Neural Vicinal Risk Minimization: Noise-robust Distillation for Noisy Labels

Hyounguk Shon[1], Seunghee Koh[1], Yunho Jeon[2] and Junmo Kim[1,*]

[1]*Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, South Korea*
[2]*Hanbat National University, 125, Dongseo-daero, Yuseong-gu, Daejeon, 34158, South Korea*

Abstract
Training deep neural networks with noisy supervision remains a challenging problem in weakly supervised learning. Mislabeled instances can severely degrade the generalization ability of classification models to unseen data. In this paper, we propose a novel regularization method coined Noise-robust Distillation (NRD) that addresses robust training under noisy supervision. NRD is motivated from a novel learning framework which we name Neural Vicinal Risk (NVR) minimization to improve the estimation quality of the data distribution and handle label noise effectively. Our framework is based upon our observation that a neural network has capability to correctly classify data sampled from vicinal distribution even when the model is overfitted to noisy label. By ensembling the predictions from the neural vicinal distribution, we obtain an accurate estimation of the class probabilities that reflects sample-wise class ambiguity. We validated our method through various noisy label benchmarks and demonstrate significant improvement in robustness to label noise.

Keywords
Learning with Label Noise, Vicinal Risk Minization, Noise-robust Loss

1. Introduction

Deep learning models have achieved remarkable success in various domains, including image classification, natural language processing, and speech recognition. However, the performance of these models heavily relies on the availability of high-quality labeled data for training. Obtaining accurately annotated labels can be a challenging and time-consuming task, often requiring human annotators to manually label large amounts of data. As a result, noisy labels may arise during the annotation process, leading to suboptimal model performance.

In this paper, we address noisy label learning as a subset of a more generic type of problem. This encompasses learning from an over-confident target probability distribution and image ambiguity [1], human annotation errors, multiple classes in an image, and out-of-distribution training examples [2] that can naturally occur due to, for example, random crop data augmentation. We show that our generic noisy label supervision algorithm can address a combination of these issues using a simple and unified approach.

We propose a noise-robust learning algorithm named Noise-Robust Distillation (NRD) to address the issue of noisy supervision during training. NRD aims to improve the generalization performance of classification models by explicitly considering the noise and ambiguity in the training labels. We motivate NRD by a novel formulation of the noisy supervision learning problem which we name Neural Vicinal Risk (NVR) minimization.

This stems from the observation that deep neural networks have the inherent capability to detect and correct noisy supervision, even when it is trained using noisy supervision. This ability is particularly evident when considering the vicinal distribution, which represents the distribution generated from perturbed versions of the training data. Despite being trained on noisy labels, neural networks can still

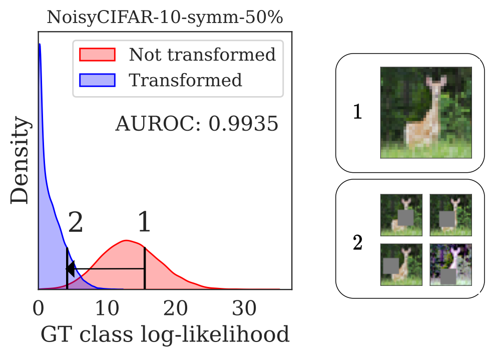


Figure 1: Averaging prediction over the novel views of a mislabeled training instance effectively mitigates memorization. The model is trained on the noisy training set and tested again using the training examples. The histogram shows the distribution of cross-entropy loss with respect to the GT labels. Red curve corresponds to standard prediction, and blue curve corresponds to ensembling over transformation views. The right side shows a training example with its original view vs. the transformed novel views. The corresponding loss is marked as “1” and “2” on the histogram.

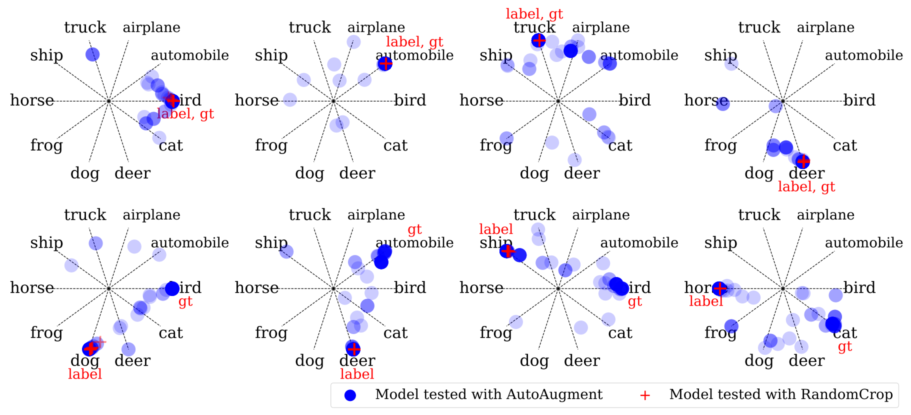
accurately model the vicinal distribution, indicating their potential to correct the noisy supervision.

Our findings suggest that the combination of perturbation-based estimation and ensembling can lead to improved model performance, even in the presence of noisy supervision. Building on these insights, we propose Noise-Robust Distillation (NRD), which is a noise-robust learning method that leverages the neural vicinal risk principle to enhance the generalization performance of classification models trained on noisy labels.

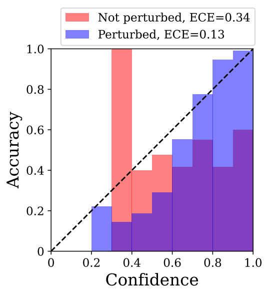
The main contributions of this work are as follows:

- We introduce the Noise-Robust Distillation (NRD), a noise-robust learning approach that comprehensively addresses the challenges posed by noisy supervision during training.
- NRD is motivated by a novel noise-robust learning framework which we name Neural Vicinal Risk (NVR) minimization. We show that NVR improves the estimation quality of the true class distribution and handles label noise effectively.
- We demonstrate the ability of neural networks to

The IJCAI-2024 AISafety Workshop, August 4, 2024, Jeju, South Korea
*Corresponding author.
hyounguk.shon@kaist.ac.kr (H. Shon); seunghee1215@kaist.ac.kr (S. Koh); yhjeon@hanbat.ac.kr (Y. Jeon); junmo.kim@kaist.ac.kr (J. Kim)
0000-0002-0867-1728 (H. Shon); 0009-0006-8662-0834 (S. Koh); 0000-0001-8043-480X (Y. Jeon); 0000-0002-7174-7932 (J. Kim)



(a) Softmax predictions of clean instances (top row) and mislabeled instances (bottom row) from the noisy training set. Each marker indicates a softmax vector projected onto a 2D decagon.



