A. Algorithms

Algorithm 1 Confidence-based Hash Merging

Input: Confidence threshold ϵ, node size threshold τ, output soft hash matrix $Z = [h_i]$ where $Z \in \mathbb{R}^{N\times L}$, $h_i \in (-1,1)^L \ \forall i \in \{1,\cdots,N\}$.

Output: Refined hash matrix $Z' = [b_i]$ where $Z' \in \mathbb{R}^{N\times L}$

Extract hash set $\mathcal{C} = \{(b, c^b) | b \in \text{sign}(Z), c^h = \text{count of } h \text{ in sign}(Z)\}$, and let $\mathcal{B} \in \mathbb{R}^{m\times L}$ with the number of detected categories m. // (Make a dictionary for all test samples (Hash, count))

for h_i in Z **do**
  $(b_i, c^b) \leftarrow \text{sign}(h_i)$
  if $c^h < \tau$ **then**
    Identify the low confidence indices $I_i = \{j | |z_{ij}| < \epsilon\}$
    $(k_c, \{P_k\}_{k=1}^{k_c}) \leftarrow$ **Creating Candidates** (Algorithm 2)
    for $k = 1,\cdots,k_c$ **do**
      $W_k \leftarrow 0$
      if $D_k \in \mathcal{B}$ **then**
        $W_k \leftarrow c^{D_k} P_k$ // (Transform the probability to the weight by multiplying the probability and count)
      end if
    end for
    $\hat{k} \leftarrow \arg\max(W_1,\cdots,W_{k_c})$ // (Find most-likely (best) candidate's index.)
    if $D_{\hat{k}} \in \mathcal{B}$ **then**
      $b_i \leftarrow D_{\hat{k}}$ // (If the best candidate is in the hash dictionary, return it.)
    else
      $b_i \leftarrow$ **Find Largest Neighbor**$(h_i, \mathcal{C})$ // (Otherwise, find the largest neighbor (Algorithm 3).)
    end if
  end if
end for

Algorithm 2 Creating Candidates

Input: Low confidence indices I_i, current soft-hash h_i and its binary hash b_i.

Output: The number of candidates k_c, the probability mapping the current instance to the candidate, $\mathcal{P}_i \leftarrow \{P_1,\cdots,P_{k_c}\}$.

$k \leftarrow 0$
for $I_i' \subseteq I_i$ **do**
  $k \leftarrow k+1$
  $D_k \leftarrow h_i$ // (Copying the current hash.)
  $d_k \leftarrow 0$ // (Initialize soft Hamming distance.)
  for $j \in I_i'$ **do**
    $D_{kj} \leftarrow 1 - b_{ij}$ // (Define candidate hash.)
    $d_k \leftarrow d_k + |h_{ij}|$ // (Compute the soft Hamming distance.)
  end for
  $S_k \leftarrow \epsilon L - d_k$ //(Define soft Hamming similarity.)
end for
$k_c \leftarrow k$ // (The number of current candidates.)
$\mathcal{D}_i \leftarrow \{D_1,\cdots,D_k\}$ // (Final candidate set of the current hash.)
$P_1,\cdots,P_{k_c} = \text{softmax}(S_1, S_2, \cdots S_{k_c})$ // (Transform the similarity to the probability.)
$\mathcal{P}_i \leftarrow \{P_1,\cdots,P_{k_c}\}$

Algorithm 3 Finding Largest Neighbor

Input: a hash instance b_i, Hash set $\mathcal{C}$

Output: largest neighbor hash $\tilde{b}_i$

$d \leftarrow 0$
while True **do**
  $d \leftarrow d+1$ // (The target Hamming distance between candidates and the current hash.)
  Create a candidate set $\mathcal{G} = \{g | \text{Hamming}(b_i, g) = d\}$ // (Find candidates.)
  if $\mathcal{G} \cap \mathcal{C} \neq \emptyset$ **then**
    return $\tilde{b}_i = \arg\max_{\hat{k}} \{\hat{k} \in 1,\cdots,|\mathcal{C}|\}$ // (If any candidates are in the hash dictionary, return the largest candidate. Otherwise, increase the target Hamming distance.)
  end if
end while

B. Ablation study

To verify the effectiveness of NCD-DLT in a dynamic long-tailed scenario, we conduct ablation studies on the long-tailed CIFAR100 dataset with $\rho = 100$.

