

KALAHash: Knowledge-Anchored Low-Resource Adaptation for Deep Hashing

Shu Zhao¹, Tan Yu², Xiaoshuai Hao³, Wenchao Ma¹, Vijaykrishnan Narayanan¹

¹The Pennsylvania State University,

²NVIDIA,

³Samsung

{smz5505, vijaykrishnan.narayanan}@psu.edu

Abstract

Deep hashing has been widely used for large-scale approximate nearest neighbor search due to its storage and search efficiency. However, existing deep hashing methods predominantly rely on abundant training data, leaving the more challenging scenario of low-resource adaptation for deep hashing relatively underexplored. This setting involves adapting pre-trained models to downstream tasks with only an extremely small number of training samples available. Our preliminary benchmarks reveal that current methods suffer significant performance degradation due to the distribution shift caused by limited training samples. To address these challenges, we introduce Class-Calibration LoRA (CLoRA), a novel plug-and-play approach that dynamically constructs low-rank adaptation matrices by leveraging class-level textual knowledge embeddings. CLoRA effectively incorporates prior class knowledge as anchors, enabling parameter-efficient fine-tuning while maintaining the original data distribution. Furthermore, we propose Knowledge-Guided Discrete Optimization (KIDDO), a framework to utilize class knowledge to compensate for the scarcity of visual information and enhance the discriminability of hash codes. Extensive experiments demonstrate that our proposed method, Knowledge-Anchored Low-Resource Adaptation Hashing (KALAHash), significantly boosts retrieval performance and achieves a 4× data efficiency in low-resource scenarios.

Code —

<https://github.com/Tree-Shu-Zhao/KALAHash.pytorch>

Introduction

Deep hashing has emerged as a powerful technique for large-scale approximate nearest neighbor search, offering significant advantages in terms of storage efficiency and search speed (Luo et al. 2023). While deep hashing methods have shown remarkable performance, they typically rely on the availability of large amounts of data for effective training, which has been a cornerstone of their success but also presents limitations in scenarios where data availability is constrained.

In this paper, we introduce a challenging scenario: low-resource adaptation for deep hashing. This setting is characterized by the need to adapt pre-trained models to the

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

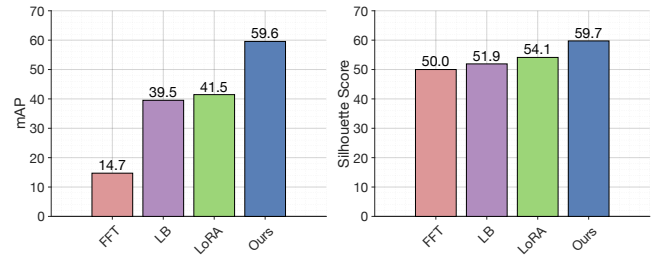


Figure 1: Performance comparison in low-resource settings (1-shot on the CIFAR-10 dataset), including mean Average Precision scores (left) and Silhouette Scores (right). FFT and LB denote Full Fine-Tuning and Lock Backbone, respectively. The increasing mAP and Silhouette Score indicate improved cluster separation and cohesion in the embedding space, demonstrating the effectiveness of our approach in addressing the distribution shift challenge. For the Silhouette Score, we normalize its range from $[-1, +1]$ to $[0, 100]$.

hashing task with extremely limited data samples available for training. The importance of this research direction is twofold. First, it addresses the critical need for efficiency and cost-effectiveness in developing retrieval systems. Annotating large datasets is often prohibitively expensive and time-consuming, especially in specialized domains (Gui, Wang, and Hebert 2017). By focusing on low-resource adaptation, we aim to reduce the resources required for effective retrieval systems while maintaining high performance. Second, this approach enables rapid adaptation to new domains or emerging topics, a crucial capability in today’s fast-paced information landscape (Cohen et al. 2022).

Despite its practical importance, this problem has received relatively little attention in the research community. Our preliminary benchmarks reveal significant challenges in low-resource adaptation for deep hashing. Specifically, we observe substantial performance degradation in existing methods when faced with limited training samples, as illustrated in Figure 1. Full Fine-Tuning (FFT) achieves a mean Average Precision (mAP) of only 14.7%. While Lock Backbone (LB) shows improvements, it also limits the ability of model fine-tuning and achieves 39.5% mAP. Even equipped with LoRA (Hu et al. 2022), an advanced technique for enabling parameter-efficient fine-tuning, it still falls short

with an mAP score of 41.5%. We argue that the performance gap is primarily attributed to the distribution shift, a mismatch between the data distributions of the pre-trained and downstream tasks, occurring when models trained on large datasets are adapted to downstream tasks with scarce data. To measure how this issue affects the distributions of hash codes in hamming space, we employ the Silhouette Score (Rousseeuw 1987) to measure how similar a class is to its own cluster compared to others. FFT achieves a Silhouette Score of 50.0%, denoting the embedding space has collapsed and all data points are close together. While LB and LoRA improve the Silhouette Score, they still cannot perform satisfactorily. The results further underscore the challenge of maintaining cohesive and well-separated clusters in the embedding space under low-resource settings, highlighting the need for more sophisticated adaptation strategies.

Therefore, we recognize the need for a novel approach that can leverage additional sources of information to compensate for the scarcity of data. Recent advancements in Vision-Language Models (VLMs) have demonstrated their ability to capture rich semantic relationships between visual concepts and textual descriptions (Radford et al. 2021; Liu et al. 2023). These models, pre-trained on vast amounts of image-text pairs, encapsulate a wealth of class-level knowledge that can potentially guide the adaptation process in low-resource settings. By tapping into this pre-existing knowledge, we hypothesize that we can mitigate the effects of distribution shift and enhance the discriminative power of hash codes, even when faced with limited training samples.

Motivated by this, we leverage the knowledge within pre-trained VLMs and propose Class-Calibration LoRA (CLoRA), a novel plug-and-play approach that dynamically constructs low-rank adaptation matrices by leveraging class-level textual knowledge embeddings. It effectively incorporates prior class knowledge as anchors, enabling parameter-efficient fine-tuning while maintaining the original data distribution. Additionally, we introduce Knowledge-Guided Discrete Optimization (KIDDO), a framework that utilizes class knowledge to compensate for the scarcity of visual information and enhance the discriminability of hash codes.

The main contributions of our work are as follows:

- We introduce and benchmark the problem of low-resource adaptation in deep hashing, highlighting its importance and challenges. Our benchmarks reveal significant performance degradation in existing methods when faced with limited training samples.
- We propose CLoRA, a novel plug-and-play approach that leverages textual knowledge embeddings as anchors for efficient adaptation in low-resource scenarios.
- We develop KIDDO, a knowledge-guided optimization framework that injects knowledge into the optimization process to enhance hash code generation.
- We demonstrate that our proposed method significantly improves retrieval performance in challenging low-resource settings through extensive experiments.

