GENERATING MASTERPRINTS

MAJOR TECHNICAL PROJECT (DP 401P)

to be submitted by

AAYUSH MISHRA

for the partial fulfilment of the degree

of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING

under the supervision of

DR. ADITYA NIGAM

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

KAMAND-175005, INDIA

MAY, 2019

Dedicated

To

Science

DECLARATION

I hereby declare that the entire work embodied in this thesis is the result of investigations carried out by me in the School of Computing and Electrical Engineering, Indian Institute of Technology Mandi, under the supervision of Dr. Aditya Nigam, and that it has not been submitted elsewhere for any degree or diploma. In keeping with the general practice, due acknowledgments have been made wherever the work described is based on the finding of other investigators.

Kamand 175005

Date: 28-05-2019

Signature:

(Aayush Mishra)

THESIS CERTIFICATE

This is to certify that the thesis titled **"Generating Masterprints"** submitted by **Aayush Mishra** to the Indian Institute of Technology Mandi, towards partial fulfillment for award of the degree of **Bachelor of Technology in Computer Science and Engineering**, is a bonafide record of the research work done by him under my supervision at the School of Computing and Electrical Engineering, Indian Institute of Technology Mandi. The contents of this thesis have not been submitted elsewhere for any Degree or Diploma.

Signature:
Name of the guide: Dr. Aditya Nigam
Date: 28-05-2019

Dr. Aditya Nigam
(Supervisor)

THESIS CERTIFICATE

This is certified that the thesis entitled **"Generating Masterprints"**, submitted by **Aayush Mishra** towards partial fulfillment for **Bachelor of Technology in Computer Science and Engineering** is based on the investigation carried out under the our supervision, and the contents of this thesis have not been submitted elsewhere for any Degree or Diploma.

(Thesis Supervisor)	Faculty Advisor (B.Tech. (CSE), 2015-2019)
School of Computing and Electrical Engineering, Indian Institute of Technology Mandi	School of Computing and Electrical Engineering, Indian Institute of Technology Mandi
(Examiner I)	(Examiner II)
School of Computing and Electrical Engineering Indian Institute of Technology Mandi	School of Computing and Electrical Engineering, Indian Institute of Technology Mandi

Acknowledgements

I would first like to thank my mentor and thesis advisor, Dr. Aditya Nigam of the School of Computing and Electrical Engineering at Indian Institute of Technology Mandi. The project idea was initiated by Prof. Nigam in the first semester of MTP and he always gave me direction when nothing seemed to be working. He urged me to carry on working on a hard problem for a long time before giving it up after exhausting all possibilities showing me how research works.

I would also like to thank my batchmate, Parinaya Chaturvedi who is a fellow undergraduate student for the degree of Bachelor of Technology in Computer Sciene and Engineering. He was involved in a research project that proved to be crucial in my work as well, without it, I would not have been able to find the results that I was ultimately able to find in my work.

At last, I would like to thank my parents for always being supportive and motivating.

Aayush Mishra

ABSTRACT

With the advent of powerful smartphones, use of biometric authentication systems has become increasingly popular. Ease of usage is their ultimate selling point, but they are not as secure as conventional security methods like passwords. There are vulnerabilities in these systems making them susceptible to presentation attacks. In this research, we explore these vulnerabilities in Fingerprint and Iris recognition systems by generating masterprints: biometric signatures capable of matching with several subjects, and compare our test results with prior art. We show that Iris is a better biometric as it does not have overlapping features between classes and is capable of handling such attacks. We also show how adversarial machine learning can be used to make these systems resistant to such attacks.

Keywords: *Fingerprint recognition, Iris recognition, Generative adversarial networks, Evolutionary computation, Optimization*

Contents

List of Figures

Abbreviations

FMR	- False Match Rate
FRR	- False Reject Rate
GAN	- Generative Adversarial Network
G	- Generator
D	- Discriminator
SAGAN	- Self-Attention Generative Adversarial Network
CMA-ES	- Covariance Matrix Adaptation - Evolution Strategy
CRR	- Correct Recognition Rate
EER	- Equal Error Rate
NIST	- National Institute of Standards and Technology
NBIS	- NIST Biometric Image Software

Chapter 1

Introduction

1.1 Motivation

Smartphones have been getting smarter every year, with hardware and software updates. They have become one of the most frequently used electronic devices during the day. They are now equipped with all computing abilities that traditionally only big desktops and laptops had. So naturally, people now store and access all kinds of personal data through their smartphones. Recently fingerprint sensors have become the gold standard for all smartphones to unlock the device. They are straightforward and intuitive to use. Some popular flagship smartphones have Iris scanners built into them as well. These biometric authentication systems have become the most popularly used way to unlock the device.

