Unsupervised Hashing Network with Hyper Quantization Tree

Sungeun Kim[1]
kimsungeun@ajou.ac.kr
Jongbin Ryu*[1,2]
jongbinryu@ajou.ac.kr

[1] Department of Artificial Intelligence, Ajou University, Republic of Korea
[2] Department of Computer Engineering, Ajou University, Republic of Korea

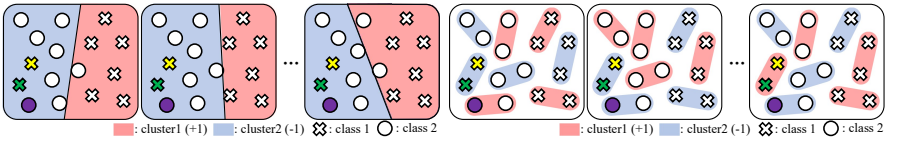
Abstract

Unsupervised hashing network commonly uses pseudo labels generated from a clustering algorithm. Therefore, the performance of the hashing network is completely oriented from the clustering algorithm, so there is no way to overcome inaccurate clustering results. To address this issue, we introduce a hyper-quantization tree method that regularizes erroneous clustering results for training robust unsupervised hashing networks. The proposed method employs a tree structure that forces leaf nodes to merge into two clusters. We refer to this forced merging method as hyper-quantization, and the merged binary cluster is used as pseudo labels that overcome the erroneous clustering results. With this hyper-quantization, we train each bit of the hash code, allowing each bit to learn a diverse feature space. As a result, our hashing network performs better due to the accurate and diversified feature representation. In our experiments, we demonstrate that the proposed hyper-quantization tree greatly enhances the performance of the state-of-the-art unsupervised hashing networks. We also provide in-depth analyses to support our claims on the diversified feature representation. Our code is publicly available at https://github.com/Lab-LVM/HQT.

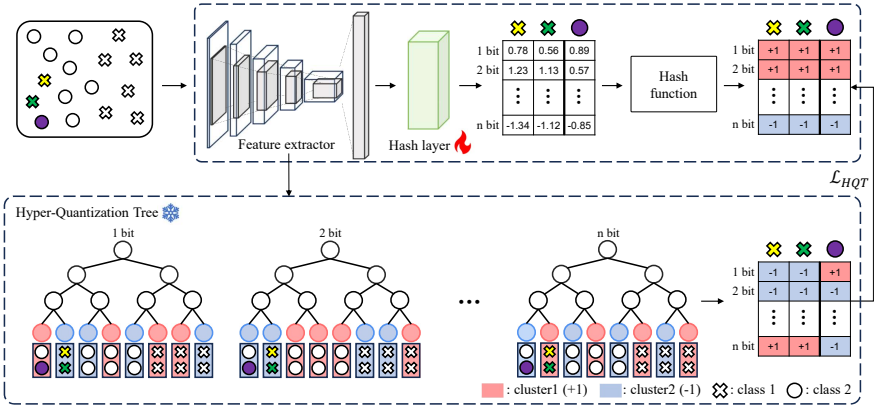
1 Introduction

Due to the expansion of social networks and the industrial revolution, huge datasets are being created, which is driving up demand for rapid information search. For this demand, hashing has been a means of transforming feature information from a high-dimensional Euclidean space into a binary hamming space. To develop such a hashing method, the self-supervision-based approach of generating pseudo labels has recently been actively studied [19, 22, 27] in recent years. This self-supervision approach of creating pseudo labels has achieved state-of-the-art performance, but it has a key drawback in that there is no chance to compensate for the incorrectly generated pseudo labels, and we overly rely on the clustering method to generate the pseudo labels. This drawback is more serious for the unsupervised hashing networks compared to the other tasks of self-supervised learning. Since, in general, for the other self-supervised learning tasks, the pretext is constructed by the pseudo labels of the self-supervision, and then downstream supervised learning is applied so that the network can possess the ground truth label information of the downstream task. On the other hand,

* Corresponding author.



(a) Clustering results of traditional algorithm (b) Clustering results of the proposed HQT



(c) Overall workflow of the proposed HQT method.

