Spatial-Aware Image Retrieval: A Hyperdimensional Computing Approach for Efficient Similarity Hashing

Sanggeon Yun[1], Ryozo Masukawa[1], SungHeon Jeong[1], and Mohsen Imani[1, †]

[1]University of California, Irvine
[†]Corresponding author, email: m.imani@uci.edu

***Abstract*—In the face of burgeoning image data, efficiently retrieving similar images poses a formidable challenge. Past research has focused on refining hash functions to distill images into compact indicators of resemblance. Initial attempts used shallow models, evolving to attention mechanism-based architectures from Convolutional Neural Networks (CNNs) to advanced models. Recognizing limitations in gradient-based models for spatial information embedding, we propose an innovative image hashing method, *NeuroHash* leveraging Hyperdimensional Computing (HDC). HDC symbolically encodes spatial information into high-dimensional vectors, reshaping image representation. Our approach combines pre-trained large vision models with HDC operations, enabling spatially encoded feature representations. Hashing with locality-sensitive hashing (LSH) ensures swift and efficient image retrieval. Notably, our framework allows dynamic hash manipulation for conditional image retrieval. Our work introduces a transformative image hashing framework enabling spatial-aware conditional retrieval. By seamlessly combining DNN-based neural and HDC-based symbolic models, our methodology breaks from traditional training, offering flexible and conditional image retrieval. Performance evaluations signify a paradigm shift in image-hashing methodologies, demonstrating enhanced retrieval accuracy.**

I. Introduction

In the era of explosive growth in image data, managing vast repositories of images, particularly in domains requiring the swift retrieval of similar images for a given query, presents an escalating challenge. Numerous research endeavors have sought to develop efficient and accurate methods for similar image retrieval. The primary focus of these investigations has been the design of adept hash functions capable of transforming images into a compact, fixed-size hash, thereby encapsulating their similarity to other images.

One early research utilized shallow machine learning models such as support vector machine (SVM) to extract discrete features from each image to hashing images [28]. As deep neural networks (DNNs) show remarkable performance on various image-based tasks, image hashing models for image retrieval based on neural networks are proposed starting from Convolutional Neural Networks (CNNs) based approaches [2], [36]. This momentum of applying DNNs to image retrieval tasks evolved towards purely attention mechanism-based models [3]. Nowadays, state-of-the-art models on image retrieval tasks only exhibit end-to-end DNN-based architectures.

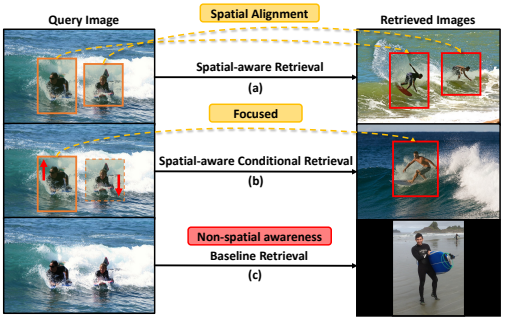

Fig. 1. **The actual image retrieval results comparing our framework's (a) spatial-aware retrieval and (b) conditional retrieval with (c) baseline retrieval**.

Despite the strides made by previous deep hashing-based methods which are using gradient-based end-to-end deep learning models with specifically designed loss functions to capture global and local information of images, this black-box manner training does not guarantee to embedding of desired information including local or spatial information. Furthermore, since such end-to-end deep learning models are trained with predetermined criteria, it has fundamental limitations from their nature on conducting image retrieval in a flexible way with additional conditions such as precise positioning of each object or the prioritization of specific objects during image retrieval.

To resolve the above limitations of previous methods, we propose an innovative image hashing method employing Hyperdimensional Computing (HDC) [13] to facilitate image retrieval with spatial structural conditions that can be easily manipulated as illustrated in Figure 1. HDC stands as an alternative paradigm inspired by essential brain functions, emphasizing high efficiency and symbolic learning capabilities. Grounded in the observation that the human brain excels in manipulating high-dimensional representations, our approach harnesses HDC operations to embed spatial structural information into a high-dimensional vector in a neuro-symbolic manner, constituting a hashed representation of the image.

Our methodology capitalizes on pre-trained large vision models to extract feature representations for individual objects, subsequently combining them into a singular representation of high-dimensional vectors through spatial encoding – a process applying HDC operations with positional information. These

representations are then hashed using the locality-sensitive hashing (LSH) [8] method to facilitate rapid image retrieval. During the retrieval process, our method replicates the spatial encoding procedure to retrieve images with similar spatial structures. Additionally, structural conditions can be controlled by incorporating HDC operations on a given query image, such as focusing on spatial information of a specific object.

In summary, our work represents a fundamentally novel contribution to the field, offering the following key advancements:

- To the best of our knowledge, we propose *NeuroHash* a novel image hashing framework fully capable of spatial-aware image retrieval. Unlike previous works on image hashing that focus on gradient-based end-to-end deep learning which depends on black box manner information embedding, our solution symbolically embedds spatial information by exploiting HDC.
- By combining DNN-based neural models with HDC-based symbolic models, our framework is capable of flexible hash value manipulation to have conditional image retrieval in a neuro-symbolic manner such as focusing on spatial information of a specific object.
- Experimental results on two benchmark datasets demonstrate that our *NeuroHash* achieves a remarkable retrieval performance compared to state-of-the-art hashing methods by outperforming mAP@5K scores. We also demonstrate spatial-aware and conditional image retrieval using our proposed metric mAP@5K$_r$ designed to measure spatial alignments with retrieved images.

II. Related Works

A. Hash-based Approximate Nearest Search

Retrieving similar vectors efficiently from abundant vector data using linear search or traditional structures is impractical. To address this, studies explore converting high-dimensional vectors into fixed-size, low-dimensional representations, with Locality Sensitive Hashing (LSH) [8] being a notable unsupervised algorithm [29]. LSH constructs a hash table using multiple functions capturing local similarity, and variations like Multilinear Hyperplane Hashing [23] specifically preserve cosine similarity in the hash space.

B. Deep Hashing Approach

In early image retrieval, methods such as supervised discrete hashing (SDH) [28] played a crucial role in reducing storage and improving retrieval speed. The integration of Convolutional Neural Networks (CNNs) brought advancements with models such as HashNet [2], building on architectures like AlexNet [16] to address discrete optimization challenges.

