Assignment-Driven Hash Learning in a Hyper-Semantic Space for On-the-Fly Category Discovery

Supplementary Material

Appendix

A. Implementation Details

A.1. Datasets Details

Dataset Details. We evaluate our approach on several benchmarks (see Table A.1), and extend the On-the-Fly Category Discovery (OCD) task to include the iNaturalist 2017 dataset. The iNaturalist 2017 collection, sourced from the citizen science platform iNaturalist, contains 675,170 images for training and validation, covering 5,089 fine-grained categories across 13 overarching classes (e.g., Plantae, Insecta, Aves, Mammalia). The pronounced intra-class variability within these super-categories underscores the difficulty of fine-grained classification under the OCD setting.

Table A.1. Statistics of datasets used in our experiments.

	CUB	Scars	Pets	Animalia	Fungi	Arachnida
$\lvert Y_S\rvert$	100	98	19	39	61	28
$\lvert Y_Q\rvert$	200	196	38	77	121	56
$\lvert \mathcal{D}_S\rvert$	1.5K	2.0K	0.9K	1.5K	1.8K	1.7K
$\lvert \mathcal{D}_Q\rvert$	4.5K	6.1K	2.7K	5.1K	5.8K	4.3K

A.2. Evaluation Metric Details

In accordance with the OCD protocol, we partition the categories of each dataset into *seen* and *unseen* subsets. For the three selected super-categories—Arachnida, Animalia, and Mollusca—we allocate 50% of samples from the seen classes to the labeled training set $\mathcal{D}_S$ and reserve the remaining seen-class samples, together with all samples from unseen classes, in the unlabeled query set $\mathcal{D}_Q$. Table A.1 summarizes the resulting splits, where $|Y_S|$ and $|Y_Q|$ denote the number of seen and query classes, respectively, and $|\mathcal{D}_S|$ and $|\mathcal{D}_Q|$ indicate the corresponding sample counts.

A.3. Algorithm Pipeline for Flexible Prototype Assignment

During each training time, for every class c we maintain two prototype vectors $p_c^{(1)}, p_c^{(2)}$ based on the Hyper-Semantic Space. Given a batch of B_c samples belonging to class c, we first extract their feature embeddings $\{z_i\}_{i=1}^{B_c}$ and stack them into $Z_c = [z_1, \ldots, z_{B_c}] \in \mathbb{R}^{D\times B_c}$. We then compute affinity scores$A_c = \frac{1}{\epsilon} P_c^T Z_c, \quad P_c = [p_c^{(1)}, p_c^{(2)}] \in \mathbb{R}^{D\times 2}$, and apply doubly-stochastic Sinkhorn–Knopp normalization to obtain soft assignment weights $W_c = \mathrm{Sinkhorn}(A_c) \in \mathbb{R}^{2\times B_c}$, $W_c = [w_{c,1}, \ldots, w_{c,B_c}]$, where $w_{c,i} = (w_{c,i}^1, w_{c,i}^2)$ indicates how strongly sample i aligns with each prototype, capturing intra-class variations.

Using these assignments, we compute the weighted mixture likelihood loss $L_{\mathrm{soft-MLE}}$ (Equ. 6) that encourages samples to cluster around their assigned prototypes . In parallel, we perform a prototype-level contrastive step: each prototype $p_c^{(k)}$ generates two augmented views $\hat{p}_c^{(k)}$, $\tilde{p}_c^{(k)}$; we pull together views of the same prototype while pushing apart views from different classes (Equ. 7). Summing these gives the FPA objective L_{FPA} (Equ. 8). After computing gradients, we update network parameters by backpropagation, while prototypes are updated by momentum averaging: $p_c^{(k)} \leftarrow \alpha\, p_c^{(k)} + (1-\alpha)\, \frac{1}{B_c} \sum_{i=1}^{B_c} w_{c,i}^k\, z_i$, normalized to unit norm. This iterative process lets the model learn flex-

Algorithm 1 Flexible Prototype Assignment

Batch features $X = \{x_i\}_{i=1}^{B}$, prototypes $P = \{p_j\}_{j=1}^{K}$, temperature ϵ, Sinkhorn iterations T, EMA momentum m Updated prototypes P and soft assignments $\widetilde{Q}$

Stage 1: Similarity and Unnormalized Assignment 1. Compute similarity matrix $S \in \mathbb{R}^{B\times K}$:

$$S_{ij} \leftarrow x_i \cdot p_j.$$

2. Compute unnormalized assignment scores:

$$Q_{ij} \leftarrow \exp\big(S_{ij}/\epsilon\big).$$

Stage 2: Global Normalization 3. Normalize Q to sum to 1:

$$Q \leftarrow Q \,/\, \sum_{i=1}^{B}\sum_{j=1}^{K} Q_{ij}.$$

Stage 3: Sinkhorn–Knopp Iterations

for $t = 1, \ldots, T$ **do** $Q_{ij} \leftarrow \frac{Q_{ij}}{\left(\sum_{i'=1}^{B} Q_{i'j}\right)/K}$ $Q_{ij} \leftarrow \frac{Q_{ij}}{\left(\sum_{j'=1}^{K} Q_{ij'}\right)/B}$ **Set** $\widetilde{Q} \leftarrow Q$ *doubly-stochastic assignments

