Supervised Discrete Hashing

Bosco Frank Paul and Vyom Raj

Abstract— Image Hashing is the process of converting the images to binary codes. The idea is to have similar images to have similar codes and dissimilar images to have dissimilar codes. Similarity of two images is measured by whether they belong to the same class. The problem is hash optimizations algorithms are particularly NP Hard and that is what makes the learning algorithm extremely challenging.In this report we implemented the Supervised Discrete Hashing Algorithm which involved learning the Binary codes through Discrete Cyclic Coordinate Descent(DCC) method for the L2 loss.The SDH method is implemented on MNIST Dataset.If we could come up with a good hashing technique for the image, then the hashed binary code could be used for linear classification of the image.

Index Terms—MNIST Dataset, Discrete Hashing, NP Hard, L2 Loss, Precision, Recall, Maximum Apriori

1 INTRODUCTION

In this project, we are trying to find the optimal hashing technique for a dataset of images, so that they can be used for linear classification. The tasks include understanding high level concepts of the paper [1]. Understand and implement the SDH algorithm and precision, recall for the same. Also try to replicate the results presented in the paper [1]. The main task of this project is to learn the Hash Function F(x) using the discrete cyclic coordinate descent with l2 loss function mentioned in the paper. The results obtained are used to find the precision, recall and MAP for different code length on MNIST dataset.

2 LITREATURE REVIEW

Hashing has witnessed an increased use for Information Retreival. There have been various implementations of Discrete Hashing ranging from the earliest being used for Peer to Peer systems to Supervised Fast Discrete Hashing [2]. Hashing is basically encoding the various structured, semi structured or unstructured data eg images, videos or documents by a set of small binary codes and making sure that the similarity of the original data has been maintained. In order to maintain highly efficient indexes and maintaining massive data we need to implement learning-based data-independent hashing methods. One of the most commonly learning based data independent hashing method is Locality Sensitive Hashing algorithm which is used for data dimensionality reduction. Dimension reduction is basically a concept in machine learning where the number of random variables are reduced by obtaining a set of principal values. The various subparts of Dimension reduction are Feature Selection and Feature Reduction. Feature Selection is a concept in which we try to find a subset of original attributes and applying either filter stratergy, wrapper stratergy or embedded stratergy. Feature Extraction is when the dimensions of the dataset is reduced. The various ways of doing dimensionality reduction are Principal Component Analysis, Linear Discriminant Analysis or LSH [3]. LSH is used for solving the approximate or the exact Nearest Neighbour Search in high Dimensional spaces.The newest data independent LSH algorithm achieves better performance as compared to the classical LSH algorithms by implementing the concept called as data independent hashing.The difference betweem the recent data independent Hashing method[5] and LSH is the use of short hash codes <200 for the available training data.The various types of hashing are Semi Supervised Hashing (SSH) , Linear Hashing(LDA)[4][5], Fast Hashing[6], Ranking based Supervised Hashing[7],Iterative Quantization[8], Isotropic Hashing[9],Supervised Minimum Loss Hashing[10] .Semi Supervised Hashing[11] handles both metric as well as semantic similarity .It basically involves minimizing the empirical error on the labeled data by maximing variance and the independence of hash bits using the labeled as well as unlabeled data.Unsupervised Hashing methods usually perform well with metric distances but when we need to do image search the performance is much lower. Generally supervised hashing methods are useful for implementing sematic similarity but when the labeled data is noisy, this method get more prone to overfitting. There are various algorithms to generate non-linear hash functions in the kernel space such as Kernel Based Supervised Hashing [12], Random Maximum Margin Hashing [13], Binary Reconstruction Embedding [14]. The problem with binary codes which are generated by the hash functions is that the generation to mixed-integer optimization problem which is NP hard problem. Therefore, in order to solve these NP Hard problem, we need to have a relaxation scheme. This relaxation scheme shall help us to deal with the optimization problem through discarding the discrete constraints and then quantizing the solution in order to obtain the approximate binary solution. The bottleneck is that these approximation schemes accumulate the quantization error in the case of learning long code lengths. Using a relaxation scheme [2][3] is also really important since directly learning the binary codes won't lead us to solutions which are manageable or scalable.

