Deep Hashing via Deep Learning

MTP Report submitted to

Indian Institute of Technology Mandi

for the award of the degree

of

B. Tech

by

DEEPAK SHARMA

under the guidance of

Dr. Aditya Nigam

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

June 2018

CERTIFICATE OF APPROVAL

Certified that the thesis entitled **Deep Hashing via Deep Learning** , submitted by **Deepak Sharma**, to the Indian Institute of Technology Mandi, for the award of the degree of **B. Tech** has been accepted after examination held today.

Date :

Faculty Advisor

Mandi, 175001

CERTIFICATE

This is to certify that the thesis titled **Deep Hashing via Deep Learning**, submitted by **Deepak Sharma**, to the Indian Institute of Technology Mandi, is a record of bona fide work under my (our) supervision and is worthy of consideration for the award of the degree of **B. Tech** of the Institute.

Date :

Supervisor(s)

Mandi, 175001

DECLARATION BY THE STUDENT

This is to certify that the thesis titled **Deep Hashing via Deep Learning** , submitted by me to the Indian Institute of Technology Mandi for the award of the degree of **B. Tech** is a bona fide record of work carried out by me under the supervision of **Dr. Aditya Nigam**. The contents of this MTP, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Date :

Deepak Sharma

Mandi, 175001

Acknowledgments

I would like to mention my sincere gratitude towards Dr. Aditya Nigam, Assistant Professor, IIT Mandi for giving me the opportunity to carry out this project. I would like to express my heart full gratitude towards my guide for his invaluable advice for the successful completion of the project. I would also like to thank my Phd guide Avantika Singh for helping me in completing my project. Finally, I would take this opportunity to thank my evaluation committee for valuable feedback and everyone else who directly or indirectly helped me in completing the project.

Deepak Sharma

Abstract

Voluminous biometric projects acquire data from multiple sensors. Like in UIDAI project of India various fingerprint sensors of different technology and model are used for data acquisition. This heterogeneity within sensors give rise to: sensor inter-operability issues ,problems in regulating sequence of commands for law-enforcement. With such increase in data also increases storage overheads and time complexity as if to find a needle in a haystack. Nowadays deep learning has achieved a remarkable performance in hashing, due to the impressive learning power. In this paper, firstly I had to develop a way to hash the biometric fingerprint images, to reduce time complexity to linear or sub-linear and also storage overheads, and later on I also had to look at the problem of sensor inter-operability. For the later part, I present FPSensorNet fingerprint sensor identification using Deep-Convolutional Neural Network(D-CNN). The proposed approach is computationally efficient and yields high accuracy.

Keywords: *D-CNN, UIDAI*

Contents

Abbreviations

CNN	- Convolutional Neural Network
LSH	- Locality Sensitive Hashing
RBM	- Restricted Boltzmann Machine
PNU	- Pixel Non Uniformity
PNRU	- Photo Response Non Uniformity
SPN	- Sensor Pattern Noise Scheme
ResNet	- Residual Network
DADH	- Dual Asymmetric Deep Hashing Learning
FVC	- Fingerprint Verification Competition
CCR	- Correctness Classification Rate
D-CNN	- Deep-Convolutional Neural Network
UIDAI	- Unique Identification Authority of India

Symbols

γ - Network parameter to control loss between real value and binary codes

τ - Network parameter to control semantic loss

η - Network parameter to control biases

List of Tables

List of Figures

Chapter 1

Introduction

With the advancement of modern technology, billions of images have been uploaded or shared online. Massive biometric projects (like Aadhaar) foment the tremendous growth in sensor technology and design. Heterogeneous sensors have been employed in voluminous projects for data acquisition in order to prevent sensor monopoly. In such cases, sensor interoperability is an issue where images from different sensors are matched using various matching algorithm. When a biometric modality is captured using different sensors, the acquired image although belonging to the same modality, differs in resolution, distortion and size. With such rapid growth, how to store data these data and make a fast search plays a fundamental role in machine learning Hashing aims to generate compact codes of a given image to project it into a hamming space. These compact codes reduces the storage overhead and help us achieve a constant or linear time complexity in searching.

1.1 Motivation

To search for an image containing a object or a group of 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 one, various leaning techniques are used to learn hashing functions which are used to map our images to compact codes.

In this project, I have been planning to use a deep hashing method [1] to learn compact binary codes for 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) there should be minimum difference between the compact real-valued code and the learned binary vector, 2) there should be even distribution of binary codes, and 3) different bits should be independent of each other.

1.2 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, palm-prints 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 accurately 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 neighbor 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.

1.2.1 Challenges

To address the above problem following are the various challenges:

- Generate Binary Vectors
- Indexing
- Matching

  Currently my main focus was on how to generate binary vectors which are all independent to each other, which can store information independently on different bits evenly and finally how to address the sensor inter-interoperability issue. Different sensor changes the quality of images captured by them, hence matching two images taken from different sensors is quite a hard task. First we need to know the origin of an image (the type of sensor used in capturing the image) to apply the matching algorithm on two different/similar images.

Chapter 2

Background and Related Work

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

2.1 Scalable Image Search

With the increase in graphic data, need for efficient nearest neighbor search algorithms has arisen. Quantization base and tree-based are the two types of methods that exists today. Many hashing algorithms has been proposed these days with excellent result in the reduction of storage overheads and even in time-complexities. Quantizations based method provide high search accuracy gains while hashing based methods provide fast image retrieval. Currently many computer vision applications are using hashing based methods such as object recognition, image retrieval, 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 uses random projections obtained from Gaussian distributions to preserve the cosine similarity of samples and finally maps them to 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.

2.2 Deep Learning

Deep learning aims to build high-level features from raw data to learn hierarchical feature representations. In recent years, Deep learning use has been extended in computer vision and machine learning and has given a variety of algorithms which are used for 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.

2.3 Sensor Interoperability

Automatically recording the command of actions for forensic and law reinforcement is a less explored research field while a lot of work has been done for solving fingerprint sensor inter-operability issues. The existing sensor identification techniques can be grouped into three main categories based on : (i) hand crafted features, (ii) sensor pattern noise and (iii) colour filter.

Bayrem et al. [9] proposed a method for sensor identification based on measuring the interpolation artifacts occurred in image using color filter arrays. Lukas et al. [10] proposed a technique in which sensor is identified by measuring the pixel non uniformity (PNU) noise of each image using wavelet based denoising. Further Barlow et al. [11] used a variant of PNU technique known as photo response non-uniformity (PRNU) for fingerprint sensor identification. Agarwal et al. [12] used handcrafted features that includes features based on entropy, texture, image quality and statistics for sensor recognition. Recently, Sudipta et al. [13] identified sensors from NIR iris images. In their work they have reported that enhanced Sensor Pattern Noise Scheme (SPN), works better for detecting image sensor than maximum likelihood and phase based SPN methods. Uhl and Holler [14] have also used P

RN U , to identify NIR iris sensor from their images. 2.1, summarizes the related work done in fingerprint sensor classification.

