Make Prototypes Perform Again: Prior-Prototypes Based Feature learning Framework for Few-Shot Hashing

Anonymous ICME submission

***Abstract*—Deep hashing methods typically rely on high-quality feature embeddings to generate compact hash codes that preserve the original semantic information. However, in supervised learning, the feature extractor can only capture the prior distribution of the training set, which results in significant performance degradation when confronted with unknown categories. To address this challenge, we propose a Prior-Prototypes based feature learning framework (PP framework) for Few-Shot hashing. Specifically, we introduce a novel Prior-Prototypes encoder (PP encoder) that generates PP embeddings by computing the differences between new samples and Prior Prototypes, thereby avoiding direct reliance on feature representations from pre-trained extractors. Furthermore, to enhance the hashing function's ability to capture subtle features, we design a Self-Consistency Diffusion Module (SCDM) that imposes self-consistency constraints during the hashing learning, and improved information exchange among samples. Extensive experiments on three benchmarks and four hashing objectives show that our method outperforms existing approaches.**

***Index Terms*—Few Shot Learning, Hashing Learning, Prototype, Diffusion Mechanism, Self consistency**

I. Introduction

With the explosive proliferation of information, multimedia computing has demonstrated vast potential across domains such as social media, healthcare, and digital agriculture. In response to the escalating demands of dense retrieval, deep hashing methods have attracted considerable attention due to their efficient and compact encoding capabilities. The quality of hash codes is intrinsically linked to the quality of feature embeddings extracted by deep models: ideally, images of the same class should cluster tightly in feature space, while those of different classes remain well-separated. This concept forms the foundation of information retrieval. Consequently, existing deep hashing approaches primarily focus on designing powerful feature extractors [1]–[3] and optimizing metric objectives [4]–[7] to maximally utilize the potential of training data.

However, supervised training is inherently constrained to enabling the feature extractor to capture the prior distribution of the training set, which makes it difficult to generate effective feature embeddings when encountering novel categories. Additionally, privacy concerns and high costs make obtaining labels challenging, further significantly constraining the performance of deep hashing models in open-world scenarios.

Fortunately, while research on Few-Shot Hashing Learning (FSHL) is sparse, Few-Shot Learning (FSL) has a strong foundation in computer vision. As illustrated in Fig. 1(1), a typical FSL paradigm separates the training and adapting stages [8]. During adaptation, probabilistic classification is transformed into a similarity evaluation between samples and class prototypes. This approach is highly adaptable and can be extended to hashing tasks, where the similarity is measured by the Hamming distance, as shown in Fig. 1(2). However, overcoming the limitations of feature encoding remains a challenge, particularly when encountering unknown categories, as the encoder is constrained by the limited diversity of the training data. This raises a fundamental question: **How can existing deep models be leveraged to generate high-quality feature embeddings, ensuring robust and efficient hash codes even with limited supervision?**

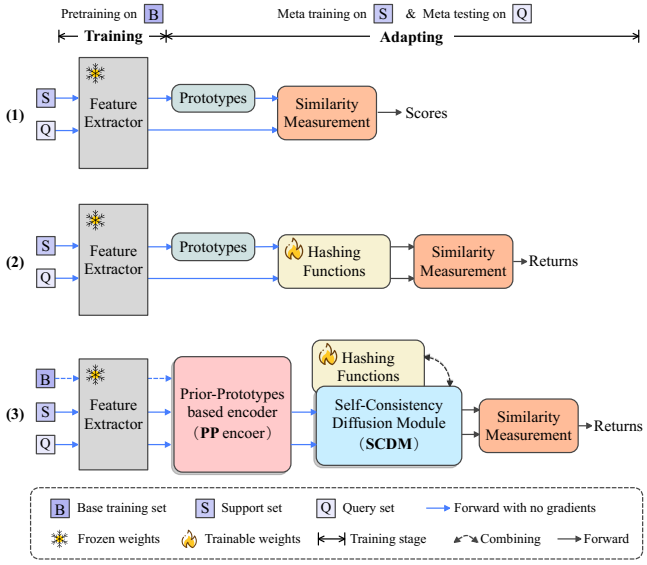


Fig. 1. FSL separates the training and adapting into two independent stages, with the extractor's well-trained weights frozen in the latter [8]: (1) FSL computes class prototypes from the support set and uses a similarity measure to assign query samples to the most similar prototype. (2) FSHL extends FSL by introducing a hash function for hashing retrieval tasks. (3) Our proposed PP framework employs a PP encoder to mitigate the degradation of the feature extractor's performance when handling unseen categories, and a SCDM to improve the hash function through incorporating a reconstruction objective.

To address this challenge, we proposes a novel Prior-Prototypes based feature learning framework (PP framework) for FSHL, as depicted in Fig. 1(3). The framework consists of two key components: a Prior-Prototypes based encoder (PP encoder) and a Self-Consistency Diffusion Module (SCDM).

