Label-consistent generalizable hash codes

Furen Zhuang, and Pierre Moulin, *Fellow, IEEE*

***Abstract*—We present a supervised semantic hashing framework, named Label-Consistent Generalized Hashing (LCGH). The main novelty of LCGH is the explicit retention of information which may be irrelevant in training, but possibly useful for generalizing to unseen test classes. This is in stark contrast to typical semantic hashing methods which seek to remove redundant feature information from their hash codes in order to maximize the margin between hash codes of dissimilar data. This typical strategy leaves hash codes narrowly viable for discerning between training classes, and inadequate in discriminating between unseen test classes. Instead of limiting the information content of hash codes to those provided by the training labels, LCGH enhances its codes with information content from both supervised and unsupervised sources, improving their ability to discriminate across a wider range of data. To do so, LCGH builds upon the foundation of first agreeing with the provided training labels (label-consistency) and then incorporating possibly useful information using a reconstruction loss. In this way, LCGH respects the reliably given label information before exploring the addition of possibly useful ones. The outcome is a hashing scheme with slightly weaker within-domain (training and test classes are the same) retrieval performance, but much stronger cross-domain (training and test classes are disjoint) performance.**

***Index Terms*—hashing, supervised, semantic, generalization, autoencoder.**

I. Introduction

THE ever growing number of images on the web has motivated the design of efficient image retrieval systems. The aim of such systems is to retrieve the most relevant visual content from a large database with low memory cost, low computational time and high accuracy. The construction of such a system has been of ongoing and active research interest within the computer vision community [1]–[12]. While conventional similarity search methods such as tree-based search and nearest neighbor search are widely used in many retrieval applications with low-dimensional data, such methods incur high memory and computational costs when applied to high-dimensional visual data.

A well-known alternative is hashing-based approximate nearest neighbor (ANN) search where the high-dimensional visual data are first mapped into a low-dimensional binary space [13]–[16]. ANN search is then performed by calculating the Hamming distance between binary vectors, an operation that can be done extremely quickly. As the distance calculation is fast and the storage cost of these short hash codes is low, the hashing-based ANN search approach is efficient in terms of both memory and time.

The recent advent of deep neural networks has enabled learning of high-level features from raw image data, via a series of non-linear transformations. These advances have allowed features to be trained which are very effective in capturing the semantic information of images. As such, end-to-end methods which combine feature training with hash code learning usually obtain better results than those which focus only on hash code learning [13]–[16].

However, it has been shown that many state-of-the-art supervised hashing methods can be outperformed by a simple neural network classifier [17] and at a much lower bit rate corresponding to the number of classes. The reason for this is two-fold. Firstly, the experiments in the paper [17] suggest that the state-of-the-art hashing methods are not encapsulating information beyond that provided by the training labels. Secondly, the reason why there is little need for regularization lies with the unique setup of the supervised hashing evaluation, whereby the database we are retrieving from is usually used in its entirety as the training set. This results in a considerable overlap between the data used in training and in test time, and a high tolerance for overfitting to the database set during training. One of the changes to the evaluation set-up suggested by [17] is a cross-domain setting whereby the training and test classes are disjoint.

The cross-domain setting is of great importance to real-world applications. The exploding diversity and availability of data has made it increasingly unrealistic to train on all possible classes. As a result, there is an urgent need for hashing methods which can generalize to classes which are not seen during training.

In the usual supervised hashing evaluation protocol, there is an advantage to removing information which are not discriminatory to training classes, because this would increase the margin between hash codes of dissimilar classes and result in fewer retrieval errors. As evidenced by the outperformance of state-of-the-art hashing methods by simple classifiers in the paper [17], these methods only retain information which are relevant to the training classes. This reduces the information content that the codes are able to hold, below the potential afforded by their hash lengths, and makes the hash codes narrowly viable to discriminate between classes seen during training and unable to discern between unseen test classes. As a result, such codes are unable to generalize beyond classes it has seen.

In this paper, we present a hashing method which better generalizes to unseen domains. This domain adaptation predicates on retaining within-class variance – instead of encouraging the hash codes to take up one of C codewords, we allow the hash codes to be different even if they are from the

This work was supported in part by the National Science Foundation under Grant CCF 12-19145 and CCF 15-27388 and in part by the American Universities International Programs Fellowship and the National Science Scholarship through Agency for Science, Technology and Research (A*STAR) Graduate Academy, Singapore.

Furen Zhuang is with the Institute for Infocomm Research (I2R), A*STAR, Singapore. Pierre Moulin is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA. (e-mails: zhuang_furen@i2r.a-star.edu.sg; moulin@ifp.uiuc.edu).

same class. We define a set of label-consistent hash codes – coding schemes where hash codes of similar classes have lower Hamming distance than codes of dissimilar classes – and select the hashing scheme within this set which can be used to reconstruct the input feature vector with the least mean square error. In so doing, the learnt hash functions agree with the known labels and also maximally retain information from the input feature vectors, which we show helps in generalization to unseen classes. We show that the hash codes demonstrate strong retrieval performance on both within-domain (test and training classes are the same) and cross-domain (test and training classes are disjoint) retrieval tasks.

II. Related Work

Prior work usually desires for hash codes to be similarity-preserving, balanced and pairwise uncorrelated [18]. Similarity-preserving means the hash codes of similar images have a shorter Hamming distance than dissimilar ones. Balanced bits means there is an equal probability that each bit value is 0 or 1. Balanced and uncorrelated bits encourage an equal number of data items to be mapped to each code [18]. These ideals help the hash codes to be maximally informative.

Learning to hash methods can be broadly split into supervised and unsupervised methods. Many of these methods are concerned with bridging the gap between a continuous relaxation used in training and the inference stage's discrete quantization to obtain hash codes [19] with little quantization error.

A. Unsupervised hashing

For unsupervised hash code learning, two approaches are prominent: graph-based approaches, and generative approaches.

Spectral Hashing [20] is a pioneering work which learns hash codes matching the graph Laplacians of the data affinity matrix. However affinity matrices might be computationally expensive to build for very large datasets. As such, Anchor Graph Hashing (AGH) [21] was proposed to use the distance between each point and a small set of anchors to approximate the graph. The obvious tradeoff for the reduced computational complexity is that the approximation of the graph by AGH leads to deterioration of hashing performance. Furthermore graph-based methods usually suffer from the 'static graph' problem [22] because the high cost of graph generation makes adaptive update of the graph during training impractical. A graph is usually pre-computed at the start of training and stays unchanged throughout training. The graph is therefore unable to update its entries to better reflect the information that is obtained during training.

Generative unsupervised hashing usually involves generating the input from the binary codes such that the reconstruction error is minimized. Notable papers include Iterative Quantization (ITQ) [3], Binary Autoencoder (BA) [23] and Stochastic Generative Hashing (SGH) [24]. ITQ iteratively learns an orthogonal rotation matrix and the binary codes such that projecting the rotated codes along the principal directions given by Principal Component Analysis (PCA) minimizes a Mean Squared Error (MSE) objective. Likewise BA and SGH utilize an auto-encoding framework, learning the binary codes as latent bottleneck variables between the encoder and the decoder.

B. Supervised hashing

Supervised hashing usually produce better results than unsupervised hashing, due to the additional information provided by the labels. The labels are very useful in helping the models overcome the semantic gap, where it is difficult to map features to the labels.