In the work "Supervised Discete Hashing" [1] the authors have proposed a novel framework which optimizes the binary hash codes adequately and conveniently. In order to build upon supervised label information, the algorithm uses a hashing framework for linear classifica-

- *First Author is with the Department of Computer Science, Arizona State University, AZ 85281. E-mail: bfpaul@asu.edu*
- *Second. Author is with the Department of Computer Science, Arizona State University, AZ 85281. E-mail: vraj5@asu.edu*

tion wherein the learned binary codes are supposed to be optimal for Classification.The learned binary codes can be seen as non-linear feature vectors of the original dataset while the label information is used for the binary feature vectors for classification. This idea uses the concept of joint optimization procedure which involves learning a binary embedding and a linear classifier together. A set of hash functions are optimized in order to fit the binary bits which are learnt. Since capturing the underlying non-linear structure is really important therefore the kernel space is used for learning the hash functions. The optimization problem deals with solving the three associated sub problems and one of the most important subproblem in the Supervised Hashing framework is solving the binary code optimization problem. The author has therefore proposed a Discrete Hashing Cyclic Descent Algorithm (DCC) [14] to generate the hash codes bit by bit. The DCC algorithm generates the hash bits in such a manner that it scales up to even massive datasets. Supervised Hashing with Decision Tree has been implemented which an efficient Graph Cut Based block [15][18][19]search approach has been used and for learning the hash functions the boosted decision trees are trained.

In the recent time Fast Supervised Discrete Hashing has been implemented which is based on the existing work Supervised Discrete Hashing. FSDH [14] uses an effective regression of class labels of the training examples of the corresponding hash code to reduce the running time of this algorithm. FSDH only requires a single hash code solving step rather than iterative which makes the algorithm even more efficient. FSDH is about 151 times faster than SDH when the number of hashing bit is 64 on the MNIST dataset. Other than FSDH there has been recent development in which Convolutional Neural Network [6] has been used to learn the hash function as deep learning method have shown superior performance as compared to the traditional methods in Image classification, image segmentation and classification. The deep learning based supervised discrete hashing algorithm outperforms the state of the art methods on benchmark image retrieval datasets.

3 DATASET DESCRIPTION

The MNIST Dataset [27] is the dataset which has been used in our project for running our Supervised Discrete Hashing Algorithm. The dataset consists of handwritten digits which is available for novel research purpose and learning the concepts of Pattern Recognition and Machine Learning.It is an excellent dataset where the focus entirely lies on learning the concepts of statistical learning rather than spending time on data extraction, formatting or preprocessing. The MNIST Dataset consists of images that are handwritten. The training set has 60,000 instances and the testing set has 10,000 instances. The MNIST dataset is actually a subset of a much larger dataset called as NIST. The MNIST test dataset has 30,000 patterns from SD-3 and 30,000 pattern from SD-1 with the help of approximately 250 writers while the test set has 5000 patterns from SD-3 and 5000 patterns from SD-1[27] which aren't disjoint.The files which we are using for our project has images and class label of both the training set and the test set.The default format of the dataset given on the website is ubyte, In our implementation of the project we converted the MNIST Dataset into a python pandas readable format known as pickle. The pixel values are organized row wise and are between 0 and 255. 0 indicates a clear white background while 1 indicates black. We have also scaled down MNIST features by dividing them by 255.

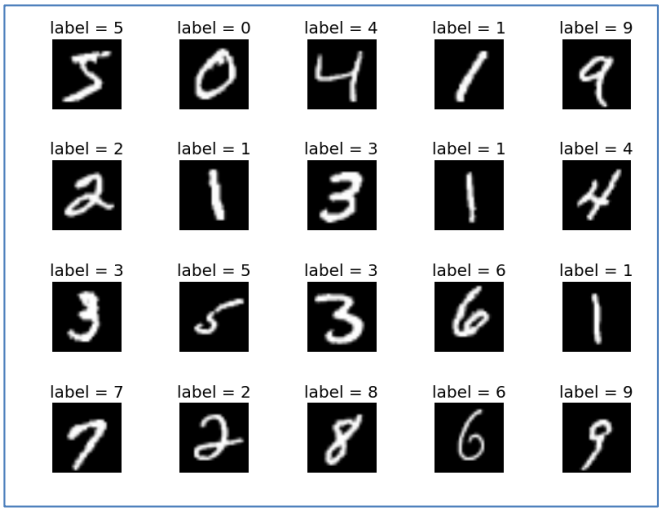

Figure1: A sample of MNIST Data Representation.

4 ALGORITHM FOR SUPERVISED DISCRETE HASHING

Algorithm: Supervised Discrete Hashing(SDH) [1]

Input: Training data $\{x_i, y_i\}_{i=1}^{n}$; code length L; number of anchor points m; maximum iteration number t; parameters λ and ν.