Related Work

Deep Hashing for Efficient Retrieval. Deep hashing has emerged as a powerful approach for large-scale visual retrieval, leveraging deep learning to project high-dimensional data into compact binary codes. The field has evolved from early two-stage methods like CNNH (Xia et al. 2014) to end-to-end frameworks such as DHHN (Lai et al. 2015), which enabled simultaneous optimization of networks and hash codes. The loss functions utilized in deep hashing can be categorized into ranking-based (Wang, Shi, and Kitani 2016; He et al. 2018), pair-wise (Li, Wang, and Kang 2016; Cao et al. 2017; Zhao et al. 2021), and point-wise methods (Yuan et al. 2020; Hoe et al. 2021; Wang et al. 2023a). To address the challenge of discrete optimization, methods like DSDH (Li et al. 2017) have proposed direct optimization of binary codes using techniques such as discrete cyclic coordinate descent. Architectural innovations, particularly asymmetric designs introduced by DAPH (Shen et al. 2017) and further developed in ADSH (Jiang and Li 2018), CCDH (Zhao et al. 2020), and CEDIH (Wu et al. 2024), have significantly improved hash learning quality and efficiency. Despite these advancements, challenges remain in scenarios with limited data. UGH (Gui, Wang, and Hebert 2017) devises a three-phase framework for the few-shot hashing. However, it needs to maintain a large hash function pool and select specific components during inference, which significantly increases the inference delay. Moreover, UGH cannot correctly select components under extremely low-resource adaptation settings, leading to significant performance degradation. Our method leverages the knowledge within the pre-trained models as anchors and complementary information to boost performance under low-resource adaptation settings.

Low-Resource Adaptation. Low-resource adaptation has gained significant attention in various machine learning domains, addressing scenarios with limited data for multi-modal large language model fine-tuning (Liu et al. 2023; Zhao et al. 2024; Zhao and Xu 2023b,a; Hao and Zhang 2023; Hao et al. 2023). Few-shot learning approaches, such as prototypical networks (Snell, Swersky, and Zemel 2017) and MAML (Finn, Abbeel, and Levine 2017), have pioneered tackling low-resource scenarios by learning transferable knowledge that can quickly adapt to new tasks with minimal data. Recently, low-rank adaptation (LoRA) (Hu et al. 2022) have demonstrated efficient parameter-tuning for large models. This approach has been particularly effective in natural language processing and is gaining traction in vision tasks. Model merging (Pan, Cai, and Zhuang 2023; Wang et al. 2023b; Yang et al. 2024) combines several model weights trained on different tasks to create a new weight that can perform all tasks simultaneously. Our work focuses on how to train a model for a specific task to achieve better performance. In the specific domain of deep hashing, low-resource adaptation remains relatively unexplored. While methods like Venkateswara et al. (2017) have addressed domain adaptation for hashing, they typically assume a substantial amount of target domain data. The challenge of adapting hash functions with extremely limited data presents a significant research opportunity.

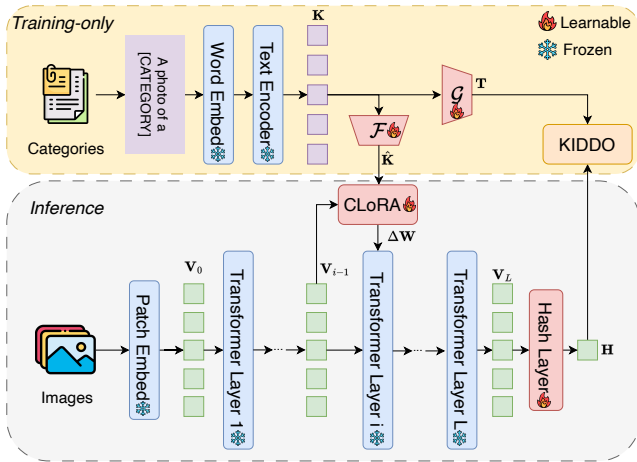


Figure 2: Architecture overview of the proposed KALAHash method, illustrating the integration of Class-Calibration LoRA (CLoRA) and Knowledge-Guided Discrete Optimization (KIDDO).

Method

Problem Formulation

Assuming models have been pre-trained on several large source datasets, our goal is to adapt these pre-trained models to learn a hash function that maps images to binary codes while preserving semantic similarity with an extremely small training set $\mathbb{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, including a set of N images and their labels.

LoRA Background

LoRA (Hu et al. 2022) is an efficient method for fine-tuning large models. It works by introducing small, trainable matrices into layers of a Transformer model (Vaswani et al. 2017). In a standard fully-connected layer, the output is calculated as

$$\mathbf{o} = \mathbf{W}\mathbf{x}, \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{d \times k}$ is the pre-trained weight matrix; $\mathbf{x} \in \mathbb{R}^{k \times 1}$ is an input vector; $\mathbf{o} \in \mathbb{R}^{d \times 1}$ is the output vector. LoRA modifies Equation (1) by adding a low-rank update:

$$\hat{\mathbf{o}} = \mathbf{W}\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{x} + \eta\mathbf{P}\mathbf{Q}\mathbf{x}. \quad (2)$$

Here, $\mathbf{Q} \in \mathbb{R}^{r \times k}$ and $\mathbf{P} \in \mathbb{R}^{d \times r}$ are small matrices that form the low-rank update. $r \ll \min(k, d)$. η is a scale factor. The key is that the number of parameters in \mathbf{P}/\mathbf{Q} is much smaller than the original weight matrix \mathbf{W} . During fine-tuning, only \mathbf{Q} and \mathbf{P} are updated, while the original model weights remain frozen. This parameter-efficient approach allows for quick adaptation of large models to new tasks with minimal additional parameters.

However, LoRA, while efficient for parameter updates, does not inherently incorporate task-specific knowledge or constraints. Its generic adaptation mechanism lacks the guidance needed to effectively map high-dimensional image features to compact binary hash codes, especially when provided with only a handful of examples per class, as illustrated in Figure 1. For deep hashing, especially with limited

data, additional guidance about class relationships or desired hash code properties are crucial for generating discriminative hash codes.

To address these limitations and provide the necessary task-specific guidance, we propose leveraging class-level textual knowledge. This approach aims to inject semantic information directly into the adaptation process, bridging the gap between the limited visual data and the rich semantic understanding required for effective hash code generation. By incorporating textual descriptions of image categories, we can provide additional context and structure to guide the learning process, even in extremely low-resource scenarios. This textual knowledge serves as a form of prior information, helping to constrain the adaptation process and ensure that the resulting hash codes maintain semantic relevance. In the following section, we detail our method for extracting and utilizing this class-level textual knowledge to enhance the deep hashing process.

