LEARNING CIRCULANT SUPPORT VECTOR MACHINES FOR FAST IMAGE SEARCH

Ramin Raziperchikolaei and Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science
University of California, Merced, USA

ABSTRACT

Binary hashing is an established approach for fast, approximate image search. It maps a query image to a binary vector so that Hamming distances approximate image similarities. Applying the hash function can be made fast by using a circulant matrix and the fast Fourier transform, but this circulant hash function must be learned optimally from training data. We show that a previously proposed learning algorithm based on optimization in the frequency domain is suboptimal. We show the problem can be solved exactly and efficiently by casting it as a convex maximum margin classification problem on a modified dataset. We confirm experimentally that this allows us to learn hash functions consisting of one or more circulant filters that provide better retrieval performance for the same query runtime as a linear hash function.

Index Terms— image retrieval, binary hashing

1. INTRODUCTION

As the dataset of images continues to grow, searching for similar images becomes a more challenging problem. Binary hashing is a fast and efficient way to solve the similarity search problems approximately [1]. The main idea is to learn a hash function that maps high-dimensional images into binary codes and search for the similar images in the binary space. If we represent each image by L-bits binary codes, then we only need L bits of memory to store each image. This makes it possible to store large datasets with millions of images in the main memory of a single machine. To search for a query, one can create a hash table, indexed by the binary codes of the training images. Given the binary code of the query, the images inside a small Hamming distance of the query can be returned using the hash table in $\mathcal{O}(1)$ [1].

The main goal in binary hashing is to learn a good hash function that can preserve the similarity between the points after mapping them to the binary space. To achieve this, different objective functions and optimization methods have been proposed [2, 3, 4, 5, 6, 7], usually based on hash functions of the form $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{W}\mathbf{x}) \in \{-1,+1\}^L$, where $\mathbf{x} \in \mathbb{R}^D$ is a D-dimensional feature vector representing the image. The time needed to generate a binary vector for a query image is $\mathcal{O}(LD)$. As we increase the number of bits L, we can preserve the similarity better and hence improve the retrieval quality, but we need more time to generate the binary codes.

A recent method, circulant binary embedding (CBE) [8], proposed to learn a hash function with a single circulant weight matrix, which gives better time and space complexity. As we show in section 2, using circulant weight matrices we need $\mathcal{O}(D\log D)$ to compute the binary codes and need $\mathcal{O}(D)$ to store the weight matrix. In [8], a classification subproblem is solved by doing optimization in the frequency domain. This has two important disadvantages: (1) it relaxes the classification problem into a regression problem (ignoring the sign function) and (2) when $L < D$ (which is the case of interest) the optimization algorithm finds a suboptimal matrix, which works worse in practice, as we will show later in our experiments.

In this paper, we provide an algorithm that does learn the hash function's circulant matrix optimally even for $L < D$ bits. In section 3, we formulate the problem as a convex maximum margin classification problem, which can be solved exactly by training a single support vector machine on a suitably modified training set. In section 4 we show experimentally that this results in better classification accuracy and generalization, and better retrieval results in binary hashing.

1.1. Related work

Binary hashing can be divided into two groups of unsupervised hashing and supervised hashing, based on how we define the similarity between the points. In supervised hashing, two points are considered similar if they are semantically similar (for example, if they have the same label) [9, 10, 3, 4, 5, 7, 11, 12]. In unsupervised hashing, two points are considered similar if they are close to each other in the high-dimensional space (for example, if the Euclidean distance between them is smaller than a threshold) [13, 14, 15, 16, 17, 8, 6]. In this paper, we focus on unsupervised hashing.

Locality sensitive hashing (LSH) [13] considers thresholded random projections as the hash functions. LSH is an example of data-independent methods. Data-dependent methods try to learn the hash functions by optimizing an objective function that is defined based on the similarity of the training points [14, 15, 16, 17, 8, 6]. Most papers focus on introducing better objective functions, better optimization methods, or more complicated hash functions to improve the retrieval results. Usually, the space and time complexity of the hash function at test time (i.e., to generate the codes for a given input image) is at least $\mathcal{O}(LD)$, where L is the number of bits and D the dimension of the original points. As L increases, they become less useful.