B.1. Impact of code length L

Although OCD [5] suggested $L = 12$, we conduct our own ablation study to identify the best code length for NCD-DLT. Consistent with OCD [5], NCD-DLT also performed best at $L = 12$ for 'New' accuracy, as shown in Figure 6. Although the highest 'All' accuracy occurred at $L = 16$, we prioritize 'New' accuracy and choose $L = 12$ due to comparable 'All' accuracy levels.

B.2. Impact of λ

In Sec.5.4.1, we used the same λ for Eq.(7) in an ablation study. However, we separately analyze the impact of $\mathcal{L}^{sd}$ and $\mathcal{L}^{dd}$ by assigning different weights, λ_s for static loss and λ_d for distillation loss, instead of a single λ. We vary λ_s and λ_d across $[1, 5, 10, 20, 100]$. As expected, larger λ_s increase 'Old' accuracy by preserving initial training knowledge (Figure 7). The 'New' accuracy peak when λ_d was set to 5 or 20. We select $\lambda_s = \lambda_d = 10$ considering both 'All' and 'New' accuracies, consistent with the main paper results.

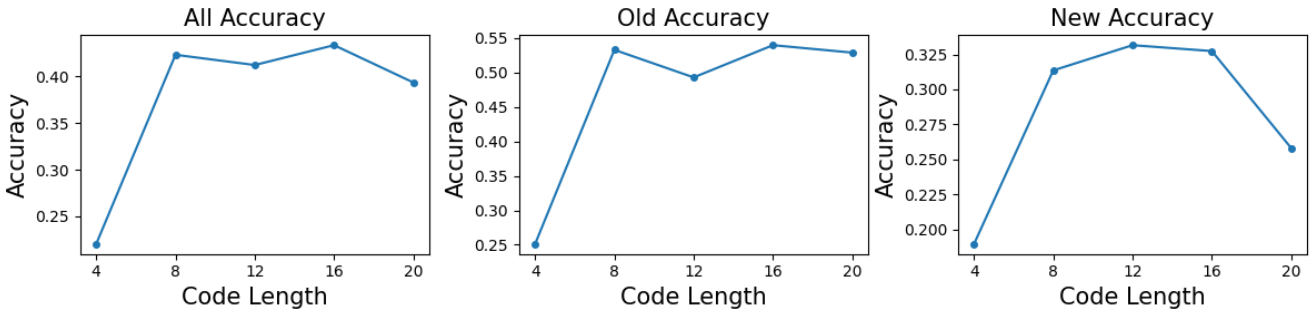

Figure 6. The ablation study on code length shows that larger L degrades 'New' accuracy. The best performance is achieved at $L = 12$.

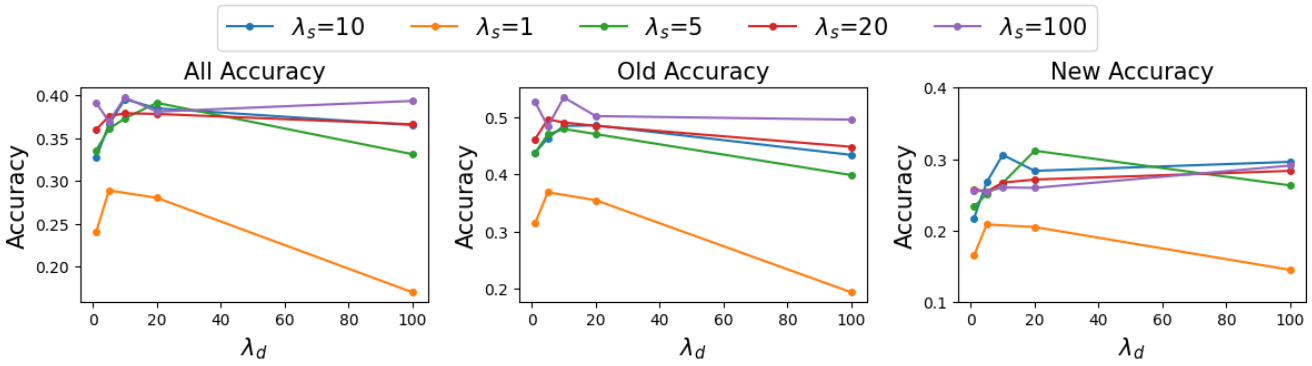

Figure 7. The best 'All' accuracy is achieved when λ_s is 10 or 100. We choose $\lambda_s = \lambda_d = 10$ due to the higher 'New' accuracy.

