bi-level Optimization Details

The proposed method follows a bi-level optimization framework that jointly learns model parameters θ and meta-parameters α, β. This hierarchical structure contains two coupled optimization objectives:

Algorithm 1: Bi-level Optimization Framework

Require: Noisy training set D_{train}, clean validation set D_{cle}

Initialize model parameters θ, meta-parameters α, β
for epoch= 1 to K **do**
  Inner Loop: Update θ via SGD
  for $t = 1$ to T **do**
    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}}$
  end for
  Outer Loop: Compute meta-gradients
  Compute θ^* from inner optimization
  Calculate $\nabla_\alpha \mathcal{L}_{\text{val}}$ and $\nabla_\beta \mathcal{L}_{\text{val}}$ via Eq. 32
  Update $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_{\text{val}}$
  Update $\beta \leftarrow \beta - \eta_\beta \nabla_\beta \mathcal{L}_{\text{val}}$
end for
Optimized θ^*, α^*, β^*

4.1. Inner and Outer Objectives

The bi-level optimization is formulated as:

- **Inner Optimization** (Model Parameter Update): Fixed meta-parameters α and β, update model parameters θ:

$$\begin{aligned} \theta^*(\alpha, \beta) &= \arg\min_\theta \mathcal{L}_{\text{train}}(\theta, \alpha, \beta) \\ &= \sum_{i=1}^{N} \alpha_i H(p_i, \hat{y}_i) + \beta_i H(p_i, \tilde{y}_i) \end{aligned} \tag{27}$$

- **Outer Optimization** (Meta-Parameter Update): Update α and β using the optimized θ^* on a clean validation set:

$$\begin{aligned} \alpha^*, \beta^* &= \arg\min_{\alpha,\beta \geq 0} \mathcal{L}_{\text{val}}(\theta^*(\alpha, \beta)) \\ &= \frac{1}{M} \sum_{i=1}^{M} H\left(f(x_i^{\text{val}}; \theta^*), y_i^{\text{val}}\right) \end{aligned} \tag{28}$$

where $H(\cdot, \cdot)$ denotes cross-entropy, $\hat{y}_i$ and $\tilde{y}_i$ represent noisy label and pseudo-label respectively, and D_{cle} contains clean validation samples.

4.2. Gradient Update Rules

The optimization alternates between two gradient update phases:

- **Inner-Level Update**: Standard gradient descent on θ with learning rate η_θ:

$$\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}} \tag{29}$$

- **Outer-Level Update**: Meta-gradient updates for α and β using implicit differentiation:

$$\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_{\text{val}} \tag{30}$$

$$\beta \leftarrow \beta - \eta_\beta \nabla_\beta \mathcal{L}_{\text{val}} \tag{31}$$

The meta-gradients are computed via chain rule:

$$\nabla_{\alpha,\beta}\mathcal{L}_{\text{val}} = \underbrace{\frac{\partial \mathcal{L}_{\text{val}}}{\partial \theta^*}}_{\text{Direct gradient}} \cdot \underbrace{\frac{\partial \theta^*}{\partial \alpha / \partial \beta}}_{\text{Implicit gradient}} \tag{32}$$

4.3. Algorithm Summary

Algorithm 1 outlines the complete procedure. The structure enables simultaneous noise-robust learning (inner loop) and automatic meta-parameter adaptation (outer loop).

5. More Experiment Results

5.1. Severe Noise Conditions

We present the classification performance of our proposed method, MEDAL, on the CIFAR10 dataset under severe label noise conditions. We compare MEDAL with the existing Union [3] at noise rates of 90%, 92%, 95%, and 98%, as summarized in Table 1.

Noise Rate (%)	90%	92%	95%	98%
Union [3]	90.81	87.61	80.82	50.63
MEDAL (Ours)	**93.92**	**90.73**	**87.82**	**59.42**

Table 1. Classification performance (%) of the proposed method on CIFAR10 under severe label noise.