(b) Calibration plot

Figure 2: On Figure 2a, model prediction of noisy samples are more sensitive to perturbation with respect to the input. NoisyCIFAR-10 dataset is used. Markers indicate the softmax scores predicted from the model trained using random crop augmentation. Red markers (+) show predictions generated using the same augmentation policy used during training, and the blue markers (●) are generated using an unseen, stronger augmentation policy. The ten-class softmax scores are visualized by projecting onto a decagon using Equiradial Projection [3]. On Figure 2b, while the model itself is heavily mis-calibrated (red bars), ensembling the predictions of the perturbed inputs significantly improves the calibration. (blue bars)

detect and correct mislabeled examples through sensitivity to perturbations in the input data, leading to improved model predictions and calibration.

- We validate the effectiveness of NRD through experiments on benchmark datasets, showing clear improvements in model performance in comparison to standard training methods under noisy supervision.

2. Related works

Noisy label learning Numerous methods tackle the challenge of training Deep Neural Networks (DNNs) on datasets that contain a mix of correctly labeled and mislabeled samples, as discussed in [4]. Some approaches focus on designing a noisy-robust loss to mitigate the impact of mislabeled samples. Mean Absolute Error (MAE) loss [5] demonstrates competitive performance. Following this, the introduction of the Generalized Cross-Entropy (GCE), Symmetric Cross-Entropy (SCE) loss, and active passive loss are proposed with improved noisy-robustness. Generalized Jensen-Shannon divergence (GJS) [6] enforces consistency between predictions from multiple augmented views of a sample to regularize training. Also, the principle of negative learning is emphasized by [7, 8]. The strategies inspired by the training dynamics of models [9] such as early stopping [10, 11] or over-parameterization [12] exploit the different convergence speeds of clean and noisy samples. Co-teaching [13] involves simultaneous training of two DNNs, where each network learns from the clean samples chosen by its counterpart. Noise identification aims to filter noisy samples from the training dataset. Noisy samples can be filtered by measuring the degree of disagreement between ensemble models, which occurs once the model is overfitted to the noisy samples. Recent algorithms [14, 15, 16] utilize the power of Semi-Supervised Learning (SSL) by following a two-step process: filtering out noisy labels first, and then treating the detected noisy samples as unlabeled for reducing the noisy learning problem into a SSL task.

Semi-supervised learning (SSL) has emerged as a powerful method for noisy label learning. Among them, consistency regularization promotes a model to make consistent outputs across data augmentations, as in Π-model, Temporal Ensembling [17] and Mean Teacher [18]. Also, FixMatch [19] integrates pseudo-labeling and and virtual adversarial training [20] utilizes adversarial attacks. MixMatch [21], adopted by DivideMix [14], generates pseudo-label with sharpening for data-augmented unlabeled examples and mixes labeled and unlabeled data using MixUp [22].

Calibration and knowledge distillation Confidence calibration [23] is the process of adjusting a model's predicted probabilities to better reflect the true likelihood. It is demonstrated that training a model with data augmentation like Mixup [22] improves model calibration and robustness to noise [24]. Meanwhile, Knowledge Distillation (KD) [25] enhances the student model by transferring knowledge contained in the prediction of the teacher model, focusing on "dark" or "hidden" knowledge, including its confident and less confident predictions.

3. Preliminaries

3.1. Notations

Consider a DNN classification model parameterized by $\theta \in \Theta$ as $f(x, \theta) : \mathcal{X} \mapsto \Delta^{C-1}$ which outputs a probability distribution $P(y|x;\theta)$. The input space is defined as $\mathcal{X} = \mathbb{R}^{H \times W \times C}$ where H, W, C are the number of height, width, and color channels of the image data. Δ^k indicates k-simplex. The model takes an image input $x \in \mathcal{X}$ and predicts a categorical distribution over $\mathcal{Y} = \{1, 2, ..., C\}$. We denote an image augmentation operation as $\mathcal{T}(x) : \mathcal{X} \to \mathcal{X}$, and the training dataset as $\mathcal{D} = \{(x_i, y_i)\}_i$. The loss function is defined as $\ell(x, y, \theta) : \mathcal{X} \times \mathcal{Y} \times \Theta \mapsto \mathbb{R}$. $\delta(\cdot)$ is the Dirac delta function and $\mathbb{1}_{\{\cdot\}}$ is the indicator function.

3.2. Empirical Risk

The expected risk $R(\theta)$ is defined as the average loss over $p(x, y)$,

$$R(\theta) = \int_{x,y} \ell(x, y, \theta) p(x, y) \, dxdy \,. \tag{1}$$

In practice, a dataset $\mathcal{D}$ is used to mimic the true distribution $p(x, y)$, which leads to the empirical risk

$$\hat{R}(\theta) = \int_{x,y} \ell(x, y, \theta)\hat{p}(x, y)\, dxdy\,. \tag{2}$$

where the corresponding empirical distribution $\hat{p}(x, y)$ is a mixture of delta masses using the observed samples, and the class distribution is a one-hot distribution given by annotations,

$$\hat{p}(x, y) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{y=y_i\}}\delta(x - x_i)\,. \tag{3}$$

Our goal is to refine the estimation of the data distribution $p(x, y)$ by utilizing the empirical distribution $\hat{p}(x, y)$. A pivotal question that arises is how to enhance the approximation of the true risk $R(\theta)$ intrinsic to a classification model. As evidenced by Equation (3), this task necessitates the accurate estimation of two orthogonal components present within the true distribution $p(x, y) = P(y|x)p(x)$: (1) the input distribution $p(x)$ and (2) the corresponding conditional distribution $P(y|x)$.

3.3. Neural Empirical Risk

Estimating $P(y|x)$ as a one-hot distribution involves assigning a single class label per sample, which is vulnerable to human annotation errors. Unfortunately, it proves challenging to enhance or secure accurate supervision signals for $P(y|x)$, as this requires multiple human annotators reviewing the same image [1] which is a prohibitively costly process. Nonetheless, enhancing the estimation quality of the true class distribution $P(y|x)$ can lead to further improvements in estimating and minimizing the true risk.