Table 2.1: SUMMARIZED SENSOR CLASSIFICATION LITERATURE REVIEW

Author	Significant Contribution
Ross & Jain [15]	Optical versus Solid State
Bartlow [11]	Photo Response Non Uniformity
Modi [16]	False non match rate, minutia count
Jia [17]	Cross Database(Fingerpass)
Lungini [18]	Optical fingerprint sensor interoperability
Agarwal [12]	Combining handcrafted features
Debiasi [19]	Multiple PRNU enhancements

Chapter 3

Work Done

3.1 During 7th Semester

Estimating correspondences between images is one the fundamental problems in computer vision [20] with applications ranging from large-scale 3D reconstruction to image manipulation and semantics segmentation [21].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 [22], 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 interoperability 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 interoperability 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 on-line 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 3.1 shows fingerprint images captured from different types of sensors. It is evident from Fig 3.1, that image quality largely depends upon underlying sensor employed.

Fig. 3.1: 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 interoperability 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.

3.1.1 Proposed Architecture

In an image classification problem the task is to predict a label of the input image among the set of the predefined labels. Traditional methods of classification uses hand-crafted features like HOG and SIFT, but these methods encode low-level characteristics and thus not able to distinguish well in case of fine grain classification problems. In present world deep learning based methods are the state of the art methods that are capable of encoding higher level characteristics

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

3.1.1.1 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 3.2. 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.

3.1.1.2 Deep Network (Resnet50 variant)

In the second set of experimentation our network is inspired by ResNet50 architecture. We had taken this network deliberately in order to know whether the presence of identity connections between the layers in the residual network architecture gives us an advantage in learning discriminative features for finger print sensor classification. We took pretrained ResNet50 model on ImageNet images and performed extensive experimentation over it using multi-sensor fingerprint images in order to fine tune it.

3.1.1.3 ResNet Description and Parameterization

The first layer of proposed architecture is an input layer which takes an input image of size 224×224. Given the fingerprint image and its corresponding label, the first convolutional layer CONV 1, filters the input image of size 224×224 using 64 kernels and pool it to an output of size 112×112. The filters of CONV 1 layer generally detects the edges and colors of the input image. The output of the CONV 1 is connected to max-pooling layer. After that Branch2, Branch3 and Branch4 blocks of ResNet50 architecture has been utilized. At the end a fully connected layer with 2048 neuron has been added along with a dropout layer in the model with a probability of 0.4, in order to avoid over-fitting as well as to force the network to learn only robust features. Finally softmax function has been used to generate the probability distribution by minimizing the categorical cross-entropy loss-function. 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 30 epochs. All the parameters that are used in this work are calculated empirically over a small validation set.

3.1.1.4 ResNet Pruning

ResNet50 is a very deep network with 50 weight layers trained over 1000 ImageNet classes. Since we have less number of classes and that too with few thousands of images, we have decide to prune the network systematically in order to avoid over-fitting. It is also famous for its notorious training hence extensive experimentation has been done to f inetune as well

as systematically prune existing ResNet50 model. During experimentation we found that *Branch*2 and *Branch*4 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 7×7. 3.2 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.

3.2 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:

- Two Neural Networks are trained to learn asymmetrically different hash functions. A semantic label information is used in a pairwise loss function to utilize the similarity between each pair of images.
- An additional asymmetric loss is applied to reveal the similarity between the binary codes and the learned features. Binary codes and real value features are bridged through an inner product, which preserves the similarity, alleviates the binary limitation, and also speeds up convergence at the training phase.
- To take advantage of the upper asymmetric properties, real values and discrete values are efficiently optimized through an alternate algorithm.