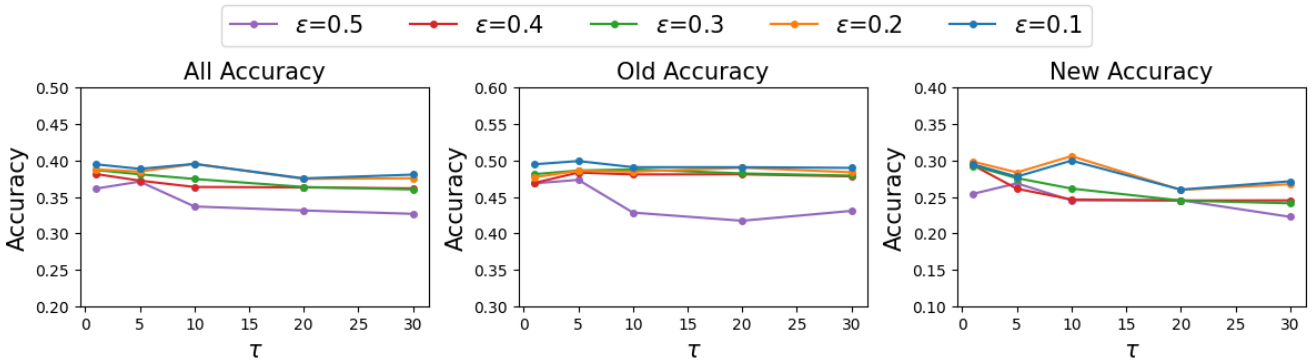

Figure 8. The impact of τ and ϵ on pseudo-labeling accuracy shows that $\epsilon \geq 0.3$ negatively affects accuracy, while $\epsilon = 0.1$ and $\epsilon = 0.2$ yield similar results. $\tau = 10$ improves 'New' accuracy. We selecte $\tau = 10$ and $\epsilon = 0.2$ for further experiments.

B.3. Impact of τ and ϵ

B.3.1 Impact of τ and ϵ in pseudo-labeling

Figure 8 illustrates the impact of τ and ϵ in graph merging for pseudo-labeling based on the results before post-processing. For ϵ, lower values lead to better accuracy, as they identify low-confidence indices where the original cluster is likely a neighbor, thereby smaller ϵ (i.e., more stringent) consistently improve accuracy. For τ, larger values merge more clusters into confident neighbor nodes, improving performance up to $\tau = 10$, after which accuracy declined. Given similar 'All' accuracy levels, we select $\tau = 10$ and $\epsilon = 0.2$, where the 'New' accuracy was highest.

B.3.2 Impact of τ and ϵ in post-processing

Unlike their effect in pseudo-labeling, the impact of τ and ϵ in post-processing follows a similar trend but peaks differently as shown in Figure 9. Excessively large values of both τ and ϵ degrade performance due to the over-merging of