Stage 4: Prototype Update (EMA)

for $j = 1, \ldots, K$ **do** 1. Aggregate weighted features:

$$\bar{x}_j \leftarrow \sum_{i=1}^{B} \widetilde{Q}_{ij}\, x_i.$$

2. EMA update of prototype:

$$p_j \leftarrow m\, p_j \;+\; (1-m)\, \bar{x}_j.$$

return $P,\ \widetilde{Q}$

ible, fine-grained prototype representations and improves both intra-class cohesion and inter-class separation. The algorithm appears in Algorithm 1.

A.4. Details of the Compared Methods.

Since the OCD task demands real-time inference and is relatively novel, traditional baselines derived from Novel Category Discovery (NCD) and Generalized Category Discovery (GCD) are not well suited for this setting. Consequently, we adopt the SMILE mode as our primary comparative baseline, alongside three hash-based competitive methods—**SMILE** baseline , Prototypical Hash Encoding **(PHE)**, Ranking Statistics **(RankStat)**, and Winner-take-all **(WTA)**—as well as an online clustering technique, **Sequential Leader Clustering (SLC)**. Below, we provide detailed descriptions of these previous methods:

- **Sequential Leader Clustering (SLC)**: A classical clustering algorithm designed specifically for analyzing sequential data streams, enabling incremental grouping without revisiting previous samples.
- **Ranking Statistics (RankStat)**: This method constructs category descriptors by identifying the top-3 ranked indices within feature embeddings, capturing salient discriminative features.
- **Winner-take-all (WTA)**: WTA encodes categories based on the indices corresponding to the maximum feature values within predefined groups, providing a robust hashing mechanism.

Moreover, the proposed two-stage mechanism enhances the *plug-and-play* capability of existing models. To demonstrate this, we integrate our mechanism with both **SMILE** and **PHE**, showcasing improvements over the original methods. The effectiveness and strong *plug-and-play* nature of our approach are analyzed and validated in Section 4. For clarity and reproducibility, we provide the detailed pseudocode in Alg 2. The three main losses can be incorporated into any OCD method to improve performance(bold in the pseudocode in Alg 2).

Table A.2. Concrete meaning of the different transformation types. "ALL" denotes using rotations of $0°$, $90°$, $180°$ and $270°$.

Name	Rotation	Color permutation	M
2 -Rotation	$0°, 180°$	RGB	2
4 -Rotation	ALL	RGB	4
3 –Color Permutation	–	RGB, GBR, BRG	3
6 –Color Permutation	–	all six channel orders	6
2 –Rot + 3 –Color	$0°, 180°$	RGB, GBR, BRG	6
4 –Rot + 3 –Color	ALL	RGB, GBR, BRG	12
Full Permutation	ALL	all six channel orders	24

A.5. Enhanced SMILE/PHE and Training Details

SMILE:

As discussed in Section 3.2, the enhanced total loss is:

$$\mathcal{L} = \mathcal{L}_{sup} + \mathcal{L}_{reg} + \alpha\cdot\mathcal{L}_{OOD} + \beta\cdot\mathcal{L}_{BHR} + \gamma\cdot\mathcal{L}_{FPA}, \quad \text{(A.1)}$$

where L_{sup} and L_{reg} are basic losses from SMILE. Following our definition, the L_{sup} formulated as:

$$\mathcal{L}_{sup} = -\frac{1}{|P_i|}\sum_{p\in P_i} \log \frac{\exp\big(\mathcal{H}(f(\mathbf{x}_i))\cdot\mathcal{H}(f(\mathbf{x}_p))\big)}{\sum_{\substack{j=1\\ j\neq i}}^{|B|}\exp\big(\mathcal{H}(f(\mathbf{x}_i))\cdot\mathcal{H}(f(\mathbf{x}_j))\big)}. \quad \text{(A.2)}$$

where the P_i is the positive set in a mini-batch and $|B|$ is batch size. Similarly, L_{reg} is formulated as:

$$\hat{\boldsymbol{h}}_i = \operatorname{hash}\left(\mathcal{H}_h\left(f\left(\mathbf{x}_i\right)\right)\right), \quad \text{(A.3)}$$

$$\mathcal{L}_{reg} = -\left|\hat{\boldsymbol{h}}_i\right|. \quad \text{(A.4)}$$

PHE:

Following the notation in Section 3.2, we define the enhanced PHE training objective as

$$\mathcal{L} = \mathcal{L}_p + \lambda_1\cdot\mathcal{L}_c + \lambda_2\cdot\mathcal{L}_f + \alpha\cdot\mathcal{L}_{OOD} + \beta\cdot\mathcal{L}_{BHR} + \gamma\cdot\mathcal{L}_{FPA}, \quad \text{(A.5)}$$

Algorithm 2 Overall Two-Stage Training Pipeline with Loss Integration

Input: Labeled set $\mathcal{L}$, Unlabeled set $\mathcal{U}$, Pretrained ViT backbone M, Projection head H, epochs E, runs R, views V, hyperparameters $\{\eta, m, \lambda\}$.
Output: Fine-tuned model parameters of M and H.

– Prototype Extraction –
`GetPrototypes`$(M, H, \mathcal{L})$:
For each class c in $\mathcal{L}$:
Extract original images and apply augmentations;
Compute features via M, H and normalize;
Average to obtain π_c^{orig} and π_c^{aug}.
return $\{\pi_c^{\text{orig}}, \pi_c^{\text{aug}}\}_c$ and class list.
$(\Pi, C_{\text{known}}) \leftarrow$ `GetPrototypes`$(M, H, \mathcal{L})$.

– Iterative Training –
Initialize optimizer SGD$(\theta_M \cup \theta_H; \eta, m, \lambda)$ and CosineAnnealingLR over E.
Instantiate HashCenterManager.

$e = 1$ **to** E
Training phase:
Set M, H to train mode.
each $(\{x_i\}, y_i) \sim \mathcal{L}$
Generate V views per x_i.
$z_i^v \leftarrow M(x_i^v)$; $(f_i^v, h_i) \leftarrow H(z_i^v)$; normalize f_i^v.

Generate latent OOD prototypes by linear interpolation over Π.
Compute OOD loss $\mathcal{L}_{\text{OOD}}$
Compute FPA loss $\mathcal{L}_{FPA}$.
Compute BHR loss $\mathcal{L}_{BHR}$.
Update hash centers via EMA with h_i.
Initialize $\mathcal{L} \leftarrow$ **other OCD methods**, like $\mathcal{L} \leftarrow \mathcal{L}_{\text{SC}} + 3\,\mathcal{L}_{\text{reg}}$ for **SMILE**, $\mathcal{L} \leftarrow \mathcal{L}_p + \lambda_1 \cdot \mathcal{L}_c + \lambda_2 \cdot \mathcal{L}_f$ for **PHE**.
$\mathcal{L} \leftarrow \mathcal{L} + \alpha\,\mathcal{L}_{\text{OOD}}$.
$\mathcal{L} \leftarrow \mathcal{L} + \beta\,\mathcal{L}_{\text{BHR}}$.
$\mathcal{L} \leftarrow \mathcal{L} + \gamma\mathcal{L}_{\text{FPA}}$.
Backpropagate and update θ_M, θ_H.

where $\mathcal{L}_p$ is the prototype generation loss, $\mathcal{L}_c = \mathcal{L}_{\text{sep}} + \mathcal{L}_q$ is the hash-center optimization loss combining center separation and quantization; $\mathcal{L}_f$ is the discriminative hash encoding loss and α and β are weighting hyperparameters balancing center optimization and hash encoding.

A.6. Data Augmentation Transform

Table A.2 specifies the concrete operations in the transformation set F used to construct the Derived–Known subspace. In particular, "2-Rotation" applies two rotations $(0°, 180°)$, "4-Rotation" applies all four cardinal rotations $(0°, 90°, 180°, 270°)$, "3-Color Permutation" permutes RGB into $\{\text{RGB}, \text{GBR}, \text{BRG}\}$, and "6-Color Permutation" uses all 3! channel orders. Combinations such as "2-Rot+3-Color" and "4-Rot+3-Color" sequentially apply rotations and the three color permutations, yielding 6 or 12 variants, respectively; "Full Permutation" uses all four rotations and six color orders ($4 \times 6 = 24$ variants). Varying the augmentation cardinality M thus enriches each class's prototype pair $(P_i^{\text{par}}, P_i^{\text{aug}})$, enabling the model to capture fine-grained intra-class diversity via Flexible Prototype Assignment and to reserve sufficient feature capacity for novel categories.