Specifically, inspired by the concept of "genus and differentia," we construct the PP encoder to characterize unknown categories based on their metric differences from known ones. It generates Prior-Prototypes using the base training set and derives feature embeddings for novel samples by measuring their discrepancies from these prototypes, referred to as PP embeddings.

Additionally, building on the self-consistency hypothesis [9], we design the SCDM to integrate seamlessly with hash functions. By incorporating reconstruction constraints, SCDM improves the quality of hash codes by refining their sensitivity to fine-grained feature differences. The diffusion mechanism within SCDM further promotes sample collisions, enhancing information exchange and refining feature encoding.

We summarize our contributions as follows:

- We propose a Prior-Prototypes based feature learning framework (PP framework) for FSHL, leveraging a PP encoder to generate PP embeddings, thereby mitigating the degradation of feature extractors when handling novel categories.
- We design a Self-Consistency Diffusion Module (SCDM) that improves the hash function's ability to capture fine-grained feature differences and generate higher-quality hash codes.
- Extensive experiments on three widely used few-shot benchmarks demonstrate that our proposed method significantly outperform mainstream approaches.

II. Related Work

A. Deep Hashing Learning

The objective of deep hashing learning is to develop an end-to-end hashing function that compresses continuous feature representations into discrete binary hash codes, reducing storage costs and enabling efficient nearest neighbor search via Hamming distance computations [10]. Key challenges in this process include designing powerful feature extractors [1], [3] and developing robust metric objectives [4]–[6], [11], [12], which have garnered significant attention. Recently, hashing methods have increasingly incorporated multimodal information and leveraged large models to enhance retrieval performance [7]. However, ensuring the performance of hashing models in the face of unknown scenarios and categories, especially without access to additional data or large model interventions, has not been sufficiently explored, with only a few studies [13], [14] addressing this challenge.

B. Few Shot Learning

Few-Shot Learning (FSL) is a challenging paradigm that aims to perform tasks with extremely limited labeled data. It is typically denoted as N-way-K-shot, where N represents the number of new categories, and K indicates the number of available training samples per category, usually 1 or 5. Existing methods often adopt a prototypical approach as the core paradigm [15]–[17], which transforms the traditional category prediction problem into a similarity measurement between new samples and category prototypes, demonstrating strong stability and generalization performance.

III. Methodology

A. Problem Setting

FSHL aims to learn a hash function $h_\theta : \mathbb{R}^d \to \mathbb{R}^b$ that can accurately retrieve new classes based on a pre-trained feature extractor $f_\theta : \mathbb{R}^D \to \mathbb{R}^d$, where d represents the dimension of the feature embedding and b denotes the length of the hash code.

Building on previous work, we divide the FSHL process into two stages: training ($\mathcal{S}t^{\text{train}}$) and adapting ($\mathcal{S}t^{\text{adapt}}$), and split the dataset into two non-overlapping subsets: the base train dataset $\mathcal{D}^{\text{train}} = \{(x_n, y_n)\}_{n=1}^{N_{\text{train}}}$ and the adapt dataset $\mathcal{D}^{\text{adapt}} = \{(x_m, y_m)\}_{m=1}^{N_{\text{adapt}}}$, where $x_n, x_m \in \mathbb{R}^D$ represent the n-th and m-th images, respectively. We ensure that the label sets from the base and adapt datasets are not duplicate: $\mathcal{Y}^{\text{base}} \cap \mathcal{Y}^{\text{adapt}} = \emptyset$.

Training stage $\mathcal{S}t^{\text{train}}$ aims to train the feature extractor f_θ using a classification loss, denoted as $f_\theta = \mathcal{S}t^{\text{train}}\left(\mathcal{D}^{\text{train}}\right)$, ensuring that the sample embeddings are well separated in the feature space.

Adapting stage $\mathcal{S}t^{\text{adapt}}$ uses multiple meta-tasks designed to evaluate the effectiveness of the adaptation algorithm $\mathcal{A}^{\text{adapt}}$. Each meta-task includes a support set $\mathcal{S}_i \subseteq \mathcal{D}^{\text{adapt}}$ and a query set $\mathcal{Q}_i \subseteq \mathcal{D}^{\text{adapt}}$, where i indicates the meta-task index. $\mathcal{S}_i = \left\{\left(x_j^s, y_j^s\right)\right\}_{j=1}^{K \cdot N_i}$ contains K samples per class across N_i classes, while $\mathcal{Q}_i = \left\{\left(x_j^q, y_j^q\right)\right\}_{j=1}^{M \cdot N_i}$ consists of M query samples from the same classes. In meta-training, we freeze the parameters of the feature extractor f_θ as f_θ^* and use $\mathcal{S}_i$ to learn the hash function h_θ :$h_\theta^{\mathcal{S}_i} = \mathcal{A}^{\text{adapt}}\ (h_\theta, \mathcal{S}_i)$.