Column sampling based Discrete Supervised Hashing (COSDISH) [10] is a method which utilizes random labeled samples to learn codes for large datasets efficiently. Strongly Constrained Discrete Hashing [12] require the model to yield binary codes and enforce the balance and decorrelation of bits.

A large class of supervised hashing methods utilize neural networks. Methods using Convolutional Neural Networks (CNN) have been shown to outperform those using precomputed ("handcrafted") features in many works [13]–[15], [25]–[30]. Of these the methods can be broadly split into those utilizing point-wise, pair-wise and triplet-wise supervision.

Typically, point-wise methods [14], [15], [31] use a feedforward network and minimize a label reconstruction loss, such as categorical cross-entropy. An intermediate (feature/bottleneck) layer is then extracted and transformed to obtain bits. Pairwise methods [13], [25], [29], [30] usually match the relaxed Hamming distance between samples to a similarity matrix calculated using their labels. Methods utilizing a triplet loss [26]–[28] encourage the relaxed Hamming distances of dissimilar images to be greater than those of similar images.

To address the gradient problem posed by the $\mathrm{sgn}(\cdot)$ function, most methods use a sigmoid/tanh relaxation with a distortion loss term encouraging the function to reach either extremes. Notably, [14], [31] uses a different approach known as structural quantization (via a block softmax) to obtain binary codes.

Recently there are methods which attempt to reduce the number of loss terms. OrthoHash [32] maximizes the cosine similarity between the continuous codes and their corresponding binary orthogonal codes to simultaneously ensure label-consistency discriminativeness and low quantization error. The Batch Normalization is used to promote code balance. The paper [19] proposes obviating the quantization constraints by matching with a discrete uniform distribution to improve code balance and quantization error. This matching is done by minimizing Wasserstein distance.

III. Importance of retaining redundant information

We establish the importance of retaining information from the original feature vector $\mathbf{X}$, even if the information is irrelevant to the label $\mathbf{y}$. We assert that the retention of such information is crucial to generalization.

For illustration, suppose $\mathbf{y}$ can take on one of two values, encoding for either apples and oranges. $\mathbf{X}$ contains a variety of information among which are the color and shade (either

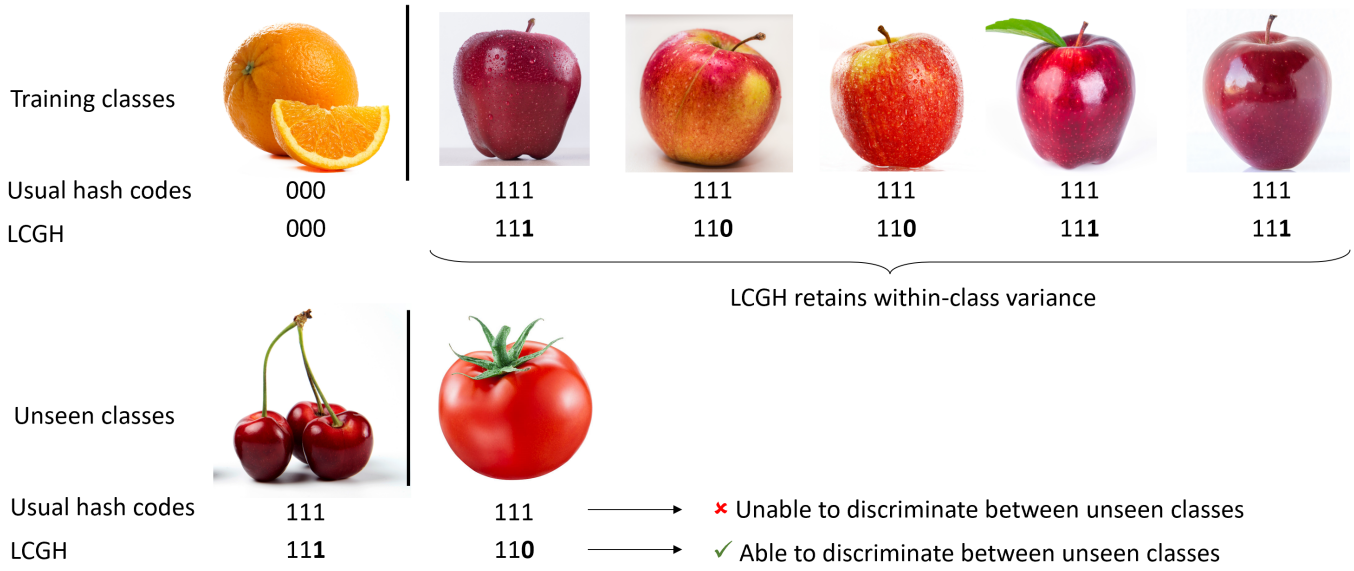


Fig. 1: Hash codes trained with categorical cross-entropy loss like LCH do not usually retain within-class variance and maps both unseen classes to the same code as they are both red. LCGH retains the 'shade of red' information resulting from the different varieties of apples, which is not useful in discriminating between training classes, but is useful in discriminating between the unseen classes. Images taken from [33].

a brighter or darker shade) of the object. To distinguish between apples and oranges, the hash code $\mathbf{b}$ requires only one bit, encoding if the object is red or not, to obtain a reasonable semantic hashing performance. The shade does not help to distinguish between the two classes and is considered redundant.

However, when the hash function h learnt with this objective on a dataset consisting of apples and oranges is used to hash unseen classes 'Tomatoes' and 'Cherries', data from both classes will be mapped to the same hash code $\mathbf{b}$, because they are both red. The hash codes are therefore unable to distinguish between them and this leads to poor retrieval performance. Had the 'shade', which is present in $\mathbf{X}$, also been encoded in $\mathbf{b}$, $\mathbf{b}$ would be able to distinguish between the brighter tomatoes and the darker cherries, and hence generalize to this unseen domain as illustrated in Fig 1.

A. Comparison against prior work

Most supervised hashing methods prioritize retaining information to discriminate between training classes. In this work we show that uncorrelated bits can be contradictory to label-consistency, and therefore this objective will lead to a decline in retrieval performance, especially when the hash code length exceeds the number of classes as seen in Fig 3. When the hash code length exceeds the number of classes, maintaining the constraint of uncorrelated bits will introduce more bits which are not discriminatory to the classes, leading to worse performance. Due to this fact, state-of-the-art supervised hashing methods are not able to fully actualize the ideal of uncorrelated bits.

Even if uncorrelated bits were achieved, in most hashing methods there is no ranking of the importance of these uncorrelated projections. The added information content may be trivial.

As such, most supervised hashing methods are not able to effectively incorporate information which are redundant during training. Instead, the evaluation protocol encourages the margin between hash codes of dissimilar data to be maximized.

In contrast, our method LCGH is able to create a hierarchy of information it needs to retain for generalization performance. LCGH moderately ensures that label-consistency is achieved before seeking out the most important features using the reconstruction loss. This results in a hashing scheme with slightly weaker within-domain (training and test classes are the same) retrieval performance, but much stronger cross-domain (training and test classes are disjoint) performance.

IV. Proposed Method

In this section, we present Label-Consistent Generalized Hashing (LCGH). We begin with a framework for within-domain hashing, named Label-Consistent Hashing (LCH) [16], and limit its margin-maximizing tendencies to encourage the retention of information in addition to those related to the training classes. We then fill the remaining information capacity of the hash codes with unsupervised information from the training data, guided by the reconstruction loss. The enhanced information content of the hash codes improves its performance on unseen test classes.