The evolution shifted towards deep learning, leveraging ResNet as a popular backbone network in approaches like CSQ [40] and DBDH [41]. As models progressed, attention turned to hybrid models like DAgH [4] and DAHP [17], using attention networks to enhance performance without increasing convolution layers. Scalability concerns led to exploration of a self-attention-based structure [3].

Recent developments expanded unsupervised deep hashing into applications like image copy detection [20] and image quality assessment [11]. Methods like DeepBit [18], Distill-Hash [38], and TBH [31] explored unsupervised learning with novel loss functions. Contrastive learning in computer vision paved the way for unsupervised hashing methods such as HAMAN [24] and MeCoQ [34], leveraging contrasting positive and negative samples for robust hash codes.

Certain unsupervised hashing methods focused on mining pairwise similarity. DistillHash [38] and SSDH [37] used data pair distillation and semantic structures, while FSCH [1] extended these approaches with fine-grained similarity structures based on global and local image representations.

C. Hyperdimensional Computing

Brain-inspired hyperdimensional computing (HDC) is based on the understanding that brains compute with patterns of neural activity that are not readily associated with numbers. Due to the huge size of the brain's circuits, neural patterns can be modeled with hypervectors [13]. HDC builds upon a well-defined set of operations with random hypervectors, is extremely robust in the presence of failures, and offers a complete computational paradigm that is easily applied to multiple learning problems, such as speech recognition [12], graph learning [14], [25], and computer vision [7], [10].

Recent literature has witnessed a growing interest in hyperdimensional computing (HDC) as a learning model, praised for its simplicity and computational efficiency. However, conventional HDC frameworks encounter issues with randomly generated and static encoders, leading to an abundance of parameters and decreased accuracy. LeHD [6], an innovative approach, employs a principled learning approach to refine model accuracy, transforming the HDC framework into an equivalent binary neural network architecture. These advancements collectively aim to overcome issues with static encoders in HDC, offering a more effective and accurate learning framework.

III. Methodology

A. HDC Basics

The core of HDC is called a hyperdimensional vector, denoted $\mathcal{H}$, which represents a vector in $\mathbb{R}^D$ with a high dimensionality of D. Hyperdimensional vectors are compared using a similarity function δ. By using this similarity measure, HDC becomes a versatile tool for cognitive tasks, including memory, classification, clustering, etc. HDC frameworks designed to support these tasks are based on three core operations that mirror brain functionalities: bundling, binding, and permutation. Here are the details of each operation:

1) **Bundling**: This operation, represented by $+$, is commonly executed as element-wise addition. If $\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2$, then both $\mathcal{H}_1$ and $\mathcal{H}_2$ exhibit similarity to $\mathcal{H}$. In terms of cognitive interpretation, this operation can be understood as a form of memorization.
2) **Binding**: This operation, denoted by $*$, is usually implemented as an element-wise multiplication. If $\mathcal{H} = \mathcal{H}_1 *$

$\mathcal{H}_2$, then $\mathcal{H}$ is dissimilar to both $\mathcal{H}_1$ and $\mathcal{H}_2$. Binding has a crucial property of similarity preservation, where for some hypervector $\mathcal{V}$, $\delta(\mathcal{V} * \mathcal{H}_1, \mathcal{V} * \mathcal{H}_2) \simeq \delta(\mathcal{H}_1, \mathcal{H}_2)$. From a cognitive point of view, this operation can be understood as an association. Binding can be used to associate different pieces of information, such as coordinates and image feature vectors, in hyperdimensional space.

3) **Permutation**: This operator, represented by ρ, is commonly executed as a rotation of vector elements. In general, $\delta(\rho(\mathcal{H}), \mathcal{H}) \simeq 0$. Permutation is frequently employed to encode the order within sequences.

Leveraging the three fundamental HDC operations provides a foundation for a hyperdimensional learning framework applicable to various tasks. In the context of classification, each step of the framework can be outlined as follows.

1) **Encoding**: The initial step within the HDC framework involves mapping the input data $\vec{F} \in U$ into a high-dimensional space through the introduction of an encoding function $\vec{\phi} : U \rightarrow H$, commonly known as *encoding*. Consider an input vector with n features, denoted as $\vec{F} = \{f_i\}$, representing features extracted from an image. The commonly used encoding function is defined as $\vec{\phi}(\vec{F}) = \cos(\vec{F} \times \vec{B} + \vec{b}) \times \sin(\vec{F} \times \vec{B})$, where $\vec{B}$ is an $n \times D$ matrix, and each element in $\vec{B}$ is sampled from an i.i.d Gaussian distribution with parameters ($\mu = 0, \sigma = 1$). Additionally, $\vec{b}$ is sampled from an i.i.d uniform distribution over the interval $[0, 2\pi]$. The $\vec{\phi}$ function preserves a notion of similarity in the input space. Consequently, for any given inputs $\vec{x}_1, \vec{x}_2 \in U$, their corresponding hypervectors, $\vec{\phi}(\vec{x}_1)$ and $\vec{\phi}(\vec{x}_2)$, exhibit similarity iif $\vec{x}_1$ is similar to $\vec{x}_2$. Such initialized encoders with parameters $\vec{B}$ and $\vec{b}$ can be further optimized by making $\vec{B}$ and $\vec{b}$ learnable parameters using a gradient descent approach.
2) **Symbolic Training**: Consider a dataset $\mathcal{D} \subset U$ where each data point $\vec{x}_i \in D$ is associated with a label $1 \leq y_i \leq m$ from a set of m classes. In traditional hyperdimensional classifier training, the process involves generating m class hypervectors through bundling: $\vec{C}_i = \sum_{y_j=i} \vec{\phi}(\vec{x}_j)$. For each data point to retrain $\vec{x}_i$, each class hypervector is updated as follows:

$$\vec{C}_l \leftarrow \vec{C}_l + \eta(1-\delta)\vec{\phi}(\vec{x}_i)$$
$$\vec{C}_{l'} \leftarrow \vec{C}_{l'} - \eta(1-\delta)\vec{\phi}(\vec{x}_i)$$

where $l = y_i, l' \neq y_i, \delta = \delta(\vec{C}_l, \vec{\phi}(\vec{x}_i))$, and η is learning rate.
3) **Symbolic Inference**: Once the class hypervectors $\vec{C}_i$ undergo updates through the initial training phases, the classification of a given query $\vec{q} \in D$ becomes a straightforward process. A class i is predicted when $\delta(\vec{C}_i, \vec{\phi}(\vec{q})) > \delta(\vec{C}_j, \vec{\phi}(\vec{q}))$ is satisfied for all $j \neq i$.