Once $h_\theta^{\mathcal{S}_i}$ converges, it generates hash codes for both the support set and the query set, denoted as $\mathcal{C}_s = \left\{h_\theta\left(x_j^s\right)\right\}_{j=1}^{K \cdot N_i}$, and $\mathcal{C}_q = \left\{h_\theta\left(x_j^q\right)\right\}_{j=1}^{M \cdot N_i}$, respectively. Next, we compute the retrieval evaluation metric, mean average precision (mAP), based on the Hamming distance between $\mathcal{C}_s$ and $\mathcal{C}_q$.

Final evaluation is performed based on the average performance across all meta-tasks in $\mathcal{D}^{\text{adapt}}$. The results are presented as the average mAP, along with the 95% confidence interval.

B. Framework Architecture

As introduced in Section I, this paper focuses on effectively utilizing pre-trained feature extractors without relying on any auxiliary data or models. Therefore, the proposed PP framework is specifically designed for $\mathcal{S}t^{\text{adapt}}$. As illustrated in Fig. 2, the framework comprises two key modules: a Prior-Prototypes based encoder (PP encoder) and a Self-Consistency Diffusion Module (SCDM).

C. Prior-Prototype based encoder

Motivation. In scenarios with extremely limited supervised information for unknown categories, it is almost impossible for deep models to identify key patterns (or critical feature representations) through iterative training. Inspired by the Aristotelian method of defining objects using *genus and*

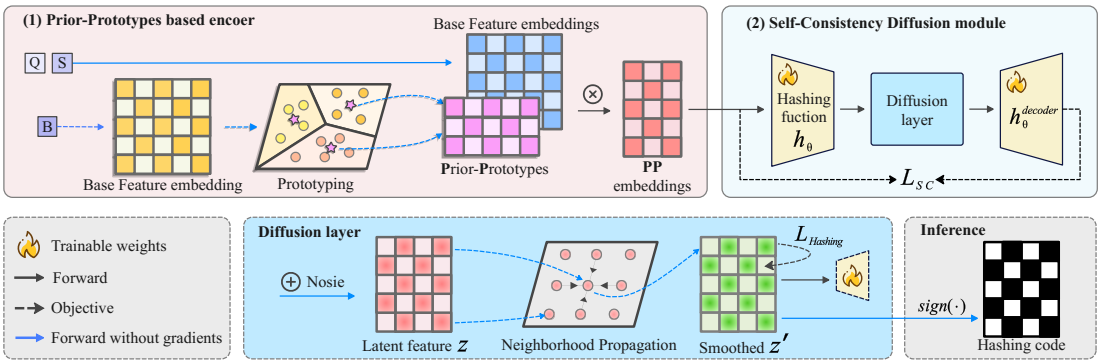


Fig. 2. Overview of PP framework. It uses PP encoder to generate PP embeddings from the base feature embeddings extracted by the feature extractor, and SCDM to enhance the hashing function by integrating an autoencoder (AE) and a diffusion layer.

differentia (e.g., a zebra is a horse with black-and-white stripes, where "horse" serves as the genus and "black-and-white stripes" the differentia), this paper introduces a Prior-Prototypes Based Encoder.

Details. As illustrated in Figure 2(1), PP encoder first utilizes the pre-trained feature extractor f_θ^* and the base train dataset $\mathcal{D}^{\text{train}}$ to generate prototype representations for base categories, referred to as Prior-Prototypes, which act as the "genus." It then computes the differences between unknown samples and each Prior-Prototype as the "differentia". These computed differences across all Prior-Prototypes form new feature representations for $\mathcal{D}^{\text{adapt}}$, referred to as PP embeddings.

a) Prior-Prototype: Genus represents the abstracted knowledge (i.e., distinctive patterns for categories) learned during the $\mathcal{S}t^{\text{train}}$ and is inherently embedded within the pre-trained f_θ^*. This feature extractor projects base samples into a latent space that captures the semantic distinctions among categories:

$$\mathcal{E}_{\mathcal{D}^{\text{train}}} = \left[f_\theta^*\left(x_1\right), f_\theta^*\left(x_2\right), \ldots, f_\theta^*\left(x_{N_{\text{train}}}\right)\right]^\top \in \mathbb{R}^{N_{\text{train}} \times d}, \tag{1}$$

where f_θ^* denotes f_θ with frozen weights, and $\mathcal{E}_{\mathcal{D}^{\text{train}}}$ represents the sample embeddings in the feature space.