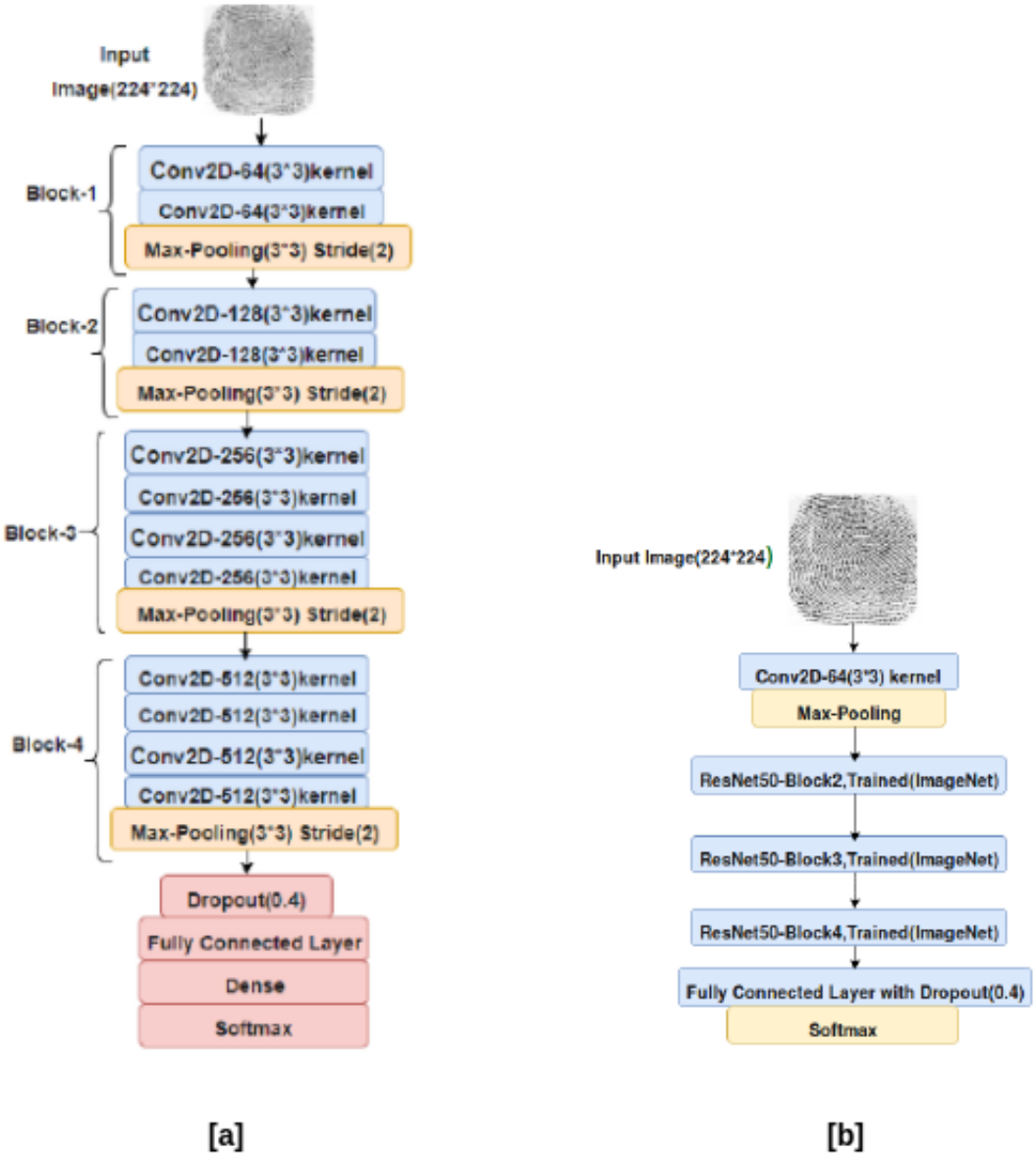

Fig. 3.2: [a] VGG-19 based shallow network, [b] Resnet50 based deep network.

3.2.1 Notations and Problem Definition

Since there are two neural networks used in this paper, X = $\{x_1, , x_i, ..., x_N\}\epsilon R^{N\times d_1\times d_2\times 3}$ and Y = $\{y_1, , y_i, ..., y_N\}\epsilon R^{N\times d_1\times d_2\times 3}$ are used 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 and both denote the same training data although represented by different symbols. Training samples X and Y are alternatively used 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 $B = [b_1, , b_i, ..., b_N]^T \ \epsilon R^{N\times k}$ and the k-bit binary code of the i-th sample as $b_i\epsilon\{1, +1\}^{k\times 1}$. 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.

They have applied the convolution neural work knowing the power of deep neural network to project the information into compact codes. Feature learning is performed using the CNN-F structure. CNN-F consists of three fully connected layers as well as five convolutional layers. The network structure is listed in Table 3.1, st. means the convolution stride, where f. means the filter, LRN means the Local Response Normalization. The last layer in the CNN-F was replaced with a k-D vector to get the final binary code and the k-bit binary codes are obtained after applying a sign operation on the output of last layer. In this proposed asymmetric structure CNN-F model is applied to both streams.

3.2.2 Dual Asymmetric Deep Hashing Learning

The main framework of the proposed method is shown in Fig 3.3. 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

Table 3.1: 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

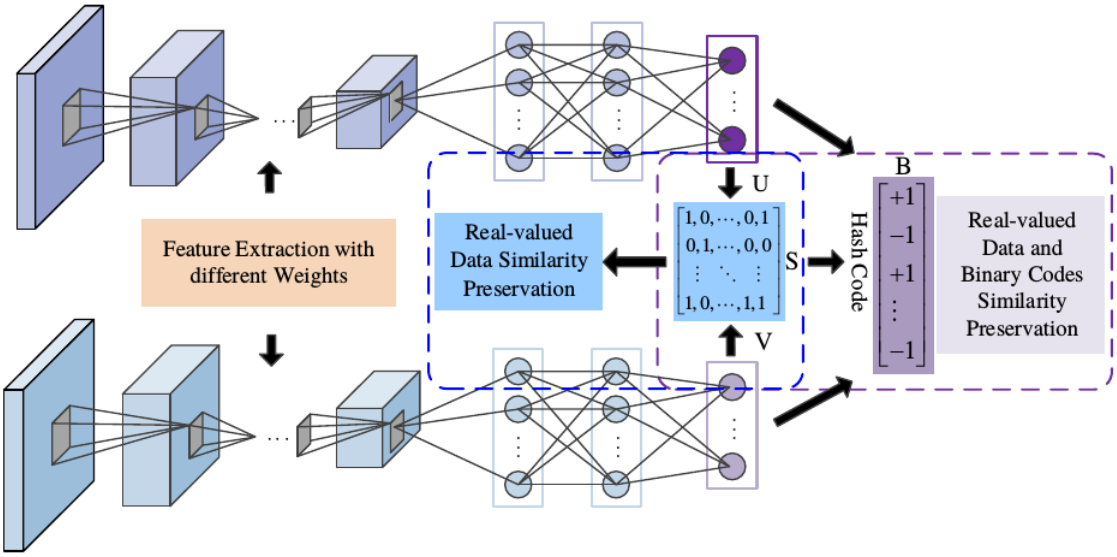

Fig. 3.3: The framework of the proposed method. Two streams with five convolution layers and two full-connected layers are used for feature extraction.

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 $f(x_i, W_f)\ \epsilon R^{k\times 1}$ 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 $F = [f_1, , f_i, f_n]^T \epsilon R^{n\times k}$ and

$G = [g_1, , g_i, g_n]^T \epsilon R^{n \times k}$ 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.

$$min||sign(f_i) - b_i||_2^2 + ||sign(g_i) - b_i||_2^2 \tag{3.1}$$

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

$$min||tanh(f_i) - b_i||_2^2 + ||tanh(g_i) - b_i||_2^2 \tag{3.2}$$

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.

$$min||tanh({f_i}^T)b_j - kS_{ij}||_2^2 + ||tanh({g_i}^T)b_j - kS_{ij}||_2^2 \tag{3.3}$$

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:

$$\begin{aligned} min_{F,G,B} L = &||tanh(F)B^T - kS||_F^2 + ||tanh(G)B^T - kS||_F^2 \\ &-\tau \sum_{i,j=1}^{n} (S_{ij}\theta_{ij} - log(1 + \exp^{\theta_{ij}})) \\ &+\gamma(||tanh(F) - B||_F^2 + ||tanh(G) - B||_F^2) \\ &+\eta(||tanh(F)^T 1||_F^2 + ||tanh(G)^T 1||_F^2) \end{aligned} \tag{3.4}$$

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.