B. Proposed Framework

1) Overall Pipeline: The overall pipeline of our proposed framework is presented in Figure 2. First, given an image I, we extract global features, which is embedding of the image, through a pre-trained image encoder model by giving the entire given image to the model (❶). Also, in order to consider local information, it extracts bounding boxes indicating objects that are presented in the image by conducting an object detection task over the image using a pre-trained object detection model (❷). Using the bounding boxes that are generated during the previous step, it extracts two types of object information: object images I_k and object positions $p_k = (x_k, y_k)$ (❸). With the extracted object images I_k, visual features $\vec{f}_k$ corresponding to each of the object images I_k are computed by using the same pre-trained image encoder model that is used during the global features extraction (❹). The resulting visual features including global and local information are sent to an HDC encoder to have visual feature hypervectors (❺). Not only considers the local visual information, to further consider local spatial information, but it also conducts spatial encoding using two positional base hypervectors each corresponding to x and y coordinate, and computes positional hypervectors $\vec{h}_j^{p_k} \in \mathbb{C}$ (❻). Finally, it combines local visual features with local positional features to have spatial embedding addition to that, it also combines global feature hypervector to have global embedding (❼). After the final hypervector embedding process, hyperdimensional representation corresponding to the given image I is generated. The generated hyperdimensional representation is now hashed into a compact binary hash representation with the optimized multilinear hyperplane hashing model $\vec{f}_h(.)$ (❽).

2) Global and Local Visual Features Extraction: In an image retrieval task, it is crucial to well-represent each image in a compact representation. Although pre-trained large image embedding models introduced so far present powerful performance in extracting visual features, simply embedding entire images can lead to insufficient interpretation of local information considering the complexity of image data. To allow solid local visual information consideration, we propose to employ a pre-trained object detection model in order to extract objects that are presented in a given image. Therefore, our proposed framework uses two pre-trained large image models: 1) object detection model $f_{obj} : \mathcal{I} \rightarrow \mathcal{B}, \mathcal{B} \subseteq \mathbb{R}^{N \times 4}$ where $\mathcal{I}$ indicates input image space and $\mathcal{B}$ indicates a set of bounding boxes each contains position and the size of the box and, 2) image embedding model $\vec{\phi}_{vis} : \mathcal{I} \rightarrow \mathcal{Z}, \mathcal{Z} \subseteq \mathbb{R}^z$ where $\mathcal{Z}$ indicates embedding space with z dimensionality. First, global visual feature vector $\vec{f}_{glob} \in \mathbb{R}^z$ is easily retrieved by passing entire image I to $\vec{\phi}_{vis}$. In the case of the local visual features extraction, f_{obj} needs to be utilized before retrieving features. By applying object detection f_{obj} over the given image $I \in \mathcal{I}$, we can obtain N bounding boxes $BB_k \in \mathbb{R}^4$ that correspond to each object in the image. Using each BB_k, local visual features $\vec{f}_k$ can be retrieved by applying cropped images I_k based on BB_k to $\vec{\phi}_{vis}$.

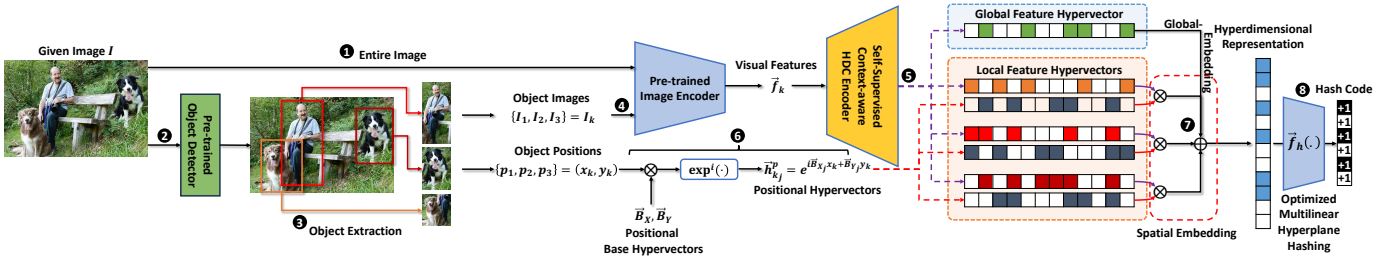

Fig. 2. **Overall pipeline of our proposed *NeuroHash* a novel framework for spatial-aware hashing and conditional image retrieval.**

3) Context-aware HDC Encoding: Inspired by the previous work LeHD [6], we designed an HDC encoder $\vec{\phi}\colon \mathbb{R}^z \rightarrow \mathbb{R}^D$ ($D \gg z$) that is capable of encoding visual features into a feature hypervector $\vec{h}_k^f$ in hyperspace while preserving contextual information in given images by enabling it to be trainable in a gradient descent-based self-supervised way. The encoder model $\vec{\phi}$ consists of two major modules: $E_{ext}\colon \mathbb{R}^z \rightarrow \mathbb{R}^{z'}$ where $z > z'$ and $E_{gen}\colon \mathbb{R}^{z'} \rightarrow \mathbb{R}^D$. For a given visual feature vector $\vec{f}_k$, $\vec{\phi}$ is applied as $\vec{h}_k^f = \vec{\phi}(\vec{f}_k) = E_{gen}(E_{ext}(\vec{f}_k)) \in \mathbb{R}^D$. E_{ext} acts as important visual feature extractor and E_{gen} acts as a mapper to hyperspace. To harness general context representation, we limit the dimensionality by z', which prevents overfitting. We train the function $\vec{\phi}$ using the following loss function: $\mathcal{L}_{Enc} = \mathcal{L}_c + \lambda_{rec}\mathcal{L}_{rec}$ where $\lambda_{rec} \in \mathbb{R}$ indicates balance coefficient. $\mathcal{L}_c$ uses pairs of M object images I_k and corresponding pseudo-labels $\tilde{y}_k$ that are generated using the pre-trained object detection model f_{obj} as shown in Equation 1 where $C \in \mathbb{R}^{D\times c}$ stands as class hypervectors with c pseudo-classes. A simple linear layer module $E_{rec}\colon \mathbb{R}^D \rightarrow \mathbb{R}^z$ is introduced in order to compute the reconstruction loss $\mathcal{L}_{rec}$ as shown in Equation 2 to force the model $\vec{\phi}$ to preserve original features' information in hyperspace.