Output: Binary Codes $\{b_i\}_{i=1}^{n} \in \{-1,1\}^{L \times n}$;
Hash function H(x)=sgn(F(x)).

1. Randomly select m samples $\{a_j\}_{j=1}^{m}$ from the training data and get the mapped training data Φ(x) via the RBF Kernel function.
2. Initialize b_i as a $\{-1,1\}^{L}$ vector randomly, ∀i.
3. Loop until converge or reach maximum iterations:
   -G-Step: Calculate W using equation (8) or multi-class SVM
   -F- Step: Compute P using (5) to form F(x).
   -B-Step: For the l_2 loss, iteratively learn$\{b_i\}_{i=1}^{n}$ bit by bit using the DCC method with equation (15);

5 METHODOLOGY USED

Our aim is to learn the set of Binary Codes $B = \{b_i\}_{i=1}^{n} \epsilon \{-1,1\}^{L\times N}$ [1]. The hypothesis in this case is that excellent binary codes are ideal for classification too. The binary codes are optimal for learned linear classifier. The multi-class classification formula used is given by.

$$y = G(b) = \mathbf{W}^{T}\mathbf{b} = [w_1^{T}b, \cdots, w_C^{T}b]^{T} \quad (1)$$

In Machine Learning multi class classification is a task in which the instances are classified into more than two classes. The classification into two classes is commonly called Binary Classification.

In equation (1) $W_k \in R^{L\times 1}, k = 1 \ldots, C$ is classification vector for class k and $y \in R^{C\times 1}$ is the label vector indicating the class of random element $x_i (i = 1, 2, \cdots n)$.

Further the method adopted has to optimize the problem.

$$\min_{B,W,F} \sum_{i=1}^{n} L(y_i, W^{T}b_i) + \lambda \| W \|^2 \quad (2)$$

$$s.t\, b_i = sgn\big(F(x_i)\big), i = 1, \cdots, n.$$

In this equation (2), the loss function is L. Loss function is a function that is basically a cost associated with the event. Our aim is to minimize the loss function. The sgn(.) is the sign function which gives +1 for positive numbers and -1 for negative numbers. The equation (2) is NP Hard and with the variable b_i exteremly hard to minimize. In order to implement regularization methods in large scale optimization techniques the equation (2) is written as:

$$\min_{B,W,F} \sum_{i=1}^{n} L(y_i, W^{T}b_i) + \lambda \| W \|^2 + \nu \sum_{i=1}^{n} \| b_i - F(X_i) \|^2$$
$$s.t.\, B \epsilon \{-1,1\}^{L\times n} \quad (3)$$

Here ν is the penalty parameter and the term $\sum_{i=1}^{n} \| b_i - F(X_i) \|^2$ models the fitting error of the binary codes b_i. The next step is to find F(x).

3.1 Approximating b_i by non-linear embedding

We can either use a linear or a non-linear learning algorithm for calculating F(X). We have used a non-linear form for our calculations [1].

$$F(x) = P^{T}\phi(x) \quad (4)$$

Here $\Phi(x)$ represents a coloumn vector which is given by the Radial Basis Function Kernel mapping and is a m-dimensional vector. RBF is a two-layer artificial neural network that uses radial basis functions as the activation function. The $\Phi(x)$ is obtained by the following equation.

$$\phi(x) = [\exp\left(-\frac{\| x - a_1 \|^2}{\sigma}\right), \cdots, \exp\left(-\frac{\| x - a_m \|^2}{\sigma}\right)]^{T}$$

Here $\{a_j\}_{j=1}^{m}$is randomly selected m anchor points from our training sample and σ is our kernel width. We have seen that the function F(x) has been used in calculating Binary Constructive Embeddings and learning the hash function [20]. The aim is similar as in the hash functions are learnt by explicitly minimizing the reconstruction error between the original distances and the Hamming distances of the corresponding Binary embeddings. Even while doing Supervised Hashing with Kernels similar concept where the idea is to map the data to compact binary codes such that the hamming distances between the similar pairs and the non-similar pairs are minimized [22,23].