Overview

Figure 2 illustrates the architecture of the proposed method. We build our approach on the pre-trained CLIP model (Radford et al. 2021), including a text and a vision encoder consisting of multiple transformer layers.

The Text Encoder pre-extracts class-level textual knowledge \mathbf{K} using category names. The Vision Encoder splits images into fixed-size patches which are projected into patch embeddings \mathbf{V}_0 by the Patch Embed module, and encodes \mathbf{V}_0 to vision tokens \mathbf{V}_L by transformer layers, where L denotes the number of transformer layers. During the encoding process, we introduce Class-Calibration LoRA (CLoRA) module to dynamically construct a weight adjustment matrix $\Delta\mathbf{W}$ by incorporating mapped knowledge $\hat{\mathbf{K}}$ and input vision tokens \mathbf{V}_{i-1} from the i -th transformer layer to guide the fine-tuning process. For simplicity, we will omit the subscripts of vision tokens in the following sections. Then, the vision tokens \mathbf{V} are mapped into hash features \mathbf{H} . To further improve the hash code generation, we employ Knowledge-Guided Discrete Optimization (KIDDO), a framework that injects the mapped textual knowledge \mathbf{T} into the optimization process.

Class-Level Textual Knowledge Generation

We use the Text Encoder to pre-extract class-level textual knowledge:

$$\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_C]^T \in \mathbb{R}^{C \times d_t}, \quad (3)$$

where C is the number of categories. Specifically, we create a prompt based on the hand-crafted template “a photo of a [CATEGORY].” For instance, given a category name *dog*, the prompt is instantiated as “a photo of a dog.” Next, the Word Embed module and Text Encoder map the prompt into a class-level textual knowledge embedding \mathbf{k}_i . Note that the knowledge generation only needs to be performed once in the whole process.

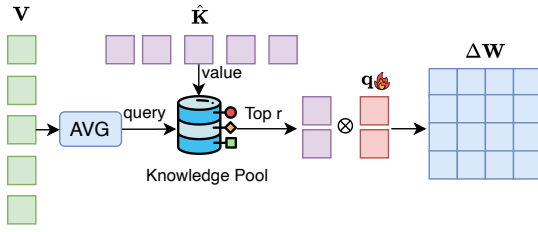


Figure 3: Architecture of the proposed CLoRA module.

Class-Calibration LoRA

We observe that the weight adjustment matrix $\Delta \mathbf{W}$ in Equation (2) can be constructed by:

$$\Delta \mathbf{W} = \eta \mathbf{P} \mathbf{Q} = \eta \sum_{i=1}^r \mathbf{p}_i \mathbf{q}_i^T, \quad (4)$$

where $\mathbf{q}_i \in \mathbb{R}^{k \times 1}$, $\mathbf{p}_i \in \mathbb{R}^{d \times 1}$.

As shown in Figure 3, to constrain the weight adjustment matrix space, spanning by $\mathbf{p}_i \mathbf{q}_i^T$, we replace \mathbf{p}_i with the class-level textual knowledge \mathbf{k}_i defined in Equation (3) as anchors:

$$\Delta \mathbf{W} = \eta \sum_{i=1}^r \hat{\mathbf{k}}_i \mathbf{q}_i^T, \quad (5)$$

where $\hat{\mathbf{k}}_i = \mathcal{F}(\mathbf{k}_i)$, \mathcal{F} is a linear layer and $\mathcal{F}(\cdot) \in \mathbb{R}^{d \times 1}$.

Considering that different inputs need different knowledge, we design a query-based strategy to dynamically select r knowledge vectors from the knowledge pool $\hat{\mathbf{K}}$:

$$\hat{\mathbf{K}}^v = \text{Top}_r(\text{avg}(\mathbf{V}), \hat{\mathbf{K}}), \quad (6)$$

where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_t]$ is the vision tokens; $\text{avg}(\cdot)$ denotes the average pooling operation. $\text{Top}_r(\cdot, \cdot)$ selects the top r vectors in $\hat{\mathbf{K}}$ with the largest cosine similarity to $\text{avg}(\mathbf{V})$, and $\hat{\mathbf{K}}^v = [\hat{\mathbf{k}}_1^v, \dots, \hat{\mathbf{k}}_r^v]$.

Finally, the weight adjustment matrix is constructed by:

$$\Delta \mathbf{W} = \eta \sum_{i=1}^r \hat{\mathbf{k}}_i^v \mathbf{q}_i^T. \quad (7)$$

Knowledge-Guided Discrete Optimization

We first employ a similarity loss function \mathcal{L}_s and a quantization loss \mathcal{L}_q that are widely used in deep hashing methods, which makes the Hamming distance of two similar points as small as possible and vice versa:

$$\mathcal{L}_s = - \sum_{s_{ij} \in \mathbf{S}} (s_{ij} \theta_{ij} - \log(1 + e^{\theta_{ij}})), \quad (8)$$

$$\mathcal{L}_q = \|\mathbf{H} - \mathbf{B}\|_2^2,$$

where s_{ij} is 1 if image i and j belong to the same category otherwise 0; $\theta_{ij} = \frac{1}{2} \mathbf{h}_i^T \mathbf{h}_j$; $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_n]^T$ are real-value image features generated from Hashing layer; $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]^T$, $\mathbf{b}_i \in \{-1, 1\}^b$ is the learned binary code and is randomly initialized before the training.

In low-resource settings, the limited number of training images may not be sufficient to cover all aspects of visual concepts, thus leading to over-fitting issues. We argue that language can be used as an abstract conceptual representation as anchor points to aid visual feature learning. For instance, while “dog” is visually represented in various ways that cannot all be covered by a limited number of images, they can be abstracted into a single linguistic concept “dog”.

Motivated by this, we add an alignment loss \mathcal{L}_a between the learned binary codes \mathbf{B} and the textual knowledge \mathbf{K} to further improve the hash code generation by leveraging the textual knowledge as anchors:

$$\mathcal{L}_a = \|\mathbf{Y} - \mathbf{T}^T \mathbf{B}\|_2^2, \quad (9)$$

where $\mathbf{T} = \mathcal{G}(\mathbf{K})$, $\mathcal{G}(\cdot)$ denotes a fully-connected layer and $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n] \in \mathbb{R}^{C \times n}$ are one-hot label vectors where C is the number of categories.

Finally, the loss function is

$$\mathcal{L} = \alpha \mathcal{L}_a + \beta \mathcal{L}_q + \gamma \mathcal{L}_s, \quad (10)$$

where α , β , and γ are scalars that balance the loss values.