$$\mathcal{L}_c = \sum_k \mathit{CrossEntropy}(\mathit{softmax}(\vec{\phi}(\vec{f}_k))^T C), \tilde{y}_k) \quad (1)$$

$$\mathcal{L}_{rec} = \frac{1}{M}\sum_k \left\| \vec{f}_k - E_{rec}(\vec{\phi}(\vec{f}_k)) \right\|^2 \quad (2)$$

4) Hyperdimensional Spatial-aware Encoding: Given global feature hypervector $\vec{h}_{glob}^f$ and local feature hypervectors $\vec{h}_k^f$ with their corresponding object positions $p_k = (x_k, y_k)$ final hyperdimensional representation $\vec{H}$ of the given image I is computed using HDC operations. First, to encode position information p_k, positional base hypervectors $\vec{B}_X$ and $\vec{B}_Y$ are randomly sampled from a normal distribution $\{\mathcal{N}(0,1)\}^D$. With the randomly sampled positional base hypervectors $\vec{B}_X$ and $\vec{B}_Y$, each x_k and y_k are projected to the hyperspace using $\vec{B}_X$ and $\vec{B}_Y$ respectively with $\exp^i(.)$ function where i indicates imaginary unit ($i = \sqrt{-1}$). Each embedding of the x-axis and y-axis dimensional information is computed by $\vec{h}_k^X = \exp^i\left(x_k \vec{B}_X\right) = \left[e^{i\vec{B}_{X_1}x_k} \quad e^{i\vec{B}_{X_2}x_k} \quad \dots \quad e^{i\vec{B}_{X_D}x_k}\right] \in \mathbb{C}^D$ and $\vec{h}_k^Y = \exp^i\left(y_k \vec{B}_Y\right) = \left[e^{i\vec{B}_{Y_1}y_k} \quad e^{i\vec{B}_{Y_2}y_k} \quad \dots \quad e^{i\vec{B}_{Y_D}y_k}\right] \in \mathbb{C}^D$ respectively. Final positional hypervectors $\vec{h}_k^p = \vec{h}_k^X * \vec{h}_k^Y = e^{i\vec{B}_{X_j}x_k + i\vec{B}_{Y_j}y_k} \in \mathbb{C}^D$ are computed by combining the two hypervectors $\vec{h}_k^X$ and $\vec{h}_k^Y$ using the binding operation to associate both (x, y)-axis dimensional information.

Additionally, we can introduce a new hyperparameter length scale w. The length scale acts like a factor that controls the standard deviation of $\vec{B}_X$ and $\vec{B}_Y$ by being placed in the $\exp^i(.)$ function as $\exp^{i/w}(.)$. With the smaller w, it affects $\vec{B}_X$ and $\vec{B}_Y$ are sampled from a normal distribution with higher standard deviation $\{\mathcal{N}(0, \frac{1}{w})\}^D$ creating sparse representation of positional hypervectors. While larger w affects $\vec{B}_X$ and $\vec{B}_Y$ are sampled from smaller standard deviations making representation of positional hypervectors more dense. Thus, by controlling w, we can adjust the magnitude of the association of spatial information.

Now, to have the final hyperdimensional representation, positional hypervectors are combined with the visual feature hypervectors that are retrieved from the global and local visual features extraction process. Each local visual feature vector $\vec{h}_k^f \in \mathbb{R}^D$ is paired with the corresponding positional hypervector $\vec{h}_k^p \in \mathbb{C}^D$. Each pair $(\vec{h}_k^f, \vec{h}_k^p)$ is associated with each other resulting in a single hypervector by the following binding operation: $\vec{h}_k^f * \vec{h}_k^p \in \mathbb{C}^D$ represents visual and positional information. Lastly, Spatial embedding by bundling $\vec{h}_k^f * \vec{h}_k^p$ for all $k = 1, 2, \cdots, N$ and global embedding by bundling $\vec{h}_{glob}^f$ with $\sum_k \vec{h}_k^f * \vec{h}_k^p$ are conducted resulting the final hyperdimensional representation $\vec{H} = \vec{h}_{glob}^f + \sum_k \vec{h}_k^f * \vec{h}_k^p$.

Furthermore, we can utilize Symbolic Training shown in subsection III-A where we merge separate symbolic representations into a single hypervector and optimize by giving weights to each symbolic hypervector to have a user desire hyperdimensional representations: $\vec{H} = \eta_{glob}\vec{h}_{glob}^f + \sum_k \eta_k \vec{h}_k^f * \vec{h}_k^p$ where $\eta_{glob} \in \mathbb{R}$ indicates weight on global features and $\eta_k \in \mathbb{R}$ indicates weight on each local features. Relatively higher weight η leads to focused hyperdimensional representation for those highly weighted symbols. Simply manipulating η_k, we can have a new customized representation that can be used for conditional image retrieval without any heavy and time-consuming gradient-based optimization. Possible ways to automate assigning η_k during hashing massive amounts of images are: by the size of bounding box BB_k, confidence score from the object detection model f_{obj}, etc. In this paper, we focus on the evaluation of manipulating query images' representation to have conditional image retrieval thus, we set the same amount of $1 = \eta_{glob} = \eta_k, \forall k$ during the hashing retrieval set.

