

Improving Unsupervised Image Clustering With Robust Learning

Sungwon Park^{*1,2} Sungwon Han^{*1,2} Sundong Kim² Danu Kim^{1,2}

Sungkyu Park² Seunghoon Hong¹ Meeyoung Cha^{2,1}

¹School of Computing, KAIST ²Data Science Group, Institute for Basic Science

Abstract

Unsupervised image clustering methods often introduce alternative objectives to indirectly train the model and are subject to faulty predictions and overconfident results. To overcome these challenges, the current research proposes an innovative model RUC that is inspired by robust learning. RUC’s novelty is at utilizing pseudo-labels of existing image clustering models as a noisy dataset that may include misclassified samples. Its retraining process can revise misaligned knowledge and alleviate the overconfidence problem in predictions. The model’s flexible structure makes it possible to be used as an add-on module to other clustering methods and helps them achieve better performance on multiple datasets. Extensive experiments show that the proposed model can adjust the model confidence with better calibration and gain additional robustness against adversarial noise.

1. Introduction

Unsupervised clustering is a core task in computer vision that aims to identify each image’s class membership without using any labels. Here, a class represents the group membership of images that share similar visual characteristics. Many studies have proposed deep learning-based algorithms that utilize distance in a feature space as the similarity metric to assign data points into classes [11, 44].

Training without ground-truth guidance, however, is prone to finding trivial solutions that are learned from low-level visual traits like colors and textures [22]. Several studies have introduced innovative ways to guide the model’s training indirectly by setting alternative objectives. For example, Hu *et al.* [20] proposed to maximize the mutual information between input and its hidden representations, and Ji *et al.* [22] proposed to learn invariant features against data augmentation. Entropy-based balancing has often been adopted to prevent degenerate solutions [17, 22, 42].

Nevertheless, these alternative objectives are bound to

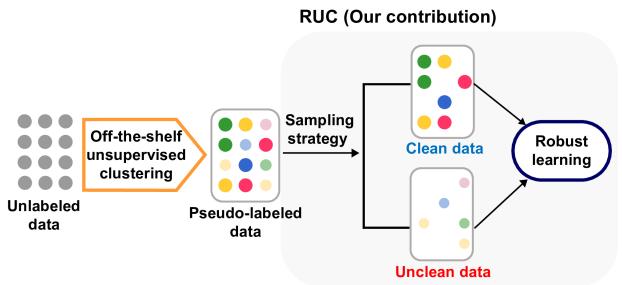


Figure 1: Illustration for this work’s basic concept: robust learning is used to separate clean data from unclean data using pseudo-labels from off-the-shelf unsupervised clustering algorithm.

producing overconfident results, i.e., low-entropy predictions, due to the dense grouping among clusters. When uncertain samples are added to a wrong cluster at an early stage of training, the model gradually becomes overconfident in its later predictions as the noise from misclassification accumulates and degrades the overall performance.

This paper introduces a novel robust learning training method, RUC (Robust learning for Unsupervised Clustering), that runs in conjunction with existing clustering models to alleviate the noise discussed above. Utilizing and treating the existing clustering model’s results as a noisy dataset that may include wrong labels, RUC updates the model’s misaligned knowledge. Bringing insights from the literature, we filter out unclean samples and apply loss correction as in Fig. 1. This process is assisted by label smoothing and co-training to reduce any wrong gradient signals from unclean labels. This retraining process with revised pseudo-labels further regularizes the model and prevents overconfident results.

RUC comprises two key components: (1) extracting clean samples and (2) retraining with the refined dataset. We propose confidence-based, metric-based, and hybrid strategies to filter out misclassified pseudo-labels. The first strategy considers samples of high prediction confidence from the original clustering model as a clean set; it filters out low confidence samples. This strategy relies on the

*Equal contribution to this work.

model’s calibration performance. The second strategy utilizes similarity metrics from unsupervised embedding models to detect clean samples with non-parametric classifiers by checking whether the given instance shares the same labels with top k -nearest samples. The third strategy combines the above two and selects samples that are credible according to both strategies.

The next step is to retrain the clustering model with the sampled dataset. We use MixMatch [5], a semi-supervised learning technique; which uses clean samples as labeled data and unclean samples as unlabeled data. We then adopt label smoothing to leverage strong denoising effects on the label noise [29] and block learning from overconfident samples [22, 42]. Finally, a co-training architecture with two networks is used to mitigate noise accumulation from the unclean samples during training and increase performance.

We evaluate RUC with rigorous experiments on datasets, including CIFAR-10, CIFAR-20, STL-10, and ImageNet-50. Combining RUC to an existing clustering model outperforms the state-of-the-art results with the accuracy of 90.3% in CIFAR-10, 54.3% in CIFAR-20, 86.7% in STL-10, and 78.5% in ImageNet-50 dataset. RUC also enhances the baseline model to be robust against adversarial noise. Our contributions are as follows:

- The proposed algorithm RUC aids existing unsupervised clustering models via retraining and avoiding overconfident predictions.
- The unique retraining process of RUC helps existing models boost performance. It achieves a 5.3pp increase for the STL-10 dataset when added to the state-of-the-art model (81.4% to 86.7%).
- The ablation study shows every component in RUC is critical, including the three proposed strategies (i.e., confidence-based, metric-based, and hybrid) that excel in extracting clean samples from noisy pseudo-labels.
- The proposed training process is robust against adversarial noise and can adjust the model confidence with better calibrations.

Implementation details of the model and codes are available at <https://github.com/deu30303/RUC>.

2. Related Work

2.1. Unsupervised Image Clustering

The main objective of clustering is to group the data points into distinct classes of similar traits [21]. Most real-world problems deal with high dimensional data (e.g., images), and thereby, setting a concrete notion of similarity while extracting low-dimensional features becomes key components for setting appropriate standards for grouping [49]. Likewise, unsupervised clustering is a line of research aiming to tackle both dimensionality reduction and boundary identification over the learned similarity met-

ric [17]. Existing research can be categorized into *sequential*, *joint*, and *multi-step refinement approach*.

Sequential approach. Sequential approach extracts features, then sequentially applies the conventional distance or density-based clustering algorithm for class assignments. For example, Ding *et al.* [11] use principal component analysis to extract low-dimensional features and then apply k -means clustering to assign classes. For feature extraction, autoencoder structures are often used to extract latent features before grouping, types of autoencoder include stacked [44], boolean [3], or variational autoencoder [24]. However, these models tend to produce features with little separation among clusters due to the lack of knowledge on subsequent assignment processes.

Joint approach. The joint approach’s characteristic is to use an end-to-end pipeline that concurrently performs feature extraction and class assignment. An example is Yang *et al.* [51], which adopt the concept of clustering loss to guarantee enough separations among clusters. End-to-end CNN pipelines are used widely to iteratively identify clusters while refining extracted features [6, 8, 49]. Recent studies have shown that a mutual information-based objective is an effective measure to improve classification accuracy [20, 22]. Nonetheless, those models still bear the problem of generating unintended solutions that depend on trivial low-level features from random initialization [17].

Multi-step refinement approach. To mitigate the unintended trivial solutions, recent approaches leverage the power of unsupervised embedding learning models to provide better initialization for downstream clustering tasks [9, 48, 50]. These methods generate feature representations to gather data points with similar visual traits and push away the rest in an embedding space. With the initialization, clustering results are elaborated in a refinement step, bringing significant gain in its class assignment quality [17, 42]. In particular, SCAN [42] first obtains high-level feature representations by feature similarity then clusters those representations by nearest neighbors, and this model has shown remarkable performance on unsupervised clustering.

Add-on modules to improve unsupervised clustering.