Figure 1: Illustration of the conceptual comparison between the traditional clustering algorithm and the proposed HQT for the proposed workflow.

in the unsupervised hashing approach with the self-supervision, there is no opportunity to be trained with ground truth labels. For this reason, the quality of generated pseudo labels from a clustering algorithm significantly impacts the performance of the hashing network. Our biggest concern with this clustering-based unsupervised hashing method is that they might learn similar inaccurate clustering results for all bits of the hash code. Although it has a randomized method in its learning process, the inherent nature of a clustering algorithm produces similar binary clustering patterns as shown in Fig. 1a. These similar clustering patterns 1) reduce the generalization capability of the hash code by increasing the correlation between each bit and 2) decrease the class discrimination capability by making hash codes completely dependent on the inaccurate clustering results.

In line with this issue, we propose the Hyper-Quantization Tree (HQT) that regularizes the clustering algorithm to create discriminant pseudo labels. In our HQT method, when producing pseudo labels, we hyper-quantize the clustering results into the binary bit code to reduce the correlation of the pseudo labels in a tree structure. After allowing the tree to grow sufficiently, we force each leaf node to merge into a binary cluster. This ensures that the binary clustering results of each tree are different from each other, which in turn reduces the correlation and diversifies the entire hash code as shown in Fig. 1b. In Fig. 1a, two class-1 samples marked with yellow and green X can not be clustered with other class-1 samples. Since the traditional clustering algorithm produces similar partitions, the two X samples can not be grouped with other class-1 samples for all cases. However, our proposed method in Fig. 1b randomly groups the leaf nodes of each tree into clusters so that two X samples can be gathered with the other class-1 samples. It is noticeable that the two class-1 samples and the purple circled class-2 samples are indistinguishable in Fig.1a, but they are distinguishable in

our HQT in Fig.1b.

Using the proposed HQT, the class-aware discriminant of a clustering algorithm is taken, and also the highly correlated pseudo label is not created by the hyper clusters for each tree. These class-aware and low-correlated pseudo labels have the benefit of enabling the discriminant hashing network. We examine these benefits using various analyses based on the Fisher criteria regarding entropy and correlation. In our analysis, we show that even while each bit is aware of the class information, the entropy and correlation criteria are optimized, which ultimately improves the discriminative performance of the model. In addition, we show that the proposed hyper-quantization method improves the performance of existing state-of-the-art hashing networks on CIFAR-10, Flickr25k, and NUS-WIDE datasets. These experimental studies support our claims on generating discriminant hashing codes using the proposed HQT. We summarize our contribution as:

- We propose a hyper-quantization tree method for clustering algorithms that regularizes the hashing networks from being overly learning the specific clustering patterns. Our method exploits the binary hyper clustering on a tree structure to create discriminant pseudo labels.
- We present a general approach that may be used for different hashing networks. Clustering algorithm-based pseudo labels are used by many hashing networks, so our approach can simply be extended with them to improve the performance even more.
- We provide the analysis of the discriminant capability of the proposed method using the Fisher criteria. In our analysis, we support our claim by showing that our HQT maximizes the entropy yet reduces the correlation of each bit, which indicates the better discriminant capability of the hash code.
- We demonstrate that the proposed HQT method achieves state-of-the-art performance in various evaluation protocols. Hashing networks with our HQT produces the finest results with considerable performance improvement in various protocols.

2 Related Work

Unsupervised learning is receiving attention as it efficiently utilizes vast amounts of data without using labels. Clustering is one of the most commonly used algorithms in unsupervised learning, aiming to classify data into distinguishable groups. It is being studied using different approaches, such as density-based[9], centroid-based[24], and graph-based[25] methods. Greedyhash[38] proposed an efficient hash layer that creates a binary sign of deep features at the end of backbone networks to maintain discrete constraints so that it improves hashing performance by directly optimizing the gradient. CIBHash[27], a hashing network recently proposed by Qiu *et al*., minimizes mutual information between generated hash codes and original data by a contrastive learning approach. In order to maximize the entropy of hash codes, the recently proposed BihalfNet[19] produces pseudo labels in each mini-batch training sample. This pseudo-label approach reduces information loss while increasing the entropy of the hash code of the full training sample, resulting in improved hashing performance. Apart from that, the majority of these unsupervised hash networks use pseudo labels to learn binary codes. Although these pseudo-label approaches performed well, we think they are limited to prevent each binary code from being highly correlated. As a result, using

the Fisher criteria, it is demonstrated that the proposed method creates high-variance binary codes while remaining discriminant to ground-truth class information.

Studies to generalize models by reducing correlation have also been conducted in ensemble methods [7, 29, 32, 34], where reducing the correlation between each weak learner improves the performance of the aggregated model. Inspired by this ensemble method, we introduce our hyper-quantization tree according to the idea that the performance of a hashing network can be improved when each bit of the hashing code learns different class discrimination.