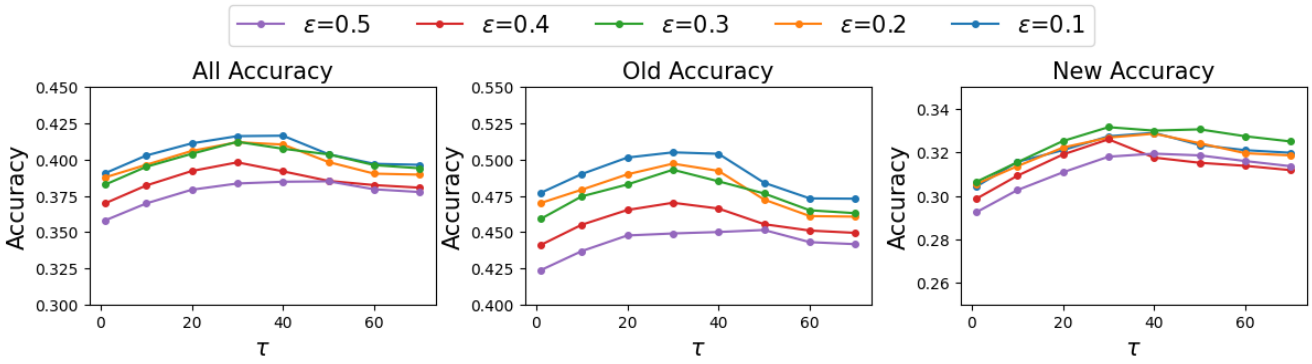

Figure 9. The impact of τ and ϵ on post-processing accuracy shows that $\tau = 30$ and $\epsilon = 0.3$ yield the best 'New' accuracy while maintaining comparable 'All' accuracy.

distinct clusters. For 'New' accuracy, $\epsilon = 0.3$ consistently provides the best results, while 'All' accuracy shows minimal variation between $\epsilon = 0.1$ and $\epsilon = 0.3$. Therefore, we select $\epsilon = 0.3$ and $\tau = 30$ as the optimal hyperparameters for post-processing.

C. Detailed of Evaluation Metrics

C.1. Strict Evaluation Metrics

Beyond the Greedy Hungarian method, as utilized by [8] and the main paper, another evaluation metric, the Strict Hungarian method is implemented by [27]. Both methods are derivatives of the algorithm introduced in [19]. The Greedy Hungarian method involves segregating samples into 'New' and 'Old' subsets based on their ground truth labels, followed by separate accuracy calculations for each subset. This approach offers a detailed view of the model's performance on both 'New' and 'Old' subsets. Conversely, the Strict Hungarian method calculates the accuracy across the entire query set upfront, thereby avoiding the potential issue of a single cluster being allocated to both 'New' and 'Old' categories simultaneously. While the results in the main paper is that of Greedy Hungarian, the experimental results with Strict Hungarian is introduced in Table 4. Even with the Strict Hungarian, NCD-DLT consistently outperform in 'All' accuracy in all scenarios.

C.2. Difference in obtaining 'All' accuracy

As discussed in the previous section, Greedy-Hungarian and Strict-Hungarian differ in their evaluation approaches: Greedy-Hungarian considers old and new classes separately, while Strict-Hungarian evaluates them together. Consequently, 'All' accuracy in Greedy-Hungarian is calculated as a weighted average of 'Old' and 'New' accuracy, whereas Strict-Hungarian calculates accuracy by running the algorithm once for all instances, without distinguishing between old and new classes. These methods are suited to different scenarios, and the choice between them depends on the user's specific needs. For instance, in a controlled setting like a medical diagnosis tool, where old diseases are well-documented but new or emerging diseases require special attention, Greedy-Hungarian allows for a more focused evaluation of the model's performance on new categories versus known ones. This approach is also beneficial when performance on new categories is critical, such as in recommendation systems or fraud detection, where failing to detect new categories can have high costs.

On the other hand, in generalized tasks like image classification for autonomous vehicles, where there is no clear boundary between old and new categories, Strict-Hungarian is more advantageous. Since it treats all categories equally, it provides a holistic evaluation of model performance across the entire dataset.

In this paper, we report the results of Greedy-Hungarian in the main text, as it emphasizes 'New' accuracy and treats new categories with special attention. However, Strict-Hungarian remains important for more general image classification tasks. The experimental results using Strict-Hungarian are presented in Table 4, demonstrating that even under this evaluation method, NCD-DLT consistently outperforms in 'All' accuracy across all scenarios.

D. Training Time Analysis

To verify the effectiveness of NCD-DLT in terms of computational cost, we compare the training time with and without knowledge distillation. Over 20 incremental stages and 10 epochs per stage, the total difference in training time between with and without knowledge distillation is 116 seconds, averaging 0.58 seconds per epoch. This is a minor increase given the significant performance improvements achieved through the use of knowledge distillation losses. Additionally, the graph-merging algorithm is highly cost-effective, with the post-processing step requiring only 0.92 seconds on a CPU during the inference stage.

