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

***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. Experiments

A. Experimental Details

Datasets. Following the setup of existing works [1], 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 [2]. Experimental results are presented on four state-of-the-art hash learning objectives (DSDH [3], HashNet [4], DPH [5], CSQ [6]) and two prototype-based few-shot methods (SimpleShot [7], Laplacian [8]).

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, 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. 1, 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). 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 a greater diversity of "genus" and corresponding "differentia"

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 [3]			Hashnet [4]			CSQ [6]			DPH [5]		
		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**

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 [3].

Results. As shown in Fig. 2, 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.

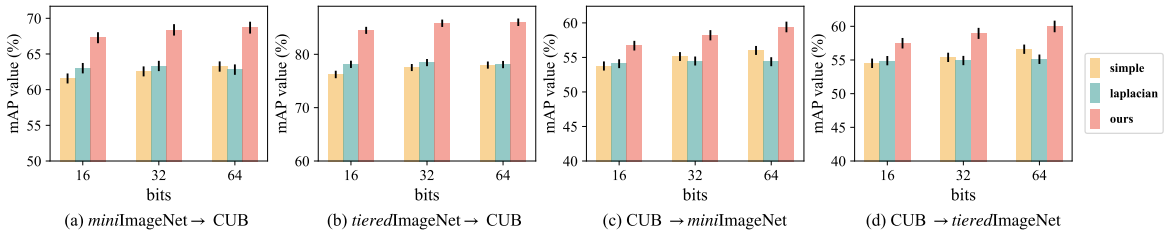
E. Results of Ablation

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. 3, 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.

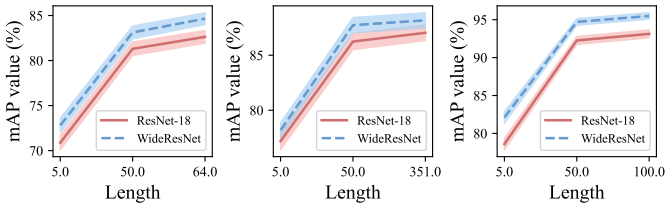
II. CONCLUSION

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.



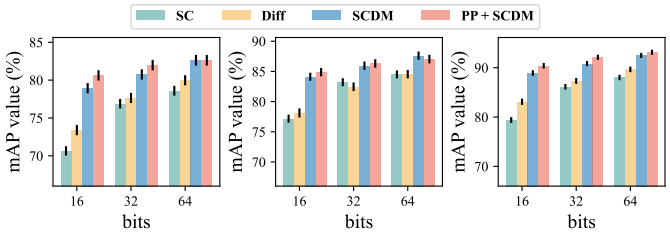
(a) *mini*ImageNet→ CUB (b) *tiered*ImageNet→ CUB (c) CUB →*mini*ImageNet (d) CUB →*tiered*ImageNet

Fig. 1. mAP (%) across datasets with 95% confidence intervals (ResNet-18 as feature extractor).



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

Fig. 2. 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. 3. 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] Tangjun Wang, Zehao Dou, Chenglong Bao, and Zuoqiang Shi, "Diffusion mechanism in residual neural network: Theory and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 2, pp. 667–680, 2024.

[2] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun, "Few-shot learning through an information retrieval lens," in *Neurips*, 2017, vol. 30.

[3] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan, "Deep supervised discrete hashing," in *Neurips*, 2017.

[4] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu, "Hashnet: Deep learning to hash by continuation," in *ICCV*, 2017, pp. 5608–5617.

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

[6] 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 *CVPR*, 2020, pp. 3083–3092.

[7] 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.

[8] Imtiaz Masud Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed, "Laplacian regularized few-shot learning," in *ICML*, 2020, pp. 11660–11670.

A. Results of Few-Shot Hashing Retrieval with 16-bit and 32-bit Hash Code Lengths

In addition to the comparison results for the 64-bit hash code length presented in SEC. I-B, we also report the comparison results for 16-bit and 32-bit hash code lengths, as shown in Tables A1 and A2, respectively. The experimental results demonstrate that our proposed PP framework achieves the best performance with both 16-bit and 32-bit hash codes, further highlighting the advantages of our PP framework in Few-Shot Hashing Retrieval.