When optimizing the loss function \mathcal{L} in Equation (10), it not only exploits the knowledge from the text encoder as complementary information to improve the image hash code generation but also enables the possibility of discrete optimization. To optimize the loss function in Equation (10), we use the standard backpropagation algorithm to learn \mathbf{H} and \mathbf{T} . To optimize \mathbf{B} , we fix all the variables except for \mathbf{B} and rewrite the optimization formula as

$$\begin{aligned} \min_{\mathbf{B}} \alpha \|\mathbf{Y} - \mathbf{T}^T \mathbf{B}\|_2^2 + \beta \|\mathbf{H} - \mathbf{B}\|_2^2 \\ \text{s.t. } \mathbf{B} \in \{-1, 1\}^{N \times b}. \end{aligned} \quad (11)$$

Then, we adopt the discrete cyclic coordinate descent (DCC) method proposed by Shen et al. (2015) to optimize \mathbf{B} column by column. The optimal solution of Equation (11) is

$$\mathbf{B}^i = \text{sign}(\mathbf{S}^i - \mathbf{B}'^T \mathbf{T}^i), \quad (12)$$

where \mathbf{B}^i is the i^{th} column of \mathbf{B} , \mathbf{B}' is the matrix of \mathbf{B} excluding \mathbf{B}^i ; \mathbf{S}^i is the i^{th} row of matrix \mathbf{S} , $\mathbf{S} = \beta \mathbf{Y} \mathbf{T} + \gamma \mathbf{H}$, \mathbf{S}' is the matrix of \mathbf{S} excluding \mathbf{S}^i ; \mathbf{T}^i is the i^{th} row of \mathbf{T} , \mathbf{T}' is the matrix of \mathbf{T} excluding \mathbf{T}^i .

In Equation (12), textual knowledge is injected into binary code \mathbf{B} , further improving the optimization process of hash code generation \mathbf{H} . In the following section, we will demonstrate the effectiveness of our proposed method.

Experiments

Datasets

We evaluate our proposed method on three standard benchmarks: NUS-WIDE (Chua et al. 2009), MS-COCO (Lin et al. 2014), and CIFAR-10 (Krizhevsky and Hinton 2009). **NUS-WIDE** is a multi-label dataset. Following Hoe et al. (2021), we adopt a subset of the original NUS-WIDE dataset, which has 195,834 images associated with the 21 most frequent classes. We randomly select 2,100 images (1,00 images per class) to form the query set, and the rest is used as the gallery set.

Method	NUS-WIDE				MS-COCO				CIFAR-10			
	1-shot	2-shot	4-shot	8-shot	1-shot	2-shot	4-shot	8-shot	1-shot	2-shot	4-shot	8-shot
HashNet (Cao et al. 2017)	65.23	66.27	70.54	73.56	58.81	62.44	65.28	67.50	41.68	44.97	69.96	76.58
DSDH (Li et al. 2017)	67.32	69.23	72.15	74.13	59.63	62.44	66.84	68.72	44.22	53.14	71.76	77.26
DCH (Cao et al. 2018)	65.55	66.04	70.92	71.48	60.32	62.26	66.51	67.70	39.53	48.69	67.03	75.59
GreedyHash (Su et al. 2018)	67.24	69.96	71.71	72.21	59.81	63.91	65.84	70.28	44.87	57.01	72.00	77.58
CSQ (Yuan et al. 2020)	65.75	67.31	70.96	71.51	59.23	63.09	66.14	70.18	46.66	60.54	69.50	77.69
OrthoHash (Hoe et al. 2021)	67.31	70.96	71.48	71.59	60.21	64.13	66.34	70.23	46.68	60.03	73.37	77.63
HSWD [†] (Doan, Yang, and Li 2022)	67.58	67.83	70.44	74.10	60.15	62.86	66.28	69.06	48.63	57.36	73.24	79.37
MDSH [‡] (Wang et al. 2023a)	67.23	68.22	70.47	72.04	58.55	59.89	60.94	63.95	47.33	58.69	73.16	78.09
KALAHash	70.69	71.26	74.11	75.24	65.32	66.43	71.98	73.96	57.54	70.00	80.14	83.00

Table 1: Comparison of mAP on NUS-WIDE, MS-COCO, and CIFAR-10 datasets for different deep hashing methods under various low-resource settings (1-shot to 8-shot).[†]: we use the HashNet-HSWD. [‡]: MSDH conducted experiments only on single-label datasets in the original paper.

Method	NUS-WIDE	MS-COCO	CIFAR-10
HashNet	65.23	58.81	41.68
+CLoRA	69.41	61.08	54.20
DSDH	67.32	59.63	44.22
+CLoRA	70.02	62.43	54.02
DCH	65.55	60.32	39.53
+CLoRA	69.48	61.83	50.55
GreedyHash	67.24	59.81	44.87
+CLoRA	70.30	60.74	51.77
CSQ	65.75	59.23	46.66
+CLoRA	69.14	60.54	49.44
OrthoHash	67.31	60.21	49.50
+CLoRA	69.61	61.39	51.58
HSWD	67.58	58.55	48.63
+CLoRA	68.85	60.75	52.63
MDSH	67.23	58.55	47.33
+CLoRA	68.24	60.34	48.29

Table 2: Plug-and-play capability of CLoRA. mAP improvements when applying CLoRA to various baseline deep hashing methods on NUS-WIDE, MS-COCO, and CIFAR-10 datasets.

MS-COCO is a multi-label dataset containing 82, 783 training images and 40, 504 validation images belonging to 80 classes. We combine the two sets of images and prune them without labels. Following Hoe et al. (2021), we randomly choose 5,000 images as the query set, and the rest are viewed as the gallery set.

CIFAR-10 consists of 60,000 images with 32×32 resolution. It has 10 classes, each containing 6,000 samples. Following Cao et al. (2018), we randomly sample 1,000 images (100 images per class) to construct the query set and the rest is used to form the gallery set.

Evaluation Protocol

In our low-rank adaptation setting, we randomly split N_K samples per class to create the training set. N_K is 1, 2, 4, or 8 in our experiments. Following the standard evaluation protocol, we report the mean Average Precision at K (mAP@ K), the mean of average precision scores of

Method	NUS-WIDE	MS-COCO	CIFAR-10
KALAHash	70.69	65.32	57.54
w.o. CLoRA	68.31	61.49	46.38
w.o. KIDDO	66.97	60.48	50.89

Table 3: Ablation study showing the impact of CLoRA and KIDDO components on mAP performance across NUS-WIDE, MS-COCO, and CIFAR-10 datasets.

the top K retrieved images, to evaluate the retrieval performance. Specifically, we report mAP@59000 for CIFAR-10, mAP@5000 for NUS-WIDE, and mAP@5000 for MS-COCO, respectively. Notably, for multi-label datasets, two images are considered similar if they share at least one common label.

Implementation Details

For a fair comparison, all methods, including baselines, use the same backbone model, optimizer, training hyperparameters, etc.

Backbone and CLoRA. We employ the CLIP ViT-B/32 as the backbone model to conduct experiments. CLoRA can be inserted into various positions in backbones. In our experiments, it is put into the key and value matrices of the multi-head attention module in the last transformer layer. η and r are set to 1.0 and 1, respectively.