Neural Empirical Risk (NER) Instead of using Equation (3), we can choose to parameterize $P(y|x)$ by a neural network $P(y|x, \phi)$ to further improve the estimation quality. First, we factorize the data distribution as $p(x, y) = P(y|x)p(x)$, and denote the corresponding empirical distributions as follows:

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i) \tag{4}$$

$$\hat{P}(y|x_i) = \mathbb{1}_{\{y=y_i\}}\,. \tag{5}$$

Instead of using $\hat{P}(y|x)$, we choose to use a distribution parameterized by a neural network trained on $\mathcal{D}$,

$$P(y|x, \mathcal{D}) = \int_{\phi} P(y|x, \phi)p(\phi|\mathcal{D})\, d\phi\,, \tag{6}$$

where $p(\phi|\mathcal{D})$ is the distribution over the function class parameterized by neural network. By plugging Equation (6) into $\hat{p}(x, y) = \hat{P}(y|x)\hat{p}(x)$, we define the *neural empirical distribution* $\hat{p}_\rho$ and the *neural empirical risk* $\hat{R}_\rho$ as

$$\hat{p}_\rho(x, y|\mathcal{D}) = P(y|x, \mathcal{D})\hat{p}(x) \tag{7}$$

$$\hat{R}_\rho(\phi) = \int_{x,y} \ell(x, y, \phi)\hat{p}_\rho(x, y|\mathcal{D})\, dxdy\,. \tag{8}$$

Here, we refer to the model $P(y|x, \phi)$ as the *teacher network* to distinguish from the model being trained, whose term is borrowed from knowledge distillation. This can provide better estimation quality than $\hat{P}(y|x)$ as is often observed in knowledge distillation, which we view as an instance of NER minimization. Knowledge distillation is known to improve generalization and calibration performance due to the dark knowledge [25].

However, when the model is over-fitted to the noisy label, it severely degrades the performance of estimating the class probabilities. Hence, in order to effectively utilize a neural network, it is necessary to employ a noise-robust method to accurately estimate the class probabilities in the presence of noisy labels.

3.4. Vicinal risk for noise-robust learning

Our motivation is based on the Vicinal Risk Minimization (VRM) principle [26], which is an alternative approximation to $p(x, y)$. The vicinal distribution $p_\nu(\tilde{x}, \tilde{y})$ constructed from the data distribution is defined as

$$p_\nu(\tilde{x}, \tilde{y}) = \int_{x,y} \nu(\tilde{x}, \tilde{y}|x, y)p(x, y)\, dxdy\,. \tag{9}$$

where $\nu(\tilde{x}, \tilde{y}|x, y)$ is the vicinity distribution around (x, y). For example, [26] used additive Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 I)$. MixUp [24] and CutMix [27] chose stochastic interpolation between samples which has also shown its effectiveness in noisy label. [24]. Using the dataset, Equation (9) is replaced by the empirical distribution as

$$\begin{aligned}\hat{p}_\nu(\tilde{x}, \tilde{y}) &= \int_{x,y} \nu(\tilde{x}, \tilde{y}|x, y)\hat{p}(x, y)\, dxdy &(10)\\ &= \frac{1}{n}\sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y}|x_i, y_i)\,. &(11)\end{aligned}$$

Neural Vicinal Risk (NVR) We propose to further improve by using a neural network to robustly approximate the data distribution by modifying Equation (9). We propose the following approximate vicinal data distribution parameterized by a deep neural network ϕ which we name *neural vicinal distribution* p_π.

$$\begin{aligned}p_\pi(\tilde{x}, \tilde{y}|\mathcal{D}) &= P(\tilde{y}|\tilde{x}; \mathcal{D})p(\tilde{x}) &(12)\\ &= \int_{\phi} P(\tilde{y}|\tilde{x}, \phi)dp(\phi|\mathcal{D}) \int_{x} \nu(\tilde{x}|x)dp(x) &(13)\\ &\approx \int_{\phi} P(\tilde{y}|\tilde{x}, \phi)d\delta(\phi - \phi^*) \int_{x} \nu(\tilde{x}|x)d\hat{p}(x) &(14)\\ &= P(\tilde{y}|\tilde{x}, \phi^*)\frac{1}{n}\sum_{i=1}^{n} \nu(\tilde{x}|x_i) &(15)\\ &= \frac{1}{n}\sum_{i=1}^{n} P(\tilde{y}|\tilde{x}, \phi^*)\nu(\tilde{x}|x_i)\,. &(16)\end{aligned}$$

Here, $\phi^* = \arg\min \hat{R}(\phi)$ is the maximum-a-posteriori (MAP) model trained on $\mathcal{D}$. It is important to note that the samples from the vicinal distribution $\nu(\tilde{x}_i|x_i)$ is not shown at the training of the model ϕ^*. Equation (14) is given by substituting the Bayesian model with the MAP model and also replacing the true distribution $p(x)$ with the empirical distribution. The true neural vicinal distribution is approximated by the ensembled MAP model predictions averaged over the samples from the vicinal distribution.

Therefore, we define the empirical *neural vicinal distribution* $\hat{p}_\pi$ as,

$$\hat{p}_\pi(\tilde{x}, \tilde{y}; \phi^*) = \frac{1}{n}\sum_{i=1}^{n} P(\tilde{y}|\tilde{x}, \phi^*)\nu(\tilde{x}|x_i)\,. \tag{17}$$

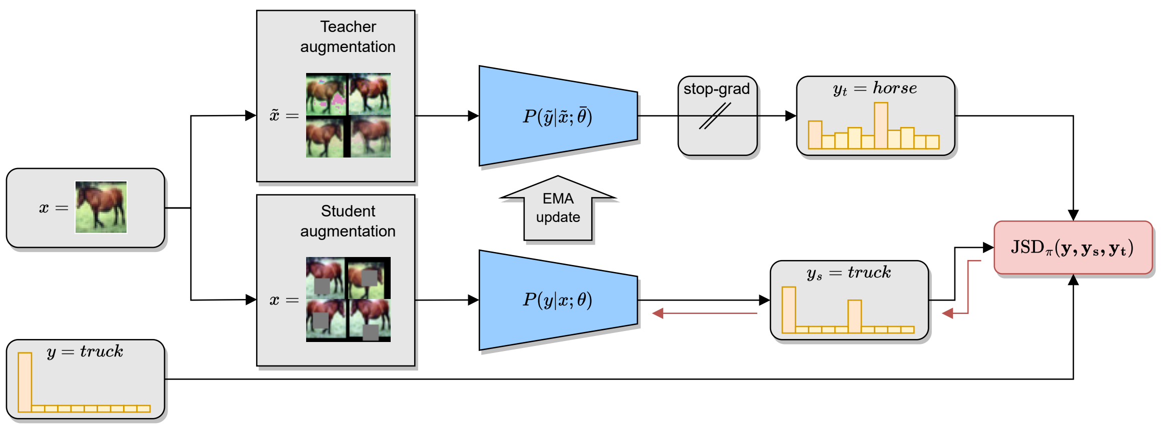


Figure 3: Illustration of the proposed Noise-robust Distillation (NRD) architecture. x is the input data to the neural network, and y is the assigned target label. Red arrows show the gradient propagation path. During training, the predictions from the original views (*student augmentation*) is regularized using the predictions generated from unseen views (*teacher augmentation*). We use asymmetric augmentation policy so that the teacher augmentation generates novel views, and the stop-gradient operation ensures that the model does not memorize the views generated from the teacher augmentation.