TABLE A1
COMPARISON OF MAP (%) RESULTS ON 16BIT BETWEEN PP FRAMEWORK AND OTHER METHODS. HIGHER VALUES INDICATE BETTER PERFORMANCE.

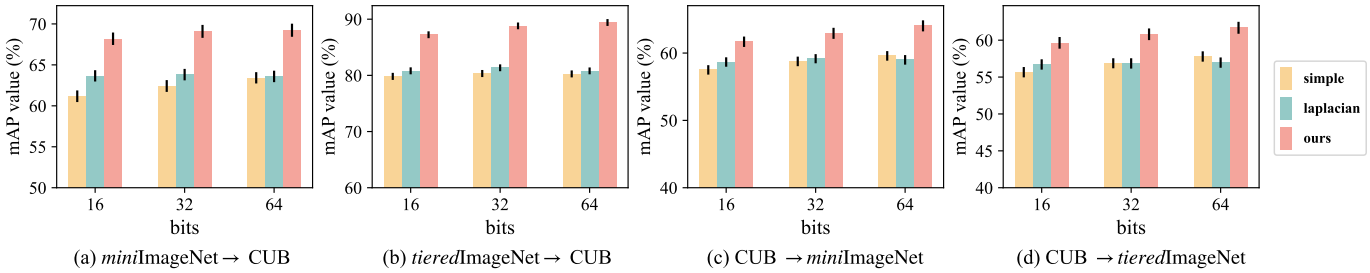
Baseline	Method	DSDH			Hashnet			CSQ			DPH		
		5-1	10-1	20-1	5-1	10-1	20-1	5-1	10-1	20-1	5-1	10-1	20-1
Results on *mini*ImageNet (16bit, mAP in %)													
ResNet-18	**simple**	75.19±0.64	60.80±0.50	41.46±0.35	75.06±0.63	60.23±0.49	44.55±0.37	73.33±0.63	61.06±0.50	45.64±0.37	75.75±0.63	61.34±0.48	44.76±0.37
	laplacian	75.47±0.63	61.67±0.50	42.19±0.38	75.31±0.63	60.47±0.51	44.79±0.36	75.56±0.63	62.08±0.50	46.39±0.37	75.80±0.62	61.53±0.51	45.07±0.36
	ours	**80.63±0.68**	**65.96±0.56**	**47.36±0.41**	**78.15±0.67**	**63.13±0.53**	**46.54±0.38**	**80.42±0.66**	**67.20±0.54**	**52.24±0.41**	**79.84±0.66**	**65.50±0.54**	**48.71±0.37**
WideResNet	**simple**	77.73±0.61	63.52±0.49	44.28±0.38	77.17±0.61	62.46±0.49	46.68±0.36	75.43±0.61	63.47±0.51	47.85±0.37	78.08±0.61	63.92±0.5	47.61±0.36
	laplacian	77.99±0.62	64.31±0.50	45.13±0.37	77.68±0.62	63.01±0.50	46.79±0.37	78.13±0.62	64.94±0.51	48.36±0.38	78.42±0.60	64.60±0.49	48.41±0.37
	ours	**83.11±0.65**	**68.44±0.56**	**49.96±0.41**	**80.04±0.66**	**65.43±0.54**	**48.92±0.39**	**82.44±0.64**	**69.63±0.54**	**54.24±0.41**	**81.93±0.65**	**67.65±0.54**	**51.03±0.39**
Results on *tiered*ImageNet (16bit, mAP in %)													
ResNet-18	**simple**	80.64±0.69	67.91±0.55	48.28±0.40	80.78±0.67	67.27±0.55	52.52±0.40	78.80±0.66	68.25±0.53	54.7±0.41	81.20±0.67	67.96±0.54	51.65±0.40
	laplacian	80.81±0.70	68.28±0.55	49.12±0.41	80.62±0.69	67.57±0.57	53.05±0.41	80.88±0.68	69.31±0.55	55.34±0.43	81.04±0.69	68.37±0.55	52.26±0.41