B. More details on Flexible Prototype Assignment and updating

B.1. Hyperspherical Mixture-of-Prototypes Model

Let $f_\theta : \mathcal{X} \to \mathbb{R}^E$ be a deep encoder and $g_\varphi : \mathbb{R}^E \to \mathbb{R}^D$ a projection head. Given an input x, we compute

$$h = f_\theta(x), \quad z' = g_\varphi(h), \quad z = \frac{z'}{\|z'\|_2}, \tag{B.1}$$

so that z lies on the unit hypersphere $\mathbb{S}^{D-1}$. We model the embedding distribution of each ID class $c \in \{1, \dots, C\}$ as a mixture of K von Mises–Fisher (vMF) components:

$$p(z \mid c) = \sum_{k=1}^{K} w_k^c \, Z_D(\kappa) \, \exp\big(\kappa \, p_k^{c\top} z\big), \tag{B.2}$$

where $\{p_k^c\}_{k=1}^K \subset \mathbb{S}^{D-1}$ are class-conditional prototypes, $\kappa > 0$ is a concentration parameter, $Z_D(\kappa)$ the normalizer, and w_k^c are nonnegative assignment weights with $\sum_k w_k^c = 1$.

B.2. Soft Prototype Assignment

Given a mini-batch $\{z_i\}_{i=1}^B$ and class-conditional prototypes $P_c = [p_1^c, \dots, p_K^c]$, we compute soft assignment weights $W_c \in \mathbb{R}^{K \times B_c}$ by solving

$$\max_{W_c \in \mathcal{W}} \; \mathrm{Tr}\big(W_c^\top P_c^\top Z_c\big) + \varepsilon \, H(W_c), \tag{B.3}$$

where $Z_c \in \mathbb{R}^{D \times B_c}$ collects all embeddings of class c, $H(W) := -\sum_{i,k} W_{k,i} \log W_{k,i}$ is entropy, and $\mathcal{W}$ enforces $\sum_i W_{k,i} = \frac{1}{K}$ and $\sum_k W_{k,i} = \frac{1}{B_c}$. In practice this yields the closed-form Sinkhorn solution

$$W_c = \mathrm{diag}(u) \, \exp\Big(\tfrac{1}{\varepsilon} \, P_c^\top Z_c\Big) \, \mathrm{diag}(v), \tag{B.4}$$

with u, v computed by a few Sinkhorn–Knopp iterations. We then prune each column of W_c to keep only the top-K' weights.

B.3. Negative Log-Likelihood Loss

Under the mixture model the probability that z_i belongs to class c is

$$p(y_i = c \mid z_i) = \frac{\sum_k w_{i,k}^c \, \exp(p_k^{c\top} z_i / \tau)}{\sum_{c'} \sum_{k'} w_{i,k'}^{c'} \, \exp(p_{k'}^{c'\top} z_i / \tau)}, \tag{B.5}$$

where $\tau = \frac{1}{\kappa}$ is the temperature.

Maximizing the joint likelihood $\prod_i p(y_i \mid z_i)$ is equivalent to minimizing the negative log-likelihood

$$\mathcal{L}_{\mathrm{Soft-MLE}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\sum_k w_{i,k}^{y_i} \, \exp\big(p_k^{y_i\top} z_i / \tau\big)}{\sum_{c'} \sum_{k'} w_{i,k'}^{c'} \, \exp\big(p_{k'}^{c'\top} z_i / \tau\big)}. \tag{B.6}$$

This encourages each z_i to lie near the subset of prototypes $\{p_k^{y_i}\}$.

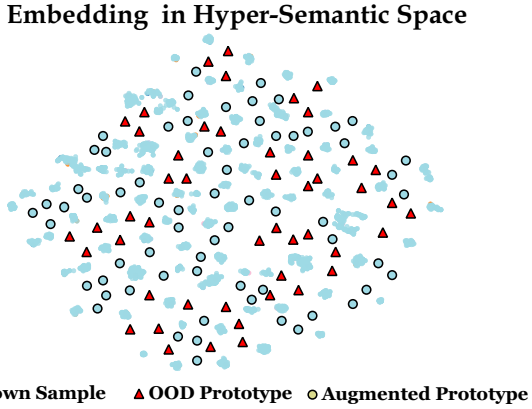

Figure B.1. Additional Examples TSNE for *Hyper-Semantci Space*

B.4. Prototype Contrastive Loss

To enforce separation among prototypes, *FPA* adds a contrastive term on $\{p_k^c\}$. Let $\tilde{p}_i, \tilde{p}_j$ denote any two distinct prototypes; then

$$\mathcal{L}_{\mathrm{proto}} = -\frac{1}{CK} \sum_{i=1}^{CK} \log \frac{\sum_{j:\, c(j)=c(i),\, j \neq i} \exp\big(\tilde{p}_i^\top \tilde{p}_j / \tau\big)}{\sum_{j:\, (c(j) \neq c(i) \,\vee\, j=i)} \exp\big(\tilde{p}_i^\top \tilde{p}_j / \tau\big)}, \tag{B.7}$$

where $c(i)$ is the class of prototype i. This drives same-class prototypes together and distinct-class apart.