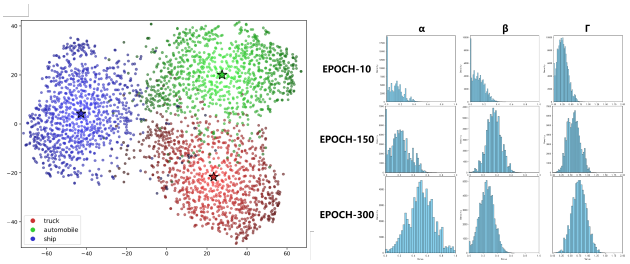
As shown in the Table 1, all methods experience a decline in classification accuracy as the noise rate increases. However, MEDAL consistently outperforms the Union method across all noise levels. Specifically, at a 90% noise rate, MEDAL achieves an accuracy of 93.92%, which is approximately 3.11% higher than Union's 90.81%. When the noise rate increases to 92% and 95%, MEDAL attains accuracies of 90.73% and 87.82%, respectively, significantly surpassing Union's 87.61% and 80.82%. Even under the extreme noise rate of 98%, MEDAL maintains an accuracy of 59.63%, exceeding Union's 50.63% by nearly 8.79%.

These results highlight the robustness of MEDAL in handling high label noise, whereas the Union method performs significantly worse as the noise rate exceeds 95%. Our approach better filters noisy data, maintaining higher classification performance under extreme conditions.

6. More Visualization Results

6.1. T-SNE Analysis of DCD-selected Samples

The t-SNE analysis (Fig. 1a) illustrates the feature distributions of three CIFAR10 classes with center c_k, where darker points represent samples with higher importance weights. Compared to traditional small-loss criteria, DCD's sample selection results demonstrate its ability to retain boundary-hard samples, which are crucial for improving model robustness and performance in noisy environments.



(a) T-SNE Analysis for selected set. (b) Distribution of α,β and Γ.

Meanwhile, we also investigated the evolution of learnable parameters in meta-learning as training epochs progress, shown in Fig. 1b. The stability of learned parameters α_i, β_i, and Γ_i reflects the model's ability to adapt and refine its sample selection process over time, further enhancing its robustness and performance.

6.2. Heatmap for All Epochs

We present heatmap visualization for all training epochs in Figure 2, highlighting the model's attention to various input regions. The extended epoch range in Figure 2 allows us to conclude that the relationship between DCD and Loss, as discussed in the main text, remains independent throughout the training process.

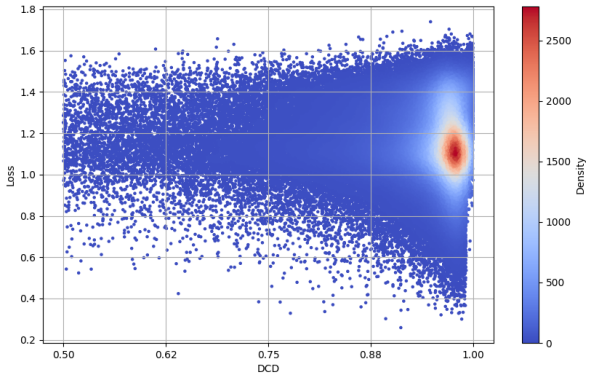


Figure 2. Heatmaps for all epochs. The heatmaps illustrate the areas the model focuses on during training.

6.3. Segmentation Results

Finally, we present additional segmentation results from the final model, highlighting its robustness against noisy labels and complex boundaries. As shown in Figure 3, the segmentation masks closely align with the ground truth, demonstrating the model's ability to effectively handle noisy annotations.