The proposed retraining process with sample selection strategy improves off-the-shelf unsupervised clustering algorithms (e.g., sequential, joint, multi-step refinement) by acting as an add-on module. Our module’s main objective is to revise the misaligned knowledge of trained clustering models via label cleansing and retraining with the refined labels. This method has not been well investigated before but has begun to be proposed recently. Gupta *et al.* [14] show that semi-supervised retraining improves unsupervised clustering. They draw a graph where data samples are nodes, and the confidence from ensemble models

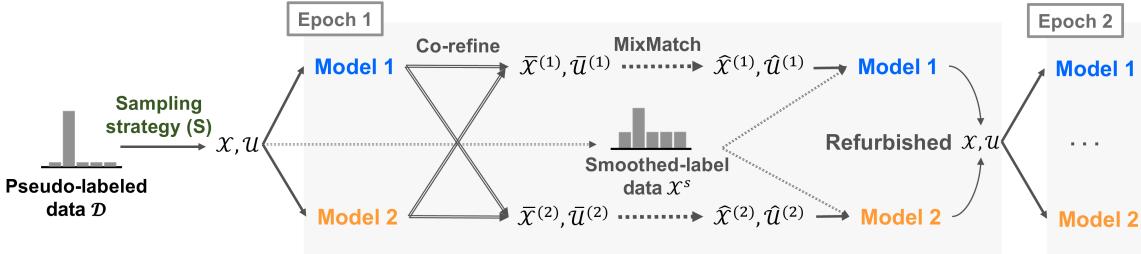


Figure 2: Illustration of the proposed model. Our model first selects clean samples as a labeled dataset \mathcal{X} and considers the remaining samples as an unlabeled dataset \mathcal{U} (Section 3.1). Next, we train two networks $f_{\theta^{(1)}}$ and $f_{\theta^{(2)}}$ in a semi-supervised fashion (Section 3.2). In each epoch, the MixMatch algorithm, along with co-training and label smoothing, is applied for training. The clean set is updated via co-refurbishing for the next epoch.

between the samples is an edge. Then, a dense sub-graph is considered as a clean set.

The main difference between Gupta *et al.* and ours is in how we treat the pseudo-labels obtained by the clustering. Gupta *et al.* treats the pseudo-label as a ground-truth for semi-supervised learning, which produces sub-optimal result if the pseudo-label is noisy (i.e., *memorization*). In contrast, we introduce the robust learning concept of label smoothing and co-training to mitigate the memorization of noisy samples, which leads to substantial improvements in the calibration and clustering performance.

2.2. Robust Learning With Label Noise

A widely used setting for robust learning is where an adversary has deliberately corrupted the labels, which otherwise arise from some clean distribution [33, 40]. According to the literature, deep networks easily overfit to the label noise during training and get a low generalization power [28]. In this light, models that prevent overfitting in a noise label environment have been studied.

Loss correction. The first representative line of work is a loss correction, which relabels unclean samples explicitly or implicitly. For example, Patrini *et al.* [35] estimate the label transition probability matrix to correct the loss and retrain the model. To estimate the transition matrix more accurately, the gold loss correction approach [19] is proposed to utilize trusted labels as additional information.

Loss reweighting. The next line of work is loss reweighting, which aims to give a smaller weight to the loss of unclean samples so that model can reduce the negative effect of label noise during training. One work computes the importance as an approximated ratio of two data distributions; clean and unclean [46]. On the other hand, the active bias approach [7] calculates the inconsistency of predictions during training and assigns a weight to penalize unclean data.

Sample selection. Relabeling the misclassified samples may cause a false correction. In this context, recent works

introduce a sample selection approach that filters out misclassified samples and only selects clean data for training [30, 32]. Notably, the small loss trick, which regards the sample with small training loss as clean, effectively separates true- and false-labeled data points [2, 23, 28]. Also, recent studies suggest diverse ways to lead additional performance by maintaining two networks to avoid accumulating sampling bias [16, 54], adopting refurbishment of false-labeled samples [39], or using a semi-supervised approach to utilize false-labeled sample maximally [27]. Our model advances some of these sample selection approaches to filter out unclean samples out of clustering results and utilize clean samples only during retraining.

3. Method

RUC is an add-on method that can be used in conjunction with the existing unsupervised clustering methods to refine mispredictions. Its key idea is at utilizing the initial clustering results as *noisy* pseudo-labels and learning to refine them with a mild clustering assumption [41] and techniques from the robust learning [27, 29].

Figure 2 and Algorithm 1 illustrate the overall pipeline of the proposed algorithm. Given the initial pseudo-labels, we first divide the training data into the two disjoint sets: clean and unclean (Section 3.1). Then treating these sets each as labeled and unlabeled data, we train a classifier in a semi-supervised manner while refurbishing the labeled and unlabeled data (Section 3.2). We guide the semi-supervised class assignment with robust learning techniques, such as co-training and label smoothing, to account for inherent label noises. These techniques are useful in handling label noises and calibrating the model’s prediction score. Below we describe the model in details.

3.1. Extracting Clean Samples

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ denote the training data, where \mathbf{x}_i is an image and $\mathbf{y}_i = g_\phi(\mathbf{x}_i)$ is a pseudo-label from

Algorithm 1 Robust learning algorithm using unsupervised clustering pseudo-label.

Input: Sampling strategy \mathcal{S} , training dataset with pseudo-labels \mathcal{D} , two networks $f_{\theta(1)}, f_{\theta(2)}$, sharpening temperature T , number of augmentations M , unsupervised loss weight λ_U , refurbish threshold τ_2 , weak- and strong augmentation ϕ_a, ϕ_A

```

/* Divide the dataset  $\mathcal{D}$  into clean and noisy set using a sampling strategy */
 $\mathcal{X}, \mathcal{U} = \mathcal{S}(\mathcal{D})$  (i.e.  $\mathcal{X} = \{(\mathbf{x}_b, \mathbf{y}_b) : b \in (1, \dots, B)\}$ ,  $\mathcal{U} = \{\mathbf{u}_b : b \in (1, \dots, B)\}$ )
for  $k \in \{1, 2\}$  do
    /* Train the two networks  $f_{\theta(1)}$  and  $f_{\theta(2)}$  iteratively */
    for  $b \in \{1, \dots, B\}$  do
         $\tilde{\mathbf{y}}_b = (1 - \epsilon) \cdot \mathbf{y}_b + \frac{\epsilon}{(C-1)} \cdot (\mathbf{1} - \mathbf{y}_b)$  // Inject uniform noise into all classes (label smoothing)
        for  $m \in \{1, \dots, M\}$  do
             $|\mathbf{x}_{b,m}, \mathbf{u}_{b,m} = \phi_a(\mathbf{x}_b), \phi_a(\mathbf{u}_b)$  // Perform weak augmentation  $M$  times
        end
         $\bar{\mathbf{y}}_b = (1 - w_b^{(c)}) \cdot \mathbf{y}_b + w_b^{(c)} \cdot f_{\theta(c)}(\mathbf{x}_b)$  // Refine the labels (( $c$ ) denotes the counter network)
         $\bar{\mathbf{y}}_b = \text{Sharpen}(\bar{\mathbf{y}}_b, T)$  // Apply sharpening to the refined label
         $\bar{\mathbf{q}}_b = \frac{1}{2M} \sum_m (f_{\theta(1)}(\mathbf{u}_{b,m}) + f_{\theta(2)}(\mathbf{u}_{b,m}))$  // Ensemble both networks' predictions to guess labels
         $\bar{\mathbf{q}}_b = \text{Sharpen}(\bar{\mathbf{q}}_b, T)$  // Apply sharpening to the guessed labels
    end
     $\mathcal{X}^s = \{(\phi_A(\mathbf{x}_b), \tilde{\mathbf{y}}_b); b \in \{1, \dots, B\}\}$  // Strongly augmented samples with smoothed labels
     $\bar{\mathcal{X}}^{(k)} = \{(\mathbf{x}_b, \bar{\mathbf{y}}_b); b \in \{1, \dots, B\}\}$  // Co-refined labeled samples
     $\bar{\mathcal{U}}^{(k)} = \{(\mathbf{u}_b, \bar{\mathbf{q}}_b); b \in \{1, \dots, B\}\}$  // Co-refined unlabeled samples
     $\hat{\mathcal{X}}^{(k)}, \hat{\mathcal{U}}^{(k)} = \text{MixMatch}(\bar{\mathcal{X}}^{(k)}, \bar{\mathcal{U}}^{(k)})$  // Apply MixMatch
     $\mathcal{L}_t = \mathcal{L}_{\mathcal{X}^s} + \mathcal{L}_{\hat{\mathcal{X}}} + \lambda_U \mathcal{L}_{\hat{\mathcal{U}}}$  // Calculate the total loss
     $\mathcal{X} \leftarrow \mathcal{X} \cup \text{Co-Refurbish}(\mathcal{U}, f_{\theta(k)}, \tau_2)$  // Refurbish noisy samples to clean samples (Eq. (18), (19))
     $\theta^{(k)} \leftarrow \text{SGD}(L_t, \theta^{(k)})$  // Update network parameters
end