A. Notation

The mathematical notation used in this paper is summarized in Table I. Let $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ be the training set with $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$. Here N is the number of examples, and $\mathbf{X}_n$ denotes the n-th input data,

TABLE I: Notation adopted in this paper

Style	Explanation
N	a scalar
$\mathbf{m}$	a random vector
$\mathbf{m}_n$	the n-th vector
$m_{n,i}$	a scalar, the i-th element of vector $\mathbf{m}_n$
W	matrix, in sans-serif
W_c	c-th row of matrix W, in sans-serif
$\|\mathbf{m}\|^2$	sum of all squared elements in $\mathbf{m}$
$\mathcal{X}$	a set
Symbol	**Explanation**
N	number of training samples
$\mathbf{X}$	input feature vector
$\mathbf{y}$	one-hot label vector
d	dimension of input feature vector
c	index of classes
$\tilde{c}_n$	class of n-th sample, scalar
C	number of classes
s_{ij}	similarity between samples i and j
h	hash function
$\mathbf{b}$	hash codes
K	length of hash code
θ	neural network parameters
σ	sigmoid function
KLD	Kullback-Leibler Divergence
$\mathbf{m}_n, \mathbf{z}_n$	latent random vectors of n-th sample
λ	temperature hyperparameter
W	trainable weight matrix
W_c	c-th row of W
bias	trainable bias vector

an image feature vector of dimension d ($\mathcal{X}_n \in \mathbb{R}^d$). Each input $\mathbf{X}_n$ has a corresponding ground truth label $\mathbf{y}_n$, where $\mathbf{y}_n = [y_{n,1}, y_{n,2}, \ldots, y_{n,C}] \in \{0,1\}^C$, with $y_{n,c} = 1$ if $\mathbf{X}_n$ belongs to class c and $y_{n,c} = 0$ otherwise.

We consider single-label datasets. where each sample of a single-label dataset only has one class associated with it and therefore the label vectors of such datasets are one-hot ($\|\mathbf{y}_n\|_1 = 1$). We define the similarity between $\mathbf{X}_i$ and $\mathbf{X}_j$ as $s_{ij} = \mathbf{y}_i^\top \mathbf{y}_j$, $\forall i, j \in \{1, \ldots, N\}$. Therefore samples with the same class are considered similar, while samples of different classes are dissimilar.

B. Label-Consistent Hashing

We aim to learn a hash function h such that the set of binary vectors $\mathbf{b}_n = h(\mathbf{X}_n) = [b_{n,1}, b_{n,2}, \ldots, b_{n,K}] \in \{-1,1\}^K$ are label-consistent, wherein similar hash codes have lower Hamming distance than dissimilar ones.

To achieve label-consistent hash codes, we train the hash function h such that the hash codes $\mathbf{b}$ are clustered around class exemplars and these class exemplars are trained to be far apart from each other. These class exemplars are represented as the row vectors in the weight matrix W of the classification linear layer of (8). Specifically, $\mathsf{W}^\top = [\mathsf{w}_1, \mathsf{w}_2, \ldots, \mathsf{w}_C], \mathsf{W} \in \mathbb{R}^{C \times K}$, and w_c is designated as the exemplar of class c. In so doing, the hash codes are encouraged to have high between-class scatter and low within-class variance. Hash codes of the same class would then have a high inner product while hash codes of dissimilar classes would have a low inner product, in turn leading to label-consistency.

This is done by taking in input $\mathbf{X}_n$ and predicting its label $\mathbf{y}_n$, similar to the framework used in a VAE. The advantage of using the VAE framework is that the network is trained to be robust to hash codes generated from the underlying distribution associated with each class and not just the observed representation obtained from each image. This encourages the network to train representations that approach negative and positive infinity values, and using the sigmoid function on these representations allows the binary hash codes to be directly trained and obtained with little quantization error.

The method involves three key steps which are integrated end-to-end into a neural network parameterized by $\Theta = \{\theta_{base}, \theta_{class}\}$, where θ_{base} represents the weights of the base neural network and θ_{class} represents the weights of the classification layer.

First we use a base neural network to obtain an intermediate representation

$$\mathbf{m}_n = f_{base}(\mathbf{X}_n, \theta_{base}) \in \mathbb{R}^K. \tag{1}$$

At the end of training, these representations are converted into hash codes,

$$\mathbf{b}_n = \text{sgn}(\mathbf{m_n}). \tag{2}$$

Next we assume there exists a succeeding latent representation $\mathbf{z}_n$ that follows a posterior distribution $p(\mathbf{z}_n|\mathbf{m}_n)$, such that $\mathbf{y}_n$ can be derived from $\text{sgn}(\mathbf{z_n})$ by obtaining logits from the linear layer in (8). Using the relaxation for discrete variables proposed in [34], we train a proposal distribution $q_\lambda(\mathbf{z}_n|\mathbf{m}_n)$ to approximate $p(\mathbf{z}_n|\mathbf{m}_n)$.

Denote by $\mathbf{u}$ the concatenation of K independent and identically distributed random variables sampled from Uniform(0,1). Under this proposal distribution, $\mathbf{z}_n$ is obtained as

$$\mathbf{z}_n = \frac{\mathbf{m}_n + \log \mathbf{u} - \log(\mathbf{1} - \mathbf{u})}{\lambda}. \tag{3}$$

The purpose of defining (3) as such is allow the output of a product Bernoulli distribution with parameters $\mathbf{m}_n$, and that the loss gradients which can be backpropagated through the network.

λ is a temperature hyperparameter. Specifically, as λ approaches 0, the distribution $q_\lambda(\sigma(\mathbf{z}_{n,i})|\mathbf{m}_{n,i})$ approaches the Bernoulli distribution with parameter $m_{n,i}$, as stated formally in Equations (6) and (7).

There is a trade-off: a low λ makes training more unstable (large loss gradients) even though the approximation to the Bernoulli distribution is better. In particular, λ is used to create a continuous relaxation of the Bernoulli distribution with the choice of λ balancing the ease of training and the fidelity to the Bernoulli distribution it is trying to approximate.

The conditional log-density of $\mathbf{z}_n$ is obtained [34] as

$$\log q_\lambda(\mathbf{z}_n|\mathbf{m}_n) = \sum_{i=1}^{K} \left[\lambda z_{n,i} + m_{n,i} + \log \frac{\lambda}{(e^{m_{n,i}} + e^{\lambda z_{n,i}})^2} \right]. \tag{4}$$

Denote by σ the sigmoid function applied element-wise on a vector,

$$\sigma(\mathbf{z}) = \frac{1}{1 + e^{-\mathbf{z}}}. \tag{5}$$

With a slight abuse of notation, we denote by $q_\lambda(\mathbf{z}_n|\mathbf{m}_n)$ the conditional distribution on $\sigma(\mathbf{z}_n)$ given $\mathbf{m}_n$. It can be shown [34], [35], that $\forall i$

$$\lim_{\lambda\to 0}\ q_\lambda(\sigma(\mathbf{z}_{n,i})=1|\mathbf{m}_{n,i})=\sigma(m_{n,i}) \tag{6}$$

$$\lim_{\lambda\to 0}\ q_\lambda(\sigma(\mathbf{z}_{n,i})=0|\mathbf{m}_{n,i})=1-\sigma(m_{n,i}). \tag{7}$$

Therefore $q_\lambda(\sigma(\mathbf{z}_n)|\mathbf{m}_n)$ is trained to be a relaxed product Bernoulli distribution which approximates the distribution $p(\text{sgn}(\mathbf{z}_n)|\mathbf{m}_n)$ from which $\mathbf{y}_n$ can be derived.