Clustering algorithms have been widely used to generate pseudo labels in unsupervised learning settings. They have been used in a variety of tasks, including self-supervised [1, 3] and unsupervised [33] learning, visual dictionary learning [31, 44], semi-supervised learning [18, 30], and unsupervised hashing networks [10, 35]. From the simple yet robust K-means[13, 24], random project tree[6], and mean-shift[4] update the networks by their clustering results. Pseudo labels are generated by the clustering results to update the deep neural networks. However, these clustering algorithms have been known to yield similar clustering results[3] in different iterations, even when they include the randomized process. This limitation is a most critical problem, especially for unsupervised hashing networks, which are completely dependent on pseudo labels. Therefore, in this paper, we propose the hyper-quantization method in a clustering algorithm to improve the hashing network performance.

3 Method

3.1 Hyper-Quantization Tree

Current unsupervised clustering approaches, such as K-means [7, 11, 14], DBSCAN[16, 35], and spectral clustering[12, 41], generate clusters by relying on the notion that data in the same cluster are more likely to belong to the same class. These attributes indicate that samples of the same class belong to the same cluster; nevertheless, there is no way to override the incorrect clustering result. This is critical in hashing networks to generate the pseudo labels due to the strong dependency on the clustering algorithm performance. Since each binary code is trained with the pseudo label from the clustering result, the hashing results may overfit the incorrectly clustered samples.

To address this, we present the HQT, which forces hyper-binary clustering by utilizing a tree structure that regularizes the clustering results. The detailed process of training HQT is described in Alg. 1. To train HQT, we first extract deep features of training images from backbone networks. Then we recursively divide the nodes of a tree via binary clustering. After the tree has grown to its maximum depth, we randomly partition leaf nodes into two hyperclusters, which become the binary pseudo labels as shown in Fig. 1c. We repeat this process of three constructions for the length of hash codes, which have diverse clustering patterns for each bit. Because we force the leaf nodes of each tree into two hyper-quantized clusters randomly, each bit code is trained with different clustering patterns, so the entire hash code can learn various patterns, as shown in Fig. 1b. Certainly, when the leaf nodes are hyper-quantized randomly, samples of different classes may exist in the same hyper-quantized cluster, but this is limited to the specific bit due to its random partition. Therefore, when we train a hashing network with the entire has code, our hyper-quantization method does not reduce the class discrimination performance; rather, our method ensures that the trained hashing network may learn a variety of clustering patterns.

Algorithm 1 Bit code generation with our HQT

INPUT: Training set X
Output: Hyper-quantized bit codes Q_1, Q_2
function GET-BIT-CODES(X)
  Tree $\leftarrow$ Build-Tree($X, 0, 1$)
  $Q_1, Q_2 \leftarrow$ Hyper-quantization(Tree)
  return Q_1, Q_2
function BUILD-TREE(X, depth, idx)
  if depth = max_depth:
    return X, idx
  $C \leftarrow$ Build Binary Cluster(X)
  Left $\leftarrow$ Build Tree($\{x \in X : C(X) == 0\}$, depth+1, $idx \cdot 2$)
  Right $\leftarrow$ Build Tree($\{x \in X : C(X) == 1\}$, depth+1, $idx \cdot 2 + 1$)
  return Left, Right
function HYPER-QUANTIZATION(Tree)
  $uidx \leftarrow$ Unique(Tree, idx)
  $uidx1, uidx2 \leftarrow$ Random Partition($uidx$)
  $Q_1 \leftarrow \{x \in$ Tree.X : Tree.idx == $uidx1\}$)
  $Q_2 \leftarrow \{x \in$ Tree.X : Tree.idx == $uidx2\}$)
  return Q_1, Q_2

3.2 Regularizing Self-supervision with HQT

The unsupervised hashing network is commonly trained by pseudo labels. For this self-supervised learning approach, creating class-aware pseudo-labels is the key to learning the discriminant unsupervised hashing network. Once the pseudo label is created, the remaining learning process is equivalent to the supervised learning approach as:

$$\mathcal{L}_{Self} = f(\hat{y}, y_p), \tag{1}$$

where $\hat{y}$ denotes generated hash code by a hashing network, y_p represents the pseudo label, and $f(\cdot)$ stands for a metric function determining the loss between $\hat{y}$ and y_p. Therefore, the performance of the self-supervised loss function $\mathcal{L}_{Self}$ is highly dependent on the pseudo label y_p. Incorrect pseudo labels might ruin the training process, which makes the hashing network overfit certain clustering patterns. To this end, we introduce a new loss function using our HQT method as:

$$\mathcal{L}_{HQT} = -(y_t \cdot log(\hat{y}) + (1 - y_t) \cdot log(1 - \hat{y})), \tag{2}$$

where y_t denotes the hyper-quantized hash codes from Alg. 1. We compare our hyper-quantized hash codes and generated hash codes $\hat{y}$ from a network using the binary cross entropy metric. This $\mathcal{L}_{HQT}$ is added to Eq. 1 to get the final loss function as:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_{Self} + \alpha\mathcal{L}_{HQT}, \tag{3}$$

where α is the weighting parameter for our $\mathcal{L}_{HQT}$, which balance loss functions of self-supervision and ours.

3.3 Random Tree Depth

The deeper the depth of a tree, the more leaf nodes it has, so that data is separated into more partitions. Therefore, the hyper-quantized bit code on the leaf nodes will contain a variety of patterns for this deeper tree depth. However, class awareness may be deteriorated by the too-diversified clustering patterns of a deeper tree depth. This property of the tree depth implies that it's crucial to strike a balance between diversified clustering patterns and class awareness by controlling the tree depth. For this reason, we investigated selecting the ideal tree depth and found that choosing the depth randomly for each tree produced the best performance. We configure the depth of each tree within a specific range and empirically discover that this random tree depth improves the performance of a hashing network. Additionally, this random tree depth has the benefit of learning more variety of class patterns than the same tree depth. Since each bit of the hash code might be trained from the different tree depths, a hashing network learns the different class patterns with its random tree depth. As a result, the proposed random tree depth is straightforward yet 1) balances the clustering pattern diversity and class awareness, and 2) helps to learn more variety of clustering patterns than the same tree depth setting.

4 Experiment

4.1 Experimental Settings

Dataset and protocol. We use three widely used datasets, CIFAR10[17], Flickr25k[15], and NUS-WIDE[5]. We report mean Average Precision(mAP), which measures ranking based on the Hamming distance between a query and a data point, as a metric to evaluate hashing networks. We follow evaluation protocols of the previous studies[19, 38, 43] for a fair comparison, summarized in Table 1. We provide results for another protocol in the supplementary material.

Implementation Details. We use the VGG-16[36] model pre-trained on the ImageNet-1k as a backbone. We configure each tree's depth at random, between 3 and 5. We set the weighting parameter α of Eq. 3 as 0.5. We apply the proposed HQT method to state-of-the-art unsupervised hashing networks such as Bi-halfNet[19], CIBHash[27], CIMON[22] and UHSCM[39]. We follow the hyper-parameter configurations of each hashing network, as given in the corresponding network. To improve reproducibility, we provide the details of hyper-parameter settings in the supplementary material.

Dataset	Train	Query	Retrieval	Class	Metric
CIFAR-10(I)[43]	5,000	10,000	50,000	10	mAP@all
CIFAR-10(II)[38]	50,000	10,000	50,000	10	mAP@1000
Flickr25k[43]	10.000	1,000	24,000	24	mAP@all
NUS-WIDE(I)[19]	10,500	5,000	181,577	10	mAP@all

Table 1: Details of the evaluation protocols.

4.2 Experimental Comparision

We compare our HQT with various previous state-of-the-art (SOTA) unsupervised hashing networks. To ensure a comprehensive comparison across various settings, we conduct our ex-

Method	CIFAR-10(I)			CIFAR-10(II)			Flickr25k			NUS-WIDE(I)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
DeepBit	0.131	0.158	0.167	0.194	0.249	0.277	0.593	0.593	0.619	0.454	0.462	0.476
SSDH	0.256	0.256	0.258	0.361	0.403	0.440	0.662	0.673	0.673	0.623	0.629	0.632
GreedyHash	0.287	0.317	0.354	0.448	0.472	0.501	0.623	0.631	0.634	0.513	0.558	0.592
Bi-half	0.428	0.432	0.441	0.561	0.576	0.595	0.714	0.723	0.731	0.671	0.681	0.682
CIBHash*	0.427	0.463	0.473	0.614	0.643	0.661	0.669	0.682	0.689	0.555	0.581	0.592
CIMON*	0.451	0.472	0.494	0.610	0.660	0.686	0.682	0.707	0.709	0.631	0.661	0.670
HQT + Bi-half (Ours)	0.475	0.468	0.491	0.581	0.623	0.644	**0.726**	**0.736**	**0.742**	**0.682**	**0.688**	**0.691**
HQT + CIBHash (Ours)	0.469	0.475	0.489	0.638	0.661	0.681	0.686	0.698	0.701	0.588	0.597	0.607
HQT + CIMON (Ours)	**0.478**	**0.520**	**0.536**	**0.646**	**0.705**	**0.740**	0.705	0.724	0.731	0.653	0.674	0.679