F step:

We calculate the projection matrix which is a square matrix that gives us a vector space projection and is calculated using our previously obtained $\emptyset(x)$ by the following equation which does not depend on our loss function.

$$P = (\phi(X)\phi(X)^{T})^{-1}\phi(X)B^{T} \quad (5)$$

3.2 Joint learning with $l_2$loss

The equation (3) can be rewritten as

$$\min_{B,W,F} \sum_{i=1}^{n} \| y_i - \mathbf{W}^{T}\mathbf{b}_i \|^2 + \lambda \| \mathbf{W} \|^2 + \nu \sum_{i=1}^{n} \| b_i - F(X_i) \|^2 \quad (6)$$

$$s.t.\, b_i \epsilon \{-1,1\}^{L}$$

Further reducing the above equation (6) we have

$$\min_{B,W,F} \sum_{i=1}^{n} \| Y - W^{T}B \|^2 + \lambda \| W \|^2 + \nu \| B - F(X) \|^2 \quad (7)$$

$$s.t.\, B \epsilon \{-1,1\}^{L\times n}$$

G Step: Using multi class SVM or by fixing B we can calculate W by using a solution given by closed form equation.

$$W = (BB^{T} + \lambda I)^{-1}BY^{T} \quad (8)$$

B Step: This is the step in which the algorithm iteratively learns $\{b_i\}_{i=1}^{n}$ bit by bit using the Discrete Cyclic coordinate descent method. The equation (7) can be rewritten as:

$$\min_{B} \| Y - W^{T} \|^2 + \nu \| B - F(X) \|^2 \quad (9)$$

$$s.t.\, B \epsilon \{-1,1\}^{L\times n}$$

The equation (9) can be solved for one row of B which can be achieved by fixing all the other rows and simultaneously learning one bit at a time. We need to reduce the equation to.

$$\min_{B} \| Y \|^2 - 2Tr(Y^{T}W^{T}B) + \| W^{T}B \|^2 + \nu(\| B \|^2 - 2Tr\big(B^{T}F(X)\big) + \| F(X) \|^2 \quad (10)$$
$$s.t.\, B \epsilon \{-1,1\}^{L\times n}$$

which is approximately equal to:

$$\min_{B} \| W^{T}B \|^2 - 2Tr(B^{T}Q) \quad (11)$$

$$s.t.\, B \epsilon \{-1,1\}^{L\times n}$$

Here Tr(.) indicates the trace norm which is the sum of diagonal elements of the matrix and Q is calculated as $Q = WY + \nu F(X)$.

Discrete Cylic Coordinate Descent (DCC Method).
We assume
z^T be the l^{th} row of B. B' be matriz B excluding z.
q^T be the l^{th} row of Q. Q' be matriz Q excluding q.
v^T be the l^{th} row of W. W' be matriz W excluding v.
also, l varies from 1 to L. Now we have

$$\begin{aligned} \| W^T B \|^2 &= \mathrm{Tr}(B^T w w^T B) \\ &= \text{const} + \| z v^T \|^2 + 2 v^T W'^T B' z \\ &= \text{const} + 2 v^T W'^T B' z \end{aligned} \tag{12}$$

In the above equation $\| z v^T \|^2 =$ constant and similarly

$$\mathrm{Tr}(B^T Q) = \text{const} + q^T z. \tag{13}$$

Adding equations (11), (12) and (13) together we get the following equation.

$$\min_z \left(V^T W'^T B' - q^T \right) z \tag{14}$$

which has a sign function as the optimal function in B', W'

$$z = \mathrm{sgn}(q - B'^T W' v) \tag{15}$$

We calculate each bit of our hashed code using z. This is done for all the datapoints and all the code (code length). We fix rest of the code in order to find the current z.

5 RESULTS AND EXPERIMENTS

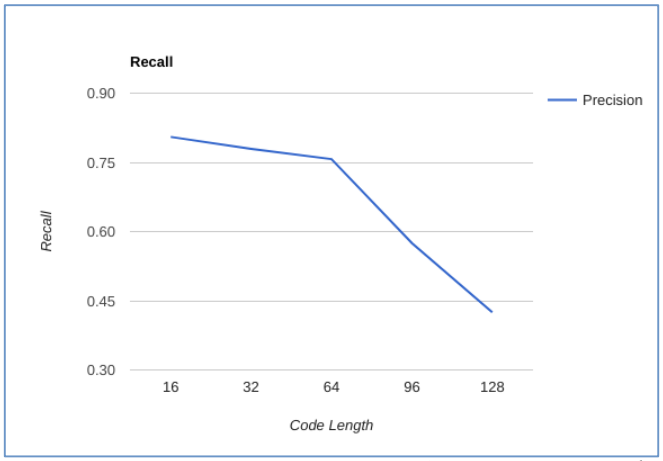

Figure 2: Recall on MNIST Dataset (various code lengths)