```

an unsupervised classifier g_ϕ . The model first divides the pseudo-labels into two disjoint sets as $\mathcal{D} = \mathcal{X} \cup \mathcal{U}$ with a specified sampling strategy. We consider \mathcal{X} as clean, whose pseudo-labels are moderately credible and thus can be used as a labeled dataset $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$ for refinement. In contrast, we consider \mathcal{U} as unclean, whose labels we discard $\mathbf{u} \in \mathcal{U}$. Designing an accurate sampling strategy is not straightforward, as there is no ground-truth to validate the pseudo-labels directly. Inspired by robust learning's clean set selection strategy, we explore three different approaches: (1) confidence-based, (2) metric-based, and (3) hybrid.

Confidence-based strategy. This approach selects clean samples based on the confidence score of the unsupervised classifier. Given a training sample $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, we consider the pseudo-label \mathbf{y} is credible if $\max(\mathbf{y}) > \tau_1$, and add it to the clean set \mathcal{X} . Otherwise, it is assigned to \mathcal{U} . This is motivated by the observation that the unsupervised classifier tends to generate overconfident predictions; thus, we trust only the most typical examples from each class while ignoring the rest. The threshold τ_1 is set substantially high to eliminate as many uncertain samples as possible.

Metric-based strategy. The limitation of the above approach is that the selection strategy still entirely depends on the unsupervised classifier. The metric-based approach leverages an additional embedding network h_ψ learned in an unsupervised manner (e.g., SimCLR [9]) and measures

the credibility of the pseudo-label based on how well it coincides to the classification results using h_ψ . For each $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, we compute its embedding $h_\psi(\mathbf{x})$, and apply the non-parametric classifier based on k -Nearest Neighbor (k -NN) by $\mathbf{y}' = k\text{-NN}(h_\psi(\mathbf{x}))$. We consider that the pseudo-label is credible if $\text{argmax}(\mathbf{y}) = \text{argmax}(\mathbf{y}')$ and add it to the clean set \mathcal{X} . Otherwise, it is assigned to the unclean set \mathcal{U} .

Hybrid strategy. This approach will add a sample to the clean set \mathcal{X} only if it is considered credible by both the confidence-based and metric-based strategies. All other samples are added to \mathcal{U} .

3.2. Retraining via Robust Learning

Given the clean set \mathcal{X} and the unclean set \mathcal{U} , our next step aims to train the refined classifier f_θ that revises incorrect predictions of the initial unsupervised classifier.

Vanilla semi-supervised learning. A naive baseline is to consider \mathcal{X} as labeled data and \mathcal{U} as unlabeled data each to train a classifier f_θ using semi-supervised learning techniques. We utilize MixMatch [5] as such a baseline, which is a semi-supervised algorithm that estimates low-entropy mixed labels from unlabeled examples using MixUp augmentation [56].¹ For unsupervised clustering, MixUp can

¹Note that our method is not dependent on the particular choice of the semi-supervised learning method and can incorporate the others.

bring additional resistance against noisy labels since a large amount of extra virtual examples from MixUp interpolation makes memorization hard to achieve [27, 56]. Specifically, given a two paired data $(\mathbf{x}_1, \mathbf{y}_1)$ and $(\mathbf{x}_2, \mathbf{y}_2)$ sampled from either labeled or unlabeled data, it augments the data using the following operations.

$$\lambda \sim Beta(\alpha, \alpha) \quad (1)$$

$$\lambda' = max(\lambda, 1 - \lambda) \quad (2)$$

$$\mathbf{x}' = \lambda' \mathbf{x}_1 + (1 - \lambda') \mathbf{x}_2 \quad (3)$$

$$\mathbf{y}' = \lambda' \mathbf{y}_1 + (1 - \lambda') \mathbf{y}_2. \quad (4)$$

In the case of unlabeled data $\mathbf{u} \in \mathcal{U}$, MixMatch is employed such that a surrogate label $\mathbf{y} = \mathbf{q}$ is obtained by averaging the model's predictions over multiple augmentations after sharpening [5]. Later, we will show that using the labels \mathbf{y} and \mathbf{q} directly in semi-supervised learning leads to a suboptimal solution and discuss how to improve its robustness.

For $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ after MixMatch (Eq. (5)), a vanilla semi-supervised learning model trains with two separate losses: the cross-entropy loss for the labeled set $\hat{\mathcal{X}}$ (Eq. (6)), and the consistency regularization for the unlabeled set $\hat{\mathcal{U}}$ (Eq. (7)). $H(p, q)$ denotes the cross-entropy between p and q .

$$\hat{\mathcal{X}}, \hat{\mathcal{U}} = \text{MixMatch}(\mathcal{X}, \mathcal{U}) \quad (5)$$

$$\mathcal{L}_{\hat{\mathcal{X}}} = \frac{1}{|\hat{\mathcal{X}}|} \sum_{\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \hat{\mathcal{X}}} H(\hat{\mathbf{y}}, f_\theta(\hat{\mathbf{x}})) \quad (6)$$

$$\mathcal{L}_{\hat{\mathcal{U}}} = \frac{1}{|\hat{\mathcal{U}}|} \sum_{\hat{\mathbf{u}}, \hat{\mathbf{q}} \in \hat{\mathcal{U}}} \|\hat{\mathbf{q}} - f_\theta(\hat{\mathbf{u}})\|_2^2 \quad (7)$$

Label smoothing. To regularize our model from being overconfident to noisy predictions, we apply label smoothing along with vanilla semi-supervised learning. Label smoothing prescribes soft labels by adding uniform noise and improves the calibration in predictions [29]. Given a labeled sample with its corresponding label $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$, we inject uniform noise into all classes as follows:

$$\tilde{\mathbf{y}} = (1 - \epsilon) \cdot \mathbf{y} + \frac{\epsilon}{(C - 1)} \cdot (\mathbf{1} - \mathbf{y}) \quad (8)$$

where C is the number of class and $\epsilon \sim \text{Uniform}(0, 1)$ is the noise. We compute cross-entropy using the soft label $\tilde{\mathbf{y}}$ and the predicted label of the strongly augmented sample $\phi_A(\mathbf{x})$ via RandAugment [10]. We find that strong augmentations minimize the memorization from noise samples.

$$\mathcal{L}_{\mathcal{X}^s} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}, \tilde{\mathbf{y}} \in \mathcal{X}} H(\tilde{\mathbf{y}}, f_\theta(\phi_A(\mathbf{x}))) \quad (9)$$

Our final objective for training can be written as:

$$\mathcal{L}(\theta; \mathcal{X}, \hat{\mathcal{X}}, \hat{\mathcal{U}}) = \mathcal{L}_{\mathcal{X}^s} + \mathcal{L}_{\hat{\mathcal{X}}} + \lambda_{\mathcal{U}} \mathcal{L}_{\hat{\mathcal{U}}}, \quad (10)$$

where $\lambda_{\mathcal{U}}$ is a hyper-parameter to control the effect of the unsupervised loss in MixMatch.

Co-training. Maintaining a single network for learning has a vulnerability of overfitting to incorrect pseudo-labels since the initial error from the network is transferred back again, and thereby, accumulated [16]. To avoid this fallacy, we additionally introduce a co-training module where the two networks $f_{\theta^{(1)}}, f_{\theta^{(2)}}$ are trained in parallel and exchange their guesses for teaching each other by adding a co-refinement step on top of MixMatch.