Table 2: Experimental comparison of benchmark protocol * indicates our reproduced results from codes released by authors.

periments using the four evaluation protocols as summarized in Table 1. For SOTA methods, we report the performance of dominant hashing networks including DeepBit[21], SSDH[42], GreedyHash[38], Bi-halfNet[19], CIBHash[27], CIMON[22], UHSCM[39], MeCoQ[40], NSH[45], PURPLE[23], CGHash[37], OH[46], and SDC[26]. In Table 2 and 4, it is confirmed that our HQT works favorably against the SOTA hashing networks. Table 2 shows that the proposed HQT yields substantial performance improvement. In addition, we perform a further experimental comparison between ours and UHSCM[39] models that demonstrated outstanding efficacy through the utilization of CLIP[28], a Vision Language Model (VLM). Table 3 shows that HQT improves performance even when VLM employs unsupervised hashing networks. Also, Table 4 shows that the proposed HQT achieves better performance than other SOTA methods on the CIFAR protocol.

Method	CIFAR10(I)			CIFAR10(II)		
	16bits	32bits	64bits	16bits	32bits	64bits
UHSCM*	0.747	0.749	0.736	0.928	0.929	0.930
HQT+UHSCM	**0.758**	**0.762**	**0.744**	**0.933**	**0.934**	**0.934**

Table 3: Experimental comparison between SOTA and our HQT with the support of vision and language model.

Method	Source	CIFAR-10(I)			CIFAR-10(II)		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
MeCoQ[Wang *et al*. 2022]	AAAI 22	0.440	0.476	0.482	0.682	0.697	0.711
NSH[Yu *et al*. 2022]	IJCAI 22	-	-	-	0.706	0.733	0.756
PURPLE[Ma *et al*. 2022]	MM 22	0.515	0.520	0.544	-	-	-
CGHash[Song *et al*. 2023]	MM 23	-	-	-	0.795	0.803	0.817
OH[Yu *et al*. 2023]	MM 23	0.443	0.553	0.576	0.743	0.758	0.776
SDC[Ng *et al*. 2023]	BMVC 23	-	-	-	0.874	0.884	0.890
HQT + UHSCM	Ours	**0.758**	**0.762**	**0.744**	**0.933**	**0.934**	**0.934**

Table 4: Experimental comparison between SOTA and our HQT on CIFAR-10

4.3 Compare Traditional Clustering Algorithm.

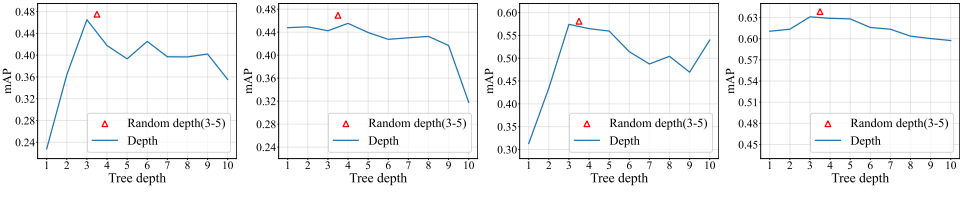
We investigate whether the HQT is effective compared to other traditional clustering algorithms. We used K-means, Agglomerative Clustering (AC), and Gaussian Mixture (GM) algorithms to train the network by directly generating hash codes without our tree structure based hyper-quantization method. For this investigation, Table 5 shows that the proposed HQT outperforms the other clustering algorithms consistently for all settings.

Dataset		K-means	AC	GM	HQT
CIFAR10(II)	16bits	0.337	0.327	0.318	**0.475**
	32bits	0.364	0.348	0.351	**0.468**
	64bits	0.381	0.402	0.383	**0.491**
Flickr25k	16bits	0.708	0.712	0.709	**0.726**
	32bits	0.717	0.720	0.719	**0.736**
	64bits	0.727	0.727	0.725	**0.742**
NUS-WIDE(I)	16bits	0.600	0.610	0.582	**0.682**
	32bits	0.617	0.618	0.637	**0.688**
	64bits	0.634	0.648	0.633	**0.691**

Table 5: Experimental comparison between traditional clustering algorithms and our HQT for training unsupervised hashing networks on Bi-halfNet[19].