Table 1
Label correction behavior for memorized training examples using transformed views. The models are trained to perfectly memorize the noisy labels, then evaluated again for training set with ground-truth labels. Due to memorization, the GT accuracy for mislabeled instances is zero and the overall accuracy is bounded by the noise rate. However, averaging the predictions from the transformed inputs shifts the prediction of the noisy examples to the ground-truth. For the transformation, AutoAugment followed by RandomErasing was used. For the dataset, NoisyCIFAR-10 with symmmetric noise was used.

		Training accuracy for GT labels (%)		
η	Transform	Clean	Mislabeled	Overall
20%	×	99.99	0.01	81.93
	○	96.01	54.72	88.55
50%	×	99.97	0.07	55.11
	○	93.09	40.58	69.51
80%	×	99.83	0.06	28.10
	○	77.29	14.82	32.38

Note that Equation (17) is a parameterized version of Equation (11) using a deep neural network. Finally, the *neural vicinal risk* is,

$$\hat{R}_\pi(\phi) = \int_{x,y} \ell(x, y, \phi)\hat{p}_\pi(\tilde{x}, \tilde{y}; \phi^*)\, dxdy\,. \qquad (18)$$

Note that $\nu(\tilde{x}|x)$ is distinct from the augmentation strategy applied to the model being trained. Similar to Equation (8), we refer to the $\nu(\tilde{x}|x)$ as *teacher augmentation.*

3.5. Self-correction for memorized instances

We further discuss the behavior of the neural vicinal distribution over a noisy training dataset. Notably, when a training dataset includes mislabeled instances, a teacher neural network can overfit to these noisy labels, where the neural empirical risk minimization fails in mitigating the impact. Interestingly, we observe that the neural vicinal distribution exhibits robustness against label noise, effectively self-correcting incoherent labels within the training set.

To understand this phenomenon, we visualize the behavior of the neural vicinal distribution in Figure 2a. Here, we compare the softmax scores from the augmented input samples, distinguishing between the neural empirical distribution (red marker) and the neural vicinal distribution (blue markers). For the transformation policy, the network was trained using random crop augmentation and AutoAugment [28] is chosen as the vicinal distribution to generate the novel views. The top row shows clean instances and the bottom row shows mislabeled instances.

The visual analysis contrasts the softmax predictions from both seen and novel views of clean and mislabeled training instances. The self-correction of the neural vicinal distribution is instance-dependent which responds differently based on if an instance is clean or mislabeled. Notably, while the teacher network's predictions for the novel views tend to shift misclassified predictions towards ground truth, they remain consistent for clean samples. This suggests that the network outputs corrected predictions by dissociating the novel views from the memorized views.

Next, in Figure 1, we analyzed the label correction behavior of the neural vicinal distribution over the dataset population. Note that the models are trained only using the noisy training set, without access to the ground-truth labels. Applying transformation (blue curve) significantly reduced the GT class cross-entropy loss compared to no transformation (red curve), and we observed a good separation between the two distributions. Also, Table 1 shows the ground-truth accuracy for the training samples where we observed significant improvements for the mislabeled instances when transformation is applied.

We additionally observed that ensembling perturbed predictions enhances the calibration, as depicted in Figure 2b. While the original model is heavily over-confident due to overfitting (red), vicinal prediction improves accuracy and reflects class ambiguities. (blue)

4. Method

Motivated from the observation in Section 3.5, we propose a novel learning method for noisy labels named Noise-robust Distillation (NRD). Our method is formulated as a simple loss function which makes it easy to employ in existing training pipeline.

For this, we combine the target loss with the neural vicinal risk loss as a regularization objective. We formulate the combined objectives into a triplet loss. We have found Jensen-Shannon divergence (JSD) to be effective which generalizes to a triplet loss. The JSD for three distributions is,

$$\text{JSD}_\pi(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}) = \sum_i \pi_i D_{\text{KL}}(\mathbf{p_i} || \mathbf{m}) \,, \tag{19}$$

where $\mathbf{m} = \sum_i \pi_i \mathbf{p_i}$. The hyperparameter $\pi \in \Delta^2$ is chosen to balance the importance weight between the distributions. Additionally, JS divergence is known to have a nice robustness property against label noise. [6] showed that JS divergence simulates MAE loss [5] in its asymptote.

Next, we derive our NRD objective step-by-step. By applying NVR to JSD loss, we have

$$\mathcal{L}(\theta; x, \mathbf{y}, \phi) = \text{JSD}_\pi(\mathbf{y}, \mathbf{y_s}, \mathbf{y}_t) \tag{20}$$
$$\mathbf{y_s} = f(x, \theta) \tag{21}$$
$$\mathbf{y_t} = \mathop{\mathbb{E}}_{\nu(\tilde{x}|x)} [f(\tilde{x}, \phi)] \,, \tag{22}$$

assuming that we have a trained teacher network ϕ. Here, $\mathbf{y}$ is the target label, $\mathbf{y_s}$ is the model output and $\mathbf{y_t}$ is the teacher network output. The loss is solved for $\min_\theta \mathcal{L}(\mathbf{y}, \mathbf{y_s}, \mathbf{y_t})$.

To improve noise-robustness, we can further employ an iterative distillation scheme which we repeat the strategy for multiple rounds of training. We set the teacher network as the model obtained from the previous training round, such that $\phi_t = \theta_{t-1}$ at the t-th training round. Applying to Equation (20),

$$\theta_t = \arg\min_\theta \mathcal{L}(\theta; x, \mathbf{y}, \theta_{t-1}) \,. \tag{23}$$

A student network obtained from previous training round is switched to the teacher role for next round. However, in practice, we found this to be unstable and difficult to converge. Instead, we take the exponential average of the historical models as the teacher and set $\phi_t = \bar{\theta}_{t-1}$.

$$\bar{\theta}_t = \beta \cdot \bar{\theta}_{t-1} + (1 - \beta) \cdot \theta_t \,. \tag{24}$$

For the decay rate, we simply set $\beta = 0.99$ for all experiments. The aggregation reduces the variance of neural vicinal risk estimation caused by stochastic gradient, and we have empirically found that it effectively stabilizes the training and lead to faster convergence.