Co-refinement is a label refinement process that aims to produce reliable labels by incorporating both networks' predictions. Following the previous literature [27], we apply co-refinement both on the label set \mathcal{X} and the unlabeled set \mathcal{U} for each network. Here, we explain the co-refinement process from the perspective of $f_{\theta^{(1)}}$. For the labeled data point \mathbf{x} , we calculate the linear sum between the original label \mathbf{y} in \mathcal{X} and the prediction from the counter network $f_{\theta^{(2)}}$ (Eq. (11)) and apply sharpening on the result to generate the refined label $\bar{\mathbf{y}}$ (Eq. (12)).

$$\bar{\mathbf{y}} = (1 - w^{(2)}) \cdot \mathbf{y} + w^{(2)} \cdot f_{\theta^{(2)}}(\mathbf{x}) \quad (11)$$

$$\bar{\mathbf{y}} = \text{Sharpen}(\bar{\mathbf{y}}, T), \quad (12)$$

where $w^{(2)}$ is the counter network's confidence value of \mathbf{x} , and T is the sharpening temperature. For the unlabeled set \mathcal{U} , we apply an ensemble of both networks' predictions to guess the pseudo-label $\bar{\mathbf{q}}$ of data sample \mathbf{u} as follows:

$$\bar{\mathbf{q}} = \frac{1}{2M} \sum_m (f_{\theta^{(1)}}(\mathbf{u}_m) + f_{\theta^{(2)}}(\mathbf{u}_m)) \quad (13)$$

$$\bar{\mathbf{q}} = \text{Sharpen}(\bar{\mathbf{q}}, T), \quad (14)$$

where \mathbf{u}_m is m -th weak augmentation of \mathbf{u} .

In place of \mathcal{X} and \mathcal{U} , co-refinement produces the refined dataset $(\mathbf{x}, \bar{\mathbf{y}}) \in \bar{\mathcal{X}}^{(1)}$, and $(\mathbf{u}, \bar{\mathbf{q}}) \in \bar{\mathcal{U}}^{(1)}$ through Eq. (11) to (14). We utilize those datasets as an input for MixMatch, and the model is eventually optimized as follows:

$$\bar{\mathcal{X}}^{(1)}, \bar{\mathcal{U}}^{(1)} = \text{Co-refinement}(\mathcal{X}, \mathcal{U}, \theta^{(1)}, \theta^{(2)}) \quad (15)$$

$$\hat{\mathcal{X}}^{(1)}, \hat{\mathcal{U}}^{(1)} = \text{MixMatch}(\bar{\mathcal{X}}^{(1)}, \bar{\mathcal{U}}^{(1)}) \quad (16)$$

$$\theta^{(1)} \leftarrow \arg \min_{\theta^{(1)}} \mathcal{L}(\theta^{(1)}; \mathcal{X}, \hat{\mathcal{X}}^{(1)}, \hat{\mathcal{U}}^{(1)}), \quad (17)$$

where \mathcal{L} is the loss defined in Eq. (10). This process is also conducted for $f_{\theta^{(2)}}$ in the same manner.

Co-refurbishing. Lastly, we refurbish the noise samples at the end of every epoch to deliver the extra clean samples across the training process. If at least one of the networks' confidence on the given unclean sample $\mathbf{u} \in \mathcal{U}$ is over the threshold τ_2 , the corresponding sample's label is updated with the network's prediction \mathbf{p} . The updated sample is then regarded as clean and appended to the labeled set \mathcal{X} .

$$\mathbf{p} = f_{\theta^{(k)}}(\mathbf{u}), \text{ where } k = \arg \max_{k'} (\max(f_{\theta^{(k')}}(\mathbf{u}))) \quad (18)$$

$$\mathcal{X} \leftarrow \mathcal{X} \cup \{(\mathbf{u}, \mathbf{1}_{\mathbf{p}}) | \max(\mathbf{p}) > \tau_2\}, \quad (19)$$

where $\mathbb{1}_p$ is a one-hot vector of p whose i -th element is 1, considering $i = \arg \max(p)$.

4. Experiments

For evaluation, we first compared the performance of our model against other baselines over multiple datasets. Then, we examined each component’s contribution to performance improvement. Lastly, we investigated how RUC helps improve existing clustering models in terms of their confidence calibration and robustness against adversarial attacks.

4.1. Unsupervised Image Clustering Task

Settings. Four benchmark datasets were used. The first two are CIFAR-10 and CIFAR-100, which contain 60,000 images of 32x32 pixels. For CIFAR-100, we utilized 20 superclasses following previous works [42]. The next is STL-10, containing 100,000 unlabeled images and 13,000 labeled images of 96x96 pixels. For the clustering problem, only 13,000 labeled images were used. Lastly, we test the model with the large-scale ImageNet-50 dataset, containing 65,000 images of 256x256 pixels.

Our model employed the ResNet18 [18] architecture following other baselines [17, 22, 42] and the model was trained for 200 epochs. Initial confidence threshold τ_1 was set as 0.99, and the number of neighbors k to divide the clean and noise samples was set to 100. The threshold τ_2 for refurbishing started from 0.9 and increased by 0.02 in every 40 epochs. The label smoothing parameter ϵ was set to 0.5. For evaluating the class assignment, the Hungarian method [25] was used to map the best bijection permutation between the predictions and ground-truth.

Result. Table 1 shows the overall performance of clustering algorithms over three datasets: CIFAR-10, CIFAR-20, and STL-10. For these datasets, the proposed model RUC, when applied to the SCAN [42] algorithm, outperforms all other baselines. Particularly for STL-10, the combined model shows a substantial improvement of 5.3 pp. Table 2 reports ImageNet-50 result on the confidence based sampling strategy, which demonstrates RUC’s applicability to large-scale dataset. Furthermore, RUC achieves consistent performance gain over another clustering model, TSUC [17]. These results confirm that our model can be successfully applied to existing clustering algorithms and improve them. We also confirm that all three selection strategies (i.e., confidence-based, metric-based, and hybrid) bring considerable performance improvement.

4.2. Component Analyses

To evaluate the model’s efficacy, we conduct an ablation study by repeatedly assessing its performance after removing each component. We also evaluate the accuracy of dif-

Method	CIFAR-10	CIFAR-20	STL-10
<i>k</i> -means [45]	22.9	13.0	19.2
Spectral clustering [55]	24.7	13.6	15.9
Triplets [37]	20.5	9.9	24.4
Autoencoder (AE) [4]	31.4	16.5	30.3
Variational Bayes AE [24]	29.1	15.2	28.2
GAN [36]	31.5	15.1	29.8
JULE [52]	27.2	13.7	27.7
DEC [49]	30.1	18.5	35.9
DAC [8]	52.2	23.8	47.0
DeepCluster [6]	37.4	18.9	33.4
ADC [15]	32.5	16.0	53.0
IIC [22]	61.7	25.7	49.9
TSUC† [17]	80.2	35.5	62.0
SCAN† [42]	88.7	50.6	81.4
TSUC + RUC (Confidence)	81.8 / 82.5	39.6 / 40.6	65.1 / 65.5
TSUC + RUC (Metric)	82.5 / 82.9	39.5 / 40.4	66.3 / 66.6
TSUC + RUC (Hybrid)	82.1 / 82.8	39.5 / 40.6	66.0 / 66.8
SCAN + RUC (Confidence)	90.3 / 90.3	53.3 / 53.5	86.7 / 86.8
SCAN + RUC (Metric)	89.5 / 89.5	53.9 / 53.9	84.7 / 85.1
SCAN + RUC (Hybrid)	90.1 / 90.1	54.3 / 54.5	86.6 / 86.7

Table 1: Performance improvement with RUC (accuracy presented in percent). Baseline results are excerpted from [17, 42] and we report the last/best accuracy. †Results obtained from our experiments with official code.

Method	SCAN (Best)	SCAN + RUC (Last / Best accuracy)
ImageNet-50	76.8	78.5 / 78.5

Table 2: Performance comparison over ImageNet-50

Setup	Last Acc	Best Acc
RUC with all components	86.7	86.8
without co-training	86.2	86.4
without label smoothing	85.5	85.8
with MixMatch only	85.2	85.4

Table 3: Ablation results of the SCAN+RUC on STL-10

ferent selection and refurbishing strategies based on precision, recall, and the F1 score.