Hash Layer. A hash layer is utilized to map the original features extracted from the backbone model to compacted hash codes. Following Hoe et al. (2021), the hash layer consists of a full-connected layer, a batch norm layer and a tanh layer. In our experiments, we use a 16-bit hash layer as default.

Training Details. We freeze all the parameters expected for the CLoRA module, \mathcal{G} , \mathcal{F} , and hash layer. We use SGD with 0.9 momentum and $1e - 5$ weight decay as the optimizer. The learning rate is set to 0.01. The batch size is set to 8. α , β , and γ are set to 0.1, 1.0, and 3.0, respectively.

If not specified, experiments are conducted on the CIFAR-10 dataset with the 1-shot setting. Detailed settings can be found in the code we provided.

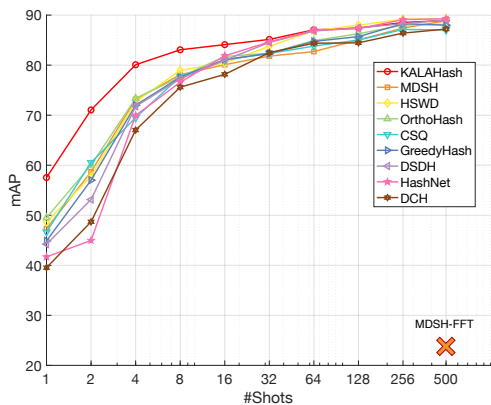


Figure 4: Performance comparison of KALAHash and baseline methods as the number of shots increases from 1 to 500 on CIFAR-10 dataset. MDSH-FFT denotes all the parameters are fine-tuned in the MDSH baseline.

Variant	NUS-WIDE	MS-COCO	CIFAR-10
CLoRA	70.69	65.32	57.54
LoRA (Hu et al. 2022)	68.24	61.63	47.05
Prompt Tuning (Zhou et al. 2022)	69.65	61.37	49.70
Bias Tuning (Zaken, Goldberg, and Ravfogel 2022)	69.90	63.26	45.81

Table 4: Comparison of CLoRA with other parameter-efficient fine-tuning techniques on NUS-WIDE, MS-COCO and CIFAR-10 datasets.

Main Results

We choose several representative deep hashing methods as baselines, including HashNet (Cao et al. 2017), DSDH (Li et al. 2017), DCH (Cao et al. 2018), GreedyHash (Su et al. 2018), CSQ (Yuan et al. 2020), OrthoHash (Hoe et al. 2021), HSWD (Doan, Yang, and Li 2022), and MDSH (Wang et al. 2023a).

Table 1 presents the mAP results for NUS-WIDE, MS-COCO, and CIFAR-10 across different low-resource settings (1-8 shots). We note that SOTA methods do not show absolute competitiveness in low-resource settings as on full-size datasets, highlighting the need for more sophisticated adaptation strategies. Our proposed KALAHash consistently outperforms all baselines across all datasets and shot settings. The performance improvements are particularly significant in the extreme low-resource scenarios (1-shot and 2-shot). For CIFAR-10, KALAHash achieves 8.91%-18.01% improvements over the baselines in the 1-shot setting. This performance gap remains substantial even as the number of shots increases, with KALAHash maintaining 3.63%-7.41% improvements in the 8-shot setting. We can also observe the same trend on the NUS-WIDE and MS-COCO datasets. The multi-label nature of these two datasets highlights KALAHash’s ability to handle complex semantic relationships even with limited data.

Plug-and-Play Capability

Table 2 demonstrates KALAHash’s plug-and-play capability by applying CLoRA to various baseline methods.

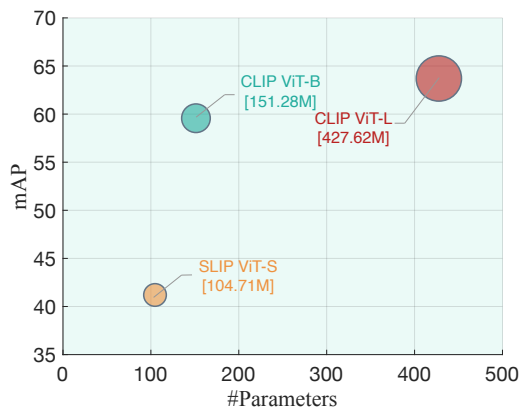


Figure 5: Performance scaling of KALAHash with different backbone models (SLIP ViT-S, CLIP ViT-B, CLIP ViT-L) in relation to the number of model parameters.

Across all baselines and datasets, adding CLoRA consistently improves performance. Specifically, the ranges of improvements are from 1.01%-4.18% on NUS-WIDE, 0.93%-2.80% on MS-COCO, and 0.96%-12.52% on CIFAR-10, respectively. The results underscore the versatility and effectiveness of our proposed method in enhancing existing deep hashing approaches in low-resource scenarios.

Ablation Studies

To understand the contribution of each component in KALAHash, we conduct ablation studies, as shown in Table 3. Removing CLoRA results in a significant performance drop across all datasets, with mAP decreasing by 2.38%, 3.83%, and 11.16% on NUS-WIDE, MS-COCO, and CIFAR-10 respectively. Similarly, removing KIDDO leads to performance degradation, with mAP decreasing by 3.72%, 4.84%, and 6.65% on NUS-WIDE, MS-COCO, and CIFAR-10 respectively, demonstrating the effectiveness of injecting textual knowledge into the optimization process.

We also compare CLoRA to other parameter-efficient fine-tuning techniques in Table 4. CLoRA outperforms standard LoRA (Hu et al. 2022), Prompt Tuning (Zhou et al. 2022), and Bias Tuning (Zaken, Goldberg, and Ravfogel 2022) across all datasets, demonstrating its effectiveness.

