MAJOR TECHNICAL PROJECT ON

Deep Hashing via Deep Learning

INTERIM PROGRESS REPORT

to be submitted by

Deepak Sharma
B14107

for the award of the degree
of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MANDI, MANDI

November 2017

1 Introduction

With the advancement of modern technology, billions of images have been uploaded or shared online. To find an image containing a particular object or partially similar objects is like finding a needle in a haystack. This has attracted great attention especially in the field of large scale visual search. The main objective of such algorithms is to retrieve the most relevant visual content from datasets consisting of corpuses size which even exceeds the main memory capacity of a single machine, in a very accurate and efficient way. There are existing algorithms like the nearest neighbor search and tree-based techniques but they are only for low dimensionality data search and are not scalable to higher dimensional data. These methods can be very expensive computationally and with regards to hardware requirements.

In order to speed up the similarity computation and save storage space in memory hashing algorithms play a vital role. The main objective of a hash function is to map each visual object into a binary feature vector, these binary feature vectors are low dimensional as compared to previous high dimensional objects. This approach makes sure that the visually similar objects are mapped with the similar binary vectors.

Currently two kinds of hashing methods are used in the industry: data-independent and data-dependent. In data-independent algorithms, random projections are used to map samples and then binarization is performed. For the second category, various statistical learning techniques are used to learn hashing functions to map samples into binary code.

In this project, I have been planning to use a deep hashing method [1] to learn compact binary codes for scalable image search which uses a deep neural network to seek multiple hierarchical non-linear transformations to learn compact binary codes. The main motives of which are: 1) the loss between the compact real-valued code and the learned binary vector is minimized, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible.

2 Background

In this section, we briefly review three related topics: 1) Large scalable image search, 2)Deep Learning.

- Scalable Image Search

  Efficient approximate nearest neighbor search algorithms are important to scalable image search. Existing algoithms include quantizations base methods and tree-based methods. Many efficient hashing methods have been proposed in the recent years due to their excellent efficiency in both the storage and search speed. Hashing based methods provide faster retrieval speed since calculating the Hamming distance only requires a bit-wise operations. Currently many computer vision applications are using hashing based methods such as object recognition, image retreival, image matching, face recognition.As discussed earlier, hashing based methods can be generally classified into: data-independent and data- dependent. The most representative method for Data-independent can be LSH , LSH preserves the cosine similarity of samples by using random projections obtained from Gaussian distributions to map samples into binary features [2]. In recent years LSH has been extended like [3] proposed an unbiased similarity estimation method by performing orthogonal random projections in a batch manner. For existing data-dependent hashing methods can be a good example. Weiss et al. [4] presented a spectral hashing method to obtain balanced binary codes by solving a spectral graph partitioning problem.

- Deep Learning

  Deep learning aims to learn hierarchical feature representations by building high-level features from raw data. In recent years, a variety of deep learning algorithms have been proposed in computer vision and machine learning and some of them have successfully applied to many visual analysis applications image classification, object detection, action recognition, face verification. Representative deep learning methods include deep stacked auto-encoder [5], deep convolutional neural networks [6], and deep belief network [7]. Deep learning has achieved great success in various visual applications including scalable image search. To my knowledge, Semantic hashing [8] is the first work on using deep learning techniques to learn hashing functions for scalable image search. They applied the stacked Restricted Boltzmann Machine (RBM) learn compact binary codes for document search. However, their model is complex and requires pre training, which is not efficient for practical applications.

3 Problem Statement

With the advancement of security, biometric has played a vital role in catching frauds, theft and also in cyber security. Several Biometric based technologies are already been floating in the market that aims at higher level of security and can be seen being used in banks, labs, houses and even vehicles nowadays are embedded with biometric security. Biometric security system have been removing the usage of passwords in various devices because they are very precise and are not easily hackable. Experiments in the field of biometric are widely performed using fingerprints, iris images, knuckles, palmprints because they are unique for each and every person and persons identity can be easily identified using such technologies. Aadhar Card nowadays have been proved to be widely accepted across the nation, over 1.171 billion as of 15 Aug 2017 aadhar cards have been issued.

Processing over all these images and estimating the identity of a person efficiently and accuratly is just like finding a needle in a haystack. These kinds of queries requires good and efficiently optimized algorithms which can extract the identity based on the biometric information provided by the person in a matter of seconds. Several matching algorithms exists in the market which can efficiently such problems like the nearest neighbour search. But processing over such large images is a bit hefty and requires a lot of computational processing, are even time consuming and may even exceed the main memory of a single system.