5) Multilinear Hyperplane Hashing Optimization: We explored that by utilizing HDC operations in hyperspace, hyperdimensional representation $\vec{H}_n \in \mathbb{C}^D$ that well represents both spatial-aware local context and global context of a given image $I_n \in \mathcal{I}$ can be driven. However, due to the high dimensionality $D \gg z$ and the high precision for representing each element $\vec{H}_{ni} \in \mathbb{C}$ it is infeasible to adapt the hyperdimensional representations in fast image retrieval task directly. Thus, it is necessary to have a hash function $\vec{f}_h \colon \vec{H} \mapsto \vec{H}'$ which maps given hyperdimensional representation $\vec{H}$ to a compact L-bit hash representation $\vec{H}' \in \{-1, +1\}^L$ that well preserves relationships in hyperspace within low-dimensional hamming space.

To have a well-performing hash function $\vec{f}_h$, we utilize the locality-sensitive hashing (LSH) method by making it trainable in a gradient descent way. Among various different variations of LSH, we target to optimize random multilinear hyperplane hashing [23] method that is specifically designed to preserve relationships in cosine similarity. The model initialization is similarly conducted as the previous work by randomly sampling $p_{ij} \sim \mathcal{N}(0, 1)$. Each $\vec{p}_i \in \mathbb{R}^{2D}$ represents a randomly sampled hyperplane that lies on the hyperspace dimensionality of $2D$. Notably, the hyperplanes lie on $2D$-dimensional space, not D, as a result of placing the hyperdimensional representation $\vec{H}_n$ which consists of complex numbers to real number space $\mathbb{R}^{2D}$ by concatenating real and imaginary parts $\Re(\vec{H}_n)^\frown\Im(\vec{H}_n) \in \mathbb{R}^{2D}$. Each hyperplane assigns a single bit value to each hyperdimensional data point by dividing them into two. Using a function *sign*(.) that returns $+1$ if a given value $x \in \mathbb{R}$ is larger or equal to 0 otherwise returns -1, this can be represented as $\vec{H}'_{ni} = \textit{sign}(\vec{p}_i \cdot \Re(\vec{H}_n)^\frown\Im(\vec{H}_n)) \in \{+1, -1\}$. As two hypervectors are divided into more common sides by hyperplanes they are considered to be also similar in the original hyperspace in terms of cosine similarity.

To optimize randomly sampled hyperplanes from a normal distribution, we generalized our hashing function as $f_h(H) = \tanh\left(HP^T + b\right) \in [-1, +1]^{M \times L}$ where $H \in \mathbb{R}^{M \times 2D}$ indicates given M concatenated hyperdimensional representations $H_n = \Re(\vec{H}_n)^\frown\Im(\vec{H}_n)$, $P \in \mathbb{R}^{L \times 2D}$ indicates L hyperplanes, and $b \in \mathbb{R}^L$ indicates bias. We used $\tanh(.)$ function instead of *sign*(.) to avoid indifferentiable characteristic of *sign*(.). Thus, the final L-bit binary representation needs to be retrieved by $B = \textit{sign}(H') \in \{+1, -1\}^{M \times L}$ where $H' = f_h(H)$. Finally, we introduce the loss function that is formulated as the following:

$$\mathcal{L}_{Hyper} = \lambda_{mse}\mathcal{L}_{mse} + \lambda_w\mathcal{L}_w + \lambda_q\mathcal{L}_q + \lambda_u\mathcal{L}_u + \lambda_o\mathcal{L}_o \quad (3)$$

The loss function that is shown in Equation 3 consists of 5 loss terms: mean square error (MSE) loss $\mathcal{L}_{mse}$, w-shape loss $\mathcal{L}_w$, quantization loss $\mathcal{L}_q$, uniform loss $\mathcal{L}_u$, and order loss $\mathcal{L}_o$. Each loss term has its balance coefficients: λ_{mse}, λ_w, λ_q, λ_u, and λ_o. These terms are formulated in order to resolve four issues:

a) Loss term for numerical correspondence.: First of all, the MSE loss term $\mathcal{L}_{mse}$ is used to match the numerical similarity between cosine similarity in hyperspace and L-precision hamming distance as shown in Equation 4.

$$\mathcal{L}_{mse} = \frac{1}{M^2} \sum_{1 \le i,j \le M} \left\| \frac{H_i H_j^T}{\|H_i\|\|H_j\|} - \frac{H'_i H'^T_j}{L} \right\|^2 \quad (4)$$

To match the hamming distance value with the similarity value, we used reversed hamming distance: $L - \sum_{1 \le k \le L} |B_{i_k} - B_{j_k}| = B_i B_j^T \approx H'_i H'^T_j$. We further adjusted the range by $-1 \le \frac{H'_i H'^T_j}{L} \le +1$ as the same as the range of cosine similarity value.

b) Loss terms for limited representation.: Due to the low precision bits representation, distance is also extremely discrete which makes indistinguishable distances between many images. To tackle this issue, we set an assumption that in most cases, boundary distance is not placed among distances that are located on either side of the edges – either distance is very close or very far. Based on this assumption, we applied another loss term we named w-shape loss presented in Equation 5. This loss function gives more penalty for the distances that are more closely located in the center. In the same context of low precision and low dimensionality, it also can cause limited unique representations. To avoid such representation collapsing, we also introduced uniform loss as shown in Equation 6. Note that $\mathbf{1}_L$ indicates L-dimension vector consists of ones. $(H'_i \mathbf{1}_L^T)^2$ will be closer to zero as the number of $+1$s and the number of -1s gets closer which forces the hashing model to generate a uniform number of binary representations.

$$\mathcal{L}_w = \frac{1}{M^2} \sum_{1 \le i,j \le M} \left(\frac{H'_i H'^T_j}{L} + 1 \right)^2 \left(\frac{H'_i H'^T_j}{L} - 1 \right)^2 \quad (5)$$

$$\mathcal{L}_u = \frac{1}{M} \sum_{1 \le i \le M} (H'_i \mathbf{1}_L^T)^2 \quad (6)$$

c) Loss term for learning binary representations.: Next, since we are using $\tanh(.)$ function instead of *sign*(.) we also applied quantization loss as shown in Equation 7. This loss term helps the hashing model f_h to generate representations that are close to the binary representation such as $H'_{ij} \approx \textit{sign}(H'_{ij})$.