The logits $\mathbf{l}_n$ are then obtained via a linear layer parameterized by $\theta_{class}=\{\mathsf{W},\mathsf{bias}\}$,

$$\mathbf{l}_n=\mathsf{W}\cdot\sigma(\mathbf{z}_n)+\ \mathsf{bias}. \tag{8}$$

Our overall loss minimization problem is stated as

$$\min_{\Theta}\ L=L_{class}+\alpha L_{KLD}+\beta\left\|\Theta\right\|^2, \tag{9}$$

where L_{class} and L_{KLD} encourages label-consistency and code balance respectively. α and β are hyper-parameters that are used to adjust the contribution of each term. We use stochastic gradient descent to numerically solve (9). The loss terms are defined as follows.

1) Promoting label-consistency: We use the categorical cross-entropy loss for training,

$$L_{class}=-\sum_{n=1}^{N}\mathbf{y}_n^\top\log\left(\hat{\mathbf{y}}_n\right) \tag{10}$$

where

$$\hat{\mathbf{y}}_n=\text{softmax}(\mathbf{l}_n)\in[0,1]^K \tag{11}$$

is the vector of predicted class probabilities.

Denote by

$$\tilde{c}_n=\text{argmax}(\mathbf{y}_n)\in\{1,\ldots,C\} \tag{12}$$

the non-zero index of the one-hot label $\mathbf{y}_n$, namely the class which $\mathbf{X}_n$ belongs to.

We show that this promotes label-consistency. Combining (10), (8), (5), and (3), we derive the update direction with respect to $\sigma(\mathbf{z}_n)$ as

$$-\frac{\partial L_{class}}{\partial\sigma(\mathbf{z}_n)}=\left(1-\frac{e^{l_{n,\tilde{c}_n}}}{\sum_{c=1}^{C}e^{l_{n,c}}}\right)\mathbf{w}_{\tilde{c}_n}. \tag{13}$$

As the term in parentheses in (13) is always positive, this loss causes each sample $\sigma(\mathbf{z}_n)$ to be updated towards its class exemplar $\mathbf{w}_{\tilde{c}_n}$.

Similarly, combining (10) and (8), we derive the update direction with respect to $\mathbf{w}_c$ as

$$-\frac{\partial L_{class}}{\partial\mathbf{w}_c}=\sum_{n:\tilde{c}_n\neq c}\left(-\frac{e^{l_{n,\tilde{c}_n}}}{\sum_{c=1}^{C}e^{l_{n,c}}}\right)\sigma(\mathbf{z}_n)+\sum_{n:\tilde{c}_n=c}\left(1-\frac{e^{l_{n,\tilde{c}_n}}}{\sum_{c=1}^{C}e^{l_{n,c}}}\right)\sigma(\mathbf{z}_n). \tag{14}$$

As the first bracketed term in (14) is always negative while the second bracketed term is always positive, this causes the exemplars $\mathbf{w}_c$ to be updated towards samples which have the same class ($\sigma(\mathbf{z}_n):\tilde{c}_n=c$) and away from samples of other classes ($\sigma(\mathbf{z}_n):\tilde{c}_n\neq c$).

Taken together, these updates encourage clustering of the trained hash codes and promote label consistency.

2) Promoting balanced bits: Finally we use a Kullback-Leibler divergence (KLD) loss to promote balanced bits. Let $\mathbf{m}$ and $\mathbf{z}$ be random variables which $\mathbf{m}_n$ and $\mathbf{z}_n$ are instances of. We specify the prior distribution of $\mathbf{z}$ as

$$p(\mathbf{z})=q_\lambda(\mathbf{z}|\mathbf{m}=\mathbf{0}), \tag{15}$$

such that from equations (6) and (7), $\sigma(\mathbf{z})$ approaches a product Bernoulli distribution with equal probabilities. The smaller

$$\text{KLD}(q_\lambda(\mathbf{z}|\mathbf{m})||p(\mathbf{z}))=\mathop{\mathbb{E}}_{\mathbf{z}\sim q_\lambda(\mathbf{z}|\mathbf{m})}\left[\log\frac{q_\lambda(\mathbf{z}|\mathbf{m})}{p(\mathbf{z})}\right] \tag{16}$$

is, the more balanced the bits will be. Therefore we include the loss term

$$L_{KLD}=\sum_{n=1}^{N}\left[\log q_\lambda(\mathbf{z}_n|\mathbf{m}_n)-\log q_\lambda(\mathbf{z}_n|\mathbf{m}_n=\mathbf{0})\right] \tag{17}$$

as part of the training objective in (9).

3) Elimination of redundant information in LCH: LCH embodies the elimination of redundant information. As we see in (13), in LCH the hash codes are continually updated towards the direction of their class exemplars. As such they are encouraged to take up one of C codewords and be invariant to within-class variations. Such a scheme works very well for semantic hashing because the Hamming distance between data of the same class is kept to a minimum, making it ideal for the condition stipulated in the problem statement: dissimilar codes should be further apart than similar codes. However it lacks information which can help it to generalize to unseen classes.

C. Label-consistent generalized hashing

We extend LCH into Label-Consistent Generalized Hashing (LCGH) to allow generalization to unseen classes.

To tolerate the retention of information which are irrelevant in training, we introduce ϵ-label consistency. Given $(\mathbf{X},\mathbf{y})$, we define the set of ϵ-label consistent hash codes $\mathcal{H}_\epsilon$ as

$$\mathcal{H}_\epsilon=\left\{h(\mathbf{X}):-\mathbf{y}_n^\top\log\left(\hat{\mathbf{y}}_n\right)<\epsilon\right\}, \tag{18}$$

where instead of requiring the hash codes be as close to their class exemplars as possible, we accept hash codes as ϵ-Label Consistent if the class loss is at most ϵ.

Within this set, we select the hash codes that minimize the reconstruction distortion with $\mathbf{X}$, via reconstruction loss L_{rec}. We use a fully-connected linear layer parameterized by θ_{rec} to obtain reconstruction $\hat{\mathbf{X}}_n=f_{rec}(\mathbf{X}_n,\theta_{rec})$ and define L_{rec} as

$$L_{rec}=\sum_{n=1}^{N}\left\|\hat{\mathbf{X}}_n-\mathbf{X}_n\right\|^2. \tag{19}$$

We define the ϵ-Label Consistency loss

$$L_{class,\epsilon}=\sum_{n=1}^{N}\text{softplus}(-\mathbf{y}_n^\top\log\left(\hat{\mathbf{y}}_n\right)-\epsilon) \tag{20}$$

TABLE II: MAP scores and training time costs (in seconds) for different hashing methods. The best results are shown in bold.