Based on $\mathcal{E}_{\mathcal{D}^{\text{train}}}$, the prototype for each base class is computed as the cluster center of the respective class:

$$\mathbf{p}_c = \frac{1}{|\mathcal{C}_c|} \sum_{x_n \in \mathcal{C}_c} f_\theta^*\left(x_n\right), \quad c \in \mathcal{Y}^{\text{base}}, \tag{2}$$

where $\mathcal{C}_c$ denotes the set of samples belonging to category c, and $\mathbf{p}_c$ is the prototype of c.

Note that, $\mathcal{D}^{\text{train}}$ is only used during the pre-training phase for $\mathcal{S}t^{\text{train}}$ $\left(\mathcal{D}^{\text{train}}\right)$ and remains unseen during the $\mathcal{S}t^{\text{adapt}}$. Since $\mathbf{p}_c$ is fixed and does not update, we refer to it as a Prior-Prototype, which is associated with each category. Acting as the "genus" of each base class, Prior-Prototypes provide embedding anchors for generating adaptive feature representations when facing unknown categories during the adaptation stage.

b) Differences Computing: Differentia characterizes the specific differences between unknown categories and each Prior-Prototype $\mathbf{p}_c$ ("genus"), enabling the clear definition of a new sample $x_m \in \mathcal{D}^{\text{adapt}}$. Euclidean distance is used to quantify these differences.

First, the base feature representation of x_m is extracted using f_θ^*:

$$\mathcal{E}_{\mathcal{D}_{\text{adappt}}} = \left[f_\theta^*\left(x_1\right), f_\theta^*\left(x_2\right), \ldots, f_\theta^*\left(x_{N_{\text{adapt}}}\right)\right]^\top \in \mathbb{R}^{N_{\text{adept}} \times d}, \tag{3}$$

where N_{adapt} denotes the number of samples in $\mathcal{D}^{\text{adapt}}$, and d is the feature dimension.

Next, the Euclidean distance between each new sample x_m and all base class prototypes is computed as:

$$d_m^c = \left\| f_\theta^*\left(x_m\right) - \mathbf{p}_c \right\|_2, \tag{4}$$

where $||\cdot||_2$ denotes the Euclidean distance.

Finally, the distances of a new sample to all Prior-Prototypes are aggregated to form its PP embedding:

$$x_m^{pp} = \left[d_m^1, d_m^2, \ldots, d_m^{N_C}\right] \in \mathcal{E}_{\mathcal{D}^{\text{adapt}}}^{pp}, \tag{5}$$

where N_C is the number of base categories.

The essence of the PP encoder lies in avoiding direct use of the base feature embeddings extracted by f_θ^*. Instead, it uses the metric differences between new samples and Prior-Prototypes as new feature embeddings. In doing so, PP embeddings bypass the failure of f_θ^* to generalize to unknown categories. This process is computationally efficient, as it relies entirely on matrix operations.

D. Self-Consistency Diffusion Module

Motivation. In scenarios with limited supervision information, a simple hashing optimization objective often struggles to capture subtle differences in feature embeddings among samples. These differences may implicitly encode critical information for semantic discrimination. To address this challenge, we propose the Self-Consistency Diffusion Module (SCDM).

Details. As illustrated in Figure 2(2). SCDM consists of two plug-and-play components: an autoencoder (AE) based on the consistency hypothesis and a diffusion layer designed for information aggregation.

a) Diffusion Mechanism: For the latent feature $z_m = h_\theta(x_m^{pp})$, we design the Diffusion Layer. It facilitates collisions between samples through an information diffusion mechanism, thereby enhancing the information exchange between samples and their neighbors. The resulting z'_m contains more patterns or perturbations, which improves the robustness of the hashing function h_θ to variations in the input data. Importantly, the Diffusion Layer does not introduce any additional parameters or labels, ensuring that the model's scale remains unchanged and avoiding the risk of data leakage.

Specifically, we perform a smoothing operation on z_m using the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the degree matrix, representing the total connection weight of each sample, and $\mathbf{W}$ is the weight matrix, updated based on Euclidean distances. The updated latent features are computed as follows:

$$z'_m = z_m - \alpha \cdot \mathbf{L} \cdot z_m, \tag{6}$$

where α is the step size, $\mathbf{L}$ is the Laplacian matrix, z_m represents the latent representation of sample m, and z'_m is the smoothed latent representation after diffusion.

b) Self Consistency: A good hashing code should contain sufficient information to recover the data prior to compression [9]. Existing works [18] have associated hashing codes with pixel-level information of images, attempting to reconstruct image details.

In our work, we design a mirrored decoder module $h_\theta^{\text{decoder}}$ based on the hashing function h_θ, forming an encoder-decoder structure. However, the reconstruction objective is not the original image but rather the PP embeddings. This reconstruction objective is formulated as follows:

$$\mathcal{L}_{\text{SC}} = \|\hat{z}'_m - x_m^{pp}\|_2 . \tag{7}$$

where $\hat{z}'_m = h_\theta^{\text{decoder}}(z'_m)$ represents the decoded output obtained from the latent feature z'_m after the diffusion layer. The optimization objective in Eq. 7 aims to minimize the reconstruction error, forcing the hashing function h_θ to learn fine-grained features in the PP embeddings.