Scalability

Figure 4 presents a comprehensive analysis of KALAHash’s performance as the number of shots increases from 1 to 500 on CIFAR-10. KALAHash consistently outperforms all baselines with limited training samples (1-16 shots). As the number of training samples increases, our approach still maintains a performance comparable to SOTA’s. This highlights the method’s effectiveness in extremely low-resource scenarios while also demonstrating its ability to maintain superior performance as more data becomes available. We note that the SOTA methods do not show absolute competitiveness. The reason may be that we lock the backbone and only fine-tune the FC layer, limiting its ability. However, even full fine-tuning of VLMs on full datasets can also lead to serious

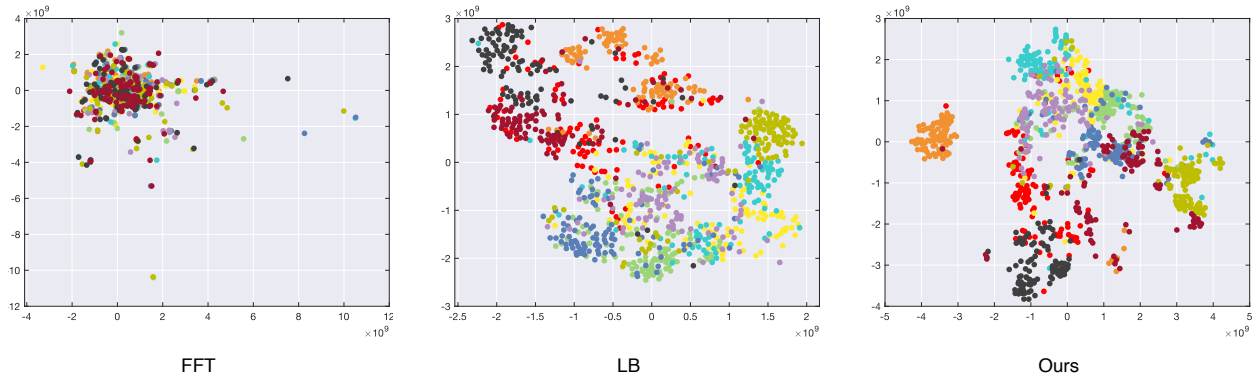


Figure 6: t-SNE visualization of learned hash codes for Full Fine-Tuning (FFT), Lock Backbone (LB), and our proposed method (KALAHash). Different colors denote different categories.

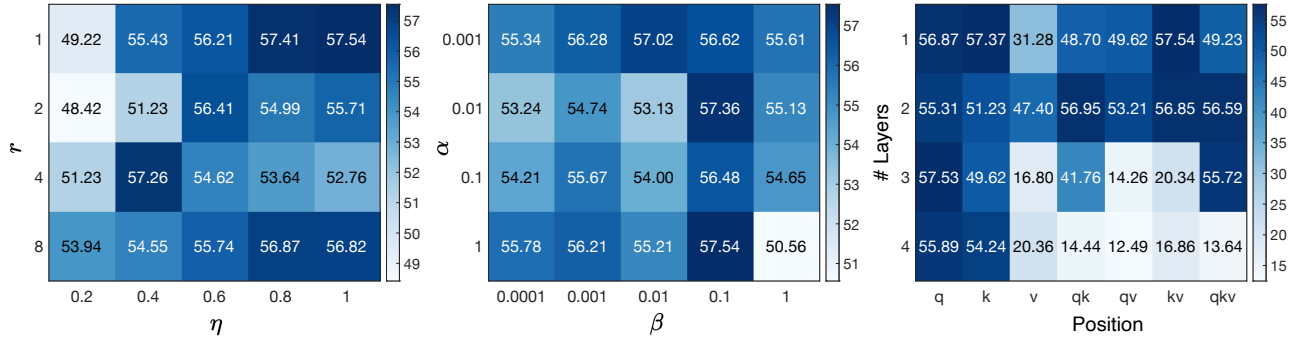


Figure 7: Parameter sensitivity analysis for KALAHash, showing mAP performance across different hyper-parameter settings.

distribution shift issue. To demonstrate this, we report the result of MDSH-FFT on the full-size CIFAR-10 dataset as an orange cross, illustrated in Figure 4.

Figure 5 illustrates the performance of KALAHash across different backbone models, including SLIP ViT-S (Mu et al. 2022), CLIP ViT-B, and CLIP ViT-L (Radford et al. 2021). The results show that KALAHash’s performance scales well with larger backbone models, achieving higher mAP scores as the number of parameters increases. This demonstrates the method’s ability to leverage more powerful pre-trained models effectively.

Qualitative Analysis

Figure 6 provides a t-SNE visualization (Van der Maaten and Hinton 2008) of the learned hash codes for Full Fine-Tuning (FFT), Lock Backbone (LB), and our proposed method. The results show that the embedding space of FFT is collapsed due to the distribution shift issue. While LB improves this, it still cannot perform satisfactorily, as the red points are scattered in the embedding space. The visualization of KALAHash clearly shows that it produces more compact and well-separated clusters than the other methods. This qualitative result supports our quantitative findings and illustrates KALAHash’s ability to learn more discriminative hash codes even in low-resource settings.

	ViT-S/16	ViT-B/32	ViT-B/16	ViT-L/14
w.o. CLoRA	2.20 ± 0.03	1.17 ± 0.02	2.20 ± 0.04	6.28 ± 0.04
w. CLoRA	2.21 ± 0.05	1.17 ± 0.05	2.23 ± 0.02	6.33 ± 0.03

Table 5: Inference time (ms) per image comparison of various VLMs with and without CLoRA. CLoRA demonstrates negligible impact on inference speeds across different model architectures.

Parameter Sensitivity

Figure 7 examines the sensitivity of KALAHash to its key hyper-parameters, where “#Layers” denotes the number of layers inserted by CLoRA, and “Position” means which attention matrices are adjusted by CLoRA. The results show that KALAHash is robust to parameter changes, maintaining strong performance across a wide range of values.

Inference Time

We conducted a comprehensive analysis of inference times to evaluate the computational efficiency of our proposed CLoRA method across various backbones with and without CLoRA. As shown in Table 5, the integration of CLoRA introduces minimal computational overhead across all tested architectures demonstrating that CLoRA maintains the efficiency of the original models while providing the benefits of knowledge-anchored adaptation.

Acknowledgment

This work was supported in part by Semiconductor Research Corporation JUMP 2.0 PRISM Center.