To overcome this problem,[17] proposed to use a bilinear projection based coding to generate the binary codes. In the case of using $L = D$ bits, the time and space complexity of the method are $\mathcal{O}(D)$ and $\mathcal{O}(D^{1.5})$, respectively. A more recent approach [8] proposed to use a circulant weight matrix as the hash function and optimize the objective in the frequency domain. Using the fast Fourier transform to generate the binary codes, the computational complexity of this method to generate the codes is $\mathcal{O}(D\log D)$.

2. HASHING WITH A CIRCULANT WEIGHT MATRIX

We quickly review circulant matrices and explain their usefulness in binary hashing. A D-dimensional vector $\mathbf{w} = (w_0, w_1, \cdots, w_{D-1})$ is the basis for the $D \times D$ circulant matrix $\mathbf{W}$, which can also be

Work supported by NSF award IIS–1423515.

regarded as a filter operating on the input image:

$$\mathbf{W} = \operatorname{circ}(\mathbf{w}) \equiv \begin{bmatrix} w_0 & w_{D-1} & \cdots & w_2 & w_1 \\ w_1 & w_0 & w_{D-1} & \cdots & w_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ w_{D-1} & w_{D-2} & \cdots & w_1 & w_0 \end{bmatrix}. \quad (1)$$

So the matrix $\mathbf{W}$ has only D (instead of D^2) free parameters and we only need $\mathcal{O}(D)$ to store it. Consider the hash function $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx})$ where $\mathbf{W}$ is circulant. We show here that $\mathbf{h}(\mathbf{x})$ can be computed in $\mathcal{O}(D \log D)$ instead of D^2. Consider $\mathcal{F}(\cdot)$ as the discrete Fourier transform, and $\mathcal{F}^{-1}(\cdot)$ as the inverse discrete Fourier transform. Since $\mathbf{W}$ is circulant, the output of the hash function can be computed as [18]:

$$\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx}) = \operatorname{sgn}\left(\mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \circ \mathcal{F}(\mathbf{w}))\right) \quad (2)$$

where $\mathcal{F}(\mathbf{x}) \circ \mathcal{F}(\mathbf{w})$ is the elementwise product of two vectors. Computing the sign and elementwise product takes $\mathcal{O}(D)$ and computing the discrete Fourier transform and the inverse Fourier transform (using the fast Fourier transform) takes $\mathcal{O}(D \log D)$. So the total time needed to generate the code for one input is $\mathcal{O}(D \log D)$.

If the hash function needs to generate $L < D$ bits, we only need the first L rows of $\operatorname{circ}(\mathbf{w})$, which we denote as $\operatorname{circ}(\mathbf{w})_L$. If we use the discrete Fourier transform, we first need to generate the D-bits codes and then use L of them, so the complexity remains $\mathcal{O}(D)$ space and $\mathcal{O}(D \log D)$ time. This is faster than directly computing $\mathbf{Wx}$ unless L is very small.

In this paper, we also propose to use multiple circulant matrices (filters) to construct the hash function, unlike CBE, which uses a single filter. Using f filters, we divide the L classification problems into f independent problems, each of them defined over L/f bits, and solve each of them independently using one circulant matrix. Using f filters, the number of free parameters increases from D to fD. This leads to better hashing results, but also increases the complexity to $\mathcal{O}(fD)$ space and $\mathcal{O}(fD \log D)$ time.

3. CIRCULANT SUPPORT VECTOR MACHINES

Assume we have N training points $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N) \in \mathbb{R}^{D \times N}$ in D-dimensional space. The goal is to learn a hash function $\mathbf{h}$: $\mathbb{R}^D \to \{-1,+1\}^L$ that maps D-dimensional points into the L-dimensional binary codes. We define $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{L \times D}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^L$ is the bias. So learning the hash function corresponds to learning $\mathbf{W}$ and $\mathbf{b}$.

We first show how learning the hash functions appears with the form of a classification problem in several hashing methods. Then, we describe our method to learn the optimal circulant weight matrix.