For addressing the above problem, hashing can be bit savior. Hashing methods reduces the dimension of the images into low dimension and maps all the similar images separately thus reducing computational processing and memory consumption. I plan on devising a hashing algorithms which can solve the problem efficiently. It should maps the images into binary vectors such that the quantizations loss is minimized. It should also be able to take care of distorted images like proper alignment, transitions. It should even index the binary vectors (obtained after hashing) properly into some kind B-tree, such that time complexity for searching an image can be reduced.

3.1 Challenges

To address the above problem following are the various challenges:

- Generate Binary Vectors
- Indexing
- Matching

4 Work Done During 7th Semester

Estimating correspondences between images is one the fundamental problems in computer vision [9] with applications ranging from large-scale 3D reconstruction to image manipulation and semantice segentation [10].In this project, My first challenge is to build on the traditional approach and develop a convolutional neural network (CNN) architecture that mimics the standard matching process and will replace the standard local features with powerful trainable convolutional neural network features [11], which allows us to handle large changes of appearance between the matched images. The outcome is a convolutional neural network architecture trainable for the end task of geometric matching,which can handle large appearance changes, and is therefore suitable for both instance-level and category-level matching problems.

My second task was to provide a matching layer at the end which can handle sensor inter-operability issues. In heterogeneous sensor environment, it is crucial to identify the source sensor by which the acquired image is captured. This is essentially required to handle sensor inter-operability issues and further in identifying various attacks on biometric systems, where biometric templates can be modified or mis-used. Another interesting application of sensor identification is in establishing the sequence of commands for law enforcement to identify spurious activities in online systems. An image can be altered or fabricated during the acquisition phase, transmission or during storage. In order to understand whether the image has been fabricated or not it is necessary to know the source that generates the image

Fingerprint sensors can be classified into various categories e.g. (i) basis of imaging technology they are classified as optical, capacitive and thermal; (ii) basis of user interaction they are classified as press, sweep and non-contacted ones. Fig 1 shows fingerprint images captured from different types of sensors. It is evident from Fig 1, that image quality largely depends upon underlying sensor employed.

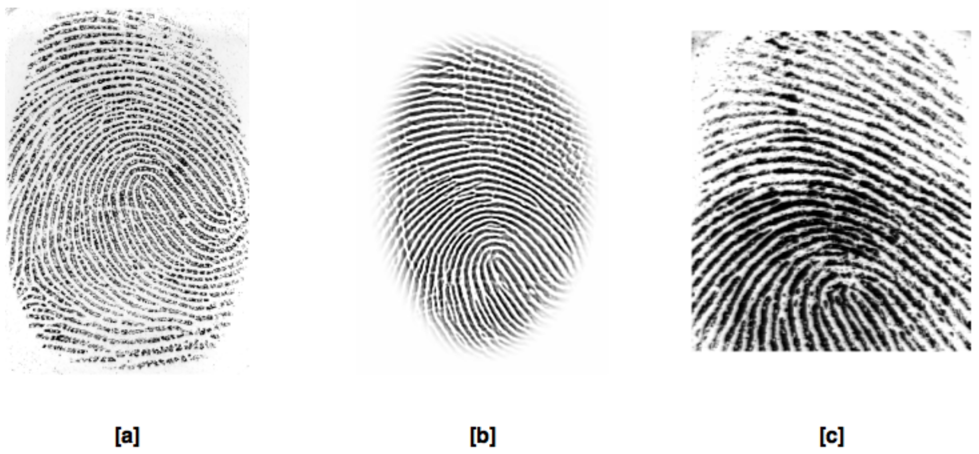


Figure 1: Example of fingerprint images taken from different sensors (a) Futronic, (b) Lumidigm, (c) SecuGen

After first evaluation my work was mainly focused on providing a matching layer which can handle sensor inter-opearbility issues. But to address this problem first we should be able to classify images on the basis of the sensors i.e. detect from which sensor the image has came. After this process the matching problem can be broken down into two (1) Inter Sensor Matching and (2) Intra Sensor Matching. The main contribution for fingerprint sensor detection is three fold, that is summarized in the following section.

- An architecture based on Deep Convolutional Neural Network is proposed that is capable of detecting input fingerprint sensor by systematically pruning and training two different types of convolutional neural networks VGG and ResNet50 namely.
- In-depth feature analysis is done to understand the real-insight of features learned by different layers.
- A highly generalized deep convolutional neural network based architecture has been proposed.