However, these biometric systems are not as secure as conventional security methods like long passwords. Also, typically, smartphone fingerprint sensors are small, so the verification has to be done for a partial print rather than full, which is less secure (Figure 1.1). Add increased false match rate (FMR) for better user experience, and the systems become less and less secure. Similarly, for the case of Iris, FMR needs to be increased, and even obscure images with lower confidence scores have to be matched if the user has to be given a pleasant experience. This makes these systems particularly susceptible to attacks that other conventional definitive methods like cryptography are not prone to (as of now). Presentation attacks on biometric security systems are, therefore, of particular interest in research. It is essential to study vulnerabilities in these systems and come up with improvements. How-

ever, to fix vulnerabilities in a security system, it is first required to attack it and see where does the vulnerability lie. Then those weak areas of the system can be improved to handle these attacks.

Fig. 1.1: Partial fingerprints. Image from [1].

One of the ways to perform presentation attacks on such systems is by using a *masterprint*. Like a master key can open many locks, a masterprint is a biometric template that can assume the identity of many subjects on a particular authentication system. But not all types of locks can be beaten by a, and not all types of keys can be used to make a master key. The advantage may lie either in the design of the locks or the keys. The analogy here is that locks correspond to the biometric authentication systems and keys correspond to the biometric in question. Some biometric traits may be better than others at being distinctive, making them a better choice for security. Similarly, some authentication systems may be better than others in terms of what features they use for matching samples and how overlapping those features are (giving rise to masterprints). Therefore, the hypothesis of the existence of masterprints for a particular biometric trait and matching algorithm needs to be tested not assumed.

In the next section, we take a look at the research that has been done so far in this area, the experiments and findings of those works and the success of these attempts. Note that we only look at presentation attacks attempted on these systems and skip other methods like jailbreaking, etc. that may have been used to break the security of such systems. We divide our work into two divisions corresponding to the two biometrics that we worked on viz.

Fingerprint and Iris. Each section having distinct information regarding the two is divided from now on.

1.2 Related Work

1.2.1 Fingerprint

Roy *et al.* [1] have shown that masterprints exist for fingerprints. They generated masterprints using various evolutionary optimization techniques [2]. They worked with minutiae templates of the fingerprints rather than the fingerprint itself and partial fingerprints, in addition to full. They sampled a few real fingerprints from the database which matched with other subjects as well, and starting from the minutiae templates of those, they generated new and better artificial templates using optimizations, which could match many more subjects. They used:

- **Hill Climbing** [1] - Generating neighbors of a particular template using random orientation change, translation, deletion and addition of minutiae and then selecting those which match with more subjects.
- **CMA-ES [3], Differential Evolution, Particle Swarm Optimization** [2] - Assuming an underlying distribution (Gaussian), sampling from it and updating its parameters according to fitness scores (~number of matches) of those samples. CMA-ES (Covariance Matrix Adaptation Evolution Strategy) tends to work the best among these.

The generated masterprints were good enough to authenticate 84% users of FVC2002 dataset at 0.1% FMR with 5 attempts using the commercial VeriFinger software for authentication (Figure 1.2). However, these masterprints based on minutiae templates cannot be used for direct spoofing as presentation attacking means that the masterprint should disguise as a real image and can not be used in any other form. A reconstruction [4] method is required to make fingerprints from these templates, which is lossy.

To overcome this problem, Bontrager *et al.* [5] used Generative Adversarial Networks [6] to make masterprints directly. They used Wasserstein GAN [7], in particular, because it has

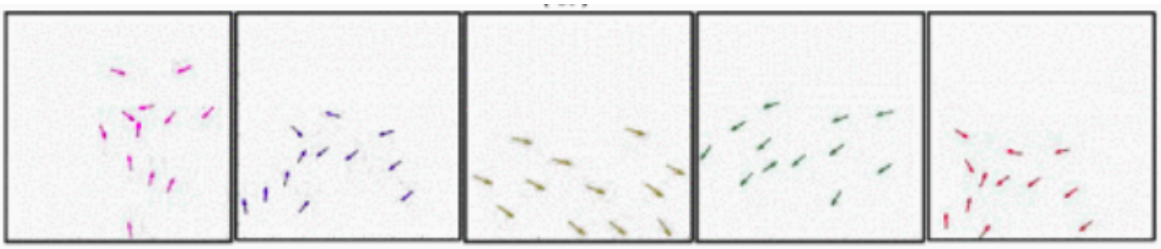


Fig. 1.2: Minutiae templates of masterprints from [2].

a smoother loss function which tends to make it converge better[1]. This was used to generate fingerprints (DeepMasterPrint) and then they evolved the latent variable that is fed to the GAN using CMA-ES and were able to match 22.5% subjects at 0.1% FMR with a single attempt on FingerPass DB7 dataset, which is way better than 6.6% on the same dataset by Roy et. al. Refer to Figure 1.3 for the method. Note that these performances are on the VeriFinger authentication system. On Bozorth3, the performance of DeepMasterPrint was poorer, and also on the NIST optical dataset of rolled images, the performance was not as good as it was on the capacitive dataset.