5 Analysis

5.1 Results on Random Tree Depth



(a) HQT+Bi-half (CIFAR(I)) (b) HQT+CIB (CIFAR(I)) (c) HQT+Bi-half (CIFAR(II)) (d) HQT+CIB (CIFAR(II))

Figure 2: Experimental results on the random tree depth for HQT+Bi-half and HQT+CIBHash methods. We randomize the depth of a tree to a value between 3 and 5. The random tree depth achieves better results compared to all the constant depth values.

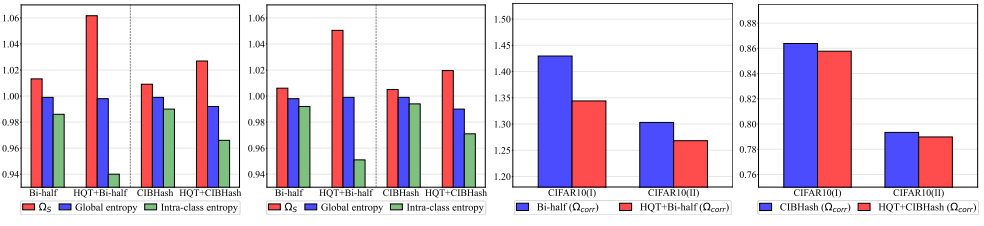
HQT has 2^{depth} leaf nodes, making the number of data partitions dependent on the depth of a tree. Therefore, we investigate the relationship between the number of leaf nodes and hashing performance of our HQT. Fig. 2 shows that the randomly selected tree depth with values from $3 \sim 5$ outperforms all the constant depth values. This result indicates that the random tree depth diversifies patterns of our hyper-quantization with different leaf node partitions to ultimately improve the hashing network performance.

5.2 Fisher Criterion

To evaluate the discriminative power of the proposed HQT, we measure the Fisher score[8] on the hash codes. We compute the Fisher score Ω using the hash codes between the inter- and intra-class samples as: $\Omega = \frac{\sum_{i=1}^{C}(\mu-\mu_i)^2}{\sum_{i=1}^{C}(\sigma_i^2)}$, where C is the number classes in a dataset, μ and μ_i are the global and class mean, and σ_i^2 represents the class variance of hash codes. We report this Fisher score in Table 6, which shows that our HQT significantly improves the score consistently. We assume that hashing networks with our hyper-quantization learn different clustering patterns for each bit, so it is able to discriminate data that cannot be handled properly using traditional clustering methods, as shown in Fig. 1.

	Dataset	Bi-half	HQT+Bi-half	CIBHash	HQT+CIBHash
Fisher score(Ω)	CIFAR10(I)	0.17	**0.78**	0.14	**0.35**
	CIFAR10(II)	0.08	**0.64**	0.07	**0.21**

Table 6: Experimental results on the Fisher score. We compare two baseline hashing networks with ours with regard to the Fisher score.



(a) Entropy of CIFAR10(I). (b) Entropy of CIFAR10(II) (c) Correlation of Bi-half (d) Correlation of CIBHash

Figure 3: Experimental results on entropy and correlation analysis

Entropy. High entropy indicates that the data can be represented by a large amount of information, so the higher the entropy, the more patterns the hash code can represent. By considering this attribute, Bi-half method[19] proposed a half-half algorithm to maximize the global entropy[20] of hash codes. In the Bi-half method, they force the hash codes of the samples in a training batch to have $+1$ bits in half and -1 bits in the other half. Similarly, our HQT divides the leaf nodes by half for $+1$ and -1 so that the entropy can be maximized in our hyper-quantization. However, there is a critical issue that Bi-half maximizes the global entropy of the samples without considering class discrimination, so the intra-class entropy is also increased in their method. We argue that the hash codes with high intra-class entropy are less effective in terms of class discrimination because samples within a class are spread out rather than clustered toward a single point due to the high intra-class entropy. Therefore, inspired by the Fisher criterion, we measure the proportions between the global entropy $\mathcal{S}(H)$ and intra-class entropy $\mathcal{S}(H_i)$ as: $\Omega_{\mathcal{S}} = \frac{\mathcal{S}(H)}{\frac{1}{C}\sum_{i=1}^{C}\mathcal{S}(H_i)}$,