$$\begin{aligned} \mathcal{L}_q &= \frac{1}{NL} \sum_{1 \le i \le M} \sum_{1 \le j \le L} \left(H'_{ij} - \textit{sign}(H'_{ij}) \right)^2 \\ &= \frac{1}{NL} \sum_{1 \le i \le M} \sum_{1 \le j \le L} \left(H'_{ij} - B_{ij} \right)^2 \end{aligned} \quad (7)$$

d) Loss term for reversed relative order.: For the last loss term, we consider the relative orders between hyperdimensional representation pairs. It targets to preserve the order of ranking that each H_i has to all the other H_j. In other words, for the pair (i, j), the number of $k \in \{1, 2, \cdots, M\}$ that satisfies $\delta(H_i, H_j) > \delta(H_i, H_k)$, should be similar as the number of $k \in \{1, 2, \cdots, M\}$ that satisfies $\delta(H'_i, H'_j) > \delta(H'_i, H'_k)$. Equation 8 shows a loss term that gives a penalty in such cases that the number of k that satisfies the above condition is not satisfied we named order loss. The function $C(.,.)$ determines

Methods	References	CIFAR10			MS COCO		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
AGH [22]	ICML11	0.333	0.357	0.358	0.596	0.625	0.631
ITQ [9]	TPAMI12	0.305	0.325	0.349	0.598	0.624	0.648
DGH [21]	NeurIPS14	0.335	0.353	0.361	0.613	0.631	0.638
SGH [5]	ICML17	0.435	0.437	0.433	0.594	0.610	0.618
BGAN [32]	AAAI18	0.525	0.531	0.562	0.645	0.682	0.707
GreedyHash [33]	NeurIPS18	0.448	0.473	0.501	0.582	0.668	0.710
DVB [30]	IJCV19	0.403	0.422	0.446	0.570	0.629	0.623
TBH [31]	CVPR20	0.497	0.524	0.529	0.706	0.735	0.722
CIB [27]	IJCAI21	0.547	0.583	0.602	0.737	0.760	0.775
HAMAN [24]	IJCAI22	-	-	-	0.722	0.775	0.787
NSH [39]	IJCAI22	0.706	0.733	0.756	0.746	0.774	0.783
FSCH [1]	TCSVT23	**0.876**	0.912	0.926	0.760	0.787	0.799
naïve (DINOv2 + LSH)		0.316	0.450	0.599	0.479	0.557	0.658
NeuroHash ($w = 1.0$)	**Ours**	0.839	**0.937**	0.927	**0.785**	**0.878**	**0.904**
NeuroHash ($w = 10.$)	**Ours**	0.827	0.912	**0.945**	0.780	0.873	0.903

TABLE I
MAP@5K RESULTS FOR DIFFERENT METHODS ON DATASETS CIFAR10 AND MS COCO.

Without	mAP@5K
$\mathcal{L}_{Hyper}$	0.623
$\mathcal{L}_{mse}$	0.505
$\mathcal{L}_{w}$	0.940
$\mathcal{L}_{q}$	0.933
$\mathcal{L}_{u}$	0.894
$\mathcal{L}_{o}$	0.942
Full Model	**0.945**

TABLE II
ABLATION STUDIES ON MULTILINEAR HYPERPLANE HASHING OPTIMIZATION OF *NeuroHash* USING THE MAP@5K METRIC ON THE CIFAR10 DATASET FOR 64 BITS WITH THE SCALE FACTOR OF $w = 10$. THE TEST IS DONE BY REMOVING EACH LOSS TERM INCLUDING THE CASE WHERE USING ONLY RANDOM HYPERPLANES ($\mathcal{L}_{Hyper}$).

order reversed cases. If the order is reduced, it gives a larger penalty to the larger hamming distance and if the order is increased, it gives a larger penalty to the smaller hamming distance otherwise, it gives zero penalty.

$$\mathcal{L}_o = \frac{1}{M^2} \sum_{1 \leq i,j \leq M} C(i,j) \left[\left(1 - \frac{{H'}_i {H'}_j^T}{L}\right)^2 \quad \left(1 + \frac{{H'}_i {H'}_j^T}{L}\right)^2 \right]^T \tag{8}$$

$$C(i,j) = \begin{cases} \begin{bmatrix} 1 & 0 \end{bmatrix} & \text{if } j \in \left\{ v \middle| |\{k| \frac{H'_i H'^T_v}{L} > \frac{H'_i H'^T_k}{L}\}| < |\{k| \frac{H_i H_v^T}{L} > \frac{H_i H_k^T}{L}\}| \right\} \\ \begin{bmatrix} 0 & 1 \end{bmatrix} & \text{if } j \in \left\{ v \middle| |\{k| \frac{H'_i H'^T_v}{L} > \frac{H'_i H'^T_k}{L}\}| > |\{k| \frac{H_i H_v^T}{L} > \frac{H_i H_k^T}{L}\}| \right\} \\ \begin{bmatrix} 0 & 0 \end{bmatrix} & \text{otherwise} \end{cases}$$

IV. EXPERIMENTS

1) Experiment settings:

1) **Implementation Details** For the object detection model $f_{obj}(.)$ we used Detectron 2 [35] and for the image embedding model $\vec{\phi}_{vis}(.)$ we used DINOv2 ViT-g/14 model [26]. Since the ViT-g/14 model uses a patch size of 14, it is necessary to transform the image size into a multiplier of 14 for both width and height. It is implemented as transforming a given image I_k of size (w, h) to $((\lfloor w/14 \rfloor + 1) \times 14, (\lfloor h/14 \rfloor + 1) \times 14)$. For the hypervectors, we used the dimensionality of $D = 10,000$.
2) **Evaluation Metrics** In order to thoroughly evaluate our proposed method and compare it to conventional baselines, we used mAP (mean Average Precision), a widely accepted metric for evaluating retrieval performance. This metric calculates the average precision (AP) for a given query and a ranked list of returned results, where mAP is determined by averaging the AP values across all queries. In our evaluation, we follow the latest convention and use mAP@5000 for CIFAR-10 and MS COCO. Higher mAP values indicate better overall performance.