3.1. Learning the hash function by learning classifiers

In many hashing papers, learning the L-bits hash function involves solving L independent classification problems [2, 3, 4, 5, 6, 7], possibly iteratively. The main idea is to define N L-dimensional binary variables $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N) \in \{-1,+1\}^{L \times N}$, and define the objective over the binary codes $\mathbf{Z}$ instead of the hash function $\mathbf{h}$. After optimizing the objective over the codes $\mathbf{Z}$, we need to learn the hash function given the codes. In [2, 3, 4], the hash function is learned a posteriori, as a final step. In [5, 6, 7], the algorithm iterates over learning codes and hash functions, which achieves better optima.

To learn the hash function given the codes, we need to solve the following problem:

$$\min_{\mathbf{W},\mathbf{b}} \sum_{l=1}^{L} \sum_{n=1}^{N} \left(\operatorname{sgn}\left(\mathbf{w}_l^T \mathbf{x}_n + b_l\right) - z_{ln}\right)^2 \quad (3)$$

where $\mathbf{w}_l^T$ is the lth row of $\mathbf{W}$, $\operatorname{sgn}\left(\mathbf{w}_l^T \mathbf{x}_n + b_l\right)$ gives the lth bit of the hash function, and $z_{ln} \in \{-1,+1\}$ is the lth bit of the nth training point. Since the rows of $\mathbf{W}$ are independent from each other, this problem can be solved by training L independent classifiers. For the lth problem, the (input,label) pairs is determined by ($\mathbf{X}$,$\mathbf{Z}_{l\cdot}$).

3.2. Using a circulant SVM classifier as the hash function

We explain our proposed method in this section. We assume that we have the binary codes $\mathbf{Z} \in \{-1,+1\}^{L \times N}$ and we try to learn the circulant matrix $\mathbf{W} = \operatorname{circ}(\mathbf{w})_L$ and the bias $\mathbf{b}$ that minimize the classification error of eq. (3). Since the weight matrix is circulant, the L classification problems are not independent: they all share the same weight values, but in different orders as shown in eq. (1).

To minimize the classification error, we consider the maximum margin formulation of support vector machines (SVMs). Consider $\mathbf{w}_l^T$ as the lth row of the matrix $\mathbf{W}$. The lth classification problem has the following form:

$$\min_{\mathbf{w}_l} \frac{1}{2}\|\mathbf{w}_l\|^2 + C\sum_{n=1}^{N} \xi_{ln} \quad \text{s.t.} \quad \begin{cases} z_{ln}(\mathbf{w}_l^T \mathbf{x}_n + b_l) \geq 1 - \xi_{ln} \\ \xi_{ln} \geq 0, \; n = 1, \dots, N \end{cases}$$

where z_{ln} and ξ_{ln} are the label and the slack variable of the nth point in the lth classification problem, $\mathbf{w}_l$ is the weight vector of the lth classifier and b_l is its bias. From eq. (1), each row of $\mathbf{W}$ is a permutation of the vector $\mathbf{w}$ (first column of $\mathbf{W}$). For this reason, we can write row l of $\mathbf{W}$ as $\mathbf{w}_l^T = \mathbf{w}^T \mathbf{P}_l$, where $\mathbf{P}_l \in \mathbb{R}^{D \times D}$ is a permutation matrix. Based on this formulation, we can rewrite the SVM formulation of the lth classification problem as:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}^T \mathbf{P}_l\|^2 + C\sum_{n=1}^{N} \xi_{ln} \quad \text{s.t.} \quad \begin{cases} z_{ln}(\mathbf{w}^T \mathbf{P}_l \mathbf{x}_n + b_l) \geq 1 - \xi_{ln} \\ \xi_{ln} \geq 0, \; n = 1, \dots, N. \end{cases}$$