Ablation study. The proposed model utilizes two robust learning techniques to cope with unclean samples: co-training and label smoothing. We remove each component from the full model to assess its efficacy. Table 3 shows the classification accuracy of each ablation on the STL-10 dataset. RUC with all components performs the best, implying that dropping any component results in performance degradation. We also compare a variant, which drops both label smoothing and co-training (i.e., MixMatch only). The effect of co-training is not evident in Table 3. Nevertheless, it improved the performance from 36.3% to 39.6% for the lowest noise ratio CIFAR-20 pseudo-labels when we set

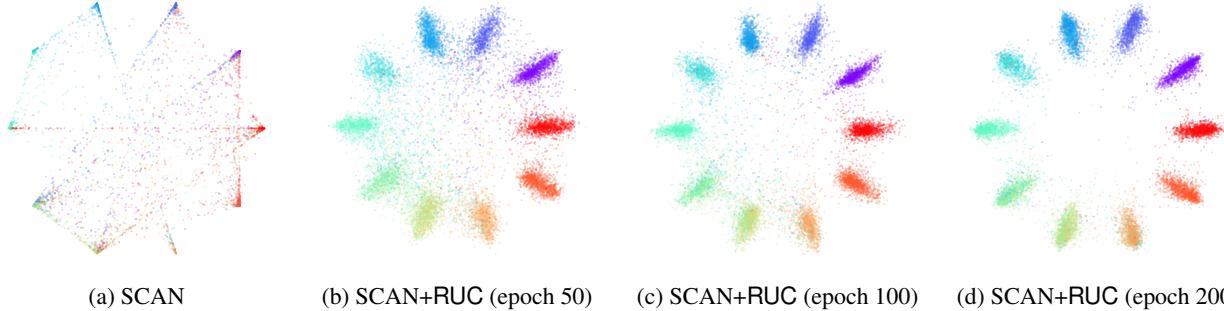


Figure 3: Visualization of clustering results on STL-10. The leftmost figure shows the results from the SCAN, while the right three figures display the intermediate results of our model on top of SCAN at different training epochs. This result demonstrates that RUC effectively alleviates the overconfident predictions while enhancing the clustering quality.

Dataset	CIFAR-10	CIFAR-20	STL-10
SCAN	88.7	50.6	81.4
SCAN + Gupta <i>et al.</i>	88.3 / 89.5	53.2 / 53.3	84.2 / 84.3
SCAN + DivideMix	86.5 / 87.9	53.5 / 53.6	80.6 / 83.9
SCAN + M-correction	81.6 / 88.6	48.5 / 50.4	81.1 / 81.3
SCAN + P-correction	87.7 / 88.7	49.5 / 50.5	81.2 / 81.4
SCAN + RUC (ours)	90.1 / 90.1	54.3 / 54.5	86.6 / 86.7

Table 4: Performance comparison with other possible add-on modules (Last / Best accuracy)

the base model as TSUC. This finding may suggest that co-training is more effective for pseudo-labels with high noise ratios. The co-training structure showed additional stability in training. Due to space limitation, we report these findings in the supplementary material.

Comparison with other possible add-on modules As an alternative of RUC, one may combine the extant robust learning algorithms (e.g., M-correction [1], P-correction [53], and DivideMix [27]) or another previously proposed add-on module (e.g., Gupta *et al.* [14]) on top of SCAN [42]. Table 1 summarizes the comparisons to four baselines. For a fair comparison, we employed SCAN as the initial clustering method and applied each baseline on top of it.² As shown in the results, improving noisy clustering is non-trivial as some baselines show even worse results after the refinement (*e.g.*, DivideMix, M-correction, P-correction). While Gupta *et al.* effectively refines the results, its improvement is limited when the initial clustering is reasonably accurate (*e.g.*, CIFAR-10). In contrast, RUC achieves consistent improvements in all datasets with non-trivial margins, showing that a carefully designed robust learning strategy is critical to the performance gain.

²For methods employing an ensemble of clusterings (*e.g.*, Gupta *et al.*), we employed multiple SCAN models with random initialization. We also applied the same semi-supervised method (*i.e.*, MixMatch).

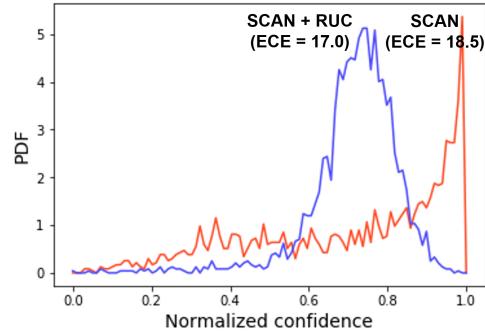


Figure 4: Confidence distribution for noise samples from the STL-10 dataset. RUC shows more widely distributed confidence and produces better calibration.

4.3. In-Depth Performance Analysis

So far, we showed that RUC improves existing baselines substantially, and its components contribute to the performance gain. We now examine RUC’s calibration effect and present a qualitative analysis by applying it to the state-of-the-art base model, SCAN [42]. We will also demonstrate the role of RUC in handling adversarially crafted noise.

Confidence calibration. Many unsupervised clustering algorithms are subject to overconfident results due to their entropy-based balancing [17, 42]. If a model is overconfident to noisy samples, separating the clean and unclean set becomes challenging, and this can induce overall performance degradation. Figure 4 shows the calibration effect of RUC. SCAN’s confidence is highly concentrated near 1, while our model’s confidence is widely distributed over [0.6, 0.8]. We also report the degree of calibration quality using Expected Calibration Error (ECE) [13]:

$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|, \quad (20)$$

	plane	bird	car	cat	deer	dog	horse	monkey	ship	truck	
plane	1226	91	7	5	9	5	6	1	8	14	
bird	3	989	0	35	32	45	16	135	0	5	
car	12	0	1256	0	0	0	6	0	3	60	
cat	0	63	0	746	29	39	3	56	1	0	
deer	0	13	0	50	907	31	21	10	0	1	
dog	2	52	6	299	65	923	86	138	0	0	
horse	1	4	1	18	189	177	1134	17	0	0	
monkey	1	88	0	136	69	73	22	938	0	1	
ship	35	0	1	7	0	4	1	5	1277	35	
truck	20	0	29	4	0	3	5	0	11	1184	

	plane	bird	car	cat	deer	dog	horse	monkey	ship	truck	
SCAN	1260	72	0	2	0	1	2	0	5	11	
SCAN + RUC	3	1130	3	23	19	21	5	87	0	3	

Figure 5: Confusion matrices of the SCAN and SCAN+RUC results on STL-10. The row names are predicted class labels, and the columns are the ground-truths.

where n is the number of data, B_m is the m -th group from equally spaced buckets based on the model confidence over the data points; $acc(B_m)$ and $conf(B_m)$ are the average accuracy and confidence over B_m . Lower ECE of RUC in Figure 4 implies that our approach led to better calibrations.

To observe this effect more clearly, we visualize the clustering confidence result at different training epochs in Figure 3. Unlike the result of SCAN in which the overly clustered sample and the uncertain sample are mixed, the result of SCAN+RUC shows that the sample’s class distribution has become smoother, and uncertain samples disappeared quickly as training continues.

Qualitative analysis. We conducted a qualitative analysis to examine how well RUC corrects the initial misclassification in pseudo-labels. Figure 5 compares the confusion matrices of SCAN and the SCAN+RUC for STL-10. A high concentration of items on the diagonal line confirms the advanced correction effect of RUC for every class. Figure 6 compares how the two models interpreted class traits based on the Grad-CAM [38] visualization on example images. The proposed model shows a more sophisticated prediction for similar images.

Robustness to adversarial noise. Clustering algorithms like SCAN introduce pseudo-labels to train the model via the Empirical Risk Minimization (ERM) method [43]. ERM is a learning principle that minimizes the averaged error over the sampled training data (i.e., empirical risk) to find the model with a small population risk (i.e., true risk). However, ERM is known to be vulnerable to adversarial examples, which are crafted by adding visually imperceptible perturbations to the input images [31, 56].