B.5. Exponential Moving Average Update

After computing soft weights $w_{i,k}^c$ for all z_i, prototypes are updated by

$$p_k^c \leftarrow \mathrm{Normalize}\Bigg(\alpha \, p_k^c + (1-\alpha) \, \frac{1}{\sum_{i:y_i=c} w_{i,k}^c} \times \sum_{i:y_i=c} w_{i,k}^c \, z_i\Bigg). \tag{B.8}$$

with a high-momentum $\alpha \approx 0.99$. This decouples prototype updates from gradient steps.

C. Additional Experiment

C.1. t-SNE Visualization on Scars

Figure B.1 presents the t-SNE embedding of SCARS features within our Hyper-Semantic Space. Known samples are plotted in blue, augmented prototypes in green circles, and OOD prototypes in red triangles. The augmented prototypes closely surround their corresponding known-class clusters, demonstrating effective modeling of intra-class variations under different transformations. The OOD prototypes occupy intermediate regions between distinct models, reserving semantically coherent areas in the embedding

Table C.1. Computation on CUB.

Metric	SMILE	SMILE+Ours	Δ
Samples	281,600	281,600	**0**
Per-sample FLOPs	1.76×10^{10}	2.11×10^{10}	**0.35**$\times10^{10}$
Total FLOPs	4.95×10^{15}	5.94×10^{15}	**0.99**$\times10^{15}$
Forward time (s)	1,024.80	1,050.82	**26.02**
Backward time (s)	4.57	5.50	**0.93**
Total time (s)	1,029.37	1,056.32	**26.95**
Avg. forward/sample (s)	0.003639	0.003732	**0.000093**
Avg. backward/sample (s)	0.000016	0.000020	**0.000004**

Table C.2. Computation on Stanford Cars

Metric	SMILE	SMILE+Ours	Δ
Samples	384,000	384,000	**0**
Per-sample FLOPs	1.76×10^{10}	2.11×10^{10}	**0.35**$\times10^{10}$
Total FLOPs	6.75×10^{15}	8.10×10^{15}	**1.35**$\times10^{15}$
Forward time (s)	1,389.88	1,414.53	**24.65**
Backward time (s)	7.47	9.34	**1.87**
Total time (s)	1,397.35	1,423.88	**26.53**
Avg. forward/sample (s)	0.003619	0.003684	**0.000065**
Avg. backward/sample (s)	0.000019	0.000024	**0.000005**

space for novel categories. For example, the OOD prototype near the lower-left exhibits combined wheel texture and windshield shape attributes, illustrating how interpolation between two parent prototypes creates a plausible "unknown" model representation. These results confirm that our Hyper-Semantic construction generalizes beyond CUB to Scars, preserving diversity within known classes and reserving capacity for emergent categories.

C.2. Additional Examples for *Hyper-Semantic Space*

In addition to the visualizations presented in Section 4.5, we include further examples illustrating the organization of feature embeddings within our Hyper-Semantic Space in Figure C.1. On the left, original images are mapped to their corresponding positions in the Derived-Known Subspace, surrounded by augmented prototypes that capture fine-grained intra-class variations. On the right, synthesized Out-of-Distribution (OOD) prototypes are shown, interpolated between parent-class prototypes to reserve semantically meaningful regions for novel categories. For instance, the OOD prototype displayed at the top right exhibits a blend of coloration and shape attributes from two known bird species—resulting in a novel olive-brown plumage with a distinct mask-like eye marking—that closely matches the appearance of the unseen Common Yellowthroat. These additional examples corroborate the effectiveness of our Hyper-Semantic Space in both preserving intra-class diversity and allocating capacity for unforeseen categories.

C.3. Justification of the choice of $K = 2$

The choice of $K = 2$ aligns with our "Parent-Derived" design, while soft assignment functions dynamically to capture diversity. We further evaluated K by randomly decomposing the augmentation pool into sub-groups on

Table C.3. Ablation study on the number of prototypes K (All ACC %). Setting $K = 2$ is sufficient to capture diversity while preventing unnecessary redundancy.

Dataset	K=2	K=3	K=4	K=5
CUB	**41.4**	40.8	39.9	40.3
Fungi	**41.0**	40.2	40.5	40.4

Table C.4. Ablation study on key components. We evaluate the impact of removing $\mathcal{L}_{OOD}$, $\mathcal{L}_{FPA}$, and $\mathcal{L}_{HBR}$ on the All ACC (%).

Dataset	w/o $\mathcal{L}_{OOD}$	w/o $\mathcal{L}_{FPA}$	w/o $\mathcal{L}_{HBR}$	Full
CUB	44.0	44.3	43.4	**45.8**
SCars	35.8	35.0	36.8	**38.8**

SMILE+ours. Results in the Tab.C.3 demonstrates $K = 2$ is sufficient to capture diversity without introducing unnecessary redundancy.