Since $\mathbf{P}_l^T \mathbf{P}_l = \mathbf{I}$, $\|\mathbf{w}^T \mathbf{P}_l\|^2 = \|\mathbf{w}\|^2$, so all L classification problems have the same margin term. Let us define $\mathbf{t}_{ln} = \mathbf{P}_l \mathbf{x}_n \in \mathbb{R}^D$. Since $\mathbf{P}_l$ is a permutation matrix, $\mathbf{P}_l \mathbf{x}_n$ does not change the values of $\mathbf{x}_n$, it only changes the order of the features. So $\mathbf{t}_{ln}$ has the same dimension and values as $\mathbf{x}_n$, but permuted based on the $\mathbf{P}_l$ matrix. Using the newly introduced vectors $\mathbf{t}_{ln}$, we can write all L classification problems in one formula as follows:

$$\min_{\mathbf{w}} \frac{L}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{L}\sum_{n=1}^{N} \xi_{ln} \text{ s.t.} \begin{cases} z_{ln}(\mathbf{w}^T \mathbf{t}_{ln} + b_l) \geq 1 - \xi_{ln}, \\ \xi_{ln} \geq 0, \; n = 1, \dots, N, \\ l = 1, \cdots, L. \end{cases} \quad (4)$$

This looks very similar to the SVM problem, where $\mathbf{w} \in \mathbb{R}^D$ is the weight vector that we try to learn and we have NL input points $\mathbf{t}_{ln}$ with labels z_{ln}. The only difference is the bias of SVMs: in this formulation we have to learn L different biases while in the traditional SVM only one bias exists. We augment the weight vector $\mathbf{w}$ with the bias vector $\mathbf{b}$ and write it as $\overline{\mathbf{w}} = [\mathbf{w}; \mathbf{b}] \in \mathbb{R}^{D+L}$. We also augment each of the inputs $\mathbf{t}_{ln}$ by $\mathbf{e}_l$ and write it as $\mathbf{y}_{ln} = [\mathbf{t}_{ln}; \mathbf{e}_l] \in \mathbb{R}^{D+L}$, where $\mathbf{e}_l \in \mathbb{R}^L$ has 1 in the lth element and zeros everywhere else. Now we can rewrite eq. (4) as :

$$\min_{\mathbf{w},\mathbf{b}} \|\mathbf{w}\|^2 + \frac{2C}{L}\sum_{l=1}^{L}\sum_{n=1}^{N} \xi_{ln} \text{ s.t.} \begin{cases} z_{ln}([\mathbf{w}; \mathbf{b}]^T \mathbf{y}_{ln}) \geq 1 - \xi_{ln}, \\ \xi_{ln} \geq 0, \; n = 1, \dots, N, \\ l = 1, \cdots, L. \end{cases} \quad (5)$$

This is now an SVM problem, with NL inputs $\mathbf{y}_{ln}$ and labels z_{ln}. To see the equivalence between eq. (4) and eq. (5), note that $[\mathbf{w}; \mathbf{b}]^T \mathbf{y}_{ln} = \mathbf{w}^T \mathbf{t}_{ln} + b_l$. The only difference between eq. (5) and the more standard SVM formulation is that the first term (inverse of the margin) is defined over the first D elements of the weight vector $\overline{\mathbf{w}} = [\mathbf{w}; \mathbf{b}]$, not over all the elements.

To summarize, we start with L classification problems as given in eq. 3, each of them defined over N training points, but which are coupled through the circulant weight matrix $\mathbf{W} = \text{circ}(\mathbf{w})$. We convert this classification problem into one maximum margin classification problem over the vector $\mathbf{w}$, with an enlarged dataset of NL points and labels (eq. (5)). The new points are L different permutations of the original points $\mathbf{X}$ and the labels are the columns of the code matrix $\mathbf{Z}$.

Advantages of our circulant SVM over the optimization in the frequency domain. CBE [8] minimizes the L classification problems of eq. (3) in the frequency domain. The main disadvantage of the CBE is that it always needs a binary matrix of size $N \times D$: each point has to have $L = D$ labels. For $L < D$, CBE adds $D - L$ labels of zero to all the points to make the code vector D-dimensional. This means that CBE returns suboptimal solutions for $L < D$. If $L \ll D$, the number of zeros in the labels becomes much more than the original labels, and the results become much worse.

Our proposed method always returns the optimal solution, even for the case of $L < D$. It formulates the classification problem as a maximum margin classification, which is a convex quadratic program. There are libraries available that solve SVM problems for a large number of points in a few seconds. Our experiments confirm that our circulant SVM always performs better than CBE.