(a) Caltech-256 {Train:Test = 26,000:3,000}

Methods/Length	8 bits		16 bits		32 bits		64 bits	
	MAP	training time	MAP	training time	MAP	training time	MAP	training time
LSH [1]	0.0191	0.007	0.0378	0.002	0.0919	0.006	0.1657	0.004
PCAH [2]	0.0655	0.597	0.1218	0.840	0.1867	1.363	0.2427	1.655
ITQ [3]	0.0842	1.238	0.1578	2.543	0.2434	4.785	0.3236	6.687
DGH [4]	0.0445	97.027	0.1012	41.100	0.1268	42.509	0.2148	58.049
SGH [5]	0.0657	3.693	0.1349	4.491	0.2156	4.940	0.2912	5.328
CCA-ITQ [3]	0.1013	3.327	0.2175	6.465	0.3033	9.052	0.3858	16.722
SDH [6]	0.1535	13.758	0.2789	22.985	0.3498	38.381	0.4102	64.244
FSDH [8]	0.1453	4.978	0.2679	7.217	0.3526	12.136	0.4381	21.123
FastH [9]	0.1847	282.138	0.3872	333.757	0.5303	550.108	0.6501	879.958
COSDISH [10]	0.1130	5.775	0.2322	9.212	0.4145	19.888	0.5699	65.364
FSSH [11]	0.1449	20.844	0.4449	21.221	0.5851	21.323	0.6338	21.898
SCDH [12]	0.1969	8.467	0.3527	4.365	0.4998	7.260	0.6220	5.804
SCDH$_K$ [12]	0.3242	16.287	0.5467	11.537	0.6538	19.075	0.7076	14.945
FSSH* [11]	0.2085	1.639	0.4635	1.594	0.5661	1.620	0.6323	1.698
SCDH* [12]	0.1881	2.540	0.3469	2.522	0.4909	2.899	0.6069	3.676
SCDH$^*_K$ [12]	0.3597	20.848	0.5129	19.916	0.5139	20.777	0.5147	22.396
SSDH* [15]	0.1645	936.677	0.5947	940.181	0.7133	945.041	0.7307	945.112
LCH (Ours)	**0.4233**	1126.321	**0.6530**	1120.947	**0.7227**	1124.729	**0.7343**	1130.720
LCGH (Ours)	0.4129	1128.403	0.6517	1119.721	0.7195	1130.886	0.7222	1132.889

(b) CIFAR10 {Train:Test = 54,000:6,000}

Methods/Length	8 bits		16 bits		32 bits		64 bits	
	MAP	training time	MAP	training time	MAP	training time	MAP	training time
LSH [1]	0.1186	0.001	0.1230	0.001	0.1408	0.012	0.1480	0.002
PCAH [2]	0.1311	0.373	0.1315	0.385	0.1276	0.438	0.1234	0.727
ITQ [3]	0.1517	1.257	0.1615	2.177	0.1674	5.459	0.1737	8.881
DGH [4]	0.1213	102.784	0.1236	121.674	0.1238	167.458	0.1230	127.135
SGH [5]	0.1388	2.483	0.1474	3.159	0.1442	4.707	0.1430	10.047
CCA-ITQ [3]	0.2087	2.689	0.2267	5.662	0.2713	7.730	0.2879	12.658
SDH [6]	0.2576	11.322	0.2868	23.149	0.3280	28.491	0.3372	49.517
FSDH [8]	0.2356	4.742	0.2932	8.164	0.3295	9.750	0.3417	11.006
HC-SDH [7]	n/a	n/a	0.5219	4.058	0.5352	4.209	0.5355	4.253
FastH [9]	0.4568	552.276	0.5463	806.598	0.6163	1258.380	0.6670	2495.000
COSDISH [10]	0.2915	7.383	0.3626	11.647	0.4717	33.977	0.5091	136.961
FSSH [11]	0.6037	100.913	0.6280	101.768	0.6738	102.850	0.6988	108.956
SCDH [12]	0.4999	4.560	0.5544	9.263	0.6116	9.901	0.6376	12.186
SCDH$_K$ [12]	0.6353	11.426	0.6773	15.499	0.7023	16.692	0.7114	26.673
FSSH* [11]	0.5869	4.915	0.6202	4.980	0.6675	5.040	0.6830	5.351
SCDH* [12]	0.5075	2.845	0.5726	3.627	0.6054	4.905	0.6256	9.699
SCDH$^*_K$ [12]	0.4624	80.083	0.4682	81.695	0.4711	87.380	0.4742	97.868
SSDH* [15]	0.6957	911.361	0.8333	915.574	0.8955	914.467	0.9273	914.171
LCH (Ours)	**0.7285**	1197.449	**0.8460**	1202.378	**0.9144**	1196.248	**0.9276**	1199.574
LCGH (Ours)	0.7265	1151.372	0.7973	1154.953	0.8467	1171.452	0.7687	1191.380

TABLE III: MAP scores and training time costs (in seconds) of different hashing methods for unseen classes.

(a) Caltech-256 {Train:Test = 70% classes:30% classes}

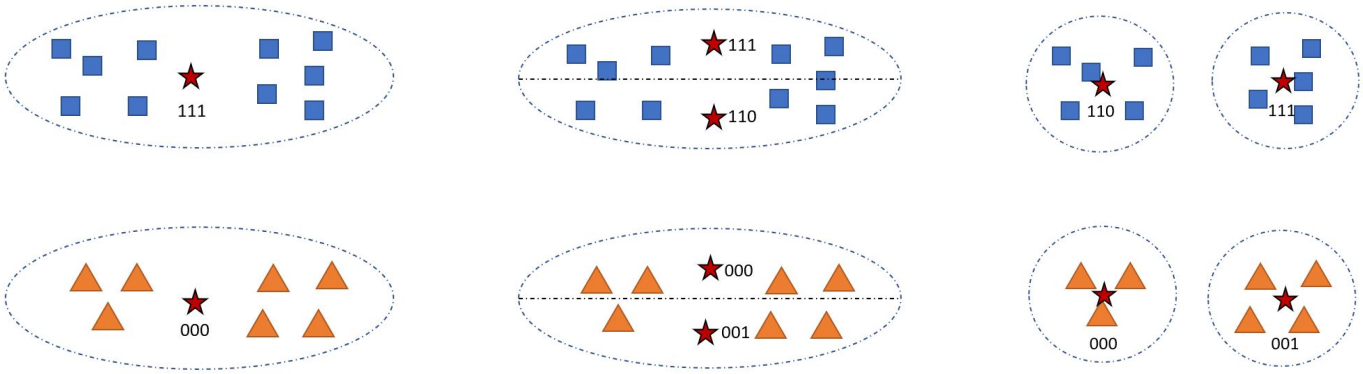
Methods/Length	8 bits		16 bits		32 bits		64 bits	
	MAP	training time	MAP	training time	MAP	training time	MAP	training time
LSH [1]	0.0154	0.002	0.0332	0.003	0.0830	0.004	0.1431	0.005
PCAH [2]	0.0548	0.493	0.1009	0.721	0.1586	1.108	0.2088	1.520
ITQ [3]	0.0750	1.094	0.1327	2.174	0.2232	4.100	0.2981	6.184
DGH [4]	0.0388	36.002	0.0834	36.649	0.1131	51.957	0.1914	80.088
SGH [5]	0.0555	3.255	0.1153	3.721	0.1974	3.957	0.2608	4.573
CCA-ITQ [3]	0.0831	2.819	0.1916	5.515	0.2657	7.784	0.3564	14.344
SDH [6]	0.1235	11.341	0.2556	20.128	0.3184	31.998	0.3591	56.078
FSDH [8]	0.1287	4.159	0.2289	6.335	0.3109	10.027	0.4066	18.889
FastH [9]	0.1484	251.928	0.3233	292.161	0.4442	449.008	0.5650	790.680
COSDISH [10]	0.0999	4.822	0.1919	7.720	0.3440	17.490	0.4927	56.555
FSSH [11]	0.1163	17.085	0.3791	17.722	0.5197	18.181	0.5783	20.421
SCDH [12]	0.1584	3.525	0.2978	5.245	0.4123	6.191	0.5424	5.986
SCDH$_K$ [12]	0.2760	9.291	0.4623	14.033	0.5992	15.576	0.6303	17.102
FSSH* [11]	0.0897	2.144	0.1583	1.957	0.2272	1.988	0.2851	2.150
SCDH* [12]	0.0431	3.256	0.0702	3.302	0.1196	3.573	0.1914	4.409
SCDH$^*_K$ [12]	0.1035	29.523	0.1186	29.725	0.1211	30.175	0.1227	31.497
SSDH* [15]	0.0732	913.257	0.1617	912.348	0.2275	913.616	0.2877	911.000
LCH (Ours)	0.1114	1108.108	0.1722	1108.360	0.2293	1112.972	0.2820	1119.616
LCGH (Ours)	**0.1119**	1157.945	**0.1797**	1145.710	**0.2340**	1191.374	**0.2958**	1128.378