C.4. Augmentation Fairness Study

Our Subspace structurally requires paired inputs (P_{par}, P_{aug}) to form the basis, making it inherently coupled with the augmentation. To demonstrate that the improvements achieved by our method stem from our architecture itself and not merely from data augmentation, we conducted more in-depth experiments. ***(1) Fairness:*** We applied $\mathcal{F}$ to generate contrastive views (SMILE) and inputs for prototype generation (PHE). The marginal gains (SMILE: 33.0%/26.8%; PHE: 38.1%/31.6% All ACC on CUB/Cars) confirm that augmentation alone is insufficient without a compatible structure. ***(2) Ablation:*** By degrading $\mathcal{F}$ to standard augmentation, our method still achieves 40.2%(SMILE) and 44.5%(PHE) All ACC on CUB, surpassing baselines. This confirms augmentation is auxiliary, while our holistic design drives the primary gains. ***(3) Effectiveness:*** Tab. 5 shows the change in four metrics with our framework on two baselines, clearly demonstrating the border effectiveness of the framework.

C.5. Ablation Study on PHE

We performed additional ablation on Scars/CUB of *PHE+ours* in Tab.C.4 , which revealed the same conclusion as *SMILE+ours*. This further validates the generalization of our plug-and-play framework.

D. Computational Consumption

Table C.1 and Table C.2 report the per-sample and total FLOPs, forward/backward runtimes, and average runtimes on CUB and Stanford Cars (SCARS), comparing the original SMILE baseline with SMILE enhanced by our three

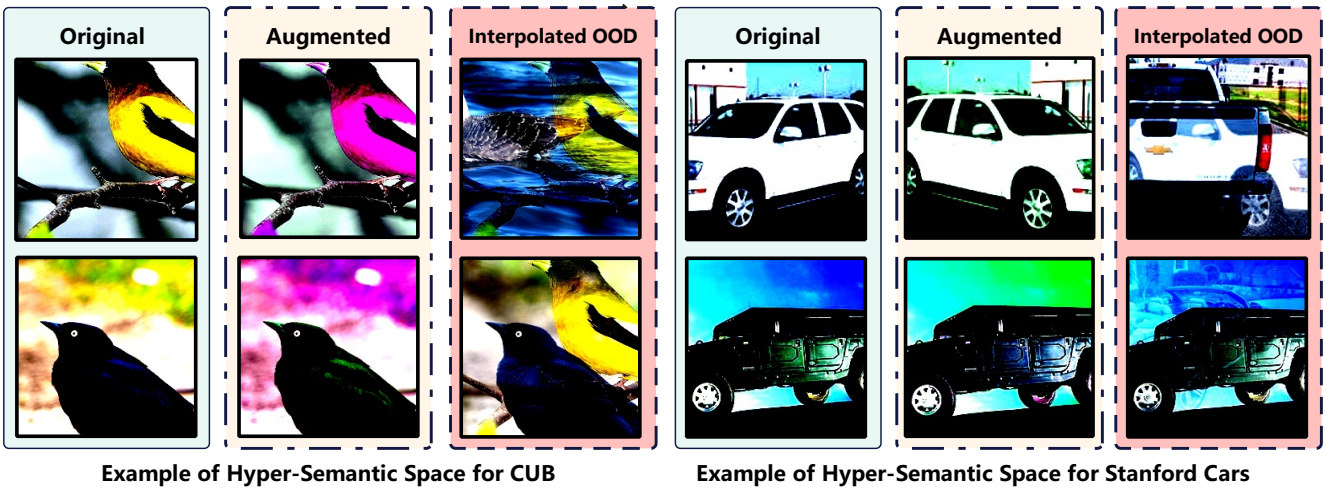

Figure C.1. Additional Examples for *Hyper-Semantci Space*

modules (FPA, OOD, BHR).

On CUB, SMILE+Ours incurs an additional 0.35×10^{10}FLOPs per sample (from 1.76×10^{10} to 2.11×10^{10}), resulting in 0.99×10^{15} extra FLOPs over 281,600 samples. Forward pass time increases by 26.02s (2.5% relative) and backward pass by 0.93s (20.4%), yielding a total training overhead of 26.95s (2.6%). Per-sample latency grows by 0.093ms forward and 0.004ms backward. On SCARS, the per-sample FLOPs increase is identical (0.35×10^{10}), yielding 1.35×10^{15} extra FLOPs over 384,000 samples. Forward time rises by 24.65s (1.8%) and backward time by 1.87s (25.0%), for a total training overhead of 26.53s (1.9%). Per-sample forward/backward latency increases by 0.065ms and 0.005ms, respectively.

Despite these overheads, the added modules deliver consistent All/Old/New ACC gains of up to 12.8% on CUB and 7.9% on SCARS (see Section 4), while the total training-time increase remains below 3%. The majority of added cost stems from the OOD distance computations and Sinkhorn normalization in FPA; hash-center updates contribute only marginal extra ops. These results demonstrate that our hyper-semantic and assignment-driven enhancements impose modest computational burdens while substantially improving On-the-fly category discovery performance.