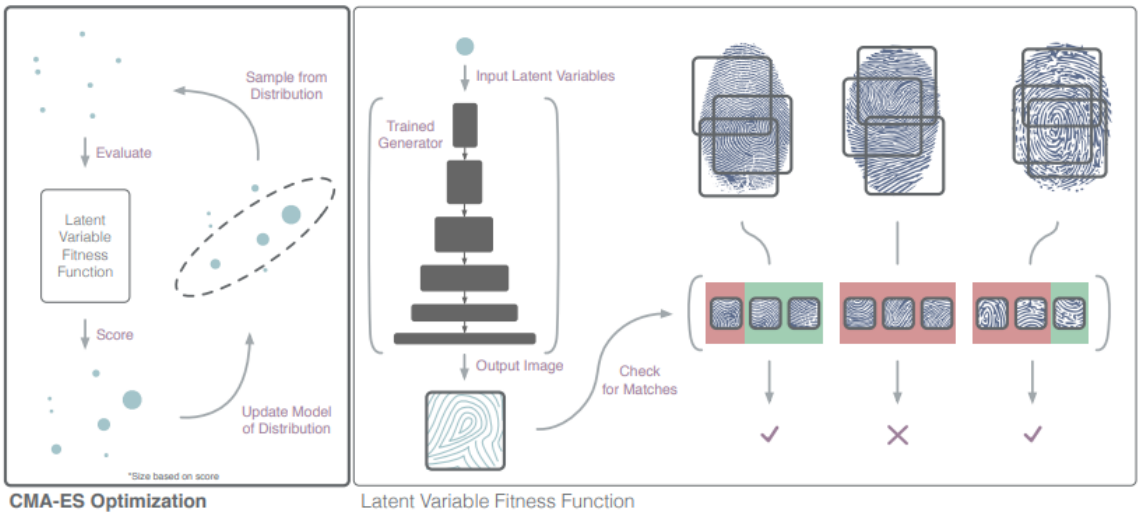


Fig. 1.3: The process of generating and selecting master prints via DeepMasterPrint [5].

1.2.2 Iris

Iris has been regarded as the ideal part of the human body for biometric identification as,

[1]The working of GANs is explained in detail in Chapter 2, Section 2.2.

- It is well protected against damage or wear, unlike fingerprints.
- Even genetically identical individuals have completely different Iris structure.
- Commonly deployed Iris Recognition algorithm (Daugman's IrisCode [8]) has unparalleled False Match Rates (10^{-11}).

Not much work has been done, however, in generating synthetic Iris images for presentation attacks. Likely reasons are:

- The complex features of the Iris. Unlike fingerprints, Iris doesn't have easily identifiable features like minutiae, and hence cannot be differentiated by an expert human's naked eye.
- All algorithms are based on mathematical transforms (Gabor Wavelet Transform in Daugman's algorithms) or machine learning.
- Tricky dataset segmentation and normalization for matching.

Kohli, *et al.* [9], have recently developed Iris images using their novel iDCGAN model (Figure 1.4), using a mixture of datasets and performed presentation attacks on VeriEye, a commercial Iris Recognition software. They generated new Iris images by removing the first quartile of bad images (low score on their quality assessment metrics) and using the rest of good images in the training of both generator and discriminator networks. They saw that by adjusting the false accept and reject rates, they were able to deviate the behavior of the verification system in favor of the presentation samples. However, they did not find masterprints in this domain. They showed that these presentation samples could be used in at least a Denial of Service attack on such systems if it is not resistant to such network threats.

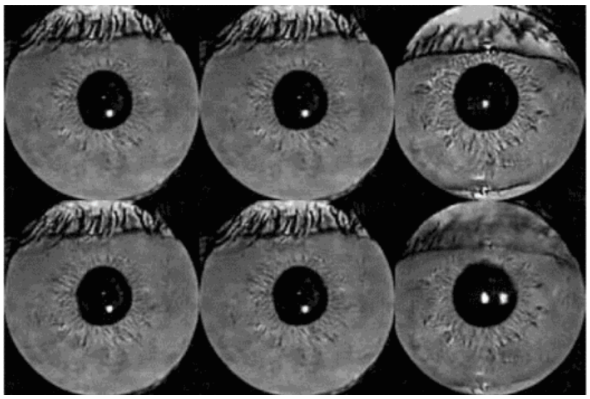


Fig. 1.4: Iris images generated from [9].

1.3 Objectives

1.3.1 Fingerprint

A lot of research has been done in the area of masterprint generation in fingerprints. There are a few natural extension to that work that we propose:

- Improving the quality of images generated using GANs by incorporating domain knowledge in their training to see if that results in better masterprints.
- Suggesting improvements in existing authentication systems to make them resistant to attacks from masterprints.
- Generalize these methods for other biometric authentication systems.

1.3.2 Iris

In Section 1.2.2, we saw that masterprint generation had not been explored so far for Iris. Given that it is the better biometric to use, we propose to:

- Generate fake Iris images using GANs.
- Test the masterprint existence hypothesis for Iris, on a particular matching system.
- Find masterprints, if they exist.
- Suggest improvements in the existing systems if masterprints exist.

1.4 Thesis Organization

This work is organized in 5 chapters.