(b) CIFAR10 {Train:Test = 70% classes:30% classes}

Methods/Length	8 bits		16 bits		32 bits		64 bits	
	MAP	training time	MAP	training time	MAP	training time	MAP	training time
LSH [1]	0.0871	0.001	0.0971	0.001	0.1160	0.001	0.1222	0.002
PCAH [2]	0.0958	0.258	0.0926	0.261	0.1034	0.313	0.1066	0.482
ITQ [3]	0.1100	0.853	0.1291	1.346	0.1400	3.574	0.1445	6.277
DGH [4]	0.0916	71.532	0.0893	78.963	0.0945	88.605	0.1016	114.303
SGH [5]	0.1099	1.652	0.1059	2.036	0.1142	3.023	0.1156	6.542
CCA-ITQ [3]	0.1607	1.718	0.1653	3.489	0.2188	5.215	0.2197	8.989
SDH [6]	0.2002	6.833	0.2045	15.385	0.2608	19.804	0.2849	33.855
FSDH [8]	0.1781	3.156	0.2292	4.995	0.2776	6.180	0.2778	7.443
HC-SDH [7]	n/a	n/a	0.4205	3.229	0.4270	3.765	0.4295	3.803
FastH [9]	0.3237	378.261	0.4324	541.176	0.5067	812.296	0.5149	1662.214
COSDISH [10]	0.2288	5.119	0.2756	7.867	0.3607	22.784	0.3846	99.903
FSSH [11]	0.4331	63.881	0.4861	67.397	0.5308	90.957	0.5597	74.212
SCDH [12]	0.3730	3.091	0.3992	4.138	0.4984	6.187	0.5108	7.020
SCDH$_K$ [12]	0.4450	7.704	0.5126	8.270	0.5609	10.314	0.5807	11.950
FSSH* [11]	0.3745	3.467	0.4927	3.500	0.4959	3.525	0.5042	3.855
SCDH* [12]	0.4343	1.899	0.4525	2.408	0.4643	4.074	0.4695	9.141
SCDH$^*_K$ [12]	0.3776	56.888	0.3787	58.405	0.3812	62.388	0.3812	70.704
SSDH* [15]	0.4704	903.656	0.5280	902.384	0.5444	904.185	0.5685	904.304
LCH (Ours)	0.4947	1196.833	0.5342	1194.428	0.5428	1194.426	0.5652	1196.592
LCGH (Ours)	**0.5001**	1167.945	**0.5573**	1158.509	**0.5895**	1180.165	**0.6194**	1170.063

(a) LCH (b) LCGH

Fig. 2: The categorical cross-entropy loss in LCH encourages the hash codes to approach its class exemplar and minimize within-class variance. Conversely in LCGH, if the categorical cross-entropy loss is less than ϵ, the loss gradient is ignored allowing within-class-variance to be retained. In the context of the example used in Fig 1, LCH encourages the hash codes of all apples to cluster around the 'red' exemplar while the hash codes trained by LCGH retains variance pertaining to their shade of red. The hash codes of the unseen classes, denoted by red squares, can be distinguished in LCGH but not LCH.



(a) LCH (b) Label Consistent + uncorrelated bits (c) LCGH

Fig. 3: LCH incorporates the softmax cross-entropy loss with bit balance but does not retain within-class variance, the codes are encouraged to be as close to their class exemplars as possible. In (b), note that the label consistency and uncorrelated bits objectives can be contradictory. To maintain label consistency, the first and second bits are correlated. Encouraging uncorrelated bits makes quantization finer but may not discover hidden attributes. (c) LCGH minimizes reconstruction distortion instead of enforcing uncorrelated bits. This allows LCGH to discover hidden clusters of attributes for generalization.

where

$$\text{softplus}(x) = \log(1 + e^x) \tag{21}$$

is a smooth approximation of the rectified linear unit (ReLU) function $\max\{0, x\}$. Appending θ_{rec} into $\Theta = \{\theta_{base}, \theta_{class}, \theta_{rec}\}$, our overall loss minimization problem for LCGH is

$$\min_{\Theta} L = L_{class,\epsilon} + \gamma L_{rec} + \alpha L_{KLD} + \beta \left\|\Theta\right\|^2. \tag{22}$$

The hash codes are obtained as $\mathbf{b}_n = \text{sgn}(\mathbf{m}_n)$, as with LCH.

1) Necessity of the reconstruction loss: In Fig 3, we illustrate the difference between encouraging uncorrelated bits using L_{KLD} and minimizing reconstruction mean-squared error using L_{rec}. While uncorrelated bits encourage finer quantization by giving rise to more bins, the bins may not group hash codes of similar concepts. Minimizing reconstruction mean-squared error discovers clusters of hidden attributes which are not reported in the labels and as illustrated in Fig 3, is not contradictory to the uncorrelated bits objective.

However the label consistency and uncorrelated bits objectives can be contradictory. From Fig (3b), if the first and second bit are not correlated, then each class will be further split into two. For example, there will exist a blue-square {1,0,0} centroid and an orange-triangle {0,0,0} centroid and their Hamming distance of 1 is lower than the Hamming distance between two blue-square {1,0,0}, {1,1,1} centroids. Therefore the codes will no longer be label-consistent. We place priority on label-consistency and relegate uncorrelated bits as a secondary objective.

2) Improved generalization ability of LCGH: LCGH seeks to learn hash codes which first agree with the training labels (label-consistency). This is because the unsupervised information from the reconstruction loss may not be able to overcome the semantic gap–the difficulty of understanding semantics directly from data without human annotation. This is reflected in the ablations, as removal of label information results in a

greater decline in performance than the removal of the reconstruction loss. As such priority is placed on label-consistency and on this foundation, unsupervised information is added to the LCGH codes. The improved information content from both supervised and unsupervised sources helps LCGH to better generalize to unseen classes, resulting in a significant improvement in performance on the cross-domain setting. However due to the reduced margin between hash codes of dissimilar classes, there is a slight decrease in performance for the within-class setting where training and test classes are the same.

V. Experiments

A. Datasets

We compare LCH and LCGH with two benchmarks found in [12]. The datasets used are Caltech-256 [36] and CIFAR10 [37]. For Caltech-256, each image is represented by a 1024-dimensional CNN feature vector, while for CIFAR10, each image is represented by a 512-dimensional GIST feature vector. These datasets were downloaded from [38].