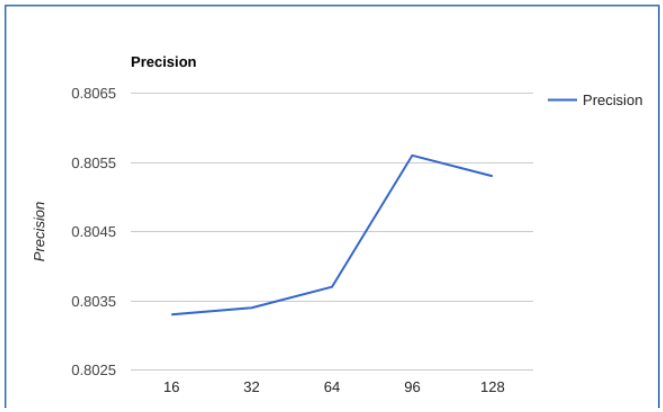

Figure 3: Precision on MNIST Dataset (various code lengths.)

We implemented the algorithm using python 2.7. We initially converted ubyte dataset to pkl file using python-mnist for ease of use in python. We used pandas as our dataframe to hold our testing and training dataset. We have used 1000 anchor points to calculate the ϕ matrix for both testing and training dataset. Using the SDH algorithm we learned both hashing values B and weight function W. We used W on our testing set to calculate the accuracy of our calculation. We got an accuracy averaging around 89% for different code length. The major difficulty was computing the l2 loss for SDH iteration [1]. Our machine ran out of memory during the same. So, we used change in norm of hashed binary code B as our basis for convsergence. Our code got completed in under 30 minutes for code length upto 64-bit. As we increased the code lenth till 128-bit, it took around 1 hour for completion.

We have computed the presion and recall values on the test dataset of 10000 objects with hamming distance of 2. As a general trend, you can see that Recall reduces with increase in code length. Whereas Precision remains more or less fixed at around 0.80. We got the maximum value of presion at 96-bit hashed code length. Recall value reducing with code length is expected. We have a fixed value of hamming distance and as the code length increases, the probability of two similar objects having same code decreases. For a realistic look at recall values, we could increase the hamming distance as our code length increases.

System Specifications
Intel Core I5, 8GB Ram, 64 bit Ubuntu 16.04 LTS

Python Libraries Used
Numpy, scipy, matplotlib, python-mnist, pandas, sklearn, python-tk

6 FUTURE SCOPE

After the implementation of Supervised Discrete Hashing, there has been many other other implementations as the growth of big data has lead to a need of an efficient design and indexing method. The most recent ones being Online Hashing wherein research has been done on to accommodate data coming from online learning. Another interesting implementation is Deep Supervised Hashing for Fast Image Retrieval [23] where Convolutional Neuaral Networks have been combinely used for learning robust image representations on various computer vision tasks [25][26]. Implementing Deep Supervised Hashing have performed exceedingly well on CIFAR-10 and NUS Wide Dataset than the other state of art algorithms. Sequential Discrete Hashing has also been implemented which is inspired by the current implementation of Supervised Discrete hashing in which each bit of a code can be sequentially learned with a discrete optimi-

zation scheme that jointly minimizes the empirical loss based on a boosting stratergy. There has also been an implementation for image search and retereival using nonlinear transformation which does not uses the concept of error prone relaxation method to learn the transformation and leverages on solving the discrete optimization problem to eliminate quantization error accumulation. The future scope of this concept of image hashing lies in either getting a new way for improving the accuracy of this present algorithm or moving to semisupervised, unsupervised or reinforcement supervised discrete hashing techniques.

7 Conclusion and Challenges

The binary codes generated by Supervised Discrete Hashing did generate promising classification performance. The precision obtained with different code lengths were different and the maximum code length accuracy was generated when the code length is 96. We noticed that precision increases or remains almost constant as the codelength increases. The recall always reduces when code length increases. This is a veryinteresting observation. The joint learning objective coupled with hash code generation and linear classifier is an excellent method for leveraging semantic label information. The challenges faced in doing the project was firstly coding the algorithm requires patience and time. Also, there is a randomization factor to the algorithmmaking it difficult to reproduce exact results. The algorithm involves lots of computation time and memory thereby making the l2 loss difficult to optimize.

Acknowledgment

Supervised Discrete Hashing is the idea presented in the paper - Supervised Discrete Hashing Fumin Shen, Chunhua Shen, Wei Liu, Heng Tao Shen Computer Vision and Pattern Recognition, 2015 [1]. We would like to thank Fumin Shen, Chunhua Shen, Wei Liu, Heng Tao Shen for this novel idea, so that we could learn by implementing it. We express our appreciation to MNIST for free access to their handwritten digits database. We also express our sincere gratitude to Professor Baoxin Li and the TA's Vijetha Gattupalli and Kevin Ding for providing us with interesting projects to work on.