Chapter 4

Exerimental Studies and Results

The proposed two architectures described in the previous section has been tested upon four publicly available benchmark single fingerprint databases viz., (i) FVC2002(4.2), (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

4.1 Intra-Model Classification

In this type of classification training and testing is performed on images acquired from same type of sensor model. Table 4.1, indicates the computed performance of two different proposed architectures on various datasets. The main phenomenon that we observed is that we

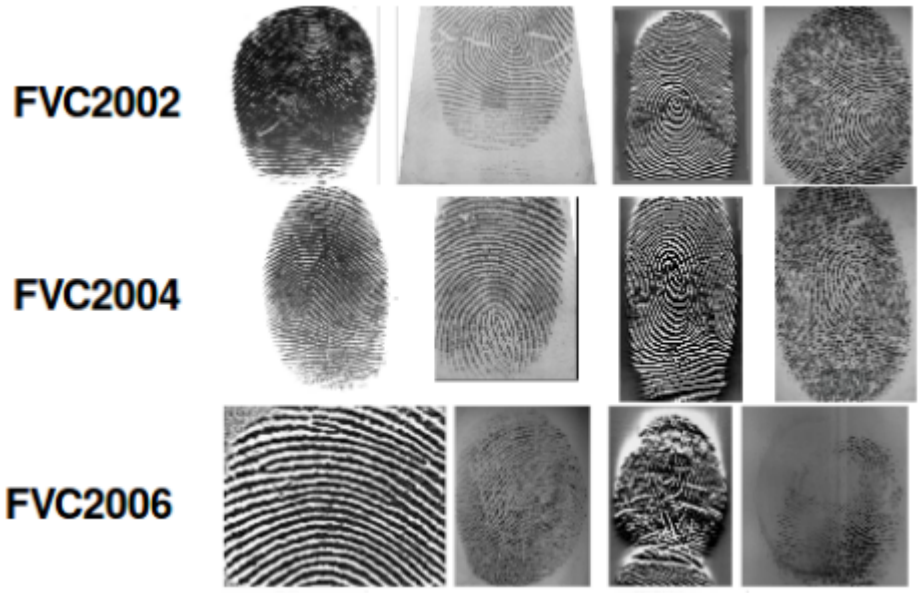

Fig. 4.1: Sample FVC2002 dataset images

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

Table 4.1: 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

4.2 Multi-Model 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 4.2 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 4.2: 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

4.3 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.

4.3.1 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 4.3. It can be inferred from Table 4.3 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 3.2

Table 4.3: 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

4.3.2 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 4.4. Surprisingly it can

be inferred from Table 4.4 that on increasing the occlusion in the image the performance of the network is not deteriorating. This abnormal behavior compel us to think what exactly our network is learning.

Table 4.4: 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

4.3.3 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 4.5 shows the result of adding random noise to our validation test data. It can be infered from Table 4.5 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 3.2 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 4.5: 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

4.4 Layer Specific Feature Analysis

As it is clearly evident from Table 4.5 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 4.2 shows the features learned by our proposed shallow network. It is clearly evident from the Fig 4.2 , 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 4.2 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 4.2 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

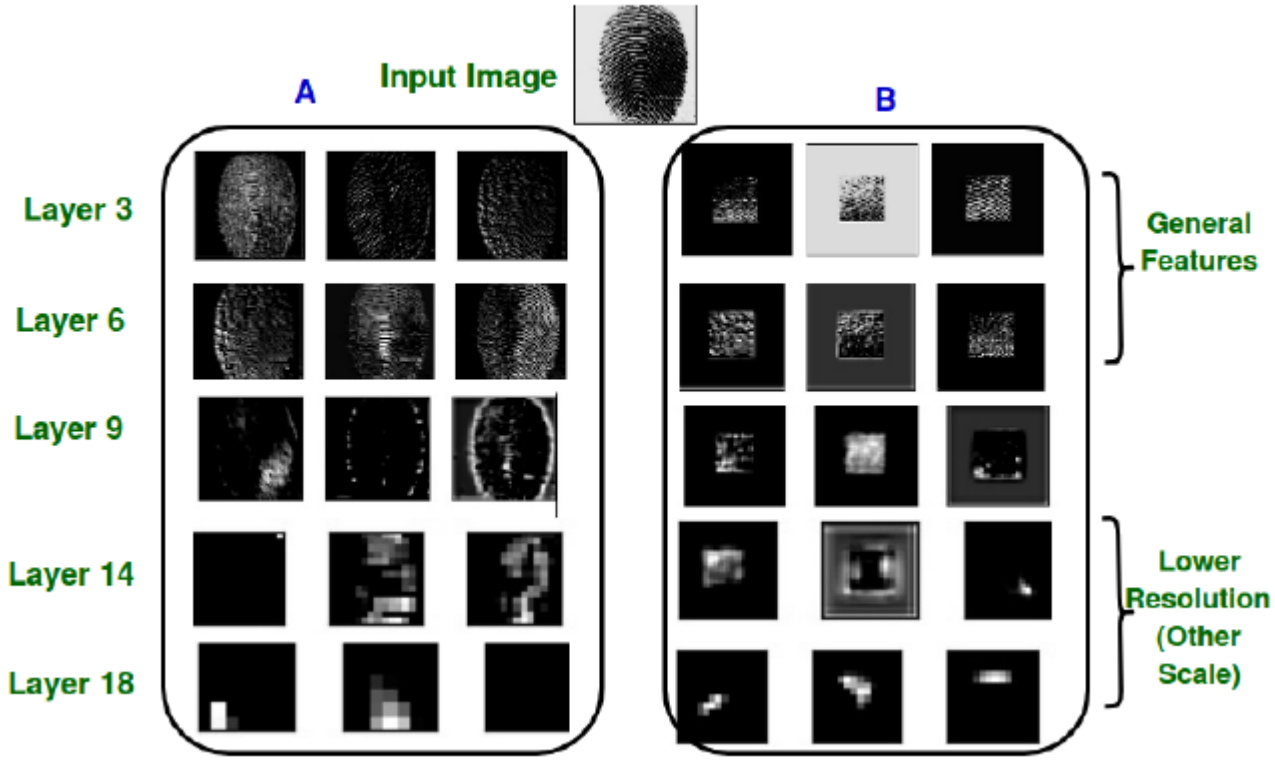

Fig. 4.2: Comparative Feature Analysis.

4.5 Training

First of all VGG was used instead of CNN-F network in the hashing network. As we don't have too much data, we used pre-trained VGG16 model trained on imagenet weights, then to fine tune the model to our fingerprint data, we used IIT-K data consisting of images of 3 different sensors with a total of about 40,000 images in the dataset. To fine tune our model on this dataset, we opted for three different ways:-

4.5.1 Auto-Encoder:

Autoencoder are used for unsupervised learning in a neural network. We try to train our VGG model using autoencoder by simply applying a decoder at the end of the VGG model. An autoencoder tries to compress the data into a short code which our VGG was doing, then decodes the short code back to the original data, thus forcing our VGG model to learn deeper feature of our dataset. The first layer in encoder might try to encode easy features like shape and qualtiy of fingerprints, the second might analyze first layer's output and then encode less local features like various patterns in our fingerprints, minutiae etc and then finally forcing our model to encode the whole fingerprint image. After training our model we did two kind of experiment to check the efficiency of our model

- We took all the positive pairs (similar pairs) in our dataset and then calculated the final output of the last layer of our hashing model (DADH) and calculated the L2 distance between them, similarly we did for the all the negative pairs and plotted both of them as shown in 4.3. But we can see that all the positive pairs and negative pairs lies closely instead of positive pairs tending towards zero and negative pairs tending away from zero. This states that our model is not properly trained with the help of autoencoder.