3.3. Time and space complexity

In the circulant case, using f filters, we need to store f circulant matrices, which gives a space complexity of $\mathcal{O}(fD)$, no matter how many bits we use. In the linear case, the space complexity is $\mathcal{O}(DL)$ to store a full $D \times L$ matrix, for L bits.

In the linear case (full matrix), the time complexity of generating the binary code of length L is c_1LD, where c_1 is a constant. For circulant matrices, considering the general case of f filters, we have to compute the Fourier transform of $\mathbf{x}$ (which takes $c_2D\log D$), Fourier transform of the f filters (takes $c_2fD\log D$), compute f elementwise products (takes c_3fD) and f inverse Fourier transform (takes $c_2fD\log D$). This gives a total runtime of $c_2(2f+1)D\log D + c_3fD$. Note the runtime of generating the binary codes is again independent of the number of bits L.

In the experiments, we fix the time needed to generate the binary codes for linear methods (full matrix) and circulant hashing methods. This means that we set the number of bits of the linear methods to $L = c_2(2f+1)\log D$. Following the experiments of [8], we set the constant $c_2 = 1.66$. Note that for practical values of D, c_3fD is much smaller than the other term and that is why we ignore it.

4. EXPERIMENTS

We use the following datasets in our experiments: (1) CIFAR-10 [19] contains 60 000 images. We consider 58 000/2 000 images as the training/test set. We extract $D = 4\,096$ VGG network [20] features, which are the output of the last fully connected layer of the VGG network. (2) ImageNet [21] dataset contains 1 000 000 images. We randomly select 500 000 images as the training set and 2 000 images as the test set. We represent each image by 4 096 dimensional VGG features (the same as CIFAR10 dataset).

Following the experiments of [8], the ground-truth set for each image consists of the first 10 nearest neighbors of the image in the original high-dimensional space. The retrieved set for each image consists of its k nearest neighbors in the Hamming space.

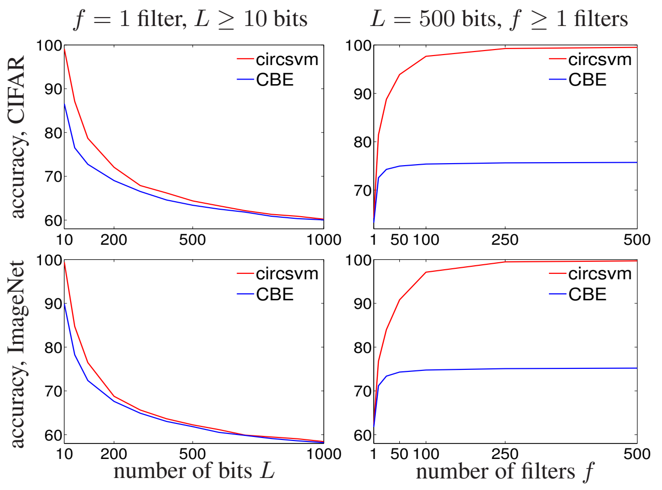

Fig. 1. Average classification accuracy of circulant SVM (circsvm) and CBE [8] on CIFAR and ImageNet datasets. *First column:* we fix the number of circulant matrices $f = 1$ and change number of bits L. *Second column:* we fix the number of bits $L = 500$ and increase the number of circulant matrices f.

We make data points zero mean before learning the classifiers or hash functions. We train the SVM classifier using the VLFeat [22] library, which uses stochastic gradient descent in the optimization. This gives us the opportunity to load a subset of data points in the memory and update the model based on them. We set the parameter λ of this library to 0.01 and the number of epochs to 10. Our circulant SVM takes around 5 minutes to solve $L = 500$ classification problems, each defined over $N = 5\,000$ points. Note that we convert the problem to one SVM problem with 2.5×10^6 points.