Finally, we formally define our NRD training objective. To reduce the training cost, we simplify each training round into a single step of stochastic gradient descent. (SGD) This simplifies the algorithm from multi-staged process into a single-staged process, and significantly accelerates the training. The NRD objective is,

$$\mathcal{L}_{\text{NRD}}(\theta; x, \mathbf{y}, \bar{\theta}) = \text{JSD}_\pi(\mathbf{y}, \mathbf{y_s}, \mathbf{y_t}) \tag{25}$$
$$\mathbf{y_s} = f(x, \theta) \tag{26}$$
$$\mathbf{y_t} = \mathop{\mathbb{E}}_{\nu(\tilde{x}|x)} \left[f(\tilde{x}, \bar{\theta})\right] \,, \tag{27}$$

with a slight abuse of notation for $\bar{\theta}$, which is not an optimization variable but continuously updated after each SGD step. This is implemented by detaching $\mathbf{y_t}$ from the

Algorithm 1 PyTorch-style pseudocode

```
ema_model = ema(model)
optimizer = sgd_optimizer(model)

for x, y in dataloader:
    x_t = teacher_aug(x)
    x_s = student_aug(x)

    # disconnect from backprop
    y_t = ema_model(x_t).detach()
    y_s = model(x_s)

    # distance between predictions
    loss = js_div(y, y_s, y_t)
    loss.backward()
    optimizer.step()
    ema_model.update()
```

backpropagation graph, which prevents the model from memorizing the teacher augmentation views. (*stop-grad* in Figure 3) For Equation (27), we found single sample per SGD step was sufficient. The overall architecture is illustrated in Figure 3 and the pseudocode is presented in Algorithm 1.

5. Experiments

5.1. Experimental settings

Benchmarking datasets For synthetic label noise benchmarks, we used NoisyCIFAR-10, NoisyCIFAR-100 [29]. For symmetric label noise, we randomly flip the ground truth label with a probability η uniformly across all categories. For asymmetric label noise, we follow the scheme in [30]. For NoisyCIFAR-10-asymm, we flip truck$\rightarrow$automobile, bird$\rightarrow$airplane, cat$\rightarrow$dog, dog$\rightarrow$cat, deer$\rightarrow$horse. For NoisyCIFAR-100-asymm, within each superclass, we randomly replace a subclass label y_i to adjacent subclass $y_i + 1$ with probability η.

For the real-world benchmark, we used WebVision [31] dataset. WebVision consists of 2.4M training examples collected via Google and Flickr image search. We used a miniaturized training set following [32] which uses only the first 50 categories in the "Google" image set. Mini-WebVision consists of 66K training and 2.5K validation examples. We additionally evaluated the trained model on ImageNet [33] validation set. The noise rate is known to be around 20%.

Baseline methods For the CIFAR benchmarks, we compare against cross-entropy (CE), bootstrapping (BS) [34], label smoothing (LS) [35], symmetric cross-entropy (SCE) [36], generalized cross-entropy (GCE) [37], normalized loss (NCE+RCE) [38], Jensen-Shannon divergence (JS, GJS) [6].

For the WebVision benchmarks, we compared our method with the state-of-the-art methods including ELR+ [10], DivideMix [14], and GJS [6]. The baseline results were adopted from [6].

Models PreActResNet-34 architecture [39] is used for all experiments conducted on CIFAR-10/100 datasets. For WebVision experiments, we used ResNet-50. All experiments were trained from random initialization.

Augmentation policy For the CIFAR experiments, we followed [6] and used RandAugment [40] chained with Cutout [41] for all methods. For the NRD teacher transformation, we used AugMix [42] in all experiments.

Hyperparameters For CIFAR-10/100 benchmarks, we used 400 epochs for each training. We used SGD optimizer with momentum 0.9 and weight decay of 10^{-4}. Learning rates

Table 2
Noisy label performance on synthetic noisy label benchmarks. We used NoisyCIFAR-10 and NoisyCIFAR-100 datasets. Values indicate clean test accuracy. All values are averaged over five independent runs. The best and second best results are highlighted in bold.

	NoisyCIFAR-10						NoisyCIFAR-100					
	Symmetric				Asymmetric		Symmetric				Asymmetric	
Noise rate	20%	40%	60%	80%	20%	40%	20%	40%	60%	80%	20%	40%
CE	91.63	87.74	81.99	66.51	92.77	87.12	65.74	55.77	44.42	10.74	66.85	49.45
BS	91.68	89.23	82.65	16.97	93.06	88.87	72.92	68.52	53.80	13.83	73.79	**64.67**
LS	93.51	89.90	83.96	67.35	92.94	88.10	74.88	68.41	54.58	26.98	73.17	57.20
SCE	94.29	92.72	89.26	**80.68**	93.48	84.98	74.21	68.23	59.28	26.80	70.86	51.12
GCE	94.24	92.82	89.37	79.19	92.83	87.00	75.02	71.54	65.21	**49.68**	72.13	51.50
NCE+RCE	94.27	92.03	87.30	77.89	93.87	86.83	72.39	68.79	62.18	31.63	71.35	57.80
JS	94.52	93.01	89.64	76.06	92.18	87.99	75.41	71.12	64.36	45.05	71.70	49.36
GJS	**95.33**	93.57	**91.64**	79.11	**93.94**	**89.65**	**78.05**	**75.71**	**70.15**	44.49	**74.60**	**63.70**
NRD (ours)	**95.43**	**94.65**	**92.45**	**85.32**	**93.90**	**91.25**	**78.54**	**76.29**	**72.43**	**60.01**	**76.07**	61.40

Table 3
Real-world noisy label benchmark on WebVision. The models are trained using the WebVision training set, and evaluated on WebVision and ImageNet validation sets. The values indicate accuracy. *IRNv2* and *RN50* indicates Inception-ResNet-V2 and ResNet-50, respectively. N indicates the number of networks used.

				WebVision		ImageNet	
Method	Arch.	Aug.	N	Top-1	Top-5	Top-1	Top-5
ELR+	IRNv2			77.78	**91.68**	70.29	89.76
DivideMix	IRNv2	MixUp	2	77.32	91.64	**75.20**	90.84
DivideMix	RN50			76.32	90.65	74.42	**91.21**
CE	RN50			70.69	88.64	67.32	88.00
JS	RN50			74.56	91.09	70.36	90.60
GJS	RN50	ColorJitter	1	**77.99**	90.62	74.33	90.33
NRD (ours)	RN50			**78.56**	**92.48**	**75.24**	**92.36**

were reduced by a factor of 0.1 after 200-th and 300-th epoch. For WebVision benchmarks, we trained the network for 300 epochs. Learning rate was reduced by a factor of 0.1 after 150 and 250 epochs. Refer to Appendix A for hyperparameter configuration details.

5.2. Results

Performance on noisy label benchmarks In Table 2, we show the performance of our method in comparison to robust loss functions. While most of the baselines shows inconsistent performance between symmetric and asymmetric noise types, our method shows consistent improvement across a wide range of noise rates and noise types. Notably, we significantly improve performance under high noise rate settings where GJS tend to underperform. For NoisyCIFAR-10 80% noise, we improve by 5%p over SCE, and for NoisyCIFAR-100-80%, we improve by 10%p over GCE.