- Now we took different subjects and tried to plot them into tsne [23] curve to see the separability between the different subjects. We predicted the output of different subjects and their similar images in group of 10 from our hashing model as shown in 4.4

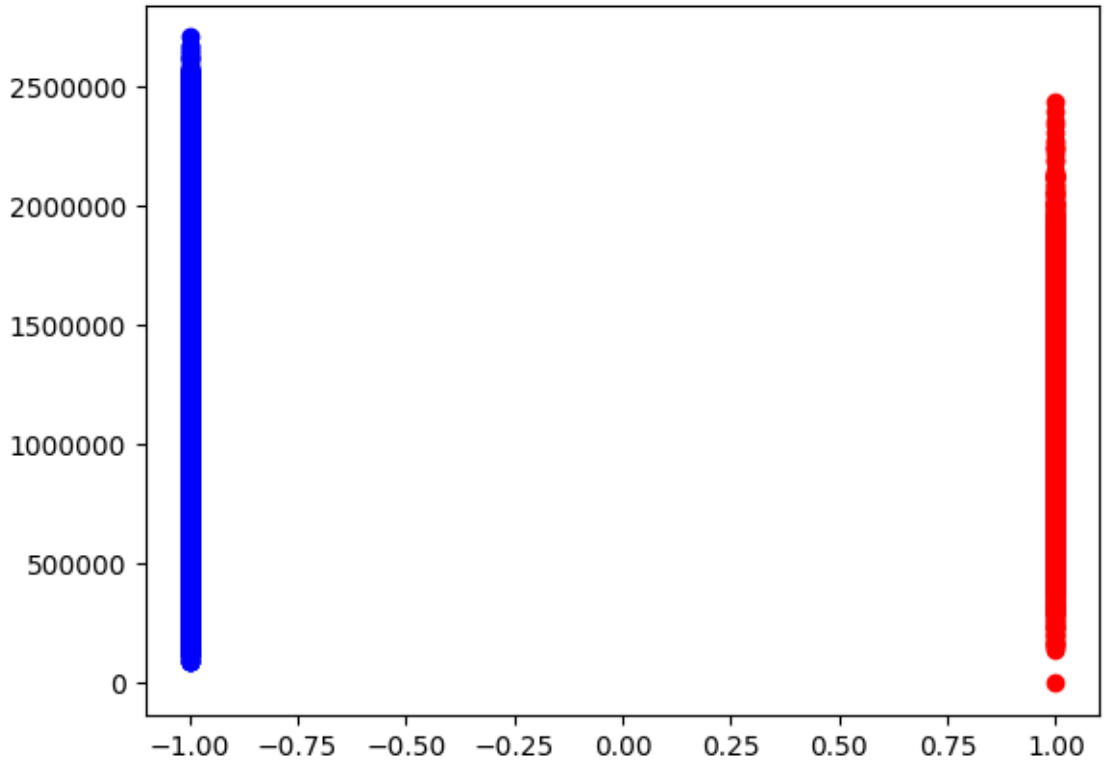

Fig. 4.3: L2 distance for all positive and negative pairs. Here X = 1 stands for positive and X = -1 stands for negative pairs. Y axis represents the value of L2 distance

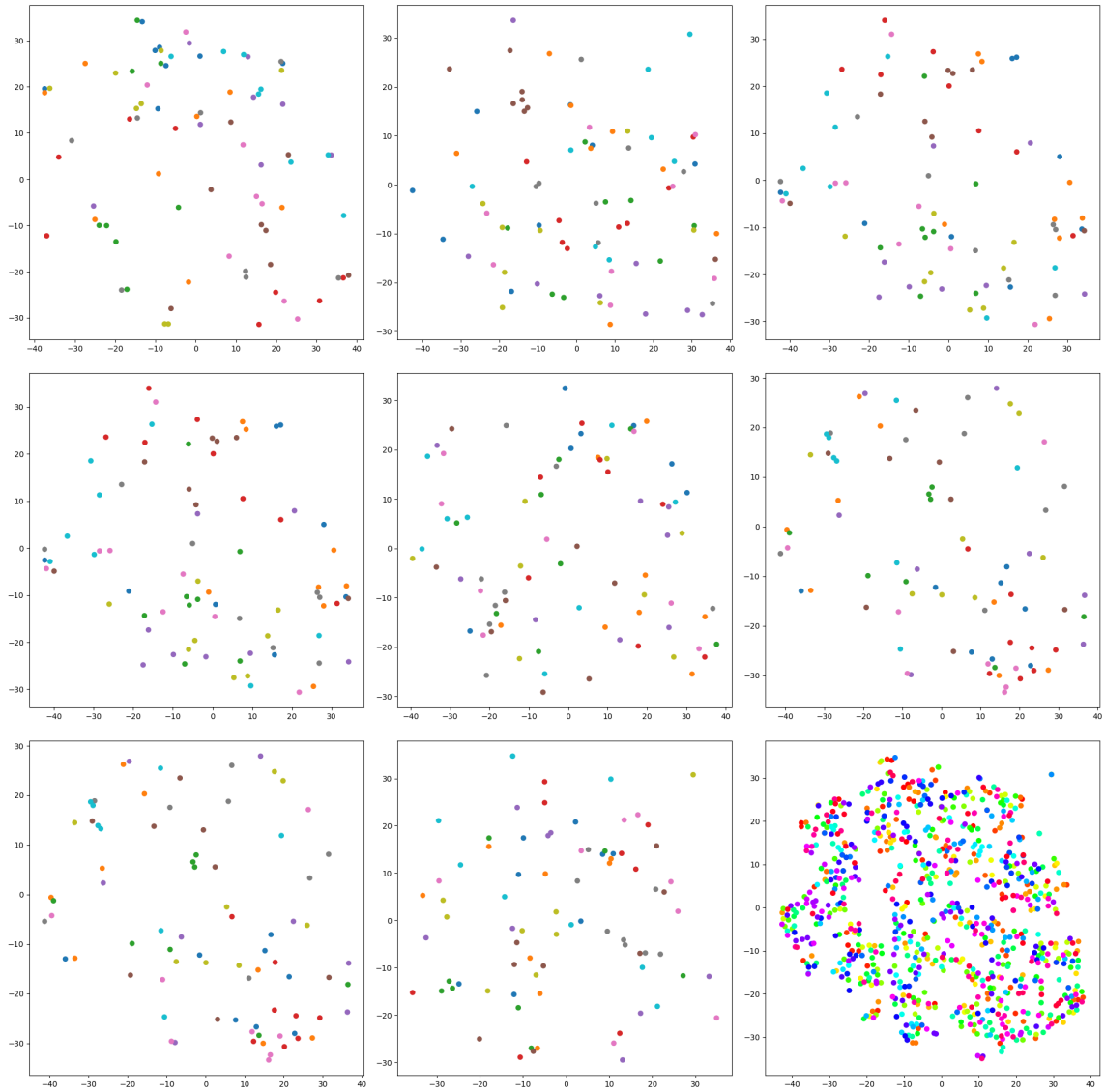

Fig. 4.4: First eight curve include graphs for 10 different subjects each, last curve shows separability of all 100 subjects.