$$\mathcal{S}(H) = -\left\{p(H^+)\log_2(p(H^+)) + p(H^-)\log_2(p(H^-))\right\}, \tag{4}$$

where H^+ and H^- denote the binary hash code of each bit $\{+1,-1\}$. Therefore, $\Omega_{entropy}$ represents the Fisher criterion on the entropy of the hash code. The results of this entropy analysis are shown in Fig. 3 (a) and (b), where it is confirmed that the proposed HQT considerably reduces the intra-class entropy while leaving the global entropy almost unchanged;

therefore, $\Omega_{entropy}$ increases in all cases. This result supports the proposal that HQT improves the performance of existing hashing networks for Fisher criterion-based entropy analysis.

5.3 Correlation

We provide empirical results of the correlation as the final analysis in this paper. In contrast to entropy, correlation should be low among all samples in a data set while being high among samples of the same class. This is because the correlation should be high between all data samples to improve the representation capability of hash codes, but the correlation should be low between samples within the same class due to the discrimination capability. Therefore, the correlation value should be minimized to improve the hashing performance expressed by the Fisher criterion as: $\Omega_{corr} = \frac{corr(H)}{\frac{1}{C}\sum_{i=1}^{C} corr(H_i)}$. Fig. 3 (c) and (d) demonstrate that the proposed HQT reduces the Ω_{corr} of the baseline hashing network, so it is shown that the proposed HQT is effective in regard to the correlation analysis of the hash codes.

6 Conclusion

In this paper, we propose the HQT method that optimizes the Fisher criterion of the hashing networks in an unsupervised learning approach. The proposed HQT diversifies the clustering patterns for each bit of the hash codes by forcing leaf nodes to be partitioned with a binary hyper-quantization method. Our HQT overcomes the limitation that existing clustering methods of unsupervised hashing networks are heavily dependent on specific clustering patterns. We show that our method can be simply applied to existing networks, which significantly improves the hashing performance. We evaluate our HQT with SOTA hashing networks using CIFAR-10, Flicker25k, and NUS-WIDE datasets. In all datasets, ours consistently improves the performance of SOTA results. Also, extensive analysis demonstrates that the proposed HQT generates effective hash codes with high entropy and low correlation. Our HQT optimizes the Fisher criterion of the unsupervised clustering method for entropy and correlation analysis. Therefore, we expect that our work can be applicable in various fields in an unsupervised learning setting.

Acknowledgements. This paper was supported in part by the Electronics and Telecommunications Research Institute (ETRI) Grant funded by Korean Government (Fundamental Technology Research for Human-Centric Autonomous Intelligent Systems) under Grant 24ZB1200, under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2024-RS-2023-00255968), Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (Artificial Intelligence Innovation Hub) under Grant RS-2021-II212068, and the National Research Foundation of Korea (NRF) from the Korea Government (MSIT) under Grant RS-2024-00356486.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *International Conference on Learn-*

ing Representations, 2020.

[2] Leo Breiman. Random forests. *Machine learning*, 2001.

[3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ACM International Conference on Image and Video Retrieval*, 2009.

[6] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *ACM Symposium on Theory of Computing*, 2008.

[7] Xiao Dong, Li Liu, Lei Zhu, Zhiyong Cheng, and Huaxiang Zhang. Unsupervised deep k-means hashing for efficient image retrieval and clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[8] Richard O Duda and Peter E Hart. Dg stork pattern classification. *John Wiely and Sons*, 2001.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

[10] Yifan Gu, Shidong Wang, Haofeng Zhang, Yazhou Yao, Wankou Yang, and Li Liu. Clustering-driven unsupervised deep hashing for image retrieval. *Neurocomputing*, 2019.

[11] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[12] Tuan Hoang, Thanh-Toan Do, Tam V Nguyen, and Ngai-Man Cheung. Unsupervised deep cross-modality spectral hashing. *IEEE Transactions on Image Processing*, 2020.

[13] Qinghao Hu, Jiaxiang Wu, Jian Cheng, Lifang Wu, and Hanqing Lu. Pseudo label based unsupervised deep discriminative hashing for image retrieval. In *ACM International Conference on Multimedia*, 2017.

[14] Qinghao Hu, Jiaxiang Wu, Jian Cheng, Lifang Wu, and Hanqing Lu. Pseudo label based unsupervised deep discriminative hashing for image retrieval. In *ACM International Conference on Multimedia*, 2017.