Original Dataset	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.2303	0.2536	0.2070	0.3125	0.3684	0.2566	0.3554	0.4128	0.2980
OCD	0.4985	**0.9820**	0.0150	0.3886	0.7296	0.0476	0.4028	**0.7336**	0.0720
GM	0.2137	0.2774	0.1500	0.2604	0.4124	0.1084	0.1904	0.2844	0.0964
BaCon	0.5815	0.8642	0.2988	0.4628	0.6292	0.2964	**0.4766**	0.5252	**0.4280**
MetaGCD	0.6375	0.3390	**0.9360**	0.3423	0.3414	**0.3432**	0.3222	0.4464	0.1980
Ours (NCD-DLT)	**0.6419**	0.8268	0.4570	**0.5045**	**0.7426**	0.2664	0.4730	0.6956	0.2504
Long-tailed ($\rho = 20$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.5582	0.3572	**0.7484**	0.1242	0.1660	0.0824	0.2623	0.2909	0.2353
OCD	0.4911	**0.9686**	0.0136	0.3434	0.6056	0.0812	0.3360	0.5928	0.0792
GM	0.1922	0.2762	0.1082	0.1993	0.3366	0.0620	0.1608	0.2244	0.0972
BaCon	0.4715	0.8294	0.1136	0.3009	0.3514	0.2504	0.3528	0.4732	0.2324
MetaGCD	0.4719	0.4354	0.5084	0.3805	0.4028	**0.3582**	0.2952	0.3396	0.2508
Ours (NCD-DLT)	**0.7311**	0.9256	0.5366	**0.4375**	**0.6114**	0.2636	**0.4268**	**0.6024**	**0.2512**
Long-tailed ($\rho = 100$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.5312	0.7586	0.6948	0.1500	0.1878	0.1122	0.2600	0.2708	**0.2509**
OCD	0.4830	**0.9586**	0.0074	0.2869	**0.4934**	0.0804	0.3012	**0.5272**	0.0752
GM	0.1603	0.1824	0.1382	0.1536	0.2404	0.0668	0.1468	0.2292	0.0644
BaCon	0.3817	0.4102	0.3532	0.3154	0.3838	0.2470	0.3126	0.3868	0.2384
MetaGCD	**0.5670**	0.1866	**0.9474**	0.3209	0.3362	**0.3056**	0.2750	0.3308	0.2192
Ours (NCD-DLT)	0.5605	0.8568	0.2642	**0.3342**	0.4144	0.2540	**0.3412**	0.4852	0.1972

Table 4. Experimental results for original and long-tailed datasets with **Strict** Hungarian. The best results are marked in **bold**, and the second best results are marked by underline.

We also compare the training and testing times on CIFAR-100 with $\rho = 100$, as shown in Table 5. NCD-DLT demonstrates relatively low total training time while maintaining superior performance. In particular, when compared to the comparable method, MetaGCD, which requires several hours for both training and testing, NCD-DLT is significantly lighter in terms of both training and inference.

E. Experimental result with k-means clustering on test set

Existing NCD works, such as GCD, MetaGCD, and BaCon, assume that all test samples are available and that the total number of classes is known; thereby, they utilize k-means clustering in the inference step. However, as discussed in Section 5.1, the experimental results reported in the main content are based on tests conducted in an on-the-fly manner, which reflects a more realistic scenario than running k-means clustering for all test samples. While we highlight the efficacy of our inference approach, we also ensure performance comparison under the same scenario used in the literature, implementing k-means clustering by assuming all the test samples are avaiable simultaneously.

The results are shown in Table 6 and Table 7. Similar to the on-the-fly inference approach as shown in Tables 2 and 4, NCD-DLT is comparable or outperforms in all scenarios in terms of 'All' accuracy. This superiority is attributed to NCD-DLT's ability to extract distinct features for different classes, and the integration of the Hash Hamming Graph with the graph-merging algorithm, which provides a more precise measurement of the total number of classes, as discussed in Section 5.3.