4.5.2 VGG as classifier

Now we took this problem as a classification problem of 100 classes (FVC dataset contains 100 subjects per sensor), and we tried to plot the t-sne [23] as well as the L2 distance between all the positive and negative pairs. We trained our VGG model on all the subjects with all the similar images as the images within same class. The plot to L2 distance and t-sne [23] are shown as 4.5 and 4.6 respectively.

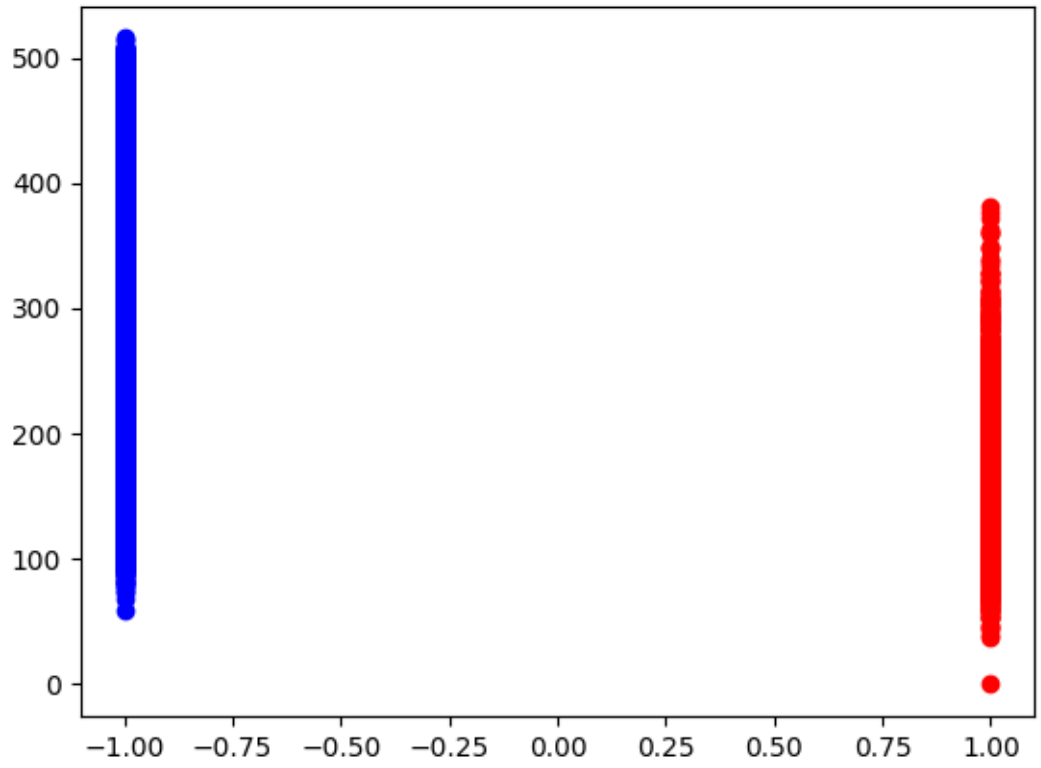

Fig. 4.5: L2 distance for all positive and negative pairs. Here X = 1 stands for positive and X = -1 stands for negative pairs. Y axis represents the value of L2 distance

4.5.3 VGG with triplet loss

Now we used the triplet loss to train our VGG model. Triplet loss contains three different images namely anchor, positive and negative. Anchor and positive are from the same subject