Furthermore, the results on large-scale real-world noisy label benchmark is shown in Table 3. Notably, we observed that our method outperforms over existing methods that uses two networks.

Performance on clean datasets The proposed method improves model generalization when applied to clean dataset training as seen in Table 4. This is because the training dataset contains visually ambiguous images that make it difficult to draw a clear decision boundary, and therefore the hard target distributions from the annotations serve as

Table 4
Performance on clean CIFAR-10 and CIFAR-100 datasets. The values indicate test accuracy.

Method	CIFAR-10	CIFAR-100
CE	94.35	77.60
GCE	94.00	77.65
GJS	94.78	79.27
NRD (ours)	**95.05**	**79.61**

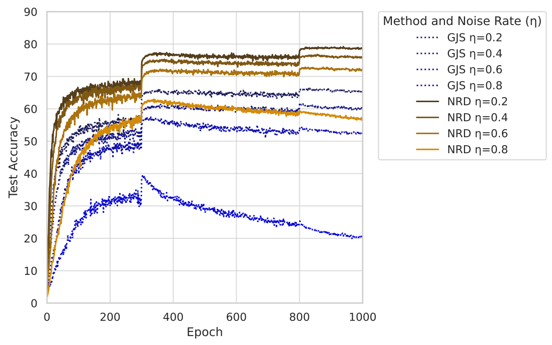
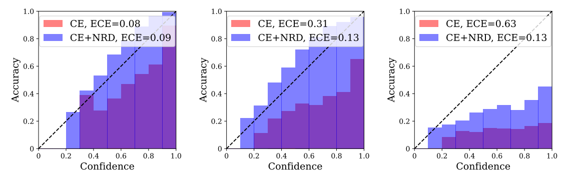


Figure 4: Comparison of overfitting behavior in consistency regularization (GJS [6]). Enforcing consistency does not fully prevent overfitting because the model memorizes the noisy labels after an extended number of epochs. In contrast, our method (NRD) effectively prevents memorization. NoisyCIFAR-100 dataset is used.

a type of noisy supervision signal. We show that applying NRD can regularize and improve the performance of the model.

Comparison to consistency regularization Consistency regularization used in GJS is a powerful technique for noise-robustness. While it is similar to NRD, however, it does not directly prevent memorization of noisy labels. Figure 4 shows that GJS suffers from overfitting when trained for an extended number of steps. This is shown by test accuracy decreasing after reaching a peak at an early epoch. In contrast, NRD significantly mitigates overfitting. Notably, in 80% noise rate setting, we improve GJS by 36%p. The key contributing factor is that our method uses stop-gradient which directly prevents the model from memorizing the views generated by the asymmetric augmentation policy.



(a) $\eta = 0.2$ (b) $\eta = 0.5$ (c) $\eta = 0.8$

Figure 5: Comparison of CE baseline and NRD models trained on NoisyCIFAR-10 with symmetric label noise, where η denotes the noise rate. Expected calibration error (ECE) is measured. NRD training consistently improves calibration across a range of noise rates.

Confidence calibration In Figure 5, we additionally evaluated the calibration performance. We observed that the regularization effect from NRD also improves calibration of the model. Our method shows consistent calibration performance across all noise rates, which aligns with the performance of our method.

6. Conclusion

Our work proposes Noise-Robust Distillation (NRD) which is a simple regularization objective that is designed to improve a wide range of noisy supervision problems in training. We motivate our method based on the novel formulation of Neural Vicinal Risk (NVR) minimization, which focuses on leveraging deep neural networks to improve empirical risk minimization under noisy supervision scenarios. A key insight of our work is the inherent capacity of deep neural networks to detect and correct mislabeled examples based on vicinal distribution, a feature we exploited to improve model predictions and calibration. We have validated our method on several noisy label learning benchmarks. The results show clear improvements in performance compared to the baselines under noisy supervision. These findings suggest that NRD offers an effective strategy for handling noisy supervision, leading to enhanced generalization performance of classification models.

Acknowledgement This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00240379).

References

[1] L. Schmarje, V. Grossmann, C. Zelenka, S. Dippel, R. Kiko, M. Oszust, M. Pastell, J. Stracke, A. Valros, N. Volkmann, et al., Is one annotation enough?-a data-centric image classification benchmark for noisy and ambiguous label estimation, in: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.

[2] S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, S. Chun, Re-labeling imagenet: from single to multi-labels, from global to localized labels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2340–2350.

[3] C. Lehman, Visualizing softmax, 2019. URL: https://charlielehman.github.io/post/visualizing-tempscaling/.

[4] H. Song, M. Kim, D. Park, Y. Shin, J.-G. Lee, Learning from noisy labels with deep neural networks: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).

[5] A. Ghosh, H. Kumar, P. S. Sastry, Robust loss functions under label noise for deep neural networks, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[6] E. Englesson, H. Azizpour, Generalized jensen-shannon divergence loss for learning with noisy labels, Advances in Neural Information Processing Systems 34 (2021) 30284–30297.

[7] Y. Kim, J. Yim, J. Yun, J. Kim, Nlnl: Negative learning for noisy labels, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[8] Y. Kim, J. Yun, H. Shon, J. Kim, Joint negative and positive learning for noisy labels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 9442–9451.

[9] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, S. Lacoste-Julien, A closer look at memorization in deep networks, in: Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 233–242.

[10] S. Liu, J. Niles-Weed, N. Razavian, C. Fernandez-Granda, Early-learning regularization prevents memorization of noisy labels, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 20331–20342.

[11] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, T. Liu, Understanding and improving early stopping for learning with noisy labels, in: Advances in Neural Information Processing Systems, volume 34, 2021, pp. 24392–24403.

[12] S. Liu, Z. Zhu, Q. Qu, C. You, Robust training under label noise by over-parameterization, in: Proceedings of the 39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 14153–14172.

[13] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, M. Sugiyama, Co-teaching: Robust training of deep neural networks with extremely noisy labels, in: Advances in Neural Information Processing Systems, volume 31, 2018.

[14] J. Li, R. Socher, S. C. Hoi, Dividemix: Learning with noisy labels as semi-supervised learning, in: International Conference on Learning Representations, 2020.

[15] J. Li, G. Li, F. Liu, Y. Yu, Neighborhood collective estimation for noisy label identification and correction, in: European Conference on Computer Vision, 2022.

[16] G. Zhao, G. Li, Y. Qin, F. Liu, Y. Yu, Centrality and consistency: Two-stage clean samples identification for learning with instance-dependent noisy labels, in: European Conference on Computer Vision, volume 13685, 2022, pp. 21–37.