Despite the apparent complexity of maintaining both parent and augmented prototypes in the *Hyper-Semantic Space*, the total number of prototypes per class remains fixed at $2K$, where K denotes the number of known classes. Consequently, the additional computational overhead for prototype storage and updates is bounded by $O(2K \times D)$, which is negligible compared to the base network's cost. This modest expense yields substantial improvements in On-the-fly Category Discovery performance.

In summary, the additional computational consumption can condlude as follow:

- **Increased Training Time.** On CUB, the addition of *Flexible Prototype Assignment (FPA)*, *Out-of-Distribution (OOD) margin*, and *Binary Hash Regularization (BHR)* modules increases total training time by 26.95s (from 1,029.37s to 1,056.32s), a 2.6% overhead for 281,600 samples (Table C.1). On Stanford Cars (SCARS), the overhead is 26.53s (from 1,397.35s to 1,423.88s), a 1.9% increase (Table C.2).
- **Additional FLOPs.** The per-sample FLOPs grow by 0.35×10^{10} (from 1.76×10^{10} to 2.11×10^{10}), adding 0.99×10^{15} extra FLOPs over CUB and 1.35×10^{15} over SCARS. Although this represents only a 20% relative increase per sample, it remains small compared to the base network's total operations.
- **Memory and Storage Overhead.** Maintaining two prototypes per class and 10 OOD prototypes requires storing $O(2K + 10) \times D$ feature vectors, which increases peak GPU memory by approximately 3%, consistent with our earlier profiling.

E. Limitation

While our proposed Assignment-Driven Hash Learning framework integrates effectively with existing On-the-Fly Category Discovery (OCD) methods and yields consistent accuracy improvements, it introduces several non-negligible computational costs that merit consideration.

These overheads pose practical constraints on large-scale or real-time deployments and motivate future work on efficient approximations, selective module activation, or mixed-precision strategies to further reduce training latency and memory footprint.

F. Related Work

F.1. Novel Category Discovery (NCD)

The original Novel Category Discovery (NCD) approach, introduced by *Deep Transfer Clustering* (DTC) [7], was designed to leverage knowledge from known categories during training to distinguish novel categories during testing. Its generalized extension, Generalized Category Discovery (GCD) [12, 15, 20, 24], expanded this framework by incorporating both old and new categories during testing, creating a more realistic and challenging scenario where models must actively determine whether instances belong to known or novel categories. Several successful methods have been proposed: SPTNet [22] proposes a two-stage strategy that combines global and spatial prompts to further fine-tune the SimGCD model; HiLo [21] decouples domain and semantic features, uses PatchMix with curriculum sampling, and robustly discovers seen and novel classes under domain shifts; and HypCD [12] embeds features in hyperbolic space and uses hybrid contrastive learning to capture hierarchies for robust generalized category discovery. While these NCD/GCD [2, 4, 6, 9, 13, 16–19, 21, 26, 28] approaches demonstrate encouraging performance, yet they still suffer from two core drawbacks: (1)These methods depend extensively on a fixed query set (the unlabeled data) throughout training, restricting their capacity to adapt to genuinely unseen instances and impairing generalization. (2)The offline batch-based processing of query data during inference renders these models unsuitable for real-time applications where inputs arrive sequentially and immediate predictions are needed. To address these constraints, Du et al. [3] proposed On-the-fly Category Discovery (OCD), which further extends the paradigm by requiring models to: (1) learn solely from known categories during training without any novel class labels, and (2) during testing, both identify novel categories and incorporate real-time feedback to update the model dynamically. This formulation presents significantly greater challenges for model training and adaptation, particularly in maintaining stability while continuously integrating new knowledge.

F.2. Deep Hashing

Deep hashing has emerged as an efficient approach for mapping high-dimensional data to compact binary codes, where similar objects are represented by proximate hash codes while dissimilar ones are separated in the Hamming space. Early methods such as HashNet [1] employed pairwise similarity measures, while DPSH [10] and DSH [11] adopted triplet-based objectives to optimize hash functions. Although these approaches aimed to learn more refined mapping functions, they often required substantial computational resources and faced convergence challenges. Recent advancements, including DPN [5], CSQ [25], and OrthoHash [8], have introduced hash centers to ensure both discriminative and well-distributed binary codes. ConceptHash [14] enables interpretable fine-grained hashing by associating sub-codes with visual concepts, Wei *et al.* [23] explore hierarchical information in hyperbolic space for semantic-aware image hashing. SMILE [3] and PHE [27] leverage hash codes for On-the-Fly Category Discovery (OCD) tasks to dynamically identify categories. These methods benefit from the effectiveness of hash codes in distinguishing categories.