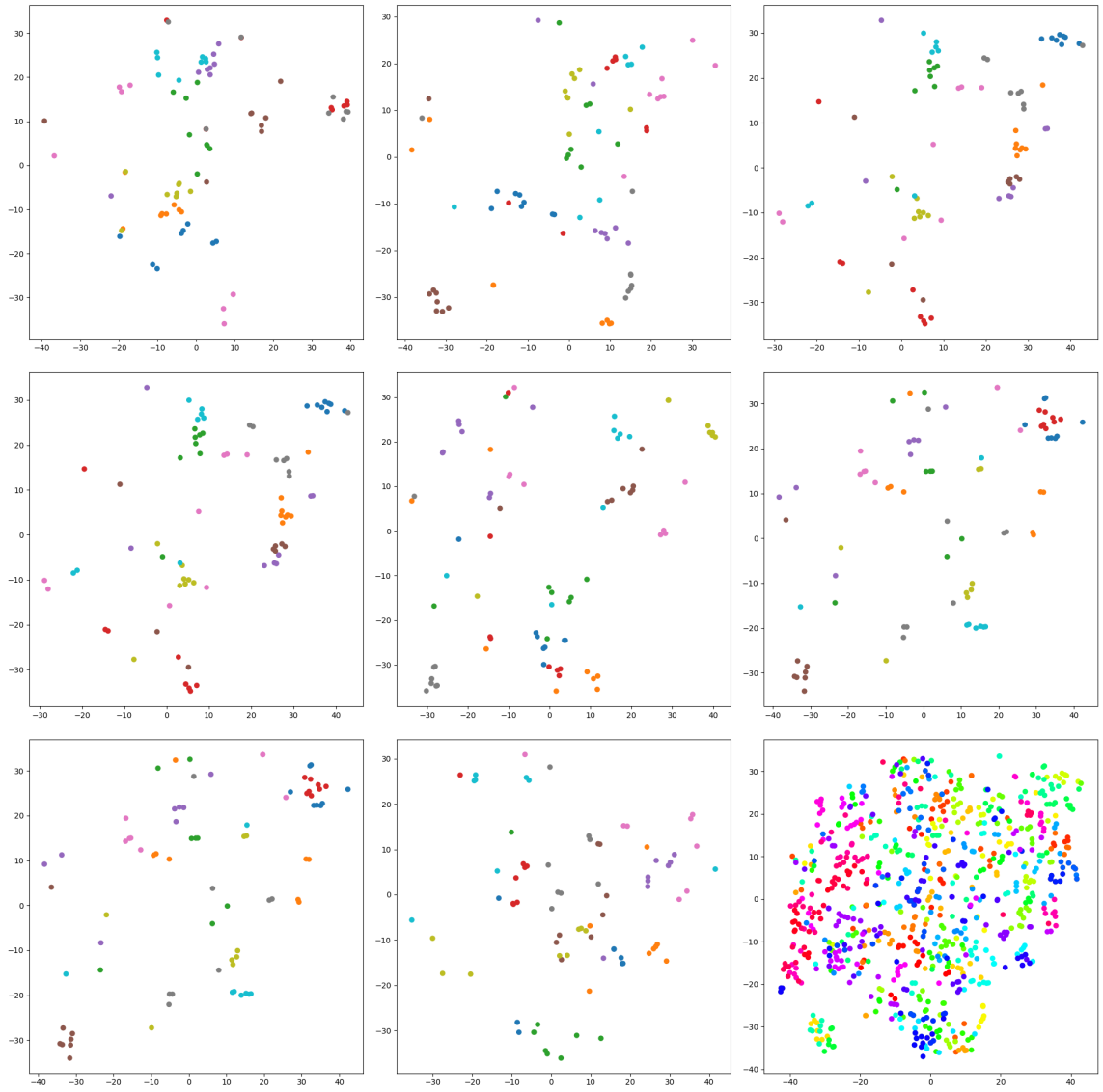

Fig. 4.6: First eight curve include graphs for 10 different subjects each, last curve shows separability of all 100 subjects.

while the negative is different from our anchor. The triplet loss can be defined as in 4.1

$$Loss = \sum_{i=1}^{N} [\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha] \tag{4.1}$$

The figures for the L2 distance and t-sne are as shown in 4.7 and 4.8 respectively

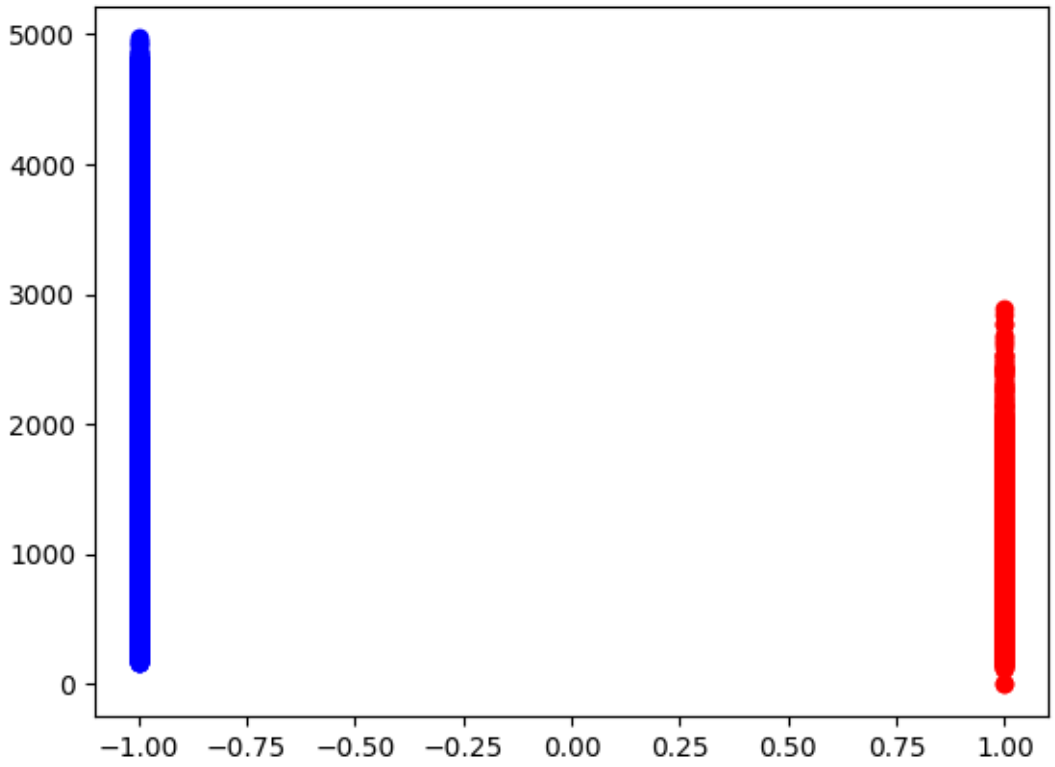

Fig. 4.7: L2 distance for all positive and negative pairs. Here X = 1 stands for positive and X = -1 stands for negative pairs. Y axis represents the value of L2 distance

Comparing all the above three methods we can say that triplet loss gives us better results as compared to others. The first two methods clearly are not capable of separating our dataset while the triplet loss has succeeded quite much (as compared to rest two).

Finally to increase the positive samples per subject we augmented our data to generate 10 different images subjecting to single image with different kinds of augmentation like Gaussian Blur, Affine, Crop, Emboss etc. Hence now our single subject contains images