In addition to conventional evaluation metrics, we introduce a novel metric called *mAP@Kr* to measure the effectiveness of our proposed spatial-aware conditional image retrieval. This metric represents a spatial-aware version of mAP and evaluates whether the coordinates of objects in the query image align with those in the retrieved image. This alignment is determined by calculating the Euclidean distance between the ground truth object coordinates and those of the retrieved image. The parameter r defines the metric's spatial sensitivity by determining correct retrieval for two objects' i and j having the same class using $(\frac{x_i}{w_i} - \frac{x_j}{w_j})^2 + (\frac{y_i}{h_i} - \frac{y_j}{h_j})^2 \leq r^2$ where x_i, y_i and x_j, x_j represent each object's coordinates in their image and w_i, h_i and w_j, h_j indicate each image's dimensionality. Consequently, a higher value of r results in a more lenient evaluation of whether the retrieved object contains similar objects at the same location. As the coordinate information is crucial for our proposed framework, we performed evaluations exclusively on the MS COCO dataset, using mAP@K$r = 0.1$, mAP@K$r = 0.2$, mAP@K$r = 0.3$ and mAP@K$r = 0.4$.

2) Datasets:

1) ***MS-COCO*** [19] has 82,783 training samples and 40,504 validation samples, with each image annotated with one or more labels from a pool of 91 categories. In this study, we follow the previous research [1], a subset of 122,218 images from 80 categories is used. Within this subset, a random sample of 5,000 images is referred to as the query dataset, while the remaining images form the retrieval set. In particular, MS COCO stands out from other datasets due to the inclusion of ground truth bounding box information, providing a unique opportunity to assess the extent to which our proposed method captures local information using our proposed metric *mAP@Kr*.

Scale Factor (w)	$mAP@5K_{r=0.1}$			$mAP@5K_{r=0.2}$			$mAP@5K_{r=0.3}$			$mAP@5K_{r=0.4}$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
$w = 0.1$	**0.698**	**0.757**	**0.776**	**0.926**	**0.945**	**0.952**	**0.976**	**0.981**	**0.983**	**0.991**	**0.992**	**0.993**
$w = 1.0$	0.626	0.632	0.634	0.885	0.888	0.889	0.964	0.965	0.965	0.988	0.988	0.988
$w = 10.$	0.622	0.631	0.632	0.882	0.887	0.887	0.962	0.964	0.964	0.988	0.988	0.988

TABLE III
EVALUATION RESULTS ON MS COCO DATASET WITH NEWLY PROPOSED METRIC $mAP@K_r$ SPECIFICALLY DESIGNED TO EVALUATE SPATIAL-WARE IMAGE RETRIEVAL PERFORMANCE.

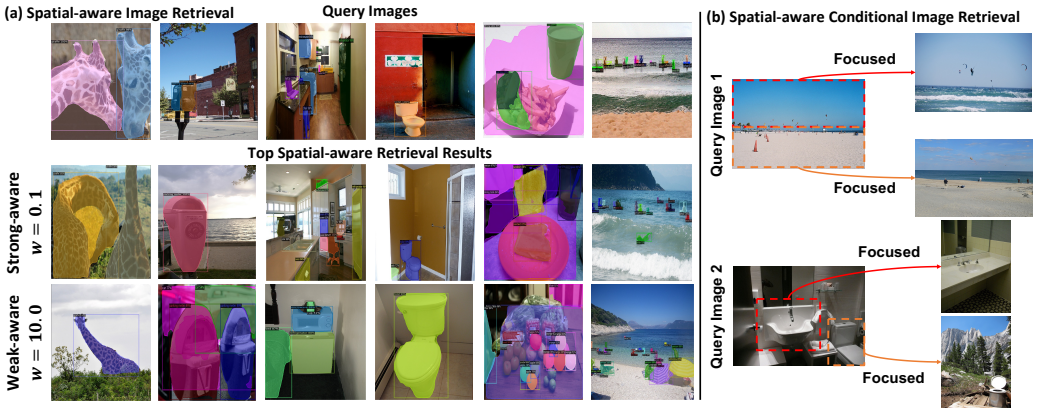

Fig. 3. **Qualitative evaluation of *NeuroHash* on (a) Spatial-aware Retrieval and (b) Spatial-aware Conditional Image Retrieval**.

2) ***CIFAR-10*** [15] involves 60,000 images distributed across 10 categories, with each class containing 6,000 images. Following the earlier study [1], we randomly chose 100 images from each class to form the query dataset, amounting to a total of 1000 images. Subsequently, we utilized the remaining images for retrieval purposes.

3) Evaluation on Weak-spatial-aware Image Retrieval: First, we evaluated our *NeuroHash* on *Weak-spatial-aware* image retrieval case with other hashing methods including current state-of-the-art models. On *Weak-spatial-aware* image retrieval, we focus on conventional image retrieval metric mAP@K which evaluates without spatial alignment of each object shown in the images. In this test, we gave high-scale factors $w = 1., 10.$ to make it less focused on spatial information. Table I shows mAP@5K results for different methods including naïve approach, which uses our backbone model DINOv2 ViT-g/14 directly with conventional hashing algorithm LSH, and ours with different scale factors w on two different datasets: CIFAR10 and MS COCO. As reported in the table, our *NeuroHash* shows strong results on the non-spatial aware image retrieval metric by outperforming other methods in most of the cases by up to 13.14% improvement.

4) Evaluation on Strong-spatial-aware Image Retrieval: To ensure the efficacy of our proposed *NeuroHash* on spatial-aware conditional image retrieval task, we conducted image retrieval evaluation on *Strong-spatial-aware* image retrieval case which aims to retrieve images with similar object positioning. In this evaluation, we use our proposed mAP@5K$_r$ metric with $r = 0.1, 1.0, 10.0$. Since the mAP@5K$_r$ metric requires ground truth labels on positional information, we used only the MS COCO dataset which provides positional annotations that other datasets are not providing. As presented in Table III, we can observe that by decreasing the scale factor w, we achieve a higher mAP@5K$_r$ score which indicates higher spatial awareness during the image retrieval.

5) Evaluation on Conditional Image Retrieval: In this conditional image retrieval section, we visually demonstrate spatial-aware image retrieval and conditional retrieval of our *NeuroHash* shown in Figure 3. On Figure 3.(a) shows the effect of controlling w to control between weak and strong spatial awareness. When $w = 0.1$, the retrieved images present higher positional alignments to the objects in the query image compared to $w = 10.0$. On Figure 3.(b), we showcase two conditional image retrieval when retrieval image set is hashed with $w = 0.1$ and $\eta_i = 1$. For the query image 1, we set $\eta_i = 10$ for the objects located in the focusing region and $\eta_i = 1$ for outside of the region. For the query image 2, we set $\eta_i = 10$ for a specific object and $\eta_i = 1$ for the others. We can observe the retrieved images are highly focused in terms of positional and visual matching on those regions or objects that have higher η_i.