References

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[2] Fernando Julio Cendra, Bingchen Zhao, and Kai Han. Promptccd: Learning gaussian mixture prompt pool for continual category discovery. In *European conference on computer vision*, pages 188–205. Springer, 2024.

[3] Ruoyi Du, Dongliang Chang, Kongming Liang, Timothy Hospedales, Yi-Zhe Song, and Zhanyu Ma. On-the-fly category discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11691–11700, 2023.

[4] Jianan Fan, Dongnan Liu, Hang Chang, Heng Huang, Mei Chen, and Weidong Cai. Seeing unseen: Discover novel biomedical concepts via geometry-constrained probabilistic modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11524–11534, 2024.

[5] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised learning of accurate binary hashing codes. In *IJCAI*, 2020.

[6] Wei Feng and Zongyuan Ge. Generalized category discovery under domain shift: A frequency domain perspective. *arXiv preprint arXiv:2511.00573*, 2025.

[7] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8401–8409, 2019.

[8] Jiun Tian Hoe, Kam Woh Ng, Tianyu Zhang, Chee Seng Chan, Yi-Zhe Song, and Tao Xiang. One loss for all: Deep hashing with a single cosine similarity based learning objective. *Advances in Neural Information Processing Systems*, 34:24286–24298, 2021.

[9] Hyungmin Kim, Sungho Suh, Daehwan Kim, Daun Jeong, Hansang Cho, and Junmo Kim. Proxy anchor-based unsupervised learning for continuous generalized category discovery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16688–16697, 2023.

[10] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[11] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2064–2072, 2016.

[12] Yuanpei Liu, Zhenqi He, and Kai Han. Hyperbolic category discovery. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9891–9900, 2025.

[13] Shijie Ma, Fei Zhu, Zhun Zhong, Wenzhuo Liu, Xu-Yao Zhang, and Cheng-Lin Liu. Happy: A debiased learning framework for continual generalized category discovery. *Advances in Neural Information Processing Systems*, 37:50850–50875, 2024.

[14] Kam Woh Ng, Xiatian Zhu, Yi-Zhe Song, and Tao Xiang. Conceptthash: Interpretable fine-grained hashing via concept discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1211–1223, 2024.

[15] Nan Pu, Zhun Zhong, and Nicu Sebe. Dynamic conceptional contrastive learning for generalized category discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[16] Sarah Rastegar, Hazel Doughty, and Cees Snoek. Learn to categorize or categorize to learn? self-coding for generalized category discovery. *Advances in Neural Information Processing Systems*, 36:72794–72818, 2023.

[17] Vaibhav Rathore, Saikat Dutta, Sarthak Mehrotra, Zsolt Kira, Biplab Banerjee, et al. When domain generalization meets generalized category discovery: An adaptive task-arithmetic driven approach. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4905–4915, 2025.

[18] Vaibhav Rathore, Divyam Gupta, and Biplab Banerjee. Hidisc: A hyperbolic framework for domain generalization with generalized category discovery. *arXiv preprint arXiv:2510.17188*, 2025.

[19] Grzegorz Rypeść, Daniel Marczak, Sebastian Cygert, Tomasz Trzciński, and Bartłomiej Twardowski. Category adaptation meets projected distillation in generalized continual category discovery. In *European Conference on Computer Vision*, pages 320–337. Springer, 2024.

[20] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7492–7501, 2022.

[21] Hongjun Wang, Sagar Vaze, and Kai Han. Hilo: A learning framework for generalized category discovery robust to domain shifts. *arXiv preprint arXiv:2408.04591*, 2024.

[22] Hongjun Wang, Sagar Vaze, and Kai Han. Sptnet: An efficient alternative framework for generalized category discovery with spatial prompt tuning. *arXiv preprint arXiv:2403.13684*, 2024.

[23] Rukai Wei, Yu Liu, Jingkuan Song, Yanzhao Xie, and Ke Zhou. Exploring hierarchical information in hyperbolic space for self-supervised image hashing. *IEEE Transactions on Image Processing*, 33:1768–1781, 2024.

[24] Xin Wen, Bingchen Zhao, and Xiaojuan Qi. Parametric classification for generalized category discovery: A baseline study. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[25] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3083–3092, 2020.

[26] Bingchen Zhao and Oisin Mac Aodha. Incremental generalized category discovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19137–19147, 2023.

[27] Haiyang Zheng, Nan Pu, Wenjing Li, Nicu Sebe, and Zhun Zhong. Prototypical hash encoding for on-the-fly fine-grained category discovery. *Advances in Neural Information Processing Systems*, 37:101428–101455, 2024.

[28] Jiaying Zhou, Yang Liu, and Qingchao Chen. Novel class discovery in chest x-rays via paired images and text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7650–7658, 2024.