- Chapter 1 motivates the problem, looks at the prior work related to it and states the objectives of this research.
- Chapter 2 provides details of the preliminary techniques used in our methodology.
- Chapter 3 discusses our proposed approach and methodology.
- Chapter 4 gives a detailed analysis of the experiments done and results found.
- Chapter 5 concludes our work and discusses the future scope of our research.
- References are given in the bibliography.

Chapter 2

Preliminaries

2.1 Introduction

In research, the strategies to solve a particular problem often fail. The flow of research is to come up with a hypothesis, design experiments to test it and find and document results for future reference. Documenting failed cases is as important as documenting the successful ones as the understanding of why an approach failed, might provide insights into the same or other problems in the future. To give up or change a strategy, it is essential to evaluate the current strategy exhaustively. For our proposed approach to the problem, several preliminary techniques need to be covered before diving into the actual solution model. Complete understanding of the techniques used in that strategy is crucial for analyzing and improving the results that our solution model generates. The following sections cover those techniques in detail, one by one.

2.2 Generative Adversarial Networks

Immediately after their introduction in 2014 by Goodfellow *et al.* [6], GANs have become one of the hottest areas of research in deep learning. They have been used in a variety of applications like generating unexisting images, music, etc., image editing and impainting, style transfer, super-resolution, etc. [10, 11, 12, 13, 14, 15] As GANs have been used in making fake Fingerprint and Iris images, it is important to know how GANs work.

Most generative models assume that the data from the real world follows a probability distribution which may be very complex. It can be considered impossible to find that distribution analytically because of an unknowable number of variables involved in the real world. It may be possible, however, to estimate that distribution by various modeling techniques. GANs model this assumed distribution using neural networks. In this section, we will limit our discussion to GANs used to generate images using Convolutional Neural Networks (CNNs).

A GAN comprises of two networks called the *Generator (G)* and the *Discriminator (D)*. This framework is similar to a zero-sum two-player game. G is trained (Figure 2.1) to generate images by picking random noise vectors ($\mathbf{z}$) from a latent space and giving an image as output corresponding to that vector. It is assumed that the real world images have a certain number of dependent variables (the dimensionality of latent space) and the probability distribution of various features of these images on the said variables is modeled using G. It typically uses de-convolution or up-sampling layers to do this. D, on the other hand, is trained to distinguish between real and fake samples. It estimates the probability of a given sample being real (coming from the training (real) data) or fake (coming from G). The two networks compete while training and finish in a Nash equilibrium with G assuming the real distribution and D equalling $\frac{1}{2}$ everywhere. The proof of convergence of the modeled distribution ($p_{\mathbf{z}}$) to the real data distribution (p_{data}) using appropriate training steps, can be found in the original paper. The framework plays the following two-player minimax game with the loss function $L(G, D)$ when trained using backpropagation,

$$\min_G \max_D L(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.1}$$

We used the GAN framework to estimate the real distribution of Fingerprint and Iris images from our datasets. They proved to be very effective. However, the training of GANs is particularly very sensitive of hyperparameters and initializations. In most cases the loss function is far from convex, and a single step in the wrong direction makes the training highly unstable which may never recover. It is very important to continuously watch over the training and save the progress. The experiments with these will be discussed in detail in

chapter 4.

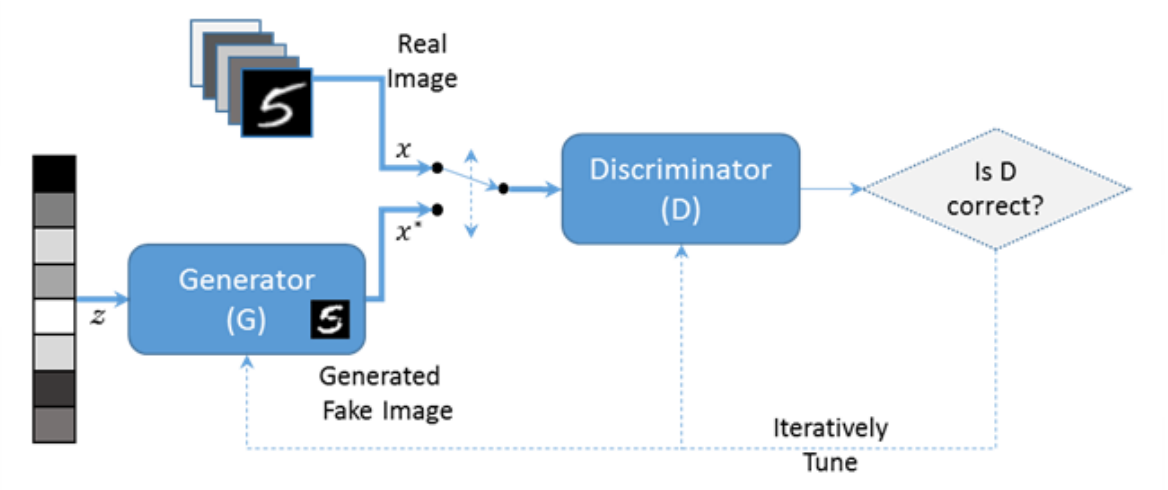


Fig. 2.1: Basic design and Training of a GAN. Image from `https://cntk.ai/pythondocs/CNTK_206B_DCGAN.html`.