IV. Experiments

A. Experimental Details

Datasets. Following the setup of existing works [19], we use three benchmarks and perform data partitioning (number of classes for $\mathcal{D}^{\text{train}}$, number of classes for $\mathcal{D}^{\text{adapt}}$): *mini*ImageNet (64, 20), *tiered*ImageNet (351, 160), and CUB-200 (100, 50). *mini*ImageNet and *tiered*ImageNet are subsets of ImageNet-1K, while CUB-200 is a fine-grained bird classification dataset.

FSHL Settings. We conduct retrieval verification in the N-way-1-shot setting following existing works [13]. Experimental results are presented on four state-of-the-art hash learning objectives (DSDH [4], HashNet [20], DPH [21], CSQ [22]) and two prototype-based few-shot methods (SimpleShot [15], Laplacian [16]).

Hyperparameter Details. All experiments use pre-trained ResNet-18 and WideResNet as feature extractors and are conducted on a single RTX 3090 GPU. During adaptation, 1,000 meta-tasks are randomly initialized, each with 100 epochs. The learning rates are set to 1 for the prototype and Laplacian methods, and 0.1 for ours. All categories of the base training data are used.

B. Results of Few Shot Hashing Retrieval

Setting. To validate the performance of our proposed PP framework, we conducted extensive experiments with a 64-bit hash code length. Given the limited attention to FSHL, we chose the prototype and Laplacian methods for quantitative comparison, as they have shown strong performance and robustness in FSL.

Results. The experimental results for FSHL retrieval are summarized in Table I (The experimental results with 16-bit and 32-bit hash codes are provided in *Supplementary Material A*.), and it shows that our PP framework achieves the best performance across various settings, which demonstrates that our method effectively addresses the encoding failure of the feature extractor when handling unknown categories and ensures the hash function's effectiveness.

C. Results of Cross Datasets

Setting. To further evaluate the performance of the proposed PP framework across different datasets, we conducted experiments using the 5-way-1-shot setting and the DSDH hashing objective, with three different hash code lengths. Since both the *mini* and *tiered* datasets are subsets of ImageNet, we designed four cross-dataset setups: *mini* → CUB (1), *tiered* → CUB (2), CUB → *mini* (3), and CUB → *tiered* (4).

Results. As shown in Fig. 3, the experimental results align with those presented in the main text. The Laplacian method demonstrated improvements over SimpleShot across all four cross-dataset setups. Our proposed PP framework outperformed existing methods across all three hash code lengths. With comparable 95% confidence intervals, the following average improvements were observed: 5.07% (ResNet18) and 5.15% (WideResNet) for setup (1); 7.22% (ResNet18) and 7.5% (WideResNet) for setup (2); 3.8% (ResNet18) and 3.94% (WideResNet) for setup (3); and 3.83% (ResNet18) and 3.87% (WideResNet) for setup (4). (Due to space limitations, the experimental results using WideResNet as the feature extractor are provided in the *Supplementary Material. B*.) These results demonstrate that the PP framework effectively generates more informative feature embeddings by leveraging the differences between new categories and prior prototypes, especially when facing larger disparities between categories.

Additionally, we observed that using CUB for pretraining the feature extractor led to poorer overall performance. This is because CUB is a fine-grained bird classification dataset, where the class distinctions are relatively subtle, limiting the number of discernible patterns the model can learn. As a result, when encoding new samples from categories with larger differences, the risk of encoding failure increases.

Furthermore, although the base training set used for pretraining in *mini* contains fewer categories than CUB, the semantic differences between the categories are larger, providing

TABLE I

COMPARISON OF MAP (%) WITH 95% CONFIDENCE INTERVAL BETWEEN THE PP FRAMEWORK AND OTHER METHODS, UNDER DIFFERENT HASHING LEARNING OBJECTIVES AND 64-BIT HASH CODES. HIGHER VALUES INDICATE BETTER PERFORMANCE.