6) Ablation Study on Multilinear Hyperplane Hashing Opt.: Table II shows an ablation study on our model. The results indicate that all loss metrics are necessary for effective hash value generation on the CIFAR10 dataset using the mAP@5K metric. As shown, the full model achieved the highest score.

V. CONCLUSIONS

In this paper, we propose *NeuroHash* a completely novel approach to hashing images in a neuro-symbolic way that enables spatial-aware hashing and conditional image retrieval. Experiments on well-known datasets for image retrieval performance benchmarking validate the efficacy of our work. In future work, we aim to evolve a more versatile approach capable of embedding various types of information, including temporal information. This future work seeks to broaden the scope of our neuro-symbolic framework, fostering its application in diverse domains beyond spatial-aware image retrieval.

ACKNOWLEDGEMENTS

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation (NSF) under Grants #2127780, #2319198, #2321840, #2312517, and #2235472, the Semiconductor Research Corporation (SRC), the Office of Naval Research through the Young Investigator Program Award, and Grants #N00014-21-1-2225 and #N00014-22-1-2067. Additionally, support was provided by the Air Force Office of Scientific Research under Award #FA9550-22-1-0253, along with generous gifts from Xilinx and Cisco.

REFERENCES

[1] H. Cao, L. Huang, J. Nie, and Z. Wei, "Unsupervised deep hashing with fine-grained similarity-preserving contrastive learning for image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[2] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5608–5617.

[3] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 357–366.

[4] Y. Chen, Z. Lai, Y. Ding, K. Lin, and W. K. Wong, "Deep supervised hashing with anchor graph," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9796–9804.

[5] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *International Conference on Machine Learning*. PMLR, 2017, pp. 913–922.

[6] S. Duan, Y. Liu, S. Ren, and X. Xu, "Lehdc: Learning-based hyperdimensional computing classifier," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1111–1116.

[7] A. Dutta, S. Gupta, B. Khaleghi, R. Chandrasekaran, W. Xu, and T. Rosing, "Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, 2022, pp. 281–286.

[8] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.

[9] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2916–2929, 2012.

[10] M. Hersche, G. Karunaratne, G. Cherubini, L. Benini, A. Sebastian, and A. Rahimi, "Constrained few-shot class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9057–9067.

[11] Z. Huang and S. Liu, "Perceptual hashing with visual content understanding for reduced-reference screen content image quality assessment," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2808–2823, 2020.

[12] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE international conference on rebooting computing (ICRC)*. IEEE, 2017, pp. 1–8.

[13] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional vectors," *Cognitive Computation*, 2009.

[14] J. Kang, M. Zhou, A. Bhansali, W. Xu, A. Thomas, and T. Rosing, "Relhd: A graph-based learning on fefet with hyperdimensional computing," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 2022, pp. 553–560.

[15] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[17] X. Li, J. Yu, Y. Wang, J.-Y. Chen, P.-X. Chang, and Z. Li, "Dahp: Deep attention-guided hashing with pairwise labels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 933–946, 2021.

[18] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun, "Unsupervised deep learning of compact binary descriptors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 6, pp. 1501–1514, 2018.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[20] S. Liu and Z. Huang, "Efficient image hashing with geometric invariant vector distance for copy detection," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 4, pp. 1–22, 2019.

[21] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," *Advances in neural information processing systems*, vol. 27, 2014.

[22] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," 2011.

[23] X. Liu, X. Fan, C. Deng, Z. Li, H. Su, and D. Tao, "Multilinear hyperplane hashing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5119–5127.

[24] Z. Ma, W. Ju, X. Luo, C. Chen, X.-S. Hua, and G. Lu, "Improved deep unsupervised hashing via prototypical learning," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 659–667.

[25] I. Nunes, M. Heddes, T. Givargis, A. Nicolau, and A. Veidenbaum, "Graphhd: Efficient graph classification using hyperdimensional computing," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1485–1490.

[26] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023.

[27] Z. Qiu, Q. Su, Z. Ou, J. Yu, and C. Chen, "Unsupervised hashing with contrastive information bottleneck," *arXiv preprint arXiv:2105.06138*, 2021.

[28] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 37–45.

[29] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 3034–3044, 2018.

[30] Y. Shen, L. Liu, and L. Shao, "Unsupervised binary representation learning with deep variational networks," *International Journal of Computer Vision*, vol. 127, no. 11-12, pp. 1614–1628, 2019.

[31] Y. Shen, J. Qin, J. Chen, M. Yu, L. Liu, F. Zhu, F. Shen, and L. Shao, "Auto-encoding twin-bottleneck hashing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2818–2827.

[32] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen, "Binary generative adversarial networks for image retrieval," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[33] S. Su, C. Zhang, K. Han, and Y. Tian, "Greedy hash: Towards fast optimization for accurate hash coding in cnn," *Advances in neural information processing systems*, vol. 31, 2018.

[34] J. Wang, Z. Zeng, B. Chen, T. Dai, and S.-T. Xia, "Contrastive quantization with code memory for unsupervised image retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 2468–2476.

[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[36] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, no. 1, 2014.

[37] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," in *Proceedings of the 27th international joint conference on artificial intelligence*, 2018, pp. 1064–1070.

[38] E. Yang, T. Liu, C. Deng, W. Liu, and D. Tao, "Distillhash: Unsupervised deep hashing by distilling data pairs," in *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, 2019, pp. 2946–2955.

[39] J. Yu, Y. Shen, M. Wang, H. Zhang, and P. H. Torr, "Learning to hash naturally sorts," *arXiv preprint arXiv:2201.13322*, 2022.

[40] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. 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.

[41] X. Zheng, Y. Zhang, and X. Lu, "Deep balanced discrete hashing for image retrieval," *Neurocomputing*, vol. 403, pp. 224–236, 2020.