2.3 Self-Attention GAN

Humans use attention extensively. Taking the example of our sense of vision, whenever we see something, our visual attention is focused on a particular area of our field of view. The rest of it generates negligible visual stimulus. Similarly, in neural networks, attention is a mechanism used to focus the network's attention on particular locations of the input to get better results by defocusing from the background that may be useless to the objective. This mechanism was initially used in the task of Neural Machine Translation and was then extended in other fields like image classification. Self-Attention is the mechanism of relating different parts of the input with itself. It is used to find a representation of the input which can correlate one part of the input with another modeling short as well as long-range dependencies. This is an important tool used in machine reading, caption generation, etc.

Recently, Zhang *et al.* [16] introduced the concept of Self-Attention in GANs. As image generation is typically done using CNNs, it is essential to see the drawbacks of CNNs. Convolution is a local operation. Therefore convolutional layers can only help provide local context. It can not model long-range dependencies in the input. For example, previous GANs have been able to generate good quality textures of a dog but not its legs, which

are long-range dependencies in an image. Increasing the filter size or the network depth makes the computations expensive and also the network inefficient. Therefore the use of Self-Attention layer in GANs tends to make them work better by modeling long-range dependencies (Figure 2.2)

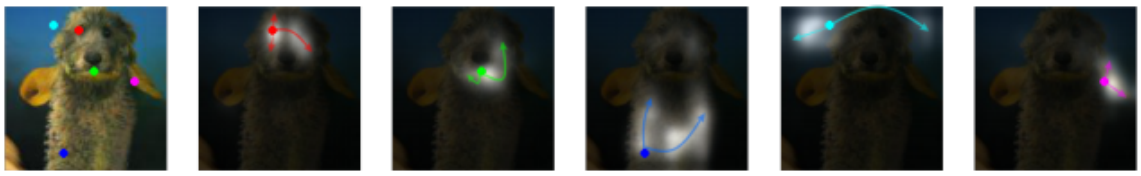


Fig. 2.2: Color coded query locations with their attention maps. Image from [16]

The Self-Attention layer is engineered as shown in Figure 2.3. Three functions namely, f, g and h are used in this operation. Using 1×1 convolution filter on the input feature maps ($\mathbf{x}$), f and g transform the input into a space suitable for making the key-value pairs ultimately generate the attention map. h uses similar filters to transform the input. The output generated ($\mathbf{o}$) by applying the attention map on h, best encorporates the intended features. The equations governing these operations are,

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})} \tag{2.2}$$

where, $s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T \boldsymbol{g}(\boldsymbol{x_j})$ and,

$$\boldsymbol{o_j} = \sum_{i=1}^{N} \beta_{j,i} \boldsymbol{h}(\boldsymbol{x_i}) \tag{2.3}$$

2.4 AutoEncoders

Autoencoders are powerful tools in the field of representation learning. They consist of an *encoder* and *decoder*. Encoder brings the input down to a lower dimensional space. The decoder then uses that embedding to generate the required output. The simplest autoencoder would be tasked with regenerating the input shown to it. The encoder in this task would learn a representation of the input image in that lower dimensional space which is required

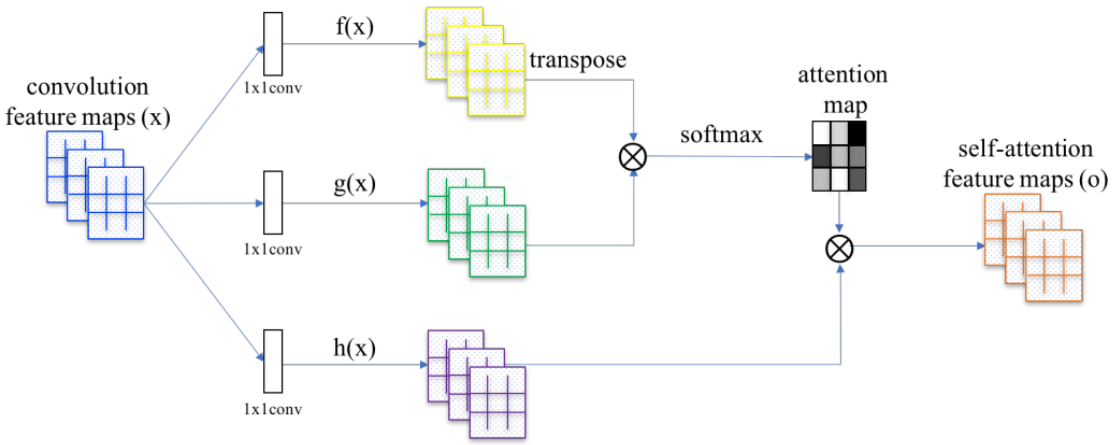


Fig. 2.3: Self-Attention operation. Image from [16]

for the task of regeneration, effectively accomplishing dimensionality reduction or compression. Another application of the autoencoder would be to reduce noise in an input image by showing it a noisy version of the image as input and training it to generate the actual denoised version. These are some applications of autoencoders in the field of self-supervised learning. The basic design of autoencoders can be seen in Figure 2.4.