[17] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, in: International Conference on Learning Representations, 2017. URL: https://openreview.net/forum?id=BJ6oOfqge.

[18] A. Tarvainen, H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, in: Advances in Neural Information Processing Systems, volume 30, 2017.

[19] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, C.-L. Li, Fixmatch: Simplifying semi-supervised learning with consistency and confidence, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 596–608.

[20] T. Miyato, A. M. Dai, I. Goodfellow, Adversarial training methods for semi-supervised text classification, in: International Conference on Learning Representations, 2017.

[21] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C. A. Raffel, Mixmatch: A holistic approach to semi-supervised learning, in: Advances in Neural Information Processing Systems, volume 32, 2019.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018.

[23] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, in: International conference on machine learning, PMLR, 2017, pp. 1321–1330.

[24] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, S. Michalak, On mixup training: Improved calibration and predictive uncertainty for deep neural networks, in: Advances in Neural Information Processing Systems, volume 32, 2019.

[25] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015. arXiv:1503.02531.

[26] O. Chapelle, J. Weston, L. Bottou, V. Vapnik, Vicinal risk minimization, Advances in neural information processing systems 13 (2000).

[27] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, Y. Yoo, Cutmix: Regularization strategy to train strong classifiers with localizable features, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.

[28] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q. V. Le, Autoaugment: Learning augmentation strategies from data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[29] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, 2009.

[30] G. Patrini, A. Rozza, A. Menon, R. Nock, L. Qu, Making neural networks robust to label noise: a loss correction approach, stat 1050 (2016) 13.

[31] W. Li, L. Wang, W. Li, E. Agustsson, L. Van Gool, Webvision database: Visual learning and understanding from web data, arXiv preprint arXiv:1708.02862 (2017).

[32] P. Chen, B. B. Liao, G. Chen, S. Zhang, Understanding and utilizing deep neural networks trained with noisy labels, in: International Conference on Machine Learning, PMLR, 2019, pp. 1062–1070.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.

[34] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, A. Rabinovich, Training deep neural networks on noisy labels with bootstrapping, arXiv preprint arXiv:1412.6596 (2014).

[35] M. Lukasik, S. Bhojanapalli, A. Menon, S. Kumar, Does label smoothing mitigate label noise?, in: International Conference on Machine Learning, PMLR, 2020, pp. 6448–6458.

[36] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, J. Bailey, Symmetric cross entropy for robust learning with noisy labels, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[37] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: Advances in Neural Information Processing Systems, volume 31, 2018.

[38] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, J. Bailey, Normalized loss functions for deep learning with noisy labels, in: Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 6543–6553.

[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016.

[40] E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, Randaugment: Practical automated data augmentation with a reduced search space. 2020 ieee, in: CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 3008–3017.

[41] T. DeVries, G. W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv preprint arXiv:1708.04552 (2017).

[42] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, AugMix: A simple data processing method to improve robustness and uncertainty, Proceedings of the International Conference on Learning Representations (ICLR) (2020).

[43] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, Springer, 2016, pp. 630–645.

[44] T. maintainers, contributors, Torchvision: Pytorch's computer vision library, https://github.com/pytorch/vision, 2016.

A. Detailed hyperparameter configurations

A.1. CIFAR-10/100 benchmarks

General training details For the network architecture, we use PreActResNet-34 [43]. For training, we use SGD optimizer with momentum 0.9, a batch size of 128, and train for 400 epochs. The learning rate is reduced by 1/10 at 50% and 75% of the training iterations.

Augmentation policy For data augmentation, we use RandAugment [40] with $N = 1, M = 3$ followed by random crop (size 32 and 4-pixel padding), random horizontal flip and Cutout [41] with length 5.

Hyperparameters See Table 5 for the details. For the baselines, we follow the same hyperparameter configurations used by [6]. 40% noise rate setting was used to find the best learning rates and weight decay rates. For the learning rates and weight decay rates for NRD, we used the same configurations as GJS. For the tuning of hyperparameters $\{\pi_1, \pi_2, \pi_3\}$ in the NRD loss, we fixed $\pi_1 = \pi_3$ so that the targets $\mathbf{y}$ and $\mathbf{y_t}$ have equal weight. We tuned $\pi_2 \in \{0.1, 0.2, ..., 0.9\}$. For the moving average decay rate, we used $\beta = 0.99$ for

Table 5
Hyperparameters for CIFAR-10/100. The hyperparameters for the baseline methods are identical to [6]. For the learning rate and weight decay, each entry denotes [LR, WD]. For the method-specific hyperparameters, each entry denotes its hyperparameters: BS (β), LS (ϵ), SCE ($[\alpha, \beta]$), GCE (q), NCE+RCE ($[\alpha, \beta]$), JS (π_1), GJS (π_1), NRD ($[\pi_1, \pi_2, \pi_3]$).