F. Comparison on Problem Setting

In Figure 10, we visualize different scenarios to highlight the difficulty of the suggested scenario in real-world circumstances. For example, (a) a scenario where the class distribution is imbalanced within the training data, but training is conducted in a static setting as assumed in BaCon [1]; (b) the inference stage of NCD, aiming to classify both known and novel classes accurately; (c) a dynamic training scenario with a uniform class distribution and data provided incrementally, as assumed in GM [33] and [29]; (d) a challenging scenario with an imbalanced data distribution and data provided incrementally at random (NCD-DLT). This

Time Comsumption (s)	Initial Training	Dynamic Training	Total Training time	Inference
GCD	1227	868	2095	63.15
GM	354	346	700	39.54
BaCon	4401	3751	8152	64.14
MetaGCD	452	83544	83996	2921.86
NCD-DLT	1187	4505	5692	48.06
NCD-DLT (w/o Distillation)	1187	4389	5576	48.06

Table 5. Comparison of training and testing times (in seconds) for NCD-DLT and other methods on CIFAR-100 with $\rho = 100$. NCD-DLT demonstrates competitive performance with significantly lower computational costs, particularly when compared to MetaGCD, which requires substantially more time for both training and testing.

Original Dataset	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.3097	0.3178	0.3016	0.3873	0.4300	0.3446	0.4290	0.4512	0.4068
OCD	0.5198	0.6786	0.3610	0.4702	0.6690	0.2714	**0.5506**	**0.7676**	0.3336
BaCon	0.7561	0.8642	0.6480	0.5426	0.6572	**0.4280**	0.4400	0.4672	**0.4128**
MetaGCD	0.8983	0.8768	0.9198	0.3940	0.4074	0.3806	0.3856	0.4372	0.3340
Ours (NCD-DLT)	**0.9714**	**0.9770**	**0.9658**	**0.6002**	**0.8388**	0.3616	0.5364	0.7288	0.3440
Long-tailed ($\rho = 20$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	**0.8726**	0.8070	**0.9382**	0.2190	0.2298	0.2082	0.3087	0.3393	0.2797
OCD	0.2938	0.3784	0.2092	0.4464	0.5524	0.3404	**0.5206**	**0.6672**	**0.3740**
BaCon	0.7171	0.8272	0.6070	0.4077	0.4790	0.3364	0.4282	0.4960	0.3604
MetaGCD	0.7575	0.7350	0.7800	0.4666	0.4978	**0.4335**	0.3426	0.3536	0.3316
Ours (NCD-DLT)	0.8653	**0.9212**	0.8094	**0.4926**	**0.6790**	0.3062	0.4848	0.6468	0.3228
Long-tailed ($\rho = 100$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.8800	0.8286	0.9314	0.2273	0.2404	0.2142	0.2954	0.3146	0.2792
OCD	0.3619	0.4370	0.2868	0.3931	0.4658	0.3204	**0.4704**	**0.5772**	**0.3636**
BaCon	0.5634	0.5798	0.5470	0.4052	0.4824	0.3280	0.4072	0.4856	0.3288
MetaGCD	0.8470	0.7494	**0.9446**	**0.4711**	0.4882	**0.4540**	0.3212	0.3164	0.3260
Ours (NCD-DLT)	**0.9008**	**0.9372**	0.8664	0.4459	**0.5502**	0.3416	0.3928	0.5272	0.2584

Table 6. Experimental results with k-means clustering for original and long-tailed datasets with **Greedy** Hungarian. The best results are marked in **bold**, and the second best results are marked by underline.

last scenario reflects real-world conditions where unlabeled new data is continuously acquired, the class distribution is imbalanced, and new classes keep emerging, such as in E-commerce.