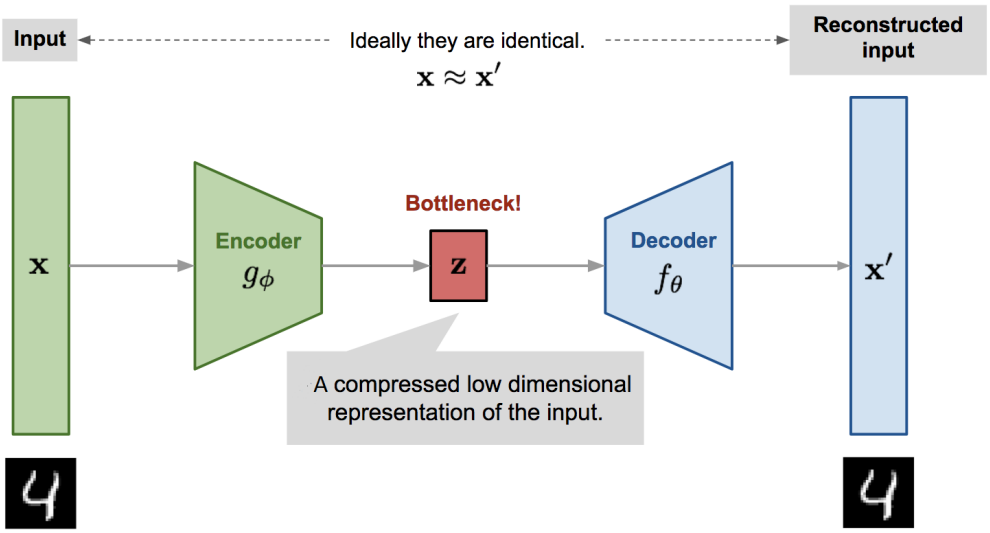


Fig. 2.4: Basic design of an AutoEncoder. Image from `https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`

Nowadays, autoencoders are made powerful using skip-connections [17], which help them accomplish many tasks like image denoising, super resolution, etc. Skip connections provide a smoother flow of gradients in deep autoencoders, helping them learn better repre-

sentations. We have used autoencoders in our research for a couple of experiments, details about which can be found in chapters 3 and 4.

2.5 Covariance Matrix Adaptation - Evolution Strategy

Given a space of samples having a *fitness* score, how can we find the sample with the best fitness score? This is an optimization problem where the global optimum needs to be found for a fitness function with infinite inputs and no analytical information about the structure of surface formed using that fitness function, which may be non-convex. After the generation of fake Fingerprint and Iris samples, our task would be to find masterprints from the latent space used to generate those fake images. This problem is solved computationally using various algorithms like Hill-Climbing, Differential Evolution Particle Swarm Optimization, etc. but as discussed in section 1.2.1, CMA-ES [3] works best for finding masterprints.

Evolutionary algorithms are based on principles of biological evolution and natural selection of the fittest, hence the term fitness function. In an infinite space, all samples cannot be evaluated on a function to find the best, and the function may be ill-conditioned in regions. For ease of computation and for finding approximate best solutions, an evolution strategy is used. Samples are chosen from a multivariate normal distribution in the given input space and then evaluated using the fitness function. The mean and covariance matrix that represents pairwise dependencies between the dependent variables are then updated, weighed according to the fitness scores of evaluated samples for sampling newer and better samples in the next generation. Here, a fitness function having a smooth surface would work extremely well, while functions with very irregular surfaces containing sporadic and solitary high fitness samples may not converge to the intended optimum. The method is illustrated in Figure 2.5. The stepwise working of this algorithm is:

1. Sample N solutions from the multivariate normal distribution having mean μ and covariance matrix C.

2. Evaluate the samples and find a new mean(μ') according to the weighted sum of previous generation samples. Similarly, update the covariance matrix to C'.

3. Make use of two evolution paths to conduct step-size control and fast convergence.

4. Repeat until convergence.

The detailed working of the algorithm and its dependence on all the parameters can be found in the tutorial by Hansen himself [18].

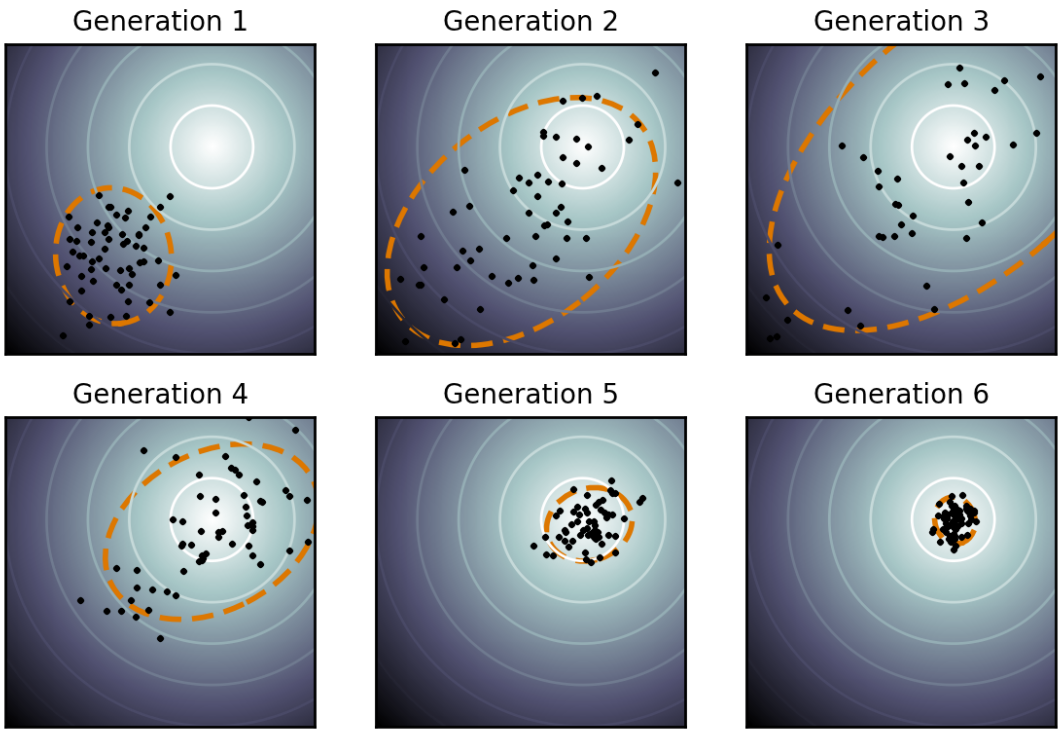
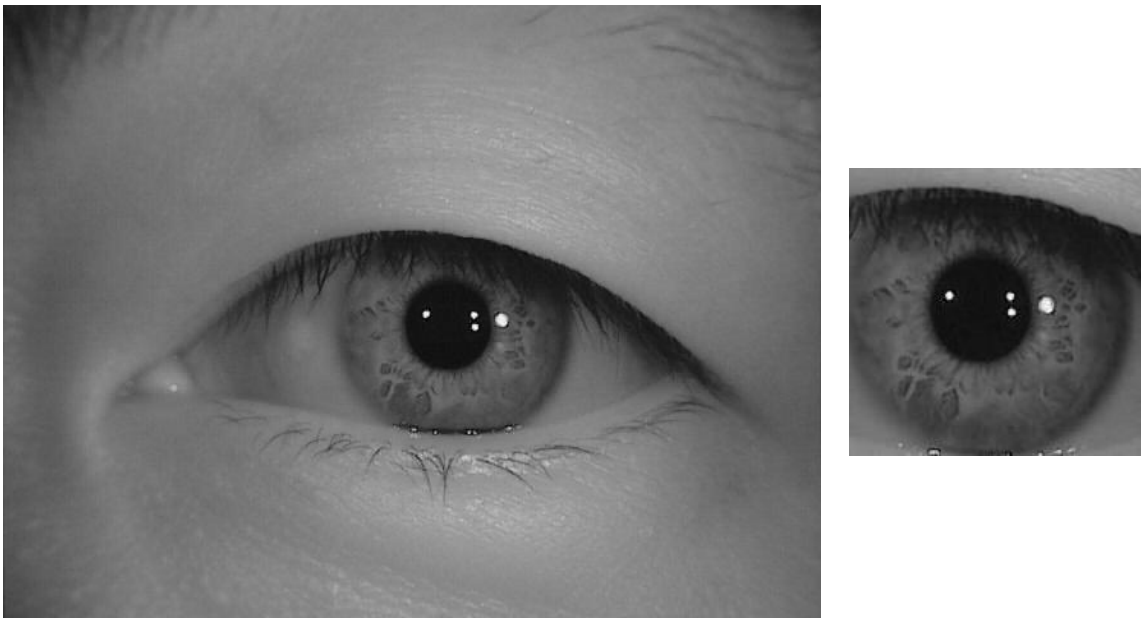


Fig. 2.5: CMA-ES illustration. Image from `https://en.wikipedia.org/wiki/CMA-ES`

2.6 Iris Segmentation and Normalization

Figure 2.6a shows an image from the LAMP dataset used in our research. It is apparent that for the task of Iris recognition, most of the image is irrelevant. For ignoring irrelevant parts and extracting important features from this image, it is first required to segment it and extract the circular portion of the Iris (Figure 2.6b). Even after this segmentation, the pupil of the image is irrelevant. For this, the image needs to be normalized by flattening the circular Iris to a rectangular strip (Figure 2.8). In the following subsections, segmentation, and normalization techniques used by us is discussed.



(a) Sample Iris image from the dataset (b) Segmented image

Fig. 2.6: Iris image segmentation.

2.6.1 Segmentation

For segmenting the Iris portion from the full image, we use the fact that it is circular and the pupil area is black. We need to find the Iris center and radius to segment it from the rest of the image. Note that we worked on grayscale images. For finding the required information, the steps followed are as follows. Some of these steps are visualized in Figure 2.7.

1. Find the color histogram of the image to binarize the image.
2. Set the binary threshold to the first maxima (pupil is black) with some slack to include the pupil region.
3. Use morphological filling to remove the specular reflection.
4. Use Sobel Edge Detection [19] to find edge points in the image and the corresponding gradients at those points.
5. Use the radius constraint and gradients at each edge point to vote for a center point [20].
6. Select the point with most votes (in the Hough Space) as the center of the pupil and use it to find the radius of pupil too.

7. Using the integro-differential operator on circular elements to get the circle with maximum gradient to in turn find the radius of the Iris.

8. Use the center and Iris radius found to crop the image at the appropriate location. This results in the final segmented Iris image.

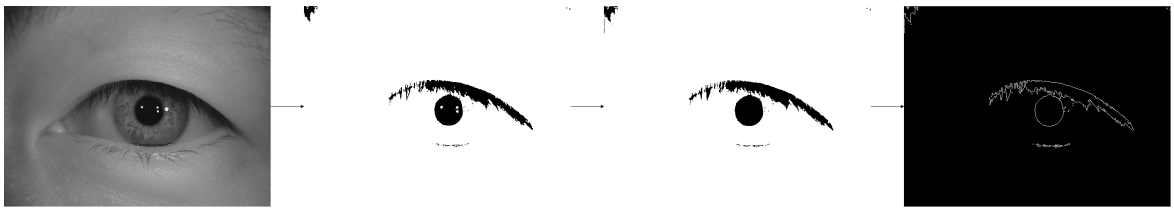


Fig. 2.7: Steps in segmentation. From left to right: original image, binary thresholding, morphological filling, edge detection.

2.6.2 Normalization

The normalization algorithm takes the image, its center point, pupil and Iris radii are taken as input and gives the normalized image of the desired size as output. The algorithm can be found in Algorithm 1. The resultant image obtained can be seen in Figure 2.8. The outline of the algorithm is as follows.

1. Use (Iris Radius − Pupil Radius) as the width (w) of resultant normalized strip. Use $2\pi w$ as the height (h).

2. Use the formula $x = rcos\theta$ and $y = rsin\theta$ to fill corresponding values of the rectangular strip in the output image row-wise from circular elements of the segmented image. Iterate θ from 0 to 2π with rotation step of $\frac{2\pi}{w}$, and r from Pupil Radius to Iris Radius with step size 1.

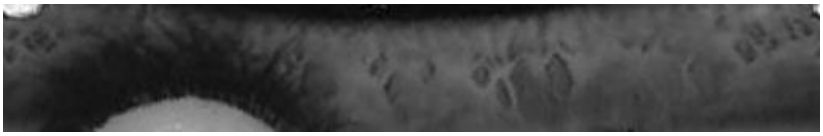


Fig. 2.8: Segmented Iris image after normalization.

Algorithm 1 Normalize a given segmented Iris image to the desired output size

Require: segmented_image, centre_point, pupil_radius, iris_radius, desired_output_size
Ensure: output_image

function NORMALIZE(seg_img, cent_point, pup_r, iris_r, des_output_size)
    $h \leftarrow iris\_r - pup\_r$
    $w \leftarrow 2\pi h$
    $output\_image \leftarrow$ **Empty image of size** (h, w)
    $rotation\_step \leftarrow 2\pi/w$
    $initial\_point \leftarrow (iris\_r/2, iris\_r/2)$
    $r \leftarrow pup\_r$
    for $i \leftarrow 1$ to h **do**
        $rotation \leftarrow \pi/2$
        for $j \leftarrow 1$ to w **do**
            $x \leftarrow initial\_point_0 + rcos(rotation)$
            $y \leftarrow initial\_point_1 + rsin(rotation)$
            $output\_image_{i,j} \leftarrow seg\_img_{x,y}$
            $rotation \leftarrow rotation + rotation\_step$
        end for
        $r \leftarrow r + 1$
    end for
    $output\_image \leftarrow$ **Resize** $output\_image$ **to** des_output_size
    return $output\_image$
end function

Chapter 3

Methodology

3.1 Introduction and General Framework

After studying the problem thoroughly, we decided to approach it with a general strategy, details of which can be found in the following sub-sections. Various experiments with respect to this methodology and their results can be found in detail in the next chapter.

3.1.1 Image Generation

The idea of presentation attacks is that one can assume the identity of another without being noticed. For that, one has to be indistinguishable from the other physically as well. In our case, the masterprints that we generate to achieve this task should also look like real Fingerprint of Iris images. This narrows our task down to finding masterprints in the space of a distribution from which all real images are assumed to come from. However, for this, we need to find the real data distribution. This is a tricky task but has been effectively handled by