4.1. Our circulant SVM improves the classification accuracy

We compare our proposed method (circsvm) and CBE [8] by reporting the average classification accuracy of L classification problems in eq. (3). We select a random subset of 5 000 points from the CIFAR10 and ImageNet datasets as the input $\mathbf{X}$. We give $\mathbf{X}$ as the input to Iterative Quantization (ITQ) [16] and consider the output of ITQ as the binary matrix $\mathbf{Z}$. Then, given $\mathbf{X}$ and $\mathbf{Z}$ as the input and the labels, we train a classifier with the circulant weights using our proposed method and CBE [8]. The objective function of CBE has an orthogonality term in addition to the classification term of eq. (3). To make the objectives of circulant SVMs and CBE the same, we set the weight of the orthogonality term (λ) to 0.

We show the results in fig. 1. In the first column of fig 1, we use 1 filter and change the number of bits from 10 to 1 000 and report the classification accuracy. We can see that circsvm performs better than CBE, specially for smaller number of bits. The reason is that circsvm finds the optimal solution (for any value of L), but CBE finds suboptimal solutions for $L < D$. As we increase the number of bits, the classification accuracy of methods decreases. The reason is that for larger L, the classification problem becomes more difficult, and the classifier needs more free parameters to solve it, but the number of free parameters of the methods is fixed to D.

In the second column of fig. 1, we fix the number of bits to 500, change the number of filters of CBE and circsvm, and report the average classification accuracy. Increasing the number of filters means that the classifier has more parameters and one would expect this to translate into a better precision. In fact, this happens for circsvm but not for CBE. The classification accuracy of circsvm improves

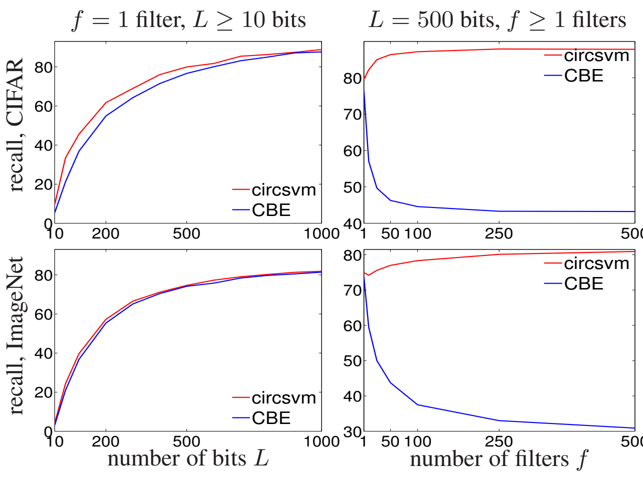

Fig. 2. We compare circsvm with CBE [8] by reporting the recall in the nearest neighbor search. *First column:* we fix the number of circulant matrices to 1 and change the number of bits. *Second column:* we fix the number of bits to 500 and increase the number of circulant matrices. Retrieved set: $k = 100$ nearest neighbors of the query.

significantly as we increase the number of filters. The classification accuracy of CBE improves a little bit, and then stops improving. The reason is that as we increase the number of filters, each circulant matrix will be trained on a smaller set of bits (in the extreme case with $f = 500$ filters, each circulant matrix is trained on 1 bit). Since CBE always needs D bits, it adds a massive number of zeros to the bits, which results in finding poor classifiers.

4.2. Better classification leads to better hashing results

In fig. 1, we trained the hash functions of circsvm and CBE and reported the classification accuracy. Now, we use those functions in the hashing setting to report the recall of the methods in fig. 2. In the first column of fig. 2, we increase the number of bits and report the recall. The recall of circsvm is always above the recall of CBE, for different values of bits. This is consistent with what we have seen in the classification results of fig. 1. Also note that even when the classification accuracy goes down as we increase the number of bits in fig. 1, the recall in fig. 2 increases. The reason is that increasing the number of bits in unsupervised hashing leads to better hashing results, as long as diversity between the hash functions exists.

The second column of fig. 2 shows the recall as we increase the number of filters. The recall of the circsvm always increases in this figure, because increasing the number of filters leads to a huge improvement in the classification results of circsvm (see fig. 1). This again shows that a better classification leads to better hashing results. Note the improvement of circsvm in the classification setting is more than in the hashing setting. The reason is that even using 1 filter, circsvm learns a reasonably good hash function.