Extensive experimentation has been done in order to decide the suitable network for our fine grain finger-print sensor classification problem. We have considered two networks (a) A shallow network (VGG) (b) A deep network (ResNet50) in order to understand the type of features, classification accuracy and generalization ability trade-off between shallow and deeper networks. As our finger-print sensor classification problem is not a trivial one, we are required to extract features at granular level. Another important point of consideration here is that our fingerprint image size is small and using a network having large kernel size will not be too useful. Considering all the above points in mind we conducted two set of experimentation

- Shallow Network (VGG-19 variant)

  In the first set of experimentation we have used a variant of VGG-19 a popular deep-convolutional neural network model as shown in Fig 2. The foremost advantage of using this network is its small kernel filter size of 3 X 3 which tries to learn high-level features at granular levels. VGG-19 network is divided into 5-blocks with 19 weight layers. It takes an input image of size 224 X 224. After doing experimentation we have found that the block-5 is not adding any discriminative information for our fingerprint sensor classification problem so we systematically prune it and add a dropout layer also to avoid overfitting . While fine tuning this network we have used adam optimizer with a mini-batch size of 64, initial learning rate has been set as 0.001 for 15 epoches. All the parameters that are used in this work are calculated empirically over a small validation set.

- Deep Network(Resnet50 variant)

  In the second set of experimentation we have considered variant of ResNet50 architecture. ResNet50 network is very deep consisting of 50 weight layers, it is approximately 2.6 times deeper than the VGG-19 model. We have deliberately chosen this network in order to find whether the knowledge of predecessor input to every layer gives it an upper hand in learning discriminative features essential for fingerprint sensor classification or not. After doing extensive experimentation we have found that Block2 and Block4 of ResNet50 are extremely important for learning discriminative information, which is necessary for differentiating between images acquired from different fingerprint sensors. We also have observed that Branch 5 and Branch 3 have similar contribution in final classification for our specific fingerprint sensor classification problem. Hence one can drop anyone layer, but dropping both have caused drastic performance deterioration. Hence we have dropped Branch 5 because generated feature map size was only 7 X 7. Fig 2 shows the proposed network architecture, which has been designed in a manner so that it can classify commonly used fingerprint sensors. In this model, we have dropped Branch 5, since in our case this branch was not learning much discriminative information. By doing so, we have decreased the computation time while retaining the performance.

4.1 Experimantal Analysis

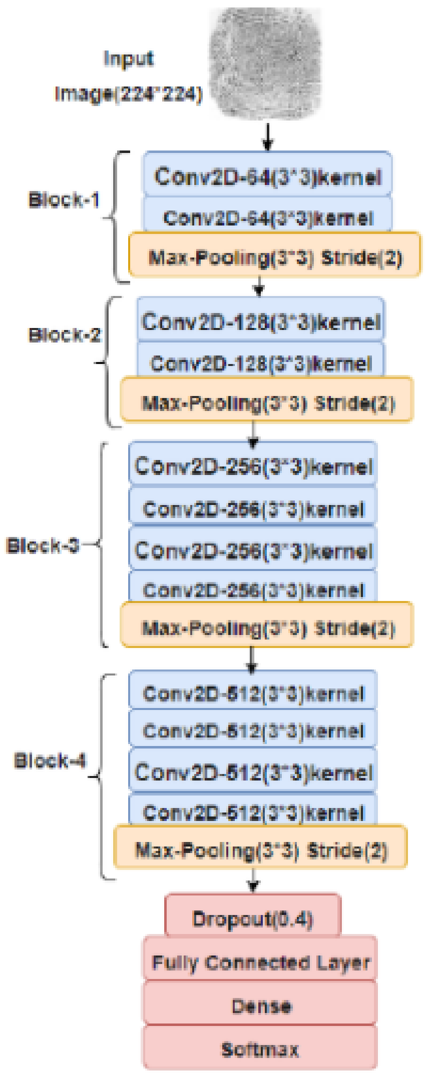
The proposed two architectures described in the previous section has been tested upon four publicly available benchmark single fingerprint databases viz., (i) FVC2002, (ii) FVC2004 and (iii) FVC2006 (iv) IITD-MOLF and on IITK database which is not publicly available and the largest dataset comprising of more than 40, 000 images so far.