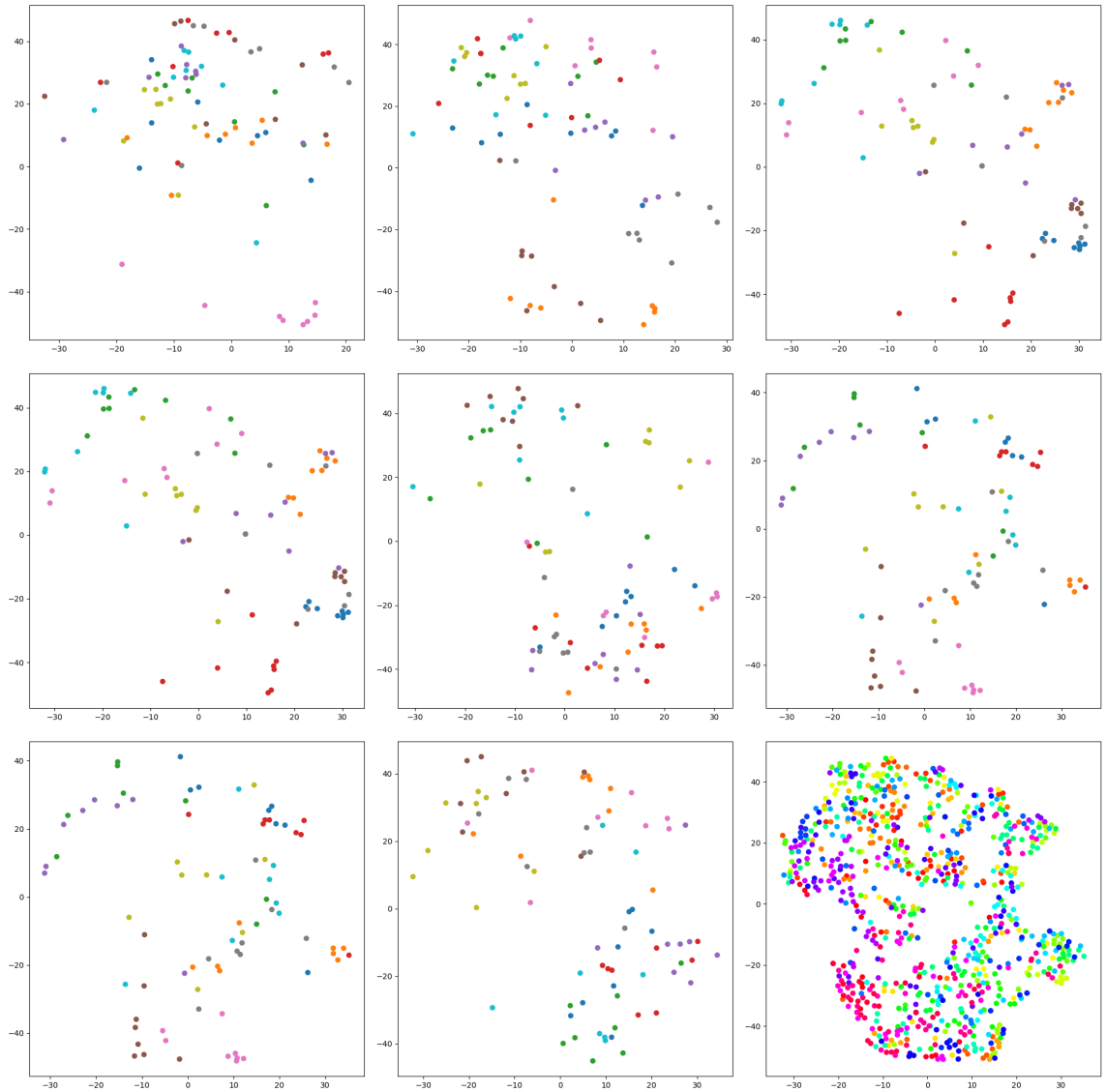

Fig. 4.8: First seven curve include graphs for 10 different subjects each, last curve shows separability of all 100 subjects.

having 10 times larger than our previous samples and to reduce our network further we used squeezeNet [24] instead of VGG16.

With the above modification, we trained our model on the FVC2002 dataset with augmentation and after 4-5 epochs we were able to generate unique hash codes of different subjects with an efficiency of 64 unique codes out of 100 subjects. We experimented keeping our anchor fixed for all the subjects in all the epochs. We were only changing the positive and negative samples of different subjects. Hence it is not guaranteed yet that the hamming distance of the generated hash codes of similar images will tend to zero, and will go further away from zero in case of negative samples.

Chapter 5

Conclusion and Future Work

The pipeline of this network starts from Geometric Alignment. Finger print images are not properly aligned, it depends on the how the person has placed his/her finger on the sensor, also there are various transformations which gets introduced into them which needs to be taken care before feeding those images to our main network. It can be done manually or a network [25] can be added at the starting of our main network that aligns the images, also transforms them if required and then those images are passed to the network. Since our images are properly aligned there is still a chance that our network will not learned features of different subjects. To handle such situations hard mining is used to train our network, in which some similar as well as some most similar samples are used to train our network. Also at the end a proper indexing algorithm is required which can allow us to easily search for our target images in an efficient manner.

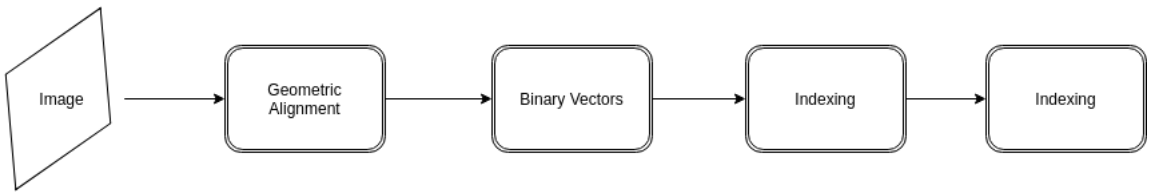

Fig. 5.1: General Pipeline

A general pipeline of our network can be shown in 5.1

References

[1] J. Li, B. Zhang, G. Lu, and D. Zhang, "Dual asymmetric deep hashing learning," *CoRR*, vol. abs/1801.08360, 2018.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 459–468.

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

[4] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, 2009, pp. 1753–1760.

[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," *CVPR 2011*, pp. 3361–3368, 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, no. 5786, pp. 504–507, 2006. [Online]. Available: http://science.sciencemag.org/content/313/5786/504

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

[9] S. Bayram, H. T. Sencar, N. D. Memon, and I. Avcibas, "Source camera identification based on cfa interpolation," *IEEE International Conference on Image Processing 2005*, vol. 3, pp. III–69, 2005.

[10] J. Lukás, J. J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 205–214, 2006.

[11] N. Bartlow, N. D. Kalka, B. Cukic, and A. Ross, "Identifying sensors from fingerprint images," *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 78–84, 2009.

[12] A. Agarwal, R. Singh, and M. Vatsa, "Fingerprint sensor classification via mélange of handcrafted features," *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 3001–3006, 2016.

[13] S. Banerjee and A. Ross, "From image to sensor: Comparative evaluation of multiple prnu estimation schemes for identifying sensors from nir iris images," in *Proceedings - 2017 5th International Workshop on Biometrics and Forensics, IWBF 2017*. Institute of Electrical and Electronics Engineers Inc., 5 2017.

[14] A. Uhl and Y. Höller, "Iris-sensor authentication using camera prnu fingerprints," *2012 5th IAPR International Conference on Biometrics (ICB)*, pp. 230–237, 2012.

[15] A. Ross and A. K. Jain, "Biometric sensor interoperability: A case study in fingerprints," in *ECCV Workshop BioAW*, 2004.

[16] S. K. Modi, S. J. Elliott, and H. Kim, "Statistical analysis of fingerprint sensor interoperability performance," in *Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on*. IEEE, 2009, pp. 1–6.

[17] X. Jia, X. Yang, Y. Zang, N. Zhang, and J. Tian, "A cross-device matching fingerprint database from multi-type sensors," *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3001–3004, 2012.

[18] L. Lugini, E. Marasco, B. Cukic, and I. Gashi, "Interoperability in fingerprint recognition: A large-scale empirical study," *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 1–6, 2013.

[19] L. Debiasi and A. Uhl, "Blind biometric source sensor recognition using advanced prnu fingerprints," *2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 779–783, 2015.

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

[21] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1939–1946, 2013.

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

[23] L. van der Maaten, G. E. Hinton, and Y. Bengio, "Visualizing data using t-sne," 2008.

[24] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size," *CoRR*, vol. abs/1602.07360, 2016.

[25] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, 2017.