Original Dataset	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.2303	0.2594	0.2012	0.3193	0.3538	0.2848	0.4150	0.4324	**0.3976**
OCD	0.5198	0.6786	0.3610	0.4611	0.6596	0.2626	**0.5230**	**0.7600**	0.2860
BaCon	0.5815	0.8642	0.2988	0.4664	0.6016	**0.3312**	0.4266	0.4492	0.3960
MetaGCD	0.6438	0.3678	**0.9198**	0.3295	0.3356	0.3234	0.3768	0.4264	0.3272
Ours (NCD-DLT)	**0.7267**	**0.8938**	0.5596	**0.5124**	**0.7592**	0.2656	0.4812	0.7156	0.2468
Long-tailed ($\rho = 20$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	0.5466	0.3490	**0.7442**	0.1423	0.1436	0.1410	0.2582	0.3061	0.2131
OCD	0.2961	0.3620	0.2302	0.4433	0.5524	**0.3342**	**0.5026**	**0.6576**	**0.3476**
BaCon	0.4732	0.8272	0.1192	0.3272	0.4012	0.2532	0.3874	0.4636	0.3112
MetaGCD	0.4802	0.4922	0.4682	0.3705	0.4126	0.3284	0.2966	0.3096	0.2836
Ours (NCD-DLT)	**0.7224**	**0.9212**	0.5236	**0.4470**	**0.6454**	0.2486	0.4414	0.6056	0.2772
Long-tailed ($\rho = 100$)	CIFAR10			CIFAR100			TinyImageNet		
Model	All	Old	New	All	Old	New	All	Old	New
GCD	**0.5520**	0.5226	0.7364	0.1586	0.2314	0.1000	0.2553	0.2792	0.2350
OCD	0.3357	0.4370	0.2344	**0.3860**	0.4658	0.3062	**0.4608**	0.5704	**0.3512**
BaCon	0.3823	0.4114	0.3532	0.3321	0.4318	0.2124	0.3490	0.4352	0.2628
MetaGCD	0.5378	0.3236	**0.7520**	0.3808	0.3850	**0.3766**	0.2766	0.2624	0.2908
Ours (NCD-DLT)	0.5035	**0.7564**	0.2506	0.3657	**0.4746**	0.2568	0.4338	**0.6140**	0.2636

Table 7. Experimental results with k-means clustering for original and long-tailed datasets with **Strict** Hungarian. The best results are marked in **bold**, and the second best results are marked by underline.

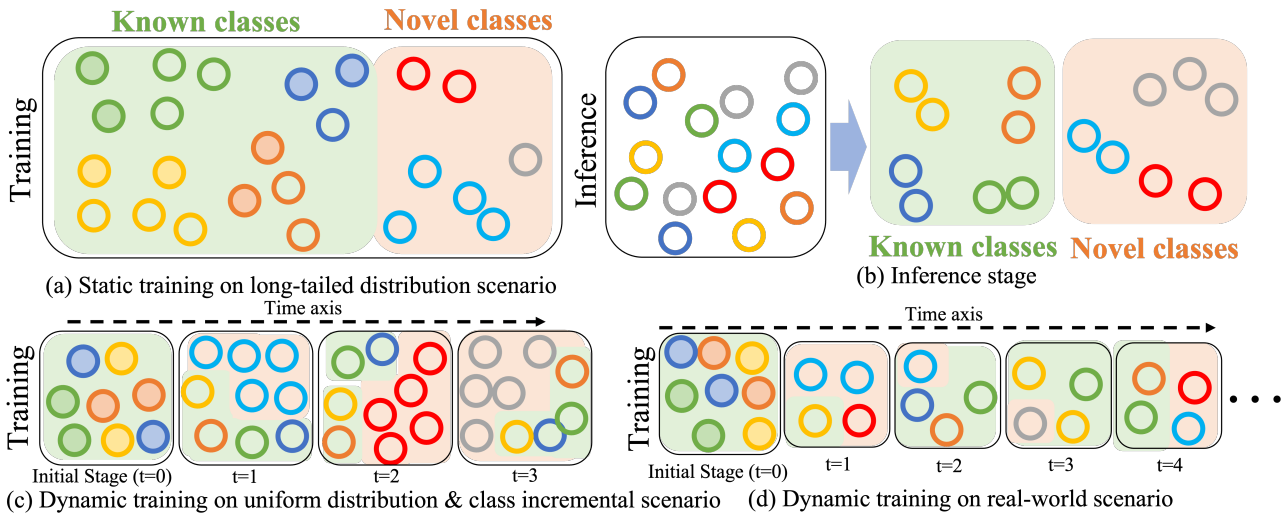

Figure 10. Filled circles represent labeled datasets, while empty circles denote unlabeled instances, with different colors indicating distinct classes. Within the Novel Category Discovery (NCD) framework, some classes' labels might never be provided during training, as illustrated by the orange background, whereas known classes have been trained at least once with labeled samples.