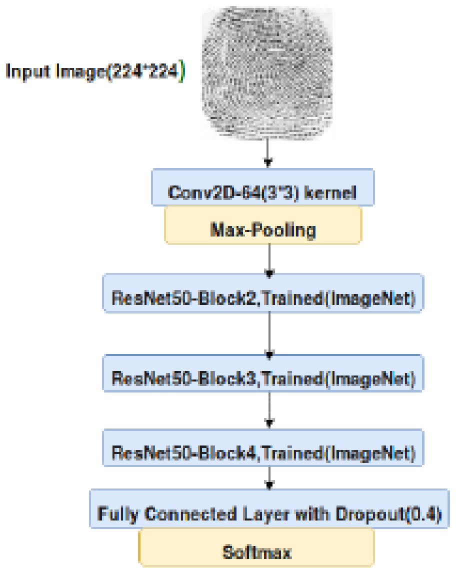
The IITK dataset of single fingerprints consists of 41, 129 images collected using three different types of sensors viz., (i) Futronic (FS88H), (ii)Lumidigm V310 (V31X) and (iii) SecuGen Hamster I. All of them are of same 500 DIP, but the light source and the image size generated is different. All FVC dataset consists of images collected from four different types of sensors. We have trained the proposed model by taking only 10% of the single finger print images as training set and rest 90% as testing set, thus adopting a very difficult protocol.

The results are computed for two different testing strategies namely (a) Intra sensor classification (b) Multi sensor classification. The Correct Classification Rate (CCR%) is computed for performance evaluation, higher the value better is the result

- Intra-sensor Classification

  In this type of classification training and testing is performed on images acquired from same type of sensor model. Table 1, indicates the computed performance of two different proposed architectures on various datasets. The main phenomenon that we observed is that we are getting almost same type of CCR(%) inspite of the fact of using two different kind of varied architectures. This can be attributed to the fact that fingerprint classification problem is a fine grain problem which does not require to learn features at the finest granular level and thus even shallow network is giving remarkable performance





Input Image(224*224)
Conv2D-64(3*3) kernel
Max-Pooling
ResNet50-Block2,Trained(ImageNet)
ResNet50-Block3,Trained(ImageNet)
ResNet50-Block4,Trained(ImageNet)
Fully Connected Layer with Dropout(0.4)
Softmax

[a] [b]

Figure 2: [a] VGG-19 based shallow network, [b] Resnet50 based deep network.

Table 1: Intra Sensor qualitative performance in CCR(%) on proposed architectures(DB1 means the images from the first sensor of the corresponding dataset and same nomenclature is used for other datasets)

Datasets	Sensor Type	Shallow	Deep
FVC 2002	DB1	100	100
	DB2	99.17	96.79
	DB3	100	99.72
	DB4	99.72	99.45
	Aggregate	99.72	98.99
FVC 2004	DB1	100	100
	DB2	100	100
	DB3	99.85	93.78
	DB4	100	99.17
	Aggregate	99.96	98.23
FVC 2006	DB1	100	99.87
	DB2	100	99.28
	DB3	100	99.80
	DB4	99.93	100
	Aggregate	99.98	99.73
IITD-MOLF	DB1	100	100
	DB2	100	100
	DB3	99.92	100
	Aggregate	99.97	100
IITK Dataset	Futronic	99.45	99.34
	Lumidigm	100	100
	SecuGen	100	100
	Aggregate	99.82	99.78

- **Multi-sensor Classification**

  In multi-sensor classification, data is fused together from various fingerprint sensors. We did this purposefully in order to check the generalizability of our proposed architectures. We have merged data from all FVC datasets i.e. FVC2002, FVC 2004 and FVC 2006. The combined dataset consists of 13, 063 images resulting from 12 different sensors and trained our network with 12 output neurons. We have trained our proposed model on 1306 images and tested on remaining 11, 757 images (i.e. 10% training and 90% testing). Table 2 indicates the computed performance in terms of CCR(%). We have observed that the obtained results are quite remarkable which depicts the high gener alizability of our proposed architectures. In this case also shallow networks are learning discriminative features quite well

Table 2: Mutli-sensor qualitative performance in CCR(%) on proposed architectures (here in xDBy: x stands for FVC dataset number and y stands for sensor type of the corresponding dataset)

Dataset	Sensor Type	Shallow	Deep
FVC Combined	2DB1	98.75	99.29
	2DB2	97.63	98.37
	2DB3	100	99.57
	2DB4	99.58	99.72
	4DB1	99.31	99.16
	4DB2	100	100
	4DB3	100	99.40
	4DB4	100	97.47
	6DB1	100	100
	6DB2	100	100
	6DB3	99.86	98.56
	6DB4	99.80	100
	Aggregate	99.58	99.29

4.2 Robustness or Generalization Analysis

In order to gain further insights in understanding the effectiveness and generalization of the proposed architecture we introduce some artifacts in the original image in the form of random-noise, rotation and occlusion. All these artifacts are done on a small validation-set comprising of 100 images for each sensor corresponding to IITK single fingerprint dataset.