Here, we show that RUC improves robustness against the adversarial noise. We conduct an experiment on STL-10 using adversarial perturbations of FGSM [12] and BIM [26] attacks, whose directions are aligned with the gradient of the loss surface of given samples. Figure 7 compares the model’s ability to handle the adversarial noise. Models based on MixMatch (Gupta *et al.*, DivideMix, RUC) out-

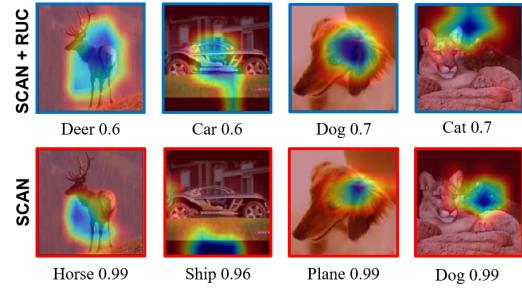


Figure 6: Class activation maps and the model’s confidence on STL-10. The highlighted area indicates where the model focused to classify the image.

perform the rest, probably because the calibration effect of MixUp prevents overconfident predictions. Among them, RUC achieves superior improvement, demonstrating that robust learning components, such as careful filtering, label smoothing, and co-training, can also handle the adversarial noise (see the supplementary material for further details).

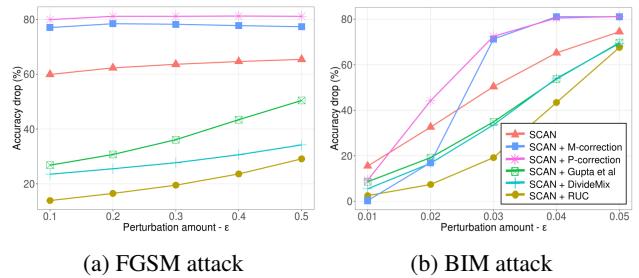


Figure 7: RUC’s robustness to adversarial attacks, experimented with different perturbation rate ϵ . Robust learning components are important in handling the adversarial noise.

5. Conclusion

This study presented RUC, an add-on approach for improving existing unsupervised image clustering models via robust learning. Retraining via robust training helps avoid overconfidence and produces more calibrated clustering results. As a result, our approach achieved a meaningful gain on top of two state-of-the-art clustering methods. Finally, RUC helps the clustering models to be robust against adversarial attacks. We expect robust learning will be a critical building block to advance real-world clustering solutions.

Acknowledgements. This work was supported in part by the IBS (IBS-R029-C2) and the Basic Science Research Program through the NRF funded by the Ministry of Science and ICT in Korea (No. NRF-2017R1E1A1A01076400), Institute of Information & communications Technology Planning & Evaluation (IITP) grant (2020-0-00153 and 2016-0-00464), Samsung Electronics, HPC support funded by MSIT & NIPA.

References

- [1] Eric Arazo, Diego Ortego, Paul Albert, Noel O’Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *Proc. of International Conference on Machine Learning*, pages 312–321, 2019. 7
- [2] Devansh Arpit, Stanislaw K Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron C Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proc. of International Conference on Machine Learning*, 2017. 3
- [3] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proc. of International Conference on Machine Learning Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012. 2
- [4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160, 2007. 6
- [5] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. MixMatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019. 2, 4, 5, 12
- [6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. of European Conference on Computer Vision*, pages 132–149, 2018. 2, 6
- [7] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1002–1012, 2017. 3
- [8] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proc. of IEEE International Conference on Computer Vision*, pages 5879–5887, 2017. 2, 6
- [9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of International Conference on Machine Learning*, 2020. 2, 4, 11
- [10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 5, 11
- [11] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proc. of International Conference on Machine Learning*, page 29, 2004. 1, 2
- [12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of International Conference on Learning Representations*, 2015. 8, 13
- [13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proc. of International Conference on Machine Learning*, pages 1321–1330, 2017. 7, 12
- [14] Divam Gupta, Ramachandran Ramjee, Nipun Kwatra, and Muthian Sivathanu. Unsupervised clustering using pseudo-semi-supervised learning. In *Proc. of International Conference on Learning Representations*, 2020. 2, 7
- [15] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *Proc. of German Conference on Pattern Recognition*, pages 18–32. Springer, 2018. 6
- [16] 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. In *Advances in Neural Information Processing Systems*, pages 8527–8537, 2018. 3, 5
- [17] Sungwon Han, Sungwon Park, Sungkyu Park, Sundong Kim, and Meeyoung Cha. Mitigating embedding and class assignment mismatch in unsupervised image classification. In *Proc. of European Conference on Computer Vision*, pages 768–784, 2020. 1, 2, 6, 7, 11, 12
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6, 11
- [19] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in Neural Information Processing Systems*, pages 10456–10465, 2018. 3
- [20] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proc. of International Conference on Machine Learning*, pages 1558–1567, 2017. 1, 2
- [21] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999. 2
- [22] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proc. of IEEE International Conference on Computer Vision*, pages 9865–9874, 2019. 1, 2, 6, 11
- [23] 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. In *Proc. of International Conference on Machine Learning*, pages 2304–2313, 2018. 3
- [24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 6
- [25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 6, 11
- [26] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016. 8, 13
- [27] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *Proc. of International Conference on Learning Representations*, 2020. 3, 5, 7, 12

- [28] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems*, 2020. 3
- [29] Michal Lukasik, Srinadh Bhojanapalli, Aditya Krishna Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *Proc. of International Conference on Machine Learning*, 2020. 2, 3, 5
- [30] Yueming Lyu and Ivor W Tsang. Curriculum loss: Robust learning and generalization against label corruption. In *Proc. of International Conference on Learning Representations*, 2020. 3
- [31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. of International Conference on Learning Representations*, 2018. 8, 13
- [32] Eran Malach and Shai Shalev-Shwartz. Decoupling” when to update” from” how to update”. In *Advances in Neural Information Processing Systems*, pages 960–970, 2017. 3
- [33] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing systems*, pages 1196–1204, 2013. 3
- [34] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016. 13
- [35] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017. 3
- [36] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. of International Conference on Learning Representations*, 2016. 6
- [37] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems*, pages 41–48, 2004. 6
- [38] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of IEEE International Conference on Computer Vision*, pages 618–626, 2017. 8, 13
- [39] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *Proc. of International Conference on Machine Learning*, pages 5907–5915, 2019. 3
- [40] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020. 3
- [41] Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020. 3
- [42] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *Proc. of European Conference on Computer Vision*, 2020. 1, 2, 6, 7, 11, 12, 13
- [43] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 8, 13
- [44] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010. 1, 2
- [45] Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):180–192, 2014. 6
- [46] Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass learning with partially corrupted labels. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2568–2580, 2017. 3
- [47] David Warde-Farley and Ian Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311, 2016. 13
- [48] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 2
- [49] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. of International Conference on Machine Learning*, pages 478–487, 2016. 2, 6
- [50] Yizhan Xu, Sungwon Han, Sungwon Park, Meeyoung Cha, and Cheng-Te Li. A Comprehensive and Adversarial Approach to Unsupervised Embedding Learning. In *Proc. of IEEE International Conference on Big Data*, 2020. 2
- [51] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proc. of International Conference on Machine Learning*, pages 3861–3870, 2017. 2
- [52] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. 6
- [53] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019. 7
- [54] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *Proc. of International Conference on Machine Learning*, pages 7164–7173, 2019. 3
- [55] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2005. 6
- [56] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. of International Conference on Learning Representations*, 2018. 4, 5, 8, 13

A. Supplementary Material

A.1. Release

Codes, training details, and the downloadable link for trained models are available at <https://github.com/deu30303/RUC>.