For CBE, the recall goes down massively as we increase the number of filters. The reason is the same as the one explained in the classification setting. Increasing the number of filters means learning circulant functions on smaller number of bits. CBE adds a massive number of zeros to the labels, which makes the inputs and labels of different hash functions very similar to each other. In this case, the L hash functions can end up being very similar to each other, which leads to losing diversity among them. As investigated in [23], lack of diversity in the set of hash functions leads to poor retrieval results. We can see that this happens for CBE in fig. 2.

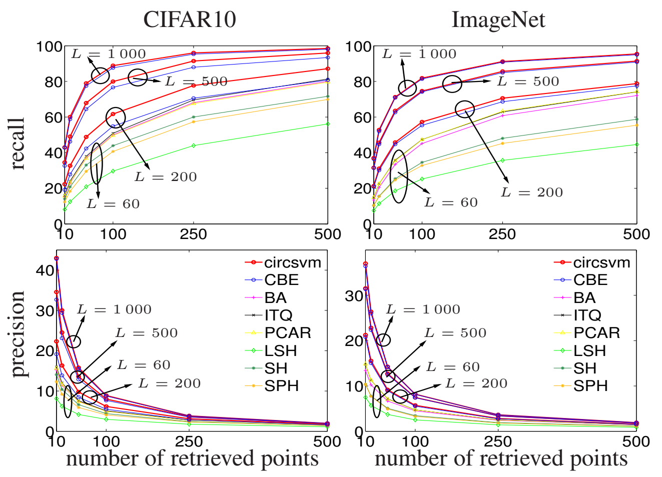

Fig. 3. Comparing circsvm with different hashing methods. We fix the time needed to generate the binary codes for different methods, which means the methods using linear hash functions use fewer bits than CBE and circsvm. Specifically, for linear hash functions $L = 60$ bits and for CBE and circsvm $L \in \{200, 500, 1\,000\}$ bits.

4.3. Comparison with other hashing methods

We compare our proposed method circsvm with several unsupervised hashing methods: Iterative Quantization (ITQ) [16], Spectral Hashing (SH) [14], Circulant Binary Embedding (CBE) [8], Binary Autoencoder (BA) [6], Spherical Hashing (SPH) [24], Locality Sensitive Hashing (LSH) [13], and thresholded rotated PCA (PCAR). Fig. 3 shows the results. We set the codes of circsvm and CBE to the output of ITQ. In this figure, we fix the number of filters $f = 1$ for circulant hashing methods (circsvm and CBE). Following [8], we use a random subset of 5 000 points for training different methods. We fix the computational time needed to generate the binary codes for different methods. Based on the discussion of sec. 3.3, the number of bits of methods using circulant hash functions can be any number less than D, and the number of bits of methods using linear hash functions should be $L = 60$. For circsvm and CBE, we report results using 200, 500 and 1 000 bits.

In almost all cases, the methods using circulant hash functions outperform those using linear ones. circsvm always beats CBE and the other methods in both datasets. This is more clear when we use smaller number of bits (e.g. $L = 200$). As we increase the number of bits, CBE and circsvm become very similar, while circsvm is still a little better. Note that achieving better results using smaller number of bits is important in binary hashing because smaller number of bits can lead to faster retrieval results.

5. CONCLUSION

Using a circulant matrix as the weight matrix of a hash function makes the computation of the binary codes very fast, $\mathcal{O}(D \log D)$ for D-dimensional inputs. This is very helpful in binary hashing, where the goal is fast image search. We showed that a previous method that learns the circulant matrix by optimizing in the frequency domain is suboptimal and its results become the more inaccurate the smaller the number of desired bits L is in the hash function. We also proposed to learn the circulant matrix in the original domain, by formulating the L classification problems using a circulant matrix as one maximum margin classification problem. This leads to a convex quadratic program whose optimal solution can be found efficiently for any desired number of bits L. This in turn gives better results in the retrieval task.

6. REFERENCES

[1] Kristen Grauman and Rob Fergus, "Learning binary hash codes for large-scale image search," in *Machine Learning for Computer Vision*, R. Cipolla, S. Battiato, and G. Farinella, Eds., pp. 49–87. Springer-Verlag, 2013.