- **Rotation**

  In order to check the robustness of the proposed architecture the input fingerprint images have been rotated with different angles and the computed CCR% is shown in Table 3. It can be inferred from Table 3 as we increase the angle of rotation the performance of SecuGen sensor decreases drastically this can be attributed to the non uniformity of the image captured through it as clearly visible from Fig 1

Table 3: Rotation qualitative performance in CCR(%) on validation set for shallow network architecture (here 2, 4, 6, 8 and 15 represents the angle of rotation in clockwise as well as in anti-clockwise direction)

Angle	Sensor	VGG	Resnet
2	Futronic	99	100
	Lumidigm	100	100
	SecuGen	94.5	100
4	Futronic	99	100
	Lumidigm	100	100
	SecuGen	84.5	100
6	Futronic	99	100
	Lumidigm	100	100
	SecuGen	80	100
8	Futronic	99	100
	Lumidigm	100	100
	SecuGen	76	95.8

- **Occlusion**

  In order to test whether our proposed architecture is invariant to occlusion or not. We consider a patch of 90 X 90 in the input image and randomly occluded some percentage of pixels in it. The results in terms of CCR% has been shown in Table 4. Surprisingly it can be inferred from Table 4 that on increasing the occlusion in the image the performance of the network is not deteriorating. This abnormal behaviour compel us to think what exactly our network is learning.

Table 4: Occlusion qualitative performance in CCR(%) on validation set for shallow network architecture (here 1%,5% and 10% are the percentage of pixels occluded in a region of 90 X 90 patch)

Percentage	Sensor	VGG	Resnet
1	Futronic	100	99.8
	Lumidigm	99.8	100
	SecuGen	99.8	100
5	Futronic	100	100
	Lumidigm	99.6	100
	SecuGen	100	100

10	Futronic	100	99.4
	Lumidigm	99.2	99.8
	SecuGen	99.2	98.6
15	Futronic	100	99.6
	Lumidigm	98.2	100
	SecuGen	98.2	98

- **Random Noise**

  In real life scenarios it is very difficult to get a perfect condition. Most of the time we end up with noise affecting our ideal conditions. In such cases it is essential that our architecture is robust for noise upto certain range. Table 5 shows the result of adding random noise to our validation test data. It can be infered from Table 5 that on increasing the random noise the performance of the Lumidigm sensor is decaying to a large extent. As it is clearly evident from Table 1 that the Lumidigm sensor captured image is highly uniform in texture, so any small alteration or artifact in its texture greatly affects our network performance.

Table 5: Random noise qualitative performance in CCR(%) on validation set for shallow network architecture (here 0.01%, 0.05%, 0.1% are the percentage of the random noise inserted in the input image of size 224 224)

Percentage	Sensor	VGG	Resnet
0.01	Futronic	99	100
	Lumidigm	100	100
	SecuGen	99	100
0.05	Futronic	99	100
	Lumidigm	94	97.8
	SecuGen	98	100
0.1	Futronic	99	100
	Lumidigm	50	85.6
	SecuGen	98	100

4.3 Layer Specific Feature Analysis

As it is clearly evident from Table 5 that on increasing the amount of occlusion on our input images the performance of our proposed network is not degrading. This compels us to think some interesting questions: like what exactly our network is learning? Is there something wrong in its learning? Is there any kind of prestidigitation in our network? In order to answer these questions we try to visualize the feature maps learned by different layers of our proposed shallow network. For visualizing the feature-maps we took the SecuGen sensor of IIIT-K single fingerprint images. Part (a) of Fig 3 shows the features learned by our proposed shallow network. It is clearly evident from the Fig 3 , that initial convolutional layers are learning general specific features, while as we go deeper and deeper more sensor specific features, localized and sparse features are learned. It can be observed in Part (a) of Fig 3 that layer 3 and layer 6 are trying to understand the textual patterns of the image, but as we move deeper in the network like in layer 9 and layer 14 more emphasis is given to learn the shape and the background of the image rather than its textual patterns. By observing this we realize that our network is smart enough instead of learning finer granular level discriminative information to distinguish between different sensors it learned the background and shape of the image instead of its textual features.