A.2. Training Details

Our model employed the ResNet18 [18] architecture following other baselines [17, 22, 42]. Before retraining, note that we randomly initialize the final fully connected layer and replace the backbone network with a newly pretrained one from an unsupervised embedding learning algorithm as done in SimCLR [9]. This random re-initialization process helps avoid the model from falling into the same local optimum. The initial confidence threshold τ_1 was set as 0.99, and the number of neighbors k to divide the clean and noise samples was set to 100. The threshold τ_2 for refurbishing started from 0.9 and increased by 0.02 in every 40 epochs. The label smoothing parameter ϵ was set to 0.5. The initial learning rate was set as 0.01, which decays smoothly by cosine annealing. The model was trained for 200 epochs using SGD with a momentum of 0.9, a weight decay of 0.0005. The batch size was 100 for STL-10 and 200 for CIFAR-10 and CIFAR-20. We chose λ_u as 25, 50, 100 for CIFAR-10, STL-10 and CIFAR-20. The w_b value was calculated by applying min-max normalization to the confidence value of the counter network $f_{\theta(c)}$. Random crop and horizontal flip were used as a weak augmentation, which does not deform images' original forms. RandAugment [10] was used as a strong augmentation. We report all transformation operations for strong augmentation strategies in Table 5. The number of transformations and magnitude for all the transformations in RandAugment was set to 2.

Transformation	Parameter	Range
AutoContrast	-	-
Equalize	-	-
Identity	-	-
Brightness	B	[0.01, 0.99]
Color	C	[0.01, 0.99]
Contrast	C	[0.01, 0.99]
Posterize	B	[1, 8]
Rotate	θ	[-45, 45]
Sharpness	S	[0.01, 0.99]
Shear X, Y	R	[-0.3, 0.3]
Solarize	T	[0, 256]
Translate X, Y	λ	[-0.3, 0.3]

Table 5: List of transformations used in RandAugment

To evaluate class assignment, the Hungarian method [25] was used to map the best bijection permutation between the predictions and ground-truth. We also note that the computational cost of RUC is not a huge burden. It took less than 12 hours to run 200 epochs with 4 TITAN Xp processors for all datasets.

A.3. Sampling strategy analysis.

We evaluate the quality of the clean set generated from three sampling strategies (See Table 6). Overall, precision was the highest for the hybrid strategy, whereas recall was the highest for the metric-based strategy. We also tested the co-refurbish accuracy over the epochs. Figure 8 displays the change of precision, recall, and the F1-score using confidence-based sampling on the STL-10 dataset. The model's precision drops slightly as the number of epochs increases, but the recall increases significantly. The F1-score, which shows the overall sampling accuracy of the clean set, increased about 5% over 200 epochs. It can be interpreted a higher rate of true-positive cases than the false-positive cases in the refurbished samples, which means that the model could successfully correct the misclassified unclean samples. Overall, we find the current hybrid selection strategy can distinguish clean sets relatively well since the selected samples benefit from both strategies' merits. This strategy, however, cannot always achieve the best performance. Further development of the selection strategy will help increase the proposed RUC model.

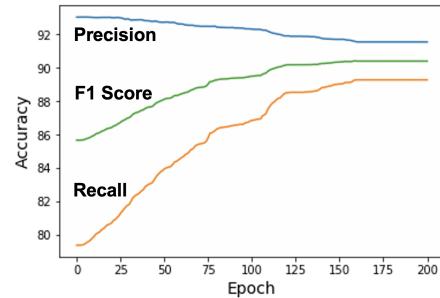


Figure 8: Changes of sampling accuracy across each epoch on our model

Strategy	CIFAR-10			CIFAR-20			STL-10		
	C	M	H	C	M	H	C	M	H
Precision	92.6	91.6	93.7	59.5	59.0	63.6	93.0	87.6	94.2
Recall	93.5	93.2	89.3	83.1	88.5	77.4	79.4	94.4	78.3
F1 Score	93.0	92.4	91.4	69.3	70.8	69.8	85.7	90.9	85.5

Table 6: Quality of the clean set (C : Confidence, M : Metric, H : Hybrid)

A.4. Hyper-parameters of Sampling Strategies

We investigate the effect of hyper-parameters from two sampling strategies: τ_1 and k . τ_1 is the threshold for selecting clean samples in the confidence-based strategy, and k is the number of neighbors for the kNN classifier in the metric-based strategy. Figure 9 summarizes the effect of each hyper-parameter. In the case of τ_1 , the final accuracy reaches the highest at $\tau_1 = 0.99$ and starts to decrease. Small τ_1 extracts clean samples with higher recall and lower precision, while large τ_1 extracts clean samples with higher precision and lower recall. Hence, balancing between the precision and recall through appropriate τ_1 can lead to better performance. Meanwhile, the number of nearest neighbors k does not significantly affect the final accuracy. Given k within the reasonable range, our model consistently produces results of high performance.

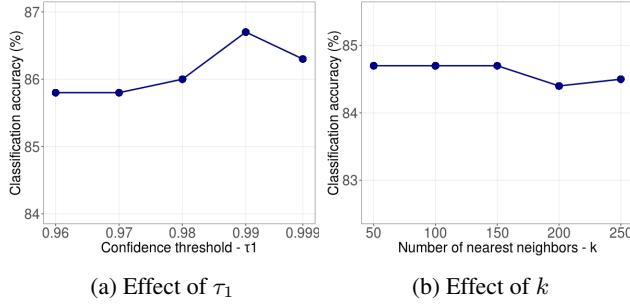


Figure 9: Analysis of the accuracy on the STL-10 dataset across two hyper-parameters: (a) τ_1 from the confidence-based strategy and (b) k from the metric-based strategy.

For remaining hyper-parameters ($\lambda_u, \epsilon, \tau_2$), our model resorted to a standard hyper-parameter setting that is commonly used in practice. For example, we set λ_u following earlier works [5, 27], choose $\epsilon = 0.5$ as the mean of Uniform(0, 1), and choose τ_2 to be a reasonably high value, similar to τ_1 . Empirically, we find the model is oblivious to these parameters (see Table 7).

λ_u	25	50	100	ϵ	0.4	0.5	0.6
STL-10	86.20	86.7	85.74	CIFAR-10	90.31	90.3	90.27
CIFAR-10	90.3	90.07	89.21	τ_2	0.85	0.9	0.95
CIFAR-20	53.00	53.05	53.50	CIFAR-10	90.28	90.3	90.28

Table 7: Hyper-parameter analyses ($\lambda_u, \epsilon, \tau_2$)

A.5. Additional Analysis for RUC on TSUC

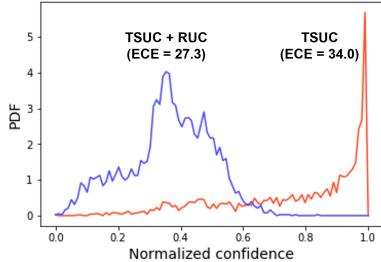


Figure 10: Confidence distribution for noise sample on STL-10 with the base model TSUC.

Many recent unsupervised clustering algorithms are subject to overconfident results because of their entropy-based balancing [17, 42]. If a model is overconfident to noisy samples, separating the clean set and the unclean set becomes challenging, which can induce the overall performance degradation. We evaluated the calibration effect of RUC on top of TSUC in Figure 10. TSUC’s confidence is highly concentrated near 1, while our model’s confidence is widely distributed. We also report the degree of calibration quality using the Expected Calibration Error (ECE) [13]:

$$ECE = \sum_{m=1}^M \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|, \quad (21)$$

where n is the number of data points, B_m is the m -th group from equally spaced buckets based on the model confidence over the data points; $acc(B_m)$ and $conf(B_m)$ are the average accuracy and confidence over B_m respectively. TSUC’s high ECE implies that TSUC is more overconfident than SCAN. Lower ECE of TSUC + RUC case in Figure 10 implies that our add-on process led to better calibrations.

We also evaluated quality of the clean set from TSUC under three sampling strategies. The results are shown in Table 8. Overall, the precision is the highest for the hybrid strategy, whereas the recall is the highest for the metric-based strategy, as same as the SCAN’s results. Meanwhile, confidence-based strategies in TSUC showed low precision, which implies that TSUC is not well-calibrated and highly overconfident.