References

- Cao, Y.; Long, M.; Liu, B.; and Wang, J. 2018. Deep Cauchy Hashing for Hamming Space Retrieval. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. HashNet: Deep Learning to Hash by Continuation. In *International Conference on Computer Vision (ICCV)*.
- Chua, T.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Conference On Image And Video Retrieval (CIVR)*.
- Cohen, N.; Gal, R.; Meir, E. A.; Chechik, G.; and Atzmon, Y. 2022. "This Is My Unicorn, Fluffy": Personalizing Frozen Vision-Language Representations. In *European Conference on Computer Vision (ECCV)*.
- Doan, K. D.; Yang, P.; and Li, P. 2022. One Loss for Quantization: Deep Hashing with Discrete Wasserstein Distributional Matching. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houshy, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.
- Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning (ICML)*.
- Gui, L.; Wang, Y.; and Hebert, M. 2017. Few-Shot Hash Learning for Image Retrieval. In *International Conference on Computer Vision Workshops (ICCV Workshops)*.
- Hao, X.; Li, R.; Zhang, H.; Li, D.; Yin, R.; Jung, S.; Park, S.; Yoo, B.; Zhao, H.; and Zhang, J. 2024a. MapDistill: Boosting Efficient Camera-Based HD Map Construction via Camera-LiDAR Fusion Model Distillation. In *European Conference on Computer Vision (ECCV)*.
- Hao, X.; Wei, M.; Yang, Y.; Zhao, H.; Zhang, H.; Zhou, Y.; Wang, Q.; Li, W.; Kong, L.; and Zhang, J. 2024b. Is Your HD Map Constructor Reliable under Sensor Corruptions? In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Hao, X.; and Zhang, W. 2023. Uncertainty-Aware Alignment Network for Cross-Domain Video-Text Retrieval. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Hao, X.; Zhang, W.; Wu, D.; Zhu, F.; and Li, B. 2023. Dual Alignment Unsupervised Domain Adaptation for Video-Text Retrieval. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- He, K.; Çakir, F.; Bargal, S. A.; and Sclaroff, S. 2018. Hashing as Tie-Aware Learning to Rank. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Hoe, J. T.; Ng, K. W.; Zhang, T.; Chan, C. S.; Song, Y.; and Xiang, T. 2021. One Loss for All: Deep Hashing with a Single Cosine Similarity based Learning Objective. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.
- Jiang, Q.; and Li, W. 2018. Asymmetric Deep Supervised Hashing. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.
- Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Li, Q.; Sun, Z.; He, R.; and Tan, T. 2017. Deep Supervised Discrete Hashing. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Li, W.; Wang, S.; and Kang, W. 2016. Feature Learning Based Deep Supervised Hashing with Pairwise Labels. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*.
- Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Luo, X.; Wang, H.; Wu, D.; Chen, C.; Deng, M.; Huang, J.; and Hua, X. 2023. A Survey on Deep Hashing Methods. *ACM Transactions on Knowledge Discovery from Data (TKDD)*.
- Mu, N.; Kirillov, A.; Wagner, D. A.; and Xie, S. 2022. SLIP: Self-supervision Meets Language-Image Pre-training. In *European Conference on Computer Vision (ECCV)*.
- Ng, K. W.; Zhu, X.; Song, Y.; and Xiang, T. 2024. ConceptHash: Interpretable Fine-Grained Hashing via Concept Discovery. In *IEEE/CVF Computer Vision and Pattern Recognition Conference Workshops (CVPRW)*.
- Pan, Z.; Cai, J.; and Zhuang, B. 2023. Stitchable neural networks. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*.

- Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*.
- Shen, F.; Gao, X.; Liu, L.; Yang, Y.; and Shen, H. T. 2017. Deep Asymmetric Pairwise Hashing. In *ACM International Conference on Multimedia (ACM MM)*.
- Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised Discrete Hashing. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Snell, J.; Swersky, K.; and Zemel, R. S. 2017. Prototypical Networks for Few-shot Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Su, S.; Zhang, C.; Han, K.; and Tian, Y. 2018. Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research (JMLR)*.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep Hashing Network for Unsupervised Domain Adaptation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Wang, L.; Pan, Y.; Liu, C.; Lai, H.; Yin, J.; and Liu, Y. 2023a. Deep Hashing with Minimal-Distance-Separated Hash Centers. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Wang, Q.; Yang, X.; Lin, S.; and Geng, X. 2023b. Learn-gene: Inheriting Condensed Knowledge from the Ancestry Model to Descendant Models. *CoRR*, abs/2305.02279.
- Wang, X.; Shi, Y.; and Kitani, K. M. 2016. Deep Supervised Hashing with Triplet Labels. In *Asian Conference on Computer Vision (ACCV)*.
- Wu, D.; Su, Q.; Li, B.; and Wang, W. 2024. Pairwise-Label-Based Deep Incremental Hashing with Simultaneous Code Expansion. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Xia, R.; Pan, Y.; Lai, H.; Liu, C.; and Yan, S. 2014. Supervised Hashing for Image Retrieval via Image Representation Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Yang, E.; Wang, Z.; Shen, L.; Liu, S.; Guo, G.; Wang, X.; and Tao, D. 2024. AdaMerging: Adaptive Model Merging for Multi-Task Learning. In *International Conference on Learning Representations (ICLR)*.
- Yuan, L.; Wang, T.; Zhang, X.; Tay, F. E. H.; Jie, Z.; Liu, W.; and Feng, J. 2020. Central Similarity Quantization for Efficient Image and Video Retrieval. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.
- Zaken, E. B.; Goldberg, Y.; and Ravfogel, S. 2022. Bit-Fit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Zhao, S.; Wu, D.; Zhang, W.; Zhou, Y.; Li, B.; and Wang, W. 2020. Asymmetric Deep Hashing for Efficient Hash Code Compression. In *ACM International Conference on Multimedia (ACM MM)*.
- Zhao, S.; Wu, D.; Zhou, Y.; Li, B.; and Wang, W. 2021. Rescuing Deep Hashing from Dead Bits Problem. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Zhao, S.; and Xu, H. 2023a. Less is More: Toward Zero-Shot Local Scene Graph Generation via Foundation Models. *CoRR*, abs/2310.01356.
- Zhao, S.; and Xu, H. 2023b. NEUCORE: Neural Concept Reasoning for Composed Image Retrieval. In *UniReps, Proceedings of Machine Learning Research*.
- Zhao, S.; Zou, X.; Yu, T.; and Xu, H. 2024. Reconstruct before Query: Continual Missing Modality Learning with Decomposed Prompt Collaboration. *CoRR*, abs/2403.11373.
- Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision (IJCV)*.

Appendix

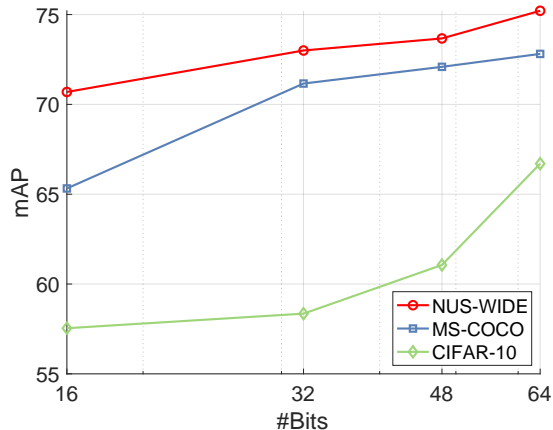


Figure 8: Performance of KALAHash as the number of bits increases from 16 to 64 on NUS-WIDE, MS-COCO, and CIFAR-10 datasets.

Scalability of the Number of Bits

Figure 8 presents a comprehensive analysis of KALAHash’s performance as the number of bits increases from 16 to 64 on NUS-WIDE, MS-COCO, and CIFAR-10. As the number of training samples increases, our approach consistently improves the retrieval performance, which demonstrates its ability to scale up the number of bits.