This could be the main reason for the exceptionally high performance of the network in case of occluded images. In order to prove it we did another set of experimentation in this we crop an input image from the middle in the size of 90 X 90 patch and paste it over a white background. By doing this we have forcefully made the background of all the images same, in-order to force our network to learn textual features instead of background. It is clearly evident from Fig 3 that now our network is learning textual patterns instead of background. It is also proved by the results that we obtained after adding occlusion on this set of images

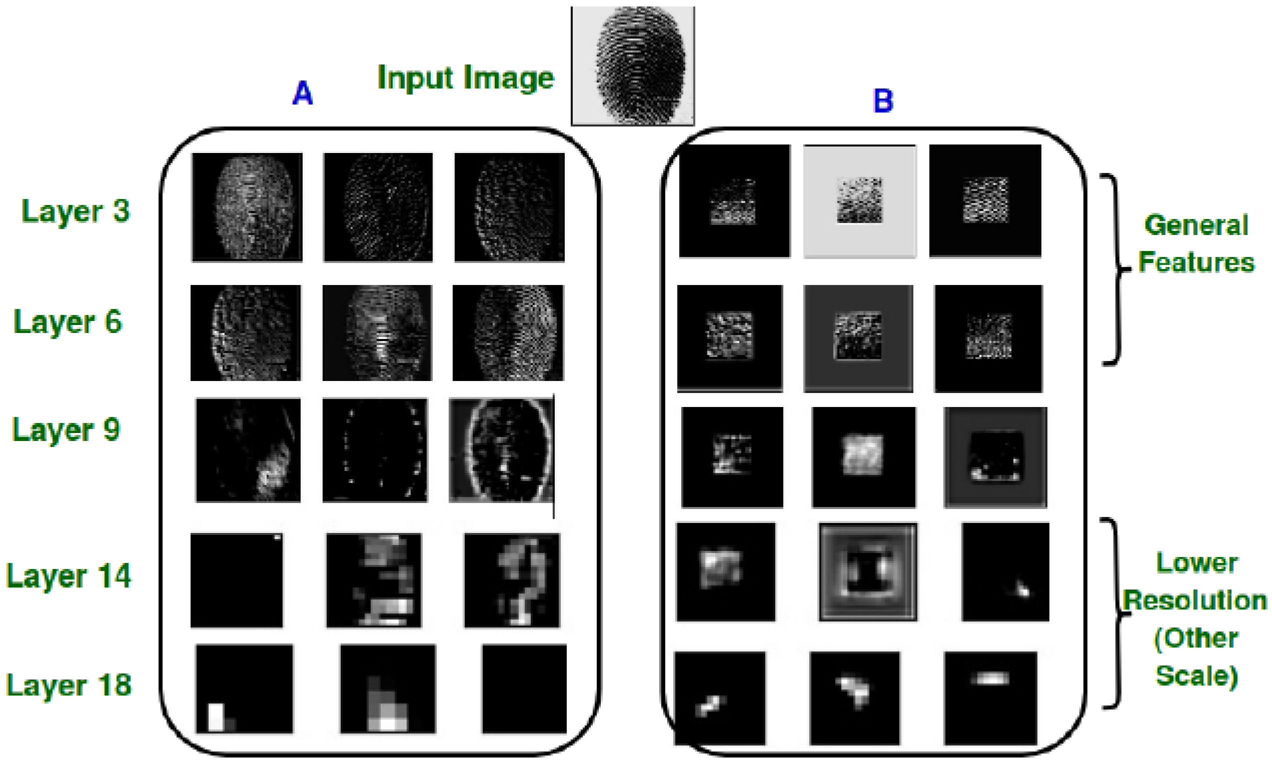


Figure 3: Comparative Feature Analysis.

5 Work Done During 8th Semester

After the start of this phase, my main aim was towards our big step Data Hashing, hence my whole work during the first month was some literature review and implementing one the papers published recently [1]. The main contributions of the proposed method are shown as follows:

- A novel asymmetric deep structure are proposed. Two streams of deep neural networks are trained to asymmetrically learn two different hash functions. The similarity between each pair images are utilized through a pairwise loss according to their semantic/label information.
- The similarity between the learned features and binary codes are also revealed through an additional asymmetric loss. Real-value features and binary codes are bridged through an inner product, which alleviates the binary limitation, better preserves the similarity, and speeds up convergence at the training phase
- By taking advantage of these two asymmetric properties, an alternative algorithm is designed to efficiently optimize the real values and discrete values.