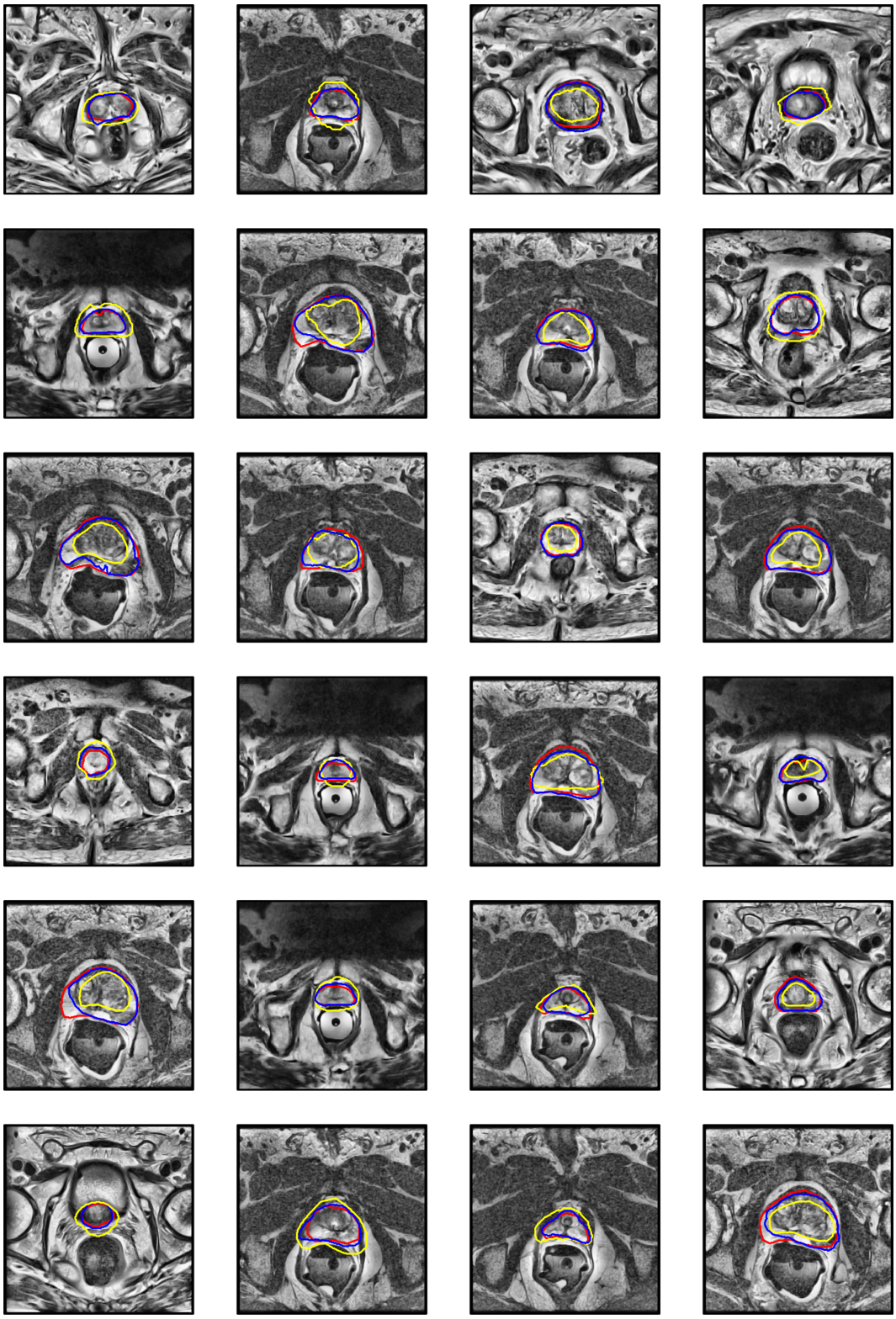


Figure 3. Segmentation results from the final model. These results show the model's accuracy in segmenting various structures in the images. The segmentation masks closely match the ground truth, indicating the model's robustness.

7. Further Discussion

7.1. Distinguishing DCD from Loss-based Methods

Symmetric noise	92%	95%	98%
DCD w/o EMA	88.91	86.15	56.78
DCD w/o LC	88.14	83.71	53.07
DCD (*Focal Loss*)	89.31 ± 0.32	85.93 ± 0.41	55.86 ± 0.49
DCD	**90.73 ± 0.29**	**87.82 ± 0.34**	**59.42 ± 0.42**

Table 2. Ablation Study of DCD on CIFAR-10 (± std. over 3 runs).

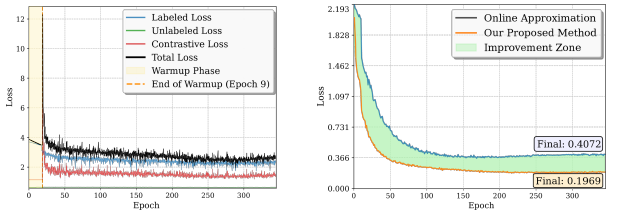
While the reweighting mechanism of DCD is inspired by L2B, their underlying motivations and solutions are fundamentally different. L2B reweights pseudo-labels and real labels to reduce the impact of erroneous pseudo-labels. In contrast, DCD adjusts the weights of easy and hard samples using dynamic center distance to effectively direct the model's focus towards more challenging examples. Unlike loss-based reweighting methods that assign high weights to large-loss samples, our approach introduces a non-loss-based reweighting mechanism driven by dynamic center distances, which can reduce the correlations between weights and losses and help prevent the amplification of high losses for mislabeled examples.