Baseline	Method	DSDH [4]			Hashnet [20]			CSQ [22]			DPH [21]		
		5-1	10-1	20-1	5-1	10-1	20-1	5-1	10-1	20-1	5-1	10-1	20-1
Results on *mini*ImageNet (64bit, mAP in %)													
ResNet-18	**simple**	77.31±0.63	64.10±0.49	50.33±0.37	77.70±0.63	64.45±0.49	51.61±0.36	74.67±0.61	63.16±0.49	51.08±0.36	77.81±0.63	64.53±0.49	50.12±0.36
	laplacian	77.43±0.63	63.85±0.49	50.43±0.38	77.52±0.63	64.12±0.49	51.07±0.37	76.84±0.62	63.76±0.50	51.18±0.36	77.72±0.63	64.04±0.49	49.58±0.37
	ours	**82.63±0.70**	**67.98±0.56**	**52.70±0.41**	**82.37±0.68**	**68.20±0.54**	**53.93±0.39**	**82.00±0.67**	**68.30±0.55**	**54.48±0.40**	**81.68±0.69**	**67.01±0.54**	**52.51±0.39**
WideResNet	**simple**	79.71±0.60	66.61±0.49	52.66±0.37	80.01±0.60	67.19±0.50	54.22±0.36	76.91±0.60	65.52±0.50	53.42±0.37	80.24±0.60	67.34±0.49	53.15±0.35
	laplacian	79.79±0.59	66.61±0.51	52.68±0.38	79.90±0.59	67.12±0.50	53.85±0.36	79.44±0.60	66.27±0.51	53.55±0.37	80.06±0.59	67.07±0.50	52.79±0.36
	ours	**84.65±0.66**	**70.46±0.57**	**54.96±0.41**	**84.42±0.65**	**70.67±0.55**	**56.39±0.39**	**83.81±0.64**	**70.84±0.55**	**56.75±0.39**	**83.83±0.65**	**69.60±0.54**	**54.90±0.39**
Results on *tiered*ImageNet (64bit, mAP in %)													
ResNet-18	**simple**	82.61±0.67	71.12±0.55	58.93±0.41	82.90±0.67	71.51±0.53	59.17±0.40	80.23±0.65	70.32±0.52	59.39±0.41	82.95±0.66	71.40±0.52	57.13±0.39
	laplacian	82.36±0.68	71.03±0.55	58.85±0.42	82.56±0.67	71.28±0.55	59.02±0.42	82.37±0.67	71.00±0.55	59.43±0.42	82.55±0.67	70.92±0.54	57.18±0.40
	ours	**87.04±0.71**	**75.20±0.61**	**61.33±0.46**	**86.44±0.68**	**75.48±0.58**	**62.79±0.45**	**85.68±0.68**	**75.28±0.59**	**63.30±0.45**	**85.99±0.70**	**75.15±0.58**	**61.56±0.45**
WideResNet	**simple**	84.21±0.63	73.68±0.53	61.81±0.41	84.33±0.64	74.00±0.52	62.36±0.39	81.70±0.64	68.39±0.57	57.65±0.41	84.38±0.63	73.44±0.52	58.74±0.39
	laplacian	84.09±0.64	73.77±0.53	61.98±0.41	84.07±0.64	73.86±0.53	62.13±0.41	83.80±0.64	73.69±0.53	62.76±0.41	84.26±0.64	73.42±0.52	59.10±0.41
	ours	**88.17±0.69**	**77.72±0.60**	**64.98±0.46**	**87.55±0.66**	**77.75±0.57**	**65.77±0.44**	**86.84±0.65**	**77.47±0.58**	**66.32±0.45**	**87.32±0.66**	**77.54±0.58**	**64.73±0.44**
Results on CUB (64bit, mAP in %)													
ResNet-18	**simple**	86.65±0.56	79.43±0.48	68.87±0.40	86.97±0.54	79.61±0.48	69.19±0.38	83.82±0.57	77.98±0.48	69.43±0.39	86.96±0.55	78.96±0.46	65.78±0.37
	laplacian	86.56±0.55	79.34±0.48	69.22±0.40	86.77±0.55	79.39±0.48	69.37±0.38	86.30±0.56	79.35±0.48	69.95±0.39	86.80±0.55	78.92±0.48	66.45±0.4
	ours	**93.13±0.52**	**85.75±0.48**	**74.77±0.43**	**92.42±0.53**	**85.48±0.48**	**74.95±0.41**	**91.30±0.53**	**85.36±0.48**	**76.02±0.42**	**92.19±0.54**	**84.65±0.48**	**72.79±0.40**
WideResNet	**simple**	90.60±0.47	85.00±0.40	75.14±0.39	90.94±0.47	85.31±0.41	75.82±0.37	87.69±0.51	83.60±0.43	75.46±0.38	90.96±0.46	84.95±0.41	73.58±0.37
	laplacian	90.75±0.47	85.05±0.43	75.64±0.39	90.83±0.47	85.35±0.41	76.28±0.37	90.49±0.48	84.84±0.43	76.21±0.38	90.87±0.47	85.01±0.41	74.38±0.38
	ours	**95.49±0.43**	**89.99±0.40**	**79.86±0.41**	**94.98±0.43**	**89.66±0.40**	**80.08±0.39**	**93.81±0.45**	**89.37±0.41**	**81.03±0.39**	**94.80±0.44**	**88.84±0.41**	**78.17±0.40**

a greater diversity of "genus" and corresponding "differentia" for the PP embeddings. This allows the PP embeddings to represent more varied and rich features. This also explains why the PP framework shows more significant improvements when using *mini* or *tiered* as the base classes to generate prior prototypes, compared to using CUB. Therefore, for Prior-Prototypes based encoding methods, sampling from more diverse and semantically distinct prior prototypes is more important than simply collecting more categories.