5.1 Notations and Problem Definition

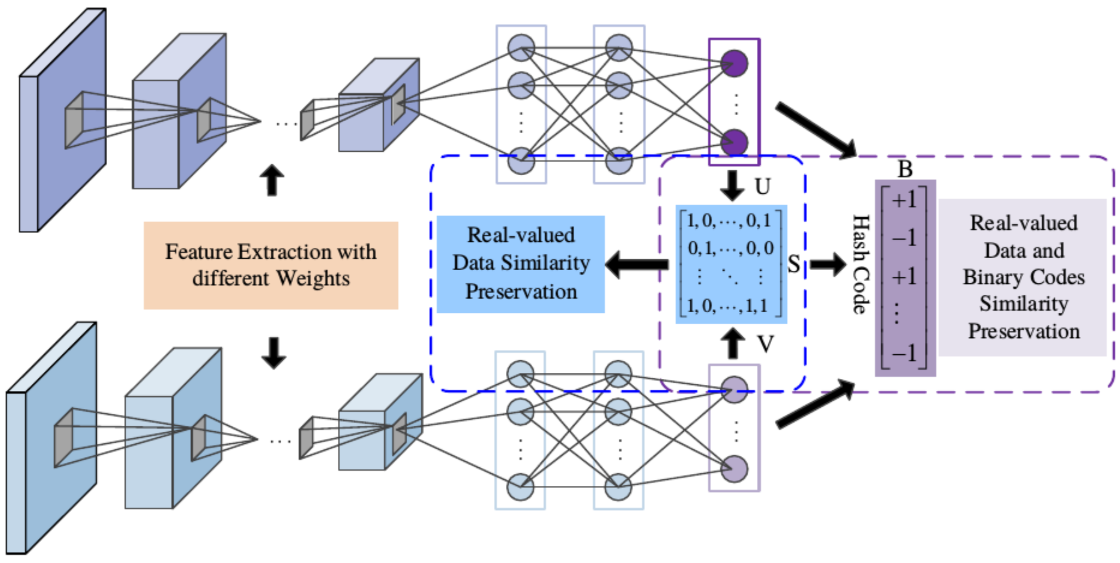
In this paper, since there are two streams in the proposed method, they have use the uppercase letters X = $\{x_1, , x_i, ..., x_N\}\epsilon R^{NXd_1Xd_2X3}$ and Y = $\{y_1, , y_i, ..., y_N\}\epsilon R^{NXd_1Xd_2X3}$ to denote the input images in the first and second deep neural networks, respectively, where N is the number of training samples, d_1 and d_2 are the length and width for each image. Note that, although X and Y are represented with different symbols, both of them denote the same training data. In our experiments, we only alternatively use training samples X and Y in the first and second networks. Since our method is supervised learning, the label information can be used. Let the uppercase letter S $\epsilon\{1, +1\}$ denote the similarity between X and Y and $S_{i,j}$ is the element in the i-th row and j-th column in S. Let $S_{i,j} = 1$ if x_i and y_j share the same semantic information or label, otherwise $S_{i,j} = 1$.

Denote the binary codes as $B = [b_1, , b_i, ..., b_N]^T \ \epsilon R^{NXk}$ and the k-bit binary code of the i-th sample as $b_i \epsilon\{1, +1\}^{kX1}$. The purpose of our model is to learn two mapping functions F and G to project X and Y into the Hamming space B. $b_i = sign(F(x_i))$ and $b_j = sign(G(y_j))$, where sign() is an element-wise sign function, and sign(x) = 1 if x 0, otherwise sign(x) = -1.

Due to the power of deep neural network in data representation, we apply the convolution neural work to learn the hash functions. Specifically, the CNN-F structure is adopted to perform feature learning. In CNN-F model, there are eight layers including five convolutional layers as well as three fully-connected layers. The network structure is listed in Table **??**, where f. means the filter, st. means the convolution stride, LRN means the Local Response Normalization. In order to get the final binary code, we replace the last layer in CNN-F with a k-D vector and the k-bit binary codes are obtained through a sign operation on the output of the last layer. In this paper, CNN-F model is applied to both streams in our proposed asymmetric structure.

Table 6: The Network Structure of CNN-F

Layer	Structure
conv1	f. 64 11 11; st. 44; pad. 0; LRN.; 2 pool
conv2	f.265 5 5; st. 11; pad. 2; LRN.; 2 pool
conv3	f. 265 3 3; st. 11; pad. 1
conv4	f.265 3 3; st. 11; pad. 1
conv5	f. 265 3 3; st. 11; pad. 1; 2 pool
full6	4096
full7	4096
full8	k-bit hash code



Feature Extraction with different Weights
Real-valued Data Similarity Preservation
U
1, 0, ···, 0, 1
0, 1, ···, 0, 0
1, 0, ···, 1, 1
S
V
B
Hash Code
+1
−1
+1
−1
Real-valued Data and Binary Codes Similarity Preservation

Figure 4: The framework of the proposed method. Two streams with five convolution layers and two full-connected layers are used for feature extraction.