To demonstrate DCD's superiority over traditional loss-based methods, we compare against Focal Loss by replacing the DCD weight with $\Gamma_i = (1 - p_i[y_i])^\gamma$ and tuning γ according to validation performance. As shown in Table 2, DCD consistently outperforms Focal Loss across all noise levels, validating our distance-based approach.

7.2. Dynamic Class Centers and Sample Reliability

Class centers are calculated based on the obtained clean examples, whose labels include both annotated and pseudo-labels, derived separately through sample selection and label correction. In our approach, α weights the annotated labels, while β weights the pseudo-labels. Thus, we use the sum of α and β to reflect the overall reliability of each sample. This differs from conventional approaches that constrain $\alpha + \beta = 1$, allowing for more flexible sample importance modeling.

The mechanism behind class centers naturally aligning closer to their true counterparts when removing the sum-to-one constraint stems from the adaptive nature of meta-learned parameters α_i and β_i. This flexibility enables dynamic calibration of label reliability weights, particularly crucial when handling varying noise levels and sample confidence. The learning process consequently shifts class prototypes toward authentic class distributions through dual mechanisms: amplifying reliable supervisory signals (true labels) while suppressing noisy samples' contributions.



(a) Training Loss Components and Convergence Dynamics of DCD (b) Online Approximation vs. DCD for Validation Convergence

Figure 4. DCD Loss Dynamics and Performance Comparison.

7.3. Meta-Learning Convergence and Training Dynamics

Both α and β are updated after each optimization step. Unlike online approximation (OA), we terminate meta-optimization when the validation loss difference falls below a threshold. Compared with OA, our method can estimate better α and β values at the cost of approximately 1.5× training time.

To demonstrate the convergence properties, Figure 4a reports the training loss for each component, showing stable convergence of all loss terms. Figure 4b compares the validation losses between online approximation and our proposed DCD, demonstrating that our approach achieves lower validation loss through more accurate parameter estimation.

7.4. Training Process and Implementation Details

In each batch, meta-learning does not update the main model's parameters. Instead, it uses a small set of clean samples to compute weights for the filtered clean samples. These weights are then used by the SSL component, which updates the main model's parameters. Moreover, meta-learning is not involved in the warm-up phase. The proposed meta-learning approach does not introduce any additional hyperparameters. The hyperparameters used for self-supervised learning follow the settings from UNICON and remain consistent across all datasets.

7.5. Clean Validation Set Construction

For all datasets, we randomly split 2% of training examples as the clean validation set. A small clean set may be feasible for certain scenarios. Moreover, existing methods such as [2] can automatically identify clean examples, making DCD potentially applicable for label-scarce domains. This addresses the practical deployment considerations while maintaining the method's effectiveness.

7.6. Ablation Study on Key Components

We conduct comprehensive ablation studies on the two key components: (1) EMA smoothing, whose removal causes an average 2.04% performance drop across CIFAR-10 noise

levels; (2) Label Correction (LC), which is applied across all noise levels and is particularly effective under heavy noise, with its removal causing a substantial 6.35% accuracy decrease under extreme noise (98%). Results presented in Table 2 demonstrate the importance of each component.

7.7. Computational Overhead

Compared to L2B, our method increases per-epoch training time by 1.5× and raises GPU memory consumption from 5.3GB to 6.9GB. This modest increase in computational requirements is justified by the consistent performance improvements across all datasets and noise levels.

7.8. Handling Clean Samples and Noise

Examples with large losses are often seen as hard examples but are also likely to be noisy. The proposed DCD method identifies hard examples based on feature distances from meta-learned class centers, offering complementary information to loss values. This approach provides a more robust distinction between genuinely hard clean samples and mislabeled noisy samples, as the distance-based metric is less susceptible to the misleading signals that affect loss-based methods.

References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, 32, 2019. 2

[2] Cheng et al. Learning with bounded instance and label-dependent label noise. In *ICML*, 2020. 6

[3] Nazmul Karim, Mamshad Nayeem Rizve, Nazanin Rahnavard, Ajmal Mian, and Mubarak Shah. Unicon: Combating label noise through uniform selection and contrastive learning. In *CVPR*, pages 9676–9686, 2022. 1, 2, 4

[4] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *NeurIPS*, 33: 18661–18673, 2020. 3

[5] Frank Klinker. Exponential moving average versus moving exponential average. *Math.*, 58:97–107, 2011. 1

[6] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *ICLR*, 2020. 2

[7] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009. 1

[8] Hongyi Zhang. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 2