Strategy	CIFAR-10			CIFAR-20			STL-10		
	C	M	H	C	M	H	C	M	H
Precision	80.9	84.2	82.7	40.9	41.8	43.0	68.2	69.0	71.4
Recall	69.9	96.4	68.0	47.4	90.3	45.5	78.3	79.4	76.2
F1 Score	69.5	89.9	74.6	43.9	85.5	44.2	72.9	73.8	73.7

Table 8: Quality of the clean set regards to sampling strategies (C : Confidence, M : Metric, H : Hybrid)

A.6. Further Discussion on Co-Training

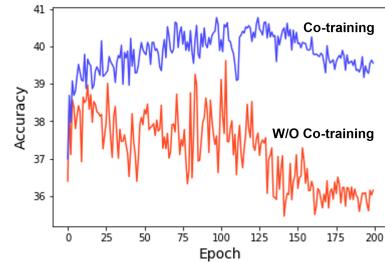


Figure 11: Changes of clustering accuracy across each epoch on CIFAR-20 with the base model TSUC

Our model architecture introduces a co-training module where the two networks exchange their guesses for teaching each other via co-refinement. Due to the different learning abilities in two networks, disagreements from networks help filter out corrupted labels, which contributes to a substantial performance increase in unsupervised classification. Besides, the co-training structure provides extra stability in the training process.

Figure 11 compares our model and the same model without co-training based on classification accuracy across the training epoch. The model without co-training shows large fluctuations in accuracy; in contrast, the full model’s accuracy remains stable and consistent throughout epochs. We speculate this extra stability comes from our model’s ensemble architecture and the effect of loss correction. Corrected labels via ensemble predictions bring additional label smoothing. Therefore, it may reduce the negative training signals from unclean samples, which can lead to abrupt updates on the model parameters.

A.7. Further Details on Adversarial Robustness

Empirical risk minimization (ERM), a learning principle which aims to minimize the averaged error over the sampled training data (i.e., empirical risk), has shown remarkable success in finding models with small population risk (i.e., true risk) in the supervised setting [43]. However, ERM-based training is also known to lead the model to memorize the entire training data and often does not guarantee to be robust on adversarial noise [31, 56]. This weakness can also be inherited from several unsupervised clustering algorithms that introduce the ERM principle with their pseudo-labels, like SCAN [42].

Adding RUC to the existing clustering models improves robustness against adversarial noise. To demonstrate this, we conducted an experiment using adversarial perturbations of the FGSM [12] and BIM [26] attacks, whose directions are aligned with the gradient of the loss surface of given samples. The details of each attack are as follows:

Fast Gradient Sign Method (FGSM) FGSM crafts adversarial perturbations by calculating the gradients of the loss function $J(\theta, \mathbf{x}, \mathbf{y})$ with respect to the input variables. The input image is perturbed by magnitude ϵ with the direction aligned with the computed gradients (Eq. (22)).

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \cdot \text{sgn}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, \mathbf{y})) \quad (22)$$

Basic Iterative Method (BIM) BIM is an iterative version of FGSM attack, which generates FGSM based adversarial noise

with small ϵ and applies the noise many times in a recursive way (Eq. (24)).

$$\mathbf{x}_0^{adv} = \mathbf{x} \quad (23)$$

$$\mathbf{x}_i^{adv} = \text{clip}_{\mathbf{x}, \epsilon}(\mathbf{x}_{i-1}^{adv} + \epsilon \cdot \text{sgn}(\nabla_{\mathbf{x}_i^{adv}} J(\theta, \mathbf{x}_{i-1}^{adv}, \mathbf{y}))) \quad (24)$$

Clip function maintains the magnitude of noise below ϵ by clipping. For BIM attack experiments, we use five iterations with an equal step size.

Figure 7 in our main manuscript compares the model’s ability to handle adversarial attacks, which confirms that adding RUC helps maintain the model accuracy better for both attack types. An investigation could guide us that this improved robustness is mainly due to the label smoothing techniques, which regularize the model to avoid overconfident results and reduce the amplitude of adversarial gradients with smoothed labels [34, 47].

A.8. Additional Examples for Qualitative Analysis

Figure 12 shows additional examples for the visual interpretation from RUC on top of SCAN via the Grad-CAM algorithm [38]. Blue framed images are the randomly chosen success cases from STL-10, and the red-framed images are example failure cases. Overall, the network trained with our model can extract key features from the images. Even though the model sometimes fails, most of the failures occurred between visually similar classes (e.g., horse-deer, cat-dog, truck-car, dog-deer).

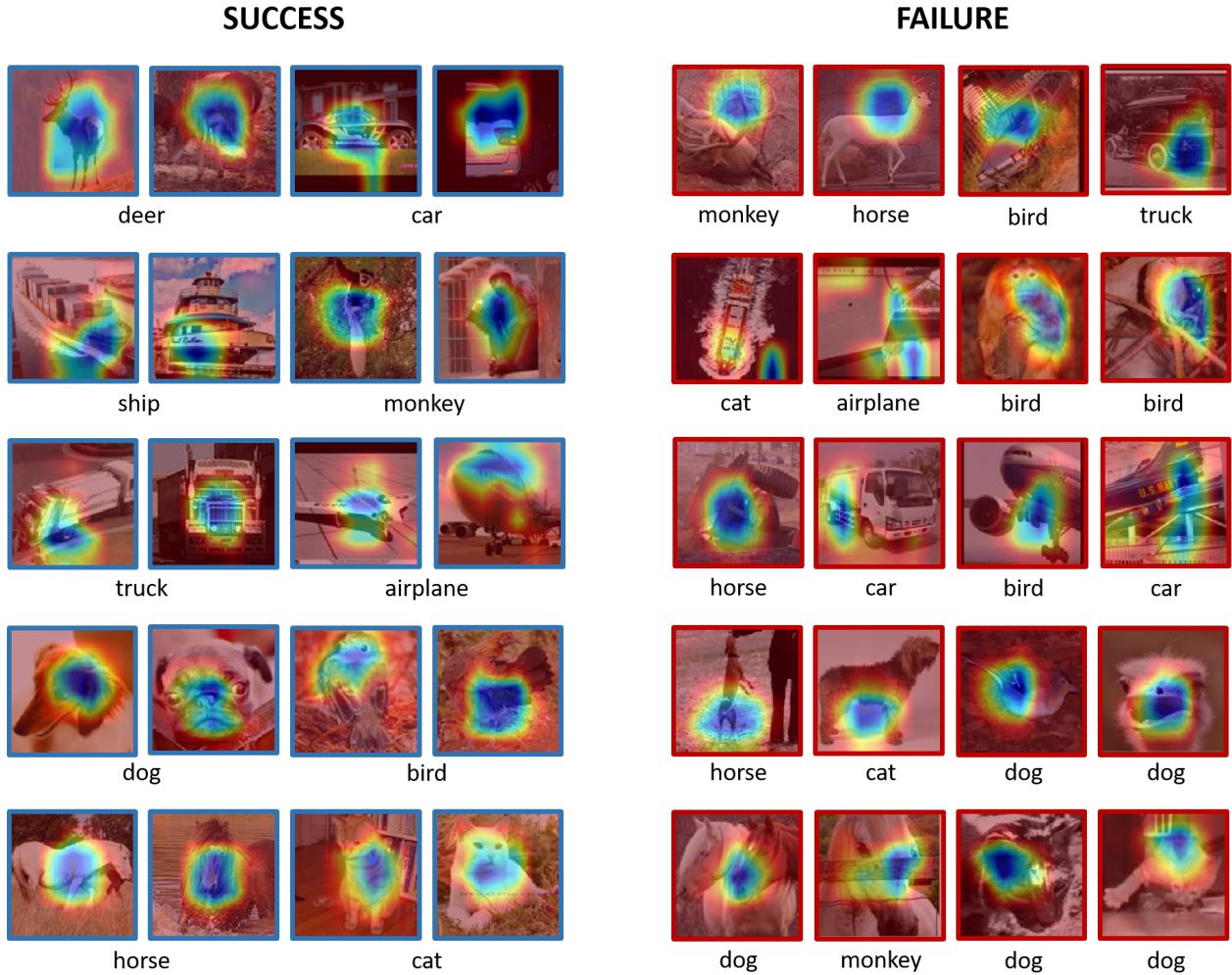


Figure 12: Additional example of successes and failures from STL-10 where the highlighted part indicates how the model interprets class traits based on the Grad-CAM method (Blue frame: success case, Red frame: failure case).