5.2 Dual Asymmetric Deep Hashing Learning

The main framework of the proposed method is shown in Fig 4. As we can see, there are two end-to-end neural networks to discriminatively represent the inputs. For a pair of outputs F and G in these two streams, their semantic information is exploit through a pairwise loss according to their predefined similarity matrix. Since the purpose is to obtain hash functions through the deep networks, the binary code B is also generated by minimizing its distance between F and G. Furthermore, in order to preserve the similarity between the learned binary codes and real-value features, and alleviate the binary limitation, another asymmetric pairwise loss is introduced by using the inner product of the hash codes B and learned features $F(G)$

Denote $f(x_i, W_f)\, \epsilon R^{kX1}$ as the output of the i-th sample in the last layer of the first stream, where W_f is the parameter of the network. To simplify the notation, we use f_i to replace $f(x_i, W_f)$. Similarly, we can obtain the output $g_j$corresponding to the j-th sample under the parameter W_g in the second network. Thus, the features $F = [f_1, , f_i, f_n]^T \epsilon R^{nXk}$ and $G = [g_1, , g_i, g_n]^T \epsilon R^{nXk}$ corresponding to the first and second networks are then gained. To learn an accurate binary code, we set $sign(f_i)$ and $sign(g_i)$ to be close to their corresponding hash code b_i . A general way is to minimize the L2 loss between them.

$$min||sign(f_i) - b_i||_2^2 + ||sign(g_i) - b_i||_2^2 \tag{1}$$

However, it is difficult to make a back-propagation for the gradient with respect to f_i or g_i since their gradients are zero anywhere. In this paper, we apply tanh() to softly approximate the sign() function. Thus, equation is transformed into

$$min||tanh(f_i) - b_i||_2^2 + ||tanh(g_i) - b_i||_2^2 \tag{2}$$

Although previous equation achieves to approximate discrete codes but the similarity between the binary codes and real-value features is ignored. To tackle this problem, another asymmetric pairwise loss is introduced.

$$min||tanh({f_i}^T)b_j - kS_{ij}||_2^2 + ||tanh({g_i}^T)b_j - kS_{ij}||_2^2 \tag{3}$$

In above Eq., the similarity between the real-valued data and binary codes is measured by their inner product. It is easy to observe that above Eq. not only encourages the $tanh(f_i)(tanh(g_i))$ and b_i to be consistent, but also preserve the similarity between them. Jointly taking all the equations into account, the objective function can be obtained as follows:

$$\begin{aligned} min_{F,G,B} L = ||tanh(F)B^T - kS||_F^2 + ||tanh(G)B^T - kS||_F^2 \\ -\tau \sum_{i,j=1}^{n} (S_{ij}\theta_{ij} - log(1 + \exp^{\theta_{ij}})) \\ +\gamma(||tanh(F) - B||_F^2 + ||tanh(G) - B||_F^2) \\ +\eta(||tanh(F)^T 1||_F^2 + ||tanh(G)^T 1||_F^2) \end{aligned} \tag{4}$$

where τ , γ and η are the non-negative parameters to make a trade-off among various terms. Note that the purpose of the forth term $||tanh(F)^T 1||_F^2 + ||tanh(G)^T 1||_F^2$ in the objective. function is to maximize the information provided by each bit. In detail, this term makes a balance for each bit, which encourages the number of -1 and +1 to be approximately similar among all training samples.

6 Tentaive Plan

As we have achieved good performance for the sensor detection problem, now the part which is left in the matching layer is to train a network to which is able to match images when they are from different images. Now I had started writing the code for the deep hashing method discussed in the previous section.Finally after the completion of this my main motive would be to find a suitable solution for indexing phase (how to arrange the clusters containing similar binary vectors) and finally training the network which can generate binary vectors. After the completion of all these phases I will try to make a single network pipeline which is backpropagable which can increase the performance of our network.

References

[1] J. Li, B. Zhang, G. Lu, and D. Zhang, "Dual asymmetric deep hashing learning," *IEEE TRANSACTIONS*, 2018.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Proc. FOCS*, pp. 459–468, 2006.

[3] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian, "Super-bit locality-sensitive hashing," *in Proc. NIPS*, pp. 108–116, 2012.

[4] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," *in Proc. NIPS*, 2008.

[5] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," *in Proc. CVPR*, pp. 3361–3368, Jun. 2011.

[6] S. Ji, W. Xu, M. Yang, , and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 221–231, Jan. 2013.

[7] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[8] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, pp. 969–978, Jul. 2009.

[9] D. A. Forsyth and J. Ponce., *Computer vision: a modern approach. Prentice Hall Professional Technical Reference*. 2002.

[10] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu., "Unsupervised joint object discovery and segmentation in internet images," *In Proc. CVPR*, 2013.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *In NIPS*, 2012.