[2] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu, "Self-taught hashing for fast similarity search," in *Proc. of the 33rd ACM Conf. Research and Development in Information Retrieval (SIGIR 2010)*, Geneva, Switzerland, July 19–23 2010, pp. 18–25.

[3] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel, "A general two-step approach to learning-based hashing," in *Proc. 14th Int. Conf. Computer Vision (ICCV'13)*, Sydney, Australia, Dec. 1–8 2013, pp. 2552–2559.

[4] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. of the 2014 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'14)*, Columbus, OH, June 23–28 2014, pp. 1971–1978.

[5] Tiezheng Ge, Kaiming He, and Jian Sun, "Graph cuts for supervised binary coding," in *Proc. 13th European Conf. Computer Vision (ECCV'14)*, Zürich, Switzerland, Sept. 6–12 2014, pp. 250–264.

[6] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei, "Hashing with binary autoencoders," in *Proc. of the 2015 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'15)*, Boston, MA, June 7–12 2015, pp. 557–566.

[7] Ramin Raziperchikolaei and Miguel Á. Carreira-Perpiñán, "Optimizing affinity-based binary hashing using auxiliary coordinates," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 640–648, MIT Press, Cambridge, MA.

[8] Felix Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang, "Circulant binary embedding," in *Proc. of the 31st Int. Conf. Machine Learning (ICML 2014)*, Eric P. Xing and Tony Jebara, Eds., Beijing, China, June 21–26 2014, pp. 946–954.

[9] Brian Kulis and Trevor Darrell, "Learning to hash with binary reconstructive embeddings," in *Advances in Neural Information Processing Systems (NIPS)*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. 2009, vol. 22, pp. 1042–1050, MIT Press, Cambridge, MA.

[10] Mohammad Norouzi and David Fleet, "Minimal loss hashing for compact binary codes," in *Proc. of the 28th Int. Conf. Machine Learning (ICML 2011)*, Lise Getoor and Tobias Scheffer, Eds., Bellevue, WA, June 28 – July 2 2011.

[11] Miguel Á. Carreira-Perpiñán and Ramin Raziperchikolaei, "An ensemble diversity approach to supervised binary hashing," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 757–765, MIT Press, Cambridge, MA.

[12] Ramin Raziperchikolaei and Miguel Á. Carreira-Perpiñán, "Learning independent, diverse binary hash functions: Pruning and locality," in *Proc. of the 17th IEEE Int. Conf. Data Mining (ICDM 2016)*, Barcelona, Spain, Dec. 12–15 2016, pp. 1173–1178.

[13] Alexandr Andoni and Piotr Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Comm. ACM*, vol. 51, no. 1, pp. 117–122, Jan. 2008.

[14] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, Daphne Koller, Yoshua Bengio, Dale Schuurmans, Leon Bottou, and Aron Culotta, Eds. 2009, vol. 21, pp. 1753–1760, MIT Press, Cambridge, MA.

[15] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Hashing with graphs," in *Proc. of the 28th Int. Conf. Machine Learning (ICML 2011)*, Lise Getoor and Tobias Scheffer, Eds., Bellevue, WA, June 28 – July 2 2011, pp. 1–8.

[16] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A Procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[17] Yunchao Gong, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. of the 2013 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'13)*, Portland, OR, June 23–28 2013, pp. 484–491.

[18] Alan V. Oppenheim and Alan S. Willsky, *Signals and Systems*, Signal Processing Series. Prentice-Hall, second edition, 1996.

[19] Alex Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. of Computer Science, University of Toronto, Apr. 8 2009.

[20] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the 3rd Int. Conf. Learning Representations (ICLR 2015)*, San Diego, CA, May 7–9 2015.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of the 2009 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'09)*, Miami, FL, June 20–26 2009, pp. 248–255.

[22] Andrea Vedaldi and Brian Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008.

[23] Ramin Raziperchikolaei and Miguel Á. Carreira-Perpiñán, "Learning supervised binary hashing: Optimization vs diversity," in *IEEE Int. Conf. Image Processing (ICIP 2017)*, Beijing, China, Sept. 17–20 2017.

[24] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon, "Spherical hashing," in *Proc. of the 2012 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'12)*, Providence, RI, June 16–21 2012, pp. 2957–2964.