Dataset	Method	Learning Rate & Weight Decay		Method-specific Hyperparameters						
		Sym Noise	Asym Noise	No Noise	Sym Noise				Asym Noise	
		20-80%	20-40%	0%	20%	40%	60%	80%	20%	40%
CIFAR-10	CE	[0.05, 1e-3]	[0.1, 1e-3]	-	-	-	-	-	-	-
	BS	[0.1, 1e-3]	[0.1, 1e-3]	0.5	0.5	0.7	0.7	0.9	0.7	0.5
	LS	[0.1, 5e-4]	[0.1, 1e-3]	0.1	0.5	0.9	0.7	0.1	0.1	0.1
	SCE	[0.01, 5e-4]	[0.05, 1e-3]	[0.2, 0.1]	[0.05, 0.1]	[0.1, 0.1]	[0.2, 1.0]	[0.1,1.0]	[0.1, 0.1]	[0.2, 1.0]
	GCE	[0.01, 5e-4]	[0.1, 1e-3]	0.5	0.7	0.7	0.7	0.9	0.1	0.1
	NCE+RCE	[0.005, 1e-3]	[0.05, 1e-4]	[10, 0.1]	[10, 0.1]	[10, 0.1]	[1.0, 0.1]	[10,1.0]	[10, 0.1]	[1.0, 0.1]
	JS	[0.01, 5e-4]	[0.1, 1e-3]	0.1	0.7	0.7	0.9	0.9	0.3	0.3
	GJS	[0.1, 5e-4]	[0.1, 1e-3]	0.5	0.3	0.9	0.1	0.1	0.3	0.3
	NRD	[0.1, 5e-4]	[0.1, 1e-3]	[0.2, 0.6, 0.2]	[0.2, 0.6, 0.2]	[0.2, 0.6, 0.2]	[0.25, 0.5, 0.25]	[0.25, 0.5, 0.25]	[0.1, 0.8, 0.1]	[0.15, 0.7, 0.15]
CIFAR-100	CE	[0.4, 1e-4]	[0.2, 1e-4]	-	-	-	-	-	-	-
	BS	[0.4, 1e-4]	[0.4, 1e-4]	0.7	0.5	0.5	0.5	0.9	0.3	0.3
	LS	[0.2, 5e-5]	[0.4, 1e-4]	0.1	0.7	0.7	0.7	0.9	0.5	0.7
	SCE	[0.2, 1e-4]	[0.4, 5e-5]	[0.1, 0.1]	[0.1, 0.1]	[0.1, 0.1]	[0.1, 1.0]	[0.1,0.1]	[0.1, 1.0]	[0.1, 1.0]
	GCE	[0.4, 1e-5]	[0.2, 1e-4]	0.5	0.5	0.5	0.7	0.7	0.7	0.7
	NCE+RCE	[0.2, 5e-5]	[0.2, 5e-5]	[20, 0.1]	[20, 0.1]	[20, 0.1]	[20, 0.1]	[20,0.1]	[20, 0.1]	[10, 0.1]
	JS	[0.2, 1e-4]	[0.1, 1e-4]	0.1	0.1	0.3	0.5	0.3	0.5	0.5
	GJS	[0.2, 5e-5]	[0.4, 1e-4]	0.3	0.3	0.5	0.9	0.1	0.5	0.1
	NRD	[0.2, 5e-5]	[0.4, 1e-4]	[0.2, 0.6, 0.2]	[0.2, 0.6, 0.2]	[0.2, 0.6, 0.2]	[0.2, 0.6, 0.2]	[0.15, 0.7, 0.15]	[0.25, 0.5, 0.25]	[0.4, 0.2, 0.4]

all experiments.

A.2. WebVision benchmark

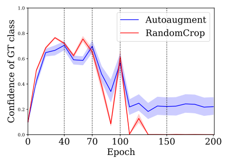
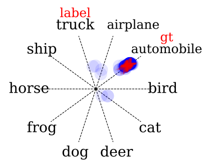
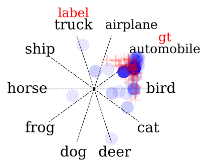
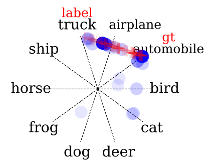
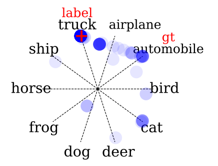
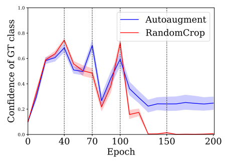
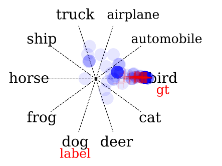
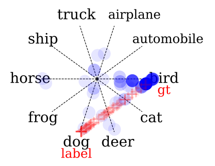
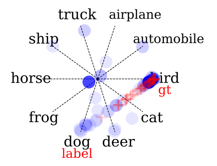
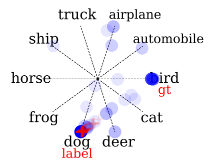
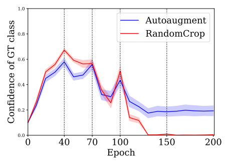
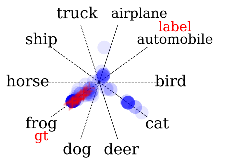
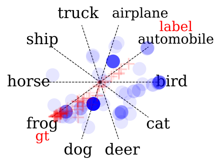
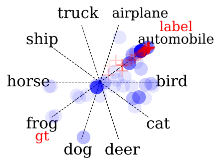
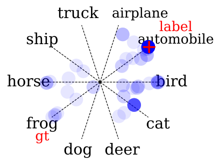
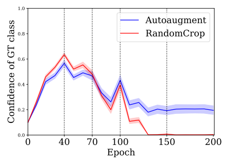
General training details For the network architecture, we use ResNet-50 with random initialization. For training, we use SGD optimizer with momentum 0.9, a batch size of 64, and train for 300 epochs. The initial learning rate was set to 0.1 and reduced by 1/10 after the 100-th and 200-th epoch.
Augmentation policy For data augmentation, we use random resized crop with size 224, random horizontal flip, and color jitter. We used the color jitter implementation from TorchVision [44] with brightness=0.4, contrast=0.4, saturation=0.4, hue=0.2. For the NRD teacher augmentation, we use AugMix [42] followed by random resize crop with size 224 and random horizontal flip.
Hyperparameters For the hyperparameters $\{\pi_1, \pi_2, \pi_3\}$ in the NRD loss, we used $\pi_1 = \pi_3 = 0.1$ and $\pi_2 = 0.8$. The moving average decay rate was set to $\beta = 0.99$.

B. Training dynamics visualization of perturbed inputs

In this section, we provide the visualized trajectory of the model prediction of the perturbed inputs throughout training. (See Table 6) These are the same plots presented in Figure 2a, albeit on different mid-training epochs. The model is trained using a standard training scheme with the cross-entropy loss on the NoisyCIFAR-10-symm-40% dataset. We observe that a significant portion of the predictions perturbed using augmentation unseen at training (AutoAugment) gradually settles to the ground truth class, whereas the predictions perturbed using the same augmentation policy used at training (RandomCrop) eventually converge to the noisy target class. The result shows that predictions from the perturbation identical to the training augmentation (red markers) are non-noise-robust distillation targets, whereas the predictions from the unseen perturbation (blue markers) are noise-robust distillation targets.

Table 6

Visualization of the model prediction over training. We randomly selected four distinct noisy samples from the training dataset, which corresponds to the four rows. The model is trained using RandomCrop and tested using RandomCrop-perturbed inputs (red) and AutoAugment-perturbed inputs (blue). Leftmost column shows the predicted confidence of the perturbed inputs with respect to the ground-truth classes. The figures on the right hand-side visualizes the softmax vectors projected onto a decagonal surface, which are analogous to Figure 2a. At the early phase of the training, both red and blue markers predict the ground-truth class. However, as the training progresses and the model overfits to the noisy labels, the red markers predict the target label, whereas a significant portion of the blue markers predicts the ground-truth markers. This shows that unseen perturbation to the input can produce noise-robust learning signal for training.

Confidence of GT class	Epoch 40	Epoch 70	Epoch 100	Epoch 150
 Confidence of GT class Autoaugment RandomCrop Epoch				
 Confidence of GT class Autoaugment RandomCrop Epoch				
 Confidence of GT class Autoaugment RandomCrop Epoch				
 Confidence of GT class Autoaugment RandomCrop Epoch