References

[1] Shen, Fumin, Chunhua Shen, Wei Liu, and Heng Tao Shen. "Supervised discrete hashing." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 37-45. 2015.

[2] Strecha, Christoph, Alex Bronstein, Michael Bronstein, and Pascal Fua. "LDAHash: Improved matching with smaller descriptors." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, no. 1 (2012): 66-78.

[3] Crammer, Koby, and Yoram Singer. "On the algorithmic implementation of multiclass kernel-based vector machines." *Journal of machine learning research* 2, no. Dec (2001): 265-292.

[4] Datar, Mayur, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. "Locality-sensitive hashing scheme based on p-stable distributions." In *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253-262. ACM, 2004.

[5] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248-255. IEEE, 2009.

[6] Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. "LIBLINEAR: A library for large linear classification." *Journal of machine learning research* 9, no. Aug (2008): 1871-1874.

[7] Gionis, Aristides, Piotr Indyk, and Rajeev Motwani. "Similarity search in high dimensions via hashing." In *VLDB*, vol. 99, no. 6, pp. 518-529. 1999.

[8] Gong, Yunchao, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik. "Learning binary codes for high-dimensional data using bilinear projections." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 484-491. 2013.

[9] Gong, Yunchao, 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* 35, no. 12 (2013): 2916-2929.

[10] A. Joly and O. Buisson. Random maximum margin hashing. In Proc. CVPR, 2011.

[11] W. Kong and W.-J. Li. Isotropic hashing. In NIPS 25, 2012.

[12] Kulis, Brian, and Trevor Darrell. "Learning to hash with binary reconstructive embeddings." In *Advances in neural information processing systems*, pp. 1042-1050. 2009.

[13] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE TPAMI, 35(12):2916–2929, 2013.

[14] Zhu, Xiaofeng, Zi Huang, Hong Cheng, Jiangtao Cui, and Heng Tao Shen. "Sparse hashing for fast multimedia search." ACM Transactions on Information Systems (TOIS) 31, no. 2 (2013): 9.

[15] Zhang, Dell, Jun Wang, Deng Cai, and Jinsong Lu. "Self-taught hashing for fast similarity search." In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pp. 18-25. ACM, 2010.

[16] Weiss, Yair, Antonio Torralba, and Rob Fergus. "Spectral hashing." In Advances in neural information processing systems, pp. 1753-1760. 2009.

[17] Weiss, Yair, Rob Fergus, and Antonio Torralba. "Multidimensional spectral hashing." Computer Vision–ECCV 2012 (2012): 340-353.

[18] Wang, Weiran, and Miguel A. Carreira-Perpinán. "The Role of Dimensionality Reduction in Classification." In AAAI, pp. 2128-2134. 2014.

[19] Wang, Jun, Wei Liu, Andy X. Sun, and Yu-Gang Jiang. "Learning hash codes with listwise supervision." In Proceedings of the IEEE International Conference on Computer Vision, pp. 3032-3039. 2013.

[20] Wang, Jun, Sanjiv Kumar, and Shih-Fu Chang. "Semi-supervised hashing for large-scale search." IEEE Transactions on Pattern Analysis and Machine Intelligence 34, no. 12 (2012): 2393-2406.

[21] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of Machine Learning Research 9, no. Nov (2008): 2579-2605.

[22] Shen, Fumin, Chunhua Shen, Qinfeng Shi, Anton Van Den Hengel, and Zhenmin Tang. "Inductive hashing on manifolds." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1562-1569. 2013.

[23] Raginsky, Maxim, and Svetlana Lazebnik. "Locality-sensitive binary codes from shift-invariant kernels." In Advances in neural information processing systems, pp. 1509-1517. 2009.

[24] Oliva, Aude, and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope." International journal of computer vision 42, no. 3 (2001): 145-175.

[25] Liu, Wei, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang. "Compact hyperplane hashing with bilinear functions." arXiv preprint arXiv:1206.4618 (2012).

[26] Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." *Neural computation* 15, no. 6 (2003): 1373-1396.

[27] Deng, Li. "The MNIST database of handwritten digit images for machine learning research [best of the web]." IEEE Signal Processing Magazine 29, no. 6 (2012): 141-142.