B. Evaluation Metric

A sample is considered relevant if it has the same class label as the query.

We use Mean Average Precision (mAP) to evaluate retrieval performance. Let Q be the number of queries, and N be the number of samples in the database. With sample q as the query, R_q as the number of relevant instances in the database, s_{iq} as the similarity of the i-th retrieved instance with the query, $\text{prec}_q(r)$ as the precision of the top r retrieved set, mAP is defined as

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\left(\frac{\sum_{i=1}^{N}(\text{prec}_q(i)\times s_{iq}}{R_q}\right). \tag{23}$$

C. CBIR experimental settings and results

The results on LSH [1], PCAH [2], ITQ [3], DGH [4], SGH [5], CCA-ITQ [3], SDH [6], HC-SDH [7], FSDH [8], FastH [9], COSDISH [10], FSSH [11], SCDH [12], SCDH$_K$ [12] are taken from [12]. Results on HC-SDH for Caltech-256 and the 8-bit evaluation on CIFAR10 were omitted because HC-SDH requires the code length to be greater than the number of classes ($K \geq C$).

We re-implement the top three methods FSSH, SCDH, SCDH$_K$ using the authors' code and mark them with an asterisk (*). The results for the top performing deep learning competitor SSDH [15] are also implemented by us and included. Our implementations are the average of ten runs with different seeds. The experiments are performed on a computer with Intel® Core™ i3-8100 3.60 GHz CPU and EVGA Nvidia Geforce GTX 1080Ti GPU.

For LCH and LCGH, we use a batch size of 500 images and optimize over 300k such random batches. We use two fully connected layers with output sizes d and K respectively for the base network. The hyperparameters used are $\lambda = 1, \alpha = 10^{-6}$ and $\beta = 0.05$. Additionally for LCGH, $\gamma = 0.1/d, \epsilon = 0.01K$. The initial learning rate is 0.01 which we reduce to 0.001 after 10k batches. SSDH was run with the same base network, learning rate schedule, batch size and number of batches. Default parameters were used for SSDH.

Following the settings in [12], we use 3000 and 6000 random samples as test set for Caltech-256 and CIFAR10 respectively in the 'seen' test setting shown in table II. The remaining images are used as the training and gallery set.

For the 'unseen' test setting shown in table III, we similarly use the 1:9 test:training split for the 'seen' test setting, with 3000 and 6000 random samples as a preliminary test set for Caltech-256 and CIFAR10 respectively and designate the rest as the preliminary training set. Additionally we use a 3:7 test:training split for classes, randomly selecting 180 training classes and 76 testing classes for Caltech-256; 7 training classes and 3 testing classes for CIFAR10. Within the 9:1 training and test sets, we further split them by the selected 7:3 training and test classes to yield sets train70, train30, test70 and test30 as in [39]. Train70 is used as the training set, train30 is the gallery set and test30 is the query set. Test70 is unused.

As we expect, LCH performs the best on the 'seen' evaluation setting while LCGH performs the best on the 'unseen' evaluation setting. We present the t-SNE plots of 64-bit LCH and LCGH on the unseen test30 set of CIFAR10 in Fig 4. LCGH is less tightly clustered than LCH because it captures more feature directions, which allows it to generalize better to unseen data. Unfortunately because it captures directions which are irrelevant to the training classes, it reduces the margin between hash codes of differing classes, making them less tightly clustered and reduces its performance on the within-domain evaluation setting.

LCH captures fewer feature directions (only those relevant to training classes) and where these directions are unable to discriminate between the unseen test classes, LCH codes of different classes become tightly (and wrongly) clustered, making discrimination difficult in the cross-domain setting. However, as LCH does not capture feature directions which are irrelevant to training classes, it performs better than LCGH on the within-domain evaluation setting.

By training embeddings which are less tightly clustered, LCGH is able to improve generalization performance at the cost of a slight reduction in the 'seen' test evaluation.

D. Ablations and Sensitivity Analysis

TABLE IV: Ablations for LCGH on caltech-256

$L_{class,\epsilon}$	α	γ	ϵ	8bit mAP	64bit mAP
	✓	✓	✓	0.071	0.297
✓		✓	✓	0.121	0.311
✓	✓		✓	0.126	0.301
✓	✓	✓		0.108	0.307
✓			✓	0.131	0.305
	✓		✓	0.024	0.024
✓	✓	✓	✓	**0.133**	**0.312**

We perform ablations for LCGH on the caltech-256 dataset for the unseen test classes, in Table IV. The experimental settings and hyperparameters are the same as the experiments

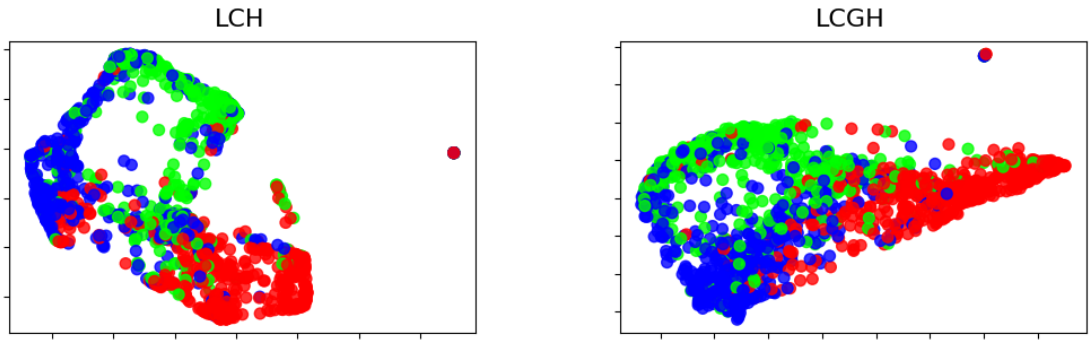


Fig. 4: t-SNE plots of 64-bit LCH and LCGH on the unseen test30 set of CIFAR10. LCGH is less tightly clustered than LCH because it captures more feature directions, which allows it to generalize better to unseen data. LCH captures fewer feature directions and where these directions are unable to discriminate between the unseen test classes, LCH codes of different classes become tightly (and wrongly) clustered, making discrimination difficult.

above, and optimized over 200k random batches for a random split of the dataset. Entries without a checkmark are set to zero in the corresponding row. As the ablation is only done once, the results are slightly different from the experiments in Tables II and III which are averaged over 10 runs.

$L_{class,\epsilon}$ is the label-consistency loss, α promotes code balance, γ correponds to the unsupervised reconstruction loss, while ϵ corresponds to the tolerance of the hash codes away from their class centers.

From the ablations, we can see that each of the terms in (22) contributes to an improvement in performance. We also see that $L_{class,\epsilon}$ and γ are the most important terms. Each of these terms are important to ensure the hash codes are able to generalize to unseen classes, as seen from the relatively high performance they can each generate on their own. When both terms are removed, the loss function is unable to capture meaningful information for generalization and results in the great fall in performance.

We also perform sensitivity analysis for LCGH on the caltech-256 dataset for the unseen test classes at 64 bit hash length, this is shown in Fig 5. The experimental settings are the same as the ablations.