	ours	**84.88±0.69**	**73.83±0.58**	**58.29±0.45**	**82.04±0.69**	**70.06±0.55**	**55.31±0.44**	**85.01±0.68**	**74.40±0.58**	**61.59±0.46**	**83.94±0.68**	**73.17±0.56**	**59.50±0.43**
WideResNet	**simple**	82.36±0.66	70.81±0.52	52.34±0.42	82.31±0.64	69.64±0.53	54.78±0.39	80.66±0.66	71.30±0.52	58.19±0.42	82.80±0.65	70.85±0.53	54.47±0.38
	laplacian	82.79±0.65	71.49±0.53	53.59±0.42	82.38±0.65	70.00±0.52	55.20±0.40	82.48±0.66	72.23±0.54	58.82±0.41	82.95±0.66	71.35±0.53	55.56±0.39
	ours	**86.17±0.65**	**76.04±0.57**	**61.46±0.45**	**83.14±0.69**	**72.29±0.56**	**58.87±0.44**	**86.12±0.66**	**76.65±0.58**	**64.58±0.46**	**85.25±0.66**	**75.02±0.56**	**62.43±0.43**
Results on CUB (16bit, mAP in %)													
ResNet-18	**simple**	84.76±0.57	75.54±0.50	55.34±0.41	84.32±0.57	73.73±0.49	58.35±0.41	81.93±0.59	75.35±0.50	62.64±0.41	85.00±0.58	75.50±0.49	58.91±0.41
	laplacian	85.00±0.58	76.74±0.50	56.89±0.41	84.73±0.57	74.52±0.50	59.30±0.42	84.60±0.58	77.20±0.49	63.67±0.41	85.23±0.57	76.09±0.50	60.16±0.42
	ours	**90.40±0.56**	**83.83±0.49**	**68.52±0.44**	**87.92±0.58**	**81.44±0.51**	**69.88±0.43**	**89.72±0.56**	**84.37±0.48**	**74.30±0.43**	**89.44±0.57**	**82.87±0.50**	**69.67±0.42**
WideResNet	**simple**	89.17±0.49	81.63±0.46	62.99±0.41	88.67±0.50	79.60±0.47	65.20±0.40	86.36±0.53	81.32±0.45	69.21±0.41	89.40±0.49	81.69±0.45	67.08±0.39
	laplacian	89.37±0.50	82.70±0.45	64.86±0.41	89.12±0.50	80.73±0.47	66.73±0.40	89.35±0.50	83.35±0.44	70.52±0.41	89.52±0.51	82.36±0.44	68.73±0.40
	ours	**93.53±0.46**	**88.33±0.41**	**74.44±0.42**	**91.63±0.47**	**86.08±0.44**	**75.66±0.42**	**92.85±0.45**	**88.59±0.42**	**79.62±0.41**	**93.01±0.44**	**87.59±0.42**	**75.46±0.41**

