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.

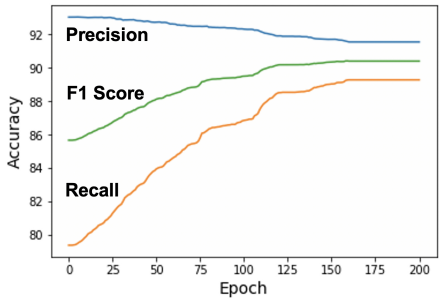


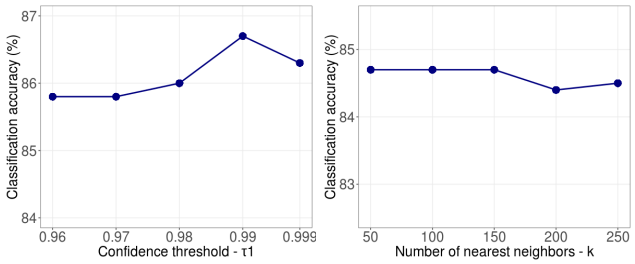
Figure 8: Changes of sampling accuracy across each epoch on our model

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.



(a) Effect of τ_1 (b) Effect of k

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 (λ_u, ϵ, τ_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
STL-10	86.20	86.7	85.74
CIFAR-10	90.3	90.07	89.21
CIFAR-20	53.00	53.05	53.50

ϵ	0.4	0.5	0.6
CIFAR-10	90.31	90.3	90.27
τ_2	0.85	0.9	0.95
CIFAR-10	90.28	90.3	90.28

Table 7: Hyper-parameter analyses (λ_u, ϵ, τ_2)

A.5. Additional Analysis for RUC on TSUC

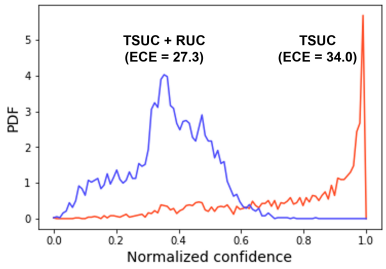


Figure 10: Confidence distribution for noise sample on STL-10 with the base model TSUC.

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]:

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

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

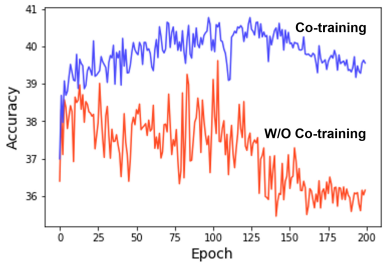


Figure 11: Changes of clustering accuracy across each epoch on CIFAR-20 with the base model TSUC

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)).

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

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)).

$$\mathbf{x}_0^{adv} = \mathbf{x} \tag{23}$$

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

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).

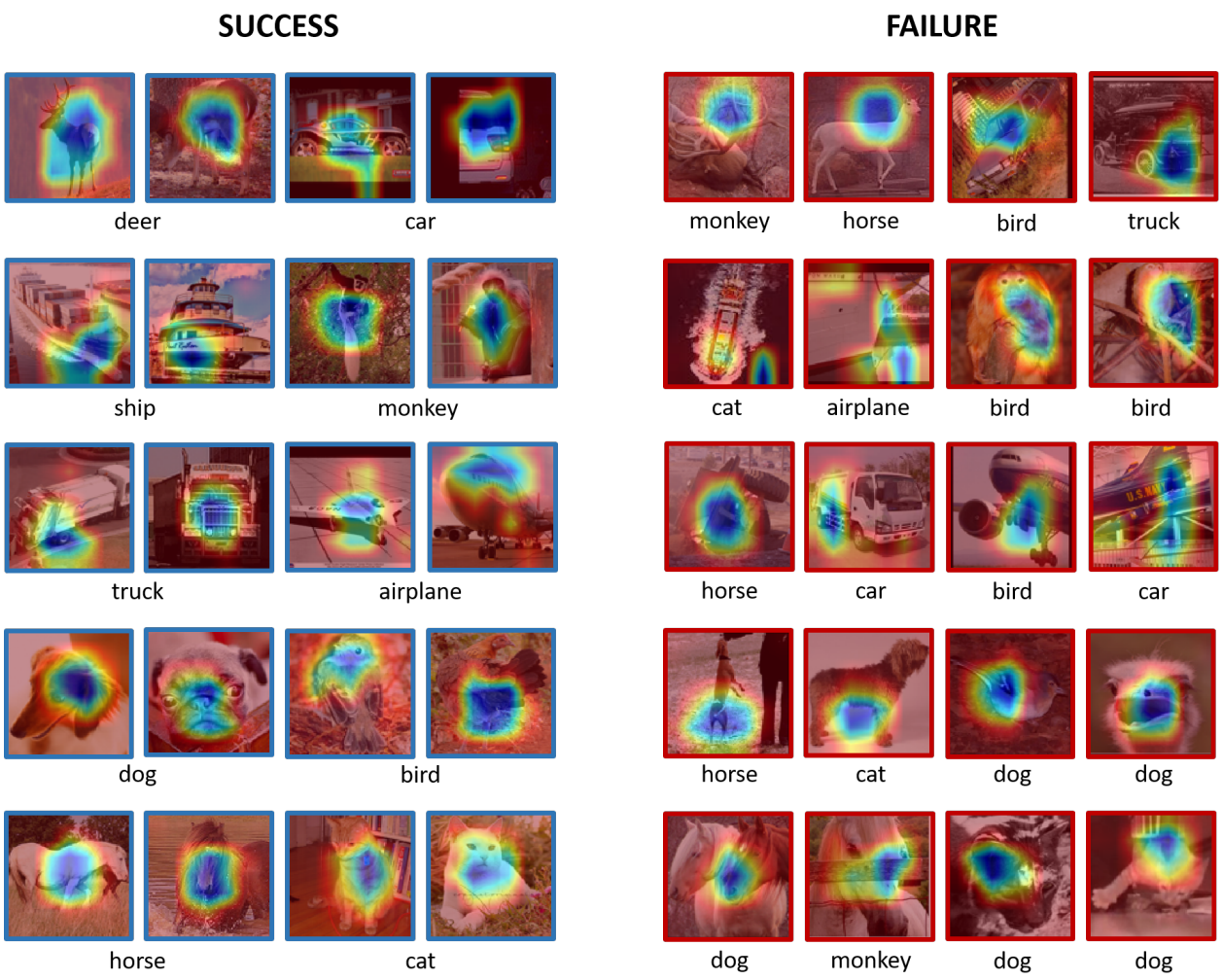


Figure 12: Additional example of successes and failures from STL-10 where the highlighted part indicates how the model interprets class traits based on the Grad-CAM method (Blue frame: success case, Red frame: failure case).