D. Results of PP embeddings with different lengths.

Setting. The dimension of PP embeddings depends on the number of Prior-Prototypes. To explore its impact on retrieval performance, we tested different PP lengths for comparison. Given the varying number of base classes across the three benchmarks, the lengths were set to (5, 50, all). The experiment is conducted in the 5-way-1-shot setting with the hash objective as DSDH [4].

Results. As shown in Fig. 4, mAP performance positively correlates with length of PP embeddings, supporting our intuition that more "genus and difference" leads to richer information in the embeddings. The PP framework demonstrated solid results even with just 5 Prior-Prototypes: 70.88±0.79 (*mini*ImageNet), 77.2±0.79 (*tiered*ImageNet), and 78.6±0.76 (CUB) with ResNet-18, and 72.8±0.79 (*mini*ImageNet), 78.19±0.79 (*tiered*ImageNet), and 82.11±0.76 (CUB) with WideResNet. With 50 Prior-Prototypes, the results were nearly identical to those obtained using all Prior-Prototypes (64, 351, 100). This finding is significant as it offers greater flexibility for scenarios with limited labeled data in the base class training set.

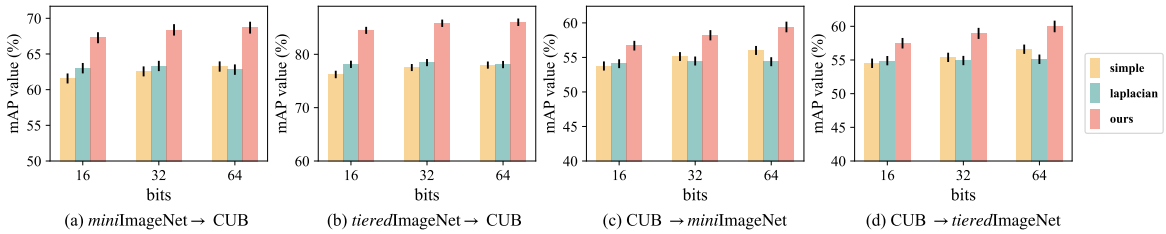
E. Results of Ablation

Setting. To validate the effectiveness of each module, we used ResNet-18 as the feature extractor and tested 5-way-1-shot retrieval with three hash code lengths (16, 32, 64 bit) for the performance of (SC, Diff, SCDM, and PP+SCDM).

Results. As shown in Fig. 5, mAP performance improves with the addition of modules and longer hash codes. For the 64-bit hash code, the PP framework slightly underperforms compared to SCDM on the *tiered*ImageNet. We attribute this to two factors: first, the feature extractor is already well-trained on 351 classes, enhancing multi-pattern recognition ability (*tiered* outperforms *mini*), thus limiting the performance gain of PP framework; second, the complexity of the PP embeddings (using 351 prior prototypes) requires a more powerful deep model for feature extraction, which will be the focus of our future work. Overall, our proposed framework effectively improves FSHL performance.

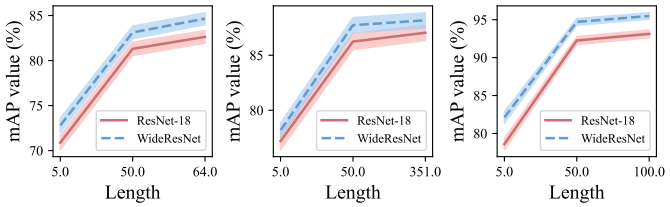
V. CONCLUSION

In this work, we propose a novel Prior-Prototypes based feature learning framework for FSHL. Central to our approach is the PP encoding method, which uses the metric difference between unknown category samples and Prior-Prototypes to generate PP embeddings. Additionally, we introduce a self-Consistency based Diffusion Module that enhances information interaction between samples and improves the expressive power of the hashing function by incorporating a reconstruction objective into the hashing learning process. Extensive experiments on three benchmarks demonstrate the superior performance of the proposed method.



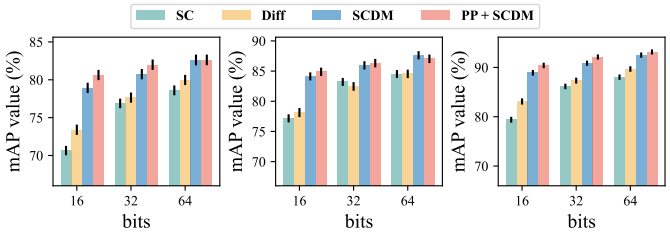
(a) *mini*ImageNet→ CUB (b) *tiered*ImageNet→ CUB (c) CUB →*mini*ImageNet (d) CUB →*tiered*ImageNet

Fig. 3. mAP (%) across datasets with 95% confidence intervals (ResNet-18 as feature extractor).