TABLE A2
COMPARISON OF MAP (%) RESULTS ON 32BIT BETWEEN PP FRAMEWORK AND OTHER METHODS. HIGHER VALUES INDICATE BETTER PERFORMANCE.

Baseline	Method	DSDH			Hashnet			CSQ			DPH		
		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 (32bit, mAP in %)													
ResNet-18	**simple**	76.73±0.64	63.12±0.49	47.77±0.37	77.15±0.62	63.52±0.49	49.68±0.36	74.07±0.61	62.44±0.50	50.17±0.36	77.03±0.63	63.49±0.49	48.68±0.35
	laplacian	76.91±0.62	63.30±0.50	48.30±0.38	77.10±0.62	63.40±0.50	49.14±0.37	76.33±0.64	63.10±0.51	50.44±0.37	77.10±0.63	63.43±0.50	48.56±0.38
	ours	**81.99±0.69**	**67.70±0.57**	**51.28±0.40**	**80.80±0.66**	**66.77±0.53**	**51.93±0.38**	**81.36±0.67**	**67.93±0.56**	**54.24±0.40**	**80.99±0.67**	**66.76±0.54**	**51.29±0.40**
WideResNet	**simple**	79.14±0.60	65.91±0.50	50.48±0.37	79.30±0.61	66.19±0.49	52.03±0.36	76.15±0.60	64.86±0.51	52.55±0.37	79.32±0.60	66.27±0.49	51.49±0.35
	laplacian	79.19±0.61	66.06±0.50	50.77±0.38	79.16±0.61	66.14±0.50	51.67±0.37	78.79±0.60	65.82±0.50	52.93±0.37	79.31±0.60	66.34±0.50	51.64±0.36
	ours	**84.08±0.66**	**69.84±0.56**	**53.52±0.41**	**83.15±0.64**	**69.06±0.53**	**54.12±0.39**	**83.37±0.64**	**70.44±0.56**	**56.46±0.40**	**83.21±0.64**	**69.00±0.54**	**53.53±0.39**
Results on *tiered*ImageNet (32bit, mAP in %)													
ResNet-18	**simple**	82.05±0.67	70.16±0.53	56.40±0.41	82.44±0.66	70.38±0.53	57.42±0.39	79.70±0.67	69.57±0.53	58.46±0.40	82.43±0.66	70.06±0.53	55.68±0.41
	laplacian	81.93±0.66	70.37±0.54	56.92±0.41	82.27±0.66	70.24±0.55	57.20±0.41	81.73±0.67	70.60±0.55	58.74±0.42	82.11±0.66	69.84±0.55	55.66±0.41
	ours	**86.32±0.70**	**75.19±0.59**	**60.49±0.45**	**84.69±0.67**	**73.93±0.57**	**60.75±0.44**	**85.25±0.68**	**74.90±0.59**	**63.04±0.44**	**85.13±0.68**	**74.46±0.57**	**60.97±0.44**
WideResNet	**simple**	83.60±0.64	73.02±0.53	59.54±0.40	83.85±0.63	73.09±0.53	59.93±0.40	81.30±0.66	72.24±0.52	61.66±0.40	83.79±0.63	72.90±0.52	58.00±0.40
	laplacian	83.57±0.65	73.21±0.53	60.13±0.42	83.72±0.64	72.9±0.54	59.85±0.41	83.28±0.65	73.10±0.53	62.08±0.41	83.65±0.64	73.01±0.54	58.41±0.41
	ours	**87.50±0.67**	**77.25±0.60**	**64.20±0.46**	**85.97±0.65**	**76.01±0.57**	**63.85±0.44**	**86.44±0.66**	**77.13±0.58**	**66.00±0.45**	**86.30±0.66**	**76.52±0.57**	**64.18±0.44**
Results on CUB (32bit, mAP in %)													
ResNet-18	**simple**	86.23±0.54	78.41±0.47	65.45±0.40	86.35±0.55	77.83±0.49	65.49±0.39	83.09±0.59	76.98±0.49	68.03±0.39	86.44±0.54	77.93±0.49	64.08±0.39
	laplacian	86.19±0.55	78.43±0.49	66.45±0.41	86.31±0.55	77.98±0.49	65.94±0.40	85.58±0.57	78.68±0.48	68.98±0.40	86.35±0.54	78.10±0.49	64.85±0.40
	ours	**92.10±0.53**	**85.22±0.49**	**73.17±0.43**	**90.46±0.54**	**84.28±0.48**	**73.37±0.41**	**90.59±0.54**	**84.96±0.48**	**75.71±0.42**	**91.06±0.55**	**84.46±0.48**	**71.75±0.42**
WideResNet	**simple**	90.40±0.47	84.10±0.43	72.43±0.40	90.47±0.47	83.80±0.43	72.50±0.37	87.06±0.51	82.82±0.44	74.43±0.39	90.47±0.48	83.94±0.43	71.96±0.38
	laplacian	90.37±0.49	84.51±0.43	73.45±0.40	90.42±0.49	84.19±0.43	73.19±0.38	90.00±0.49	84.28±0.43	75.55±0.39	90.54±0.48	84.34±0.42	72.97±0.39
	ours	**94.84±0.44**	**89.42±0.41**	**78.39±0.41**	**93.67±0.44**	**88.43±0.42**	**78.92±0.41**	**93.46±0.44**	**89.01±0.41**	**80.92±0.40**	**94.08±0.44**	**88.63±0.42**	**77.44±0.41**

B. *Results of Cross Datasets with WideResNet*

This subsection supplements SEC. I-C in the main text, providing a quantitative comparison of mAP values and 95% confidence intervals for different methods using WideResNet as the feature extractor under a 5-way-1-shot setting. The results are shown for three different hash code lengths and with DSDH as the hashing objective. Higher values indicate better performance.



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

Fig. B1. mAP (%) across datasets with 95% confidence intervals (WideResNet as feature extractor).