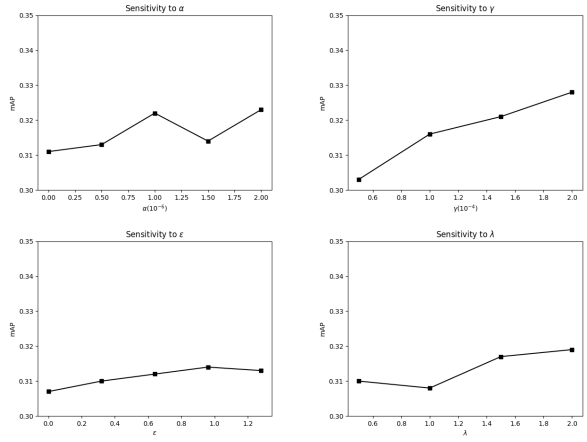


Fig. 5: Sensitivity analysis

From the sensitivity analysis, we see that performance is relatively stable around the chosen hyperparameter values and further improvement can be obtained by increasing $\gamma, \epsilon, \lambda$ from the values used in the experiments. However since there was no validation set to guide the selection of hyperparameters, the hyperparameters were chosen conservatively.

VI. Conclusion

We have developed an end-to-end method, Label Consistent Generalized Hashing (LCGH) which obtains good generalization performance by retaining information which are redundant during training but possibly helpful for generalization. LCGH is based on Label Consistent Hashing (LCH), which allows hash codes to be trained with low quantization error, and obtains state-of-the-art retrieval performance on within-domain settings. We show that LCH and LCGH perform the best on two recent benchmarks and that LCGH achieves the best retrieval performance on 'unseen' test classes while retaining comparable performance on 'seen' test classes.

References

[1] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *Vldb*, 1999, vol. 99, pp. 518–529.

[2] Yusuke Matsushita and Toshikazu Wada, "Principal component hashing: An accelerated approximate nearest neighbor search," in *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2009, pp. 374–385.

[3] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent 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.

[4] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang, "Discrete graph hashing," *Advances in neural information processing systems*, vol. 27, 2014.

[5] Qing-Yuan Jiang and Wu-Jun Li, "Scalable graph hashing with feature transformation," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[6] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 37–45.

[7] Gou Koutaki, Keiichiro Shirai, and Mitsuru Ambai, "Hadamard coding for supervised discrete hashing," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5378–5392, 2018.

[8] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan, "Fast supervised discrete hashing," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 490–496, 2017.

[9] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1963–1970.
[10] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou, "Column sampling based discrete supervised hashing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30.
[11] Xin Luo, Liqiang Nie, Xiangnan He, Ye Wu, Zhen-Duo Chen, and Xin-Shun Xu, "Fast scalable supervised hashing," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 735–744.
[12] Yong Chen, Zhibao Tian, Hui Zhang, Jun Wang, and Dell Zhang, "Strongly constrained discrete hashing," *IEEE Transactions on Image Processing*, vol. 29, pp. 3596–3611, 2020.
[13] Q.-Y. Jiang and W.-J. Li, "Asymmetric Deep Supervised Hashing," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[14] V. E. Liong, J. Lu, L. Y. Duan, and Y. Tan, "Deep Variational and Structural Hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
[15] H. F. Yang, K. Lin, and C. S. Chen, "Supervised Learning of Semantics-preserving Hash via Deep Convolutional Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 437–451, 2018.
[16] Furen Zhuang and Pierre Moulin, "A new variational method for deep supervised semantic image hashing," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4532–4536.
[17] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 1732–1736.
[18] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al., "A Survey on Learning to Hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2017.
[19] Khoa D Doan, Peng Yang, and Ping Li, "One loss for quantization: Deep hashing with discrete wasserstein distributional matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9447–9457.
[20] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," *Advances in neural information processing systems*, vol. 21, 2008.
[21] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Hashing with graphs," in *Icml*, 2011.
[22] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao, "Auto-encoding twin-bottleneck hashing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2818–2827.
[23] Miguel A Carreira-Perpinán and Ramin Raziperchikolaei, "Hashing with binary autoencoders," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 557–566.
[24] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song, "Stochastic generative hashing," in *International Conference on Machine Learning*. PMLR, 2017, pp. 913–922.
[25] W.-J. Li, S. Wang, and W.-C. Kang, "Feature Learning Based Deep Supervised Hashing with Pairwise Labels," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2016, IJCAI'16, pp. 1711–1717, AAAI Press.
[26] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous Feature Learning and Hash Coding with Deep Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.
[27] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable Deep Hashing with Regularized Similarity Learning for Image Retrieval and Person Re-identification," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.
[28] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep Semantic Ranking Based Hashing for Multi-label Image Retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1556–1564.
[29] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep Supervised Hashing for Fast Image Retrieval," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.
[30] T. T. Do, A. D. Doan, and N. M. Cheung, "Learning to Hash with Binary Deep Neural Network," in *European Conference on Computer Vision*. Springer, 2016, pp. 219–234.
[31] H. Jain, J. Zepeda, P. Pérez, and R. Gribonval, "Subic: A Supervised, Structured Binary Code for Image Search," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 833–842.
[32] Jiun Tian Hoe, Kam Woh Ng, Tianyu Zhang, Chee Seng Chan, Yi-Zhe Song, and Tao Xiang, "One loss for all: Deep hashing with a single cosine similarity based learning objective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24286–24298, 2021.
[33] "Freepik images," https://www.freepik.com, Accessed: 2022-07-25.
[34] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *International Conference on Learning Representations*, 2017.
[35] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.
[36] Gregory Griffin, Alex Holub, and Pietro Perona, "Caltech-256 object category dataset," 2007.
[37] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.
[38] Yong Yuan, "Habir: Hashing baseline for image retrieval," https://github.com/willard-yuan/hashing-baseline-for-image-retrieval.
[39] Thanh-Toan Do, Khoa Le, Tuan Hoang, Huu Le, Tam V Nguyen, and Ngai-Man Cheung, "Simultaneous feature aggregating and hashing for compact binary code learning," *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 4954–4969, 2019.

Furen Zhuang is a Research Scientist with the Machine Intellection Department, Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR). He holds a B.A. degree in Chemical Engineering from the University of Cambridge, and a PhD degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign (UIUC).

Pierre Moulin (Fellow, IEEE) received his doctoral degree in 1990, after which he joined Bell Communications Research as a Research Scientist. In 1996, he joined the University of Illinois at Urbana-Champaign, where he is currently Professor in the Department of Electrical and Computer Engineering, Research Professor at the Coordinated Science Laboratory and the Beckman Institute, and Affiliate Professor in the Department of Statistics.

His fields of professional interest include statistical decision theory, statistical signal processing and modeling, machine learning, information security, and Shannon theory. Dr. Moulin has served on the editorial boards of the IEEE TRANSACTIONS ON INFORMATION THEORY, the IEEE TRANSACTIONS ON IMAGE PROCESSING, and the PROCEEDINGS OF IEEE. He was co-founding Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (2005-2008), member of the IEEE Signal Processing Society Board of Governors (2005-2007), member of the IEEE Information Theory Society Board of Governors (2016-2018) and has served IEEE in various other capacities. He is co-recipient of two best paper awards from the IEEE Signal processing Society and was plenary speaker for ICASSP, ICIP, and several other conferences. He is an IEEE Fellow (2003) and was Distinguished Lecturer of the IEEE Signal Processing Society for 2012-2013 and co-chair of the technical program for ISIT 2015. He was UIUC Sony Faculty Scholar and is the recipient of the 2018 Ronald W. Pratt Faculty Outstanding Teaching Award.