(a) *mini*ImageNet (b) *tiered*ImageNet (c) CUB

Fig. 4. mAP (%) of PP embeddings with different lengths and 95% confidence intervals. Higher values indicate better performance.



(a) *mini*ImageNet (b) *tiered*ImageNet (c) CUB

Fig. 5. Ablation study of four variations (mAP (%) with 95% confidence intervals): SC uses the reconstruction objective, Diff uses the diffusion layer, SCDM combines self-consistency with the diffusion layer, and PP + SCDM represents our complete PP framework.

REFERENCES

[1] Shiv Ram Dubey, Satish Kumar Singh, and Wei-Ta Chu, "Vision transformer hashing for image retrieval," in *2022 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2022, pp. 1–6.

[2] Yang Xu, Lei Zhu, Jingjing Li, Fengling Li, and Heng Tao Shen, "Temporal social graph network hashing for efficient recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[3] Feiran Hu, Chenlin Zhang, Jiangliang Guo, Xiu-Shen Wei, Lin Zhao, Anqi Xu, and Lingyan Gao, "An asymmetric augmented self-supervised learning method for unsupervised fine-grained image hashing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17648–17657.

[4] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan, "Deep supervised discrete hashing," *Advances in neural information processing systems*, vol. 30, 2017.

[5] 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*, vol. 34, pp. 24286–24298, 2021.

[6] Chengyin Xu, Zenghao Chai, Zhengzhuo Xu, Hongjia Li, Qiruyi Zuo, Lingyu Yang, and Chun Yuan, "Hhf: Hashing-guided hinge function for deep hashing retrieval," *IEEE Transactions on Multimedia*, vol. 25, pp. 7428–7440, 2022.

[7] Fengling Li, Bowen Wang, Lei Zhu, Jingjing Li, Zheng Zhang, and Xiaojun Chang, "Cross-domain transfer hashing for efficient cross-modal retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[8] Xu Luo, Hao Wu, Ji Zhang, Lianli Gao, Jing Xu, and Jingkuan Song, "A closer look at few-shot classification again," in *Proceedings of the 40th International Conference on Machine Learning*. 2023, ICML'23, JMLR.org.

[9] Yi Ma, Doris Tsao, and Heung-Yeung Shum, "On the principles of parsimony and self-consistency for the emergence of intelligence," *Frontiers of Information Technology & Electronic Engineering*, vol. 23, no. 9, pp. 1298–1323, 2022.

[10] Xiao Luo, Haixin Wang, Daqing Wu, Chong Chen, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua, "A survey on deep hashing methods," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 1, pp. 1–50, 2023.

[11] Jian Zhu, Xiaohu Ruan, Yongli Cheng, Zhangmin Huang, Yu Cui, and Lingfang Zeng, "Deep metric multi-view hashing for multimedia retrieval," in *Proceedings of the 2023 IEEE International Conference on Multimedia and Expo*, 2023, pp. 1955–1960.

[12] Hai Su, Jianwei Fang, Weixing Liu, Songsen Yu, and Huan Yang, "A deep hashing method of likelihood function adaptive mapping," *Neural Computing and Applications*, vol. 35, no. 8, pp. 5903–5921, 2023.

[13] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun, "Few-shot learning through an information retrieval lens," *Advances in neural information processing systems*, vol. 30, 2017.

[14] Yu-Xiong Wang, Liangke Gui, and Martial Hebert, "Few-shot hash learning for image retrieval," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1228–1237.

[15] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," *arXiv preprint arXiv:1911.04623*, 2019.

[16] Imtiaz Masud Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed, "Laplacian regularized few-shot learning," in *Proceedings of the 37th International Conference on Machine Learning*. 2020, ICML'20, JMLR.org.

[17] Jieqi Sun and Jian Li, "Few-shot classification with fork attention adapter," *Pattern Recognition*, vol. 156, pp. 110805, 2024.

[18] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen, "Binary generative adversarial networks for image retrieval," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.

[19] Tangjun Wang, Zehao Dou, Chenglong Bao, and Zuoqiang Shi, "Diffusion mechanism in residual neural network: Theory and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 2, pp. 667–680, 2024.

[20] 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*, 2017, pp. 5608–5617.

[21] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao, "Adversarial examples for hamming space search," *IEEE transactions on cybernetics*, vol. 50, no. 4, pp. 1473–1484, 2018.

[22] 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*, 2020, pp. 3083–3092.