[15] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *ACM International Conference on Multimedia Information Retrieval*, 2008.

[16] Heinrich Jiang, Jennifer Jang, and Jakub Lacki. Faster dbscan via subsampled similarity queries. *Neural Information Processing Systems*, 2020.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Xiang Li, Yao Wu, Martin Ester, Ben Kao, Xin Wang, and Yudian Zheng. Semi-supervised clustering in attributed heterogeneous information networks. In *ACM International World Wide Web Conference*, 2017.

[19] Yunqiang Li and Jan van Gemert. Deep unsupervised image hashing by maximizing bit entropy. In *Association for the Advancement of Artificial Intelligence*, 2021.

[20] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 1991.

[21] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[22] Xiao Luo, Daqing Wu, Zeyu Ma, Chong Chen, Minghua Deng, Jinwen Ma, Zhongming Jin, Jianqiang Huang, and Xian-Sheng Hua. Cimon: Towards high-quality hash codes. In *International Joint Conferenceon Artificial Intelligence*, 2021.

[23] Zeyu Ma, Wei Ju, Xiao Luo, Chong Chen, Xian-Sheng Hua, and Guangming Lu. Improved deep unsupervised hashing via prototypical learning. In *ACM International Conference on Multimedia*, 2022.

[24] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, 1967.

[25] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Neural Information Processing Systems*, 2001.

[26] KamWoh Ng, Xiatian Zhu, Jiun Tian Hoe, Chee Seng Chan, Tianyu Zhang, Yi-Zhe Song, and Tao Xiang. Unsupervised hashing with similarity distribution calibration. In *British Machine Vision Conference*, 2023.

[27] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. Unsupervised hashing with contrastive information bottleneck. *International Joint Conferenceon Artificial Intelligence*, 2021.

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[29] Victor Francisco Rodriguez-Galiano, Bardan Ghimire, John Rogan, Mario Chica-Olmo, and Juan Pedro Rigol-Sanchez. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2012.

[30] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery*, 2010.

[31] Jongbin Ryu and Hyun S Yang. Locality-preserving descriptor for robust texture feature representation. *Neurocomputing*, 2016.

[32] Jongbin Ryu, Gitaek Kwon, Ming-Hsuan Yang, and Jongwoo Lim. Generalized convolutional forest networks for domain generalization and visual recognition. In *International conference on learning representations*, 2019.

[33] Jongbin Ryu, Ming-Hsuan Yang, and Jongwoo Lim. Unsupervised feature learning for self-tuning neural networks. *Neural Networks*, 2021.

[34] Jongbin Ryu, Dongyoon Han, and Jongwoo Lim. Gramian attention heads are strong yet efficient vision learners. In *IEEE International Conference on Computer Vision*, 2023.

[35] Ye Shiqiu and Zhu Qingsheng. Dbscan clustering algorithm based on locality sensitive hashing. In *Journal of Physics: Conference Series*. IOP Publishing, 2019.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Zhenpeng Song, Qinliang Su, and Jiayang Chen. Unsupervised hashing with contrastive learning by exploiting similarity knowledge and hidden structure of data. In *ACM International Conference on Multimedia*, 2023.

[38] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. *Neural Information Processing Systems*, 2018.

[39] Rong-Cheng Tu, Xian-Ling Mao, Kevin Qinghong Lin, Chengfei Cai, Weize Qin, Wei Wei, Hongfa Wang, and Heyan Huang. Unsupervised hashing with semantic concept mining. *Proceedings of the ACM on Management of Data*, 2023.

[40] Jinpeng Wang, Ziyun Zeng, Bin Chen, Tao Dai, and Shu-Tao Xia. Contrastive quantization with code memory for unsupervised image retrieval. In *Association for the Advancement of Artificial Intelligence*, 2022.

[41] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. *Neural Information Processing Systems*, 2008.

[42] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. Semantic structure-based unsupervised deep hashing. In *International Joint Conferenceon Artificial Intelligence*, 2018.

[43] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. Distillhash: Unsupervised deep hashing by distilling data pairs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[44] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the International Workshop on Multimedia Information Retrieval*, 2007.

[45] Jiaguo Yu, Yuming Shen, Menghan Wang, Haofeng Zhang, and Philip HS Torr. Learning to hash naturally sorts. *International Joint Conferenceon Artificial Intelligence*, 2022.

[46] Jiaguo Yu, Yuming Shen, and Haofeng Zhang. Hashing one with all. In *ACM International Conference on Multimedia*, 2023.