Silhouette Score

Table 6 shows the results of the Silhouette Score. KALAHash consistently outperforms baseline methods, demonstrating the effectiveness of our proposed method. Besides, we notice that methods using pair-wise loss achieve a higher score than those using point-wise loss. The reason may be that contrastive loss has a stronger ability to push hash codes belonging to different classes to different locations in the embedding space.

Parameter Sensitivity

Figure 9 examines the sensitivity of KALAHash to γ . KALAHash maintains relatively high mAP scores when $\gamma \leq 3$. There is a noticeable drop in performance when $\gamma \geq 4$, indicating that extremely high values may affect the optimization progress, leading to a suboptimal result.

PR Curve

Figure 10 illustrates the Precision-Recall (PR) curves for KALAHash and baseline methods on NUS-WIDE, MS-COCO, and CIFAR-10 datasets. These curves provide a comprehensive view of the model’s performance across different precision and recall thresholds. The results demonstrate that KALAHash consistently exhibits competitive performance across all three datasets, maintaining a good balance between precision and recall. The consistent performance across varied datasets highlights the versatility and robustness of our proposed method.

Method	Silhouette Score
HashNet (Cao et al. 2017)	57.48
DSDH (Li et al. 2017)	57.50
DCH (Cao et al. 2018)	55.96
GreedyHash (Su et al. 2018)	52.25
CSQ (Yuan et al. 2020)	52.36
OrthoHash (Hoe et al. 2021)	51.84
HSWD (Doan, Yang, and Li 2022)	55.54
MDSH (Wang et al. 2023a)	52.07
KALAHash	59.76

Table 6: Silhouette Scores for various hashing methods on CIFAR-10.

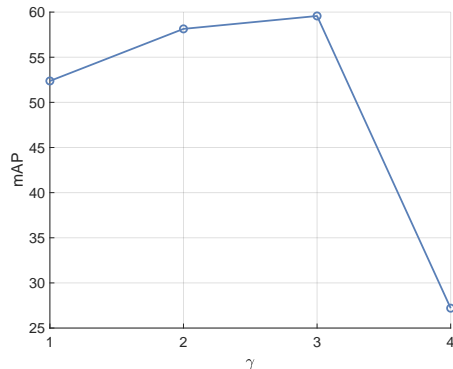


Figure 9: Sensitivity analysis of KALAHash with respect to γ on CIFAR-10.

Various Backbones

Pretrained Backbones are vital for downstream tasks (Ng et al. 2024; Hao et al. 2024a,b). Table 7 presents the performance of various baseline methods across different backbone architectures. We evaluate the methods using ResNet-18 (He et al. 2016), ResNet-50 (He et al. 2016), ImageNet21k ViT-B/32 (Dosovitskiy et al. 2021), and CLIP ViT-B/32 (Radford et al. 2021) as backbone networks. Notably, the performance improves when moving from CNN-based architectures (ResNet-18 and ResNet-50) to transformer-based architectures (ImageNet21k ViT-B/32 and CLIP ViT-B/32). This improvement is particularly pronounced with the CLIP ViT-B/32 backbone, which is trained on a large corpus containing image-text pairs.

Time Complexity Analysis

To delve deeper into the time complexity of KALAHash for highlighting the efficiency and scalability of our approach, we provide the inference time compared to LoRA (Hu et al. 2022), Bias Tuning (Zaken, Goldberg, and Ravfogel 2022), and Prompt Tuning (Zhou et al. 2022). From Table 8, LoRA and Bias Tuning do not alter the architecture or inputs. They do not introduce any overhead. Prompt Tuning adds learnable tokens, resulting in a slight increase in computational time. Our KALAHash does introduce some overhead in Equation (6) and Equation (7), but these operations are simple and add negligible inference time (0.01 – 0.05

Method	ResNet-18	ResNet-50	ImageNet21k	ViT-B/32	CLIP ViT-B/32
HashNet (Cao et al. 2017)	15.25	23.58	31.35		41.68
DSDH (Li et al. 2017)	14.27	16.24	24.50		44.22
DCH (Cao et al. 2018)	16.17	19.08	21.79		39.53
GreedyHash (Su et al. 2018)	16.01	19.26	24.24		44.87
CSQ (Yuan et al. 2020)	15.01	18.10	22.59		46.66
OrthoHash (Hoe et al. 2021)	16.01	19.73	27.56		46.68
HSWD (Doan, Yang, and Li 2022)	14.86	23.20	30.70		48.63
MDSH (Wang et al. 2023a)	16.19	18.08	24.28		47.33

Table 7: Comparison of different hashing methods using various backbone architectures.

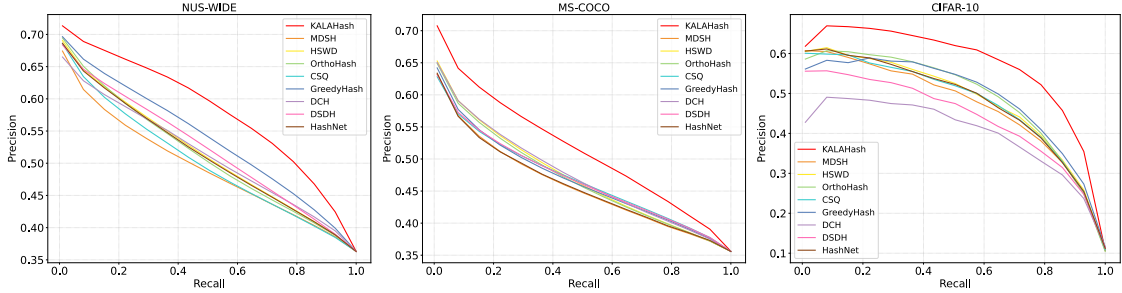


Figure 10: Precision-Recall curves for KALAHash and baseline methods on NUS-WIDE, MS-COCO, and CIFAR-10 datasets.

Variants	Inference Time (ms/per query)
LoRA	1.16
Bias Tuning	1.16
Prompt Tuning	1.19
CLoRA ($N = 10$)	1.17
CLoRA ($N = 1,000$)	1.18
CLoRA ($N = 100,000$)	1.26

Table 8: Inference time (ms) per image comparison of PEFT techniques.

Method	mAP@1000
<i>ImageNet-100</i>	
MDSH (Wang et al. 2023a)	24.69
KALAHash	30.77
<i>CUB-200</i>	
ConceptHash (Ng et al. 2024)	1.65
KALAHash	9.54

Table 9: mAP@1000 on ImageNet-100 and CUB-200.

ms) across various architectures. We compare the inference times of KALAHash with other PEFT techniques, considering the knowledge pool size N , to demonstrate its efficiency.

More Results

Following Cao et al. (2017), we report mAP@1000 on the 1-shot ImageNet-100 dataset. We also compare our method to ConceptHash (Ng et al. 2024), the best paper award of CVPRW24 FGVC11, on the 1-shot CUB-200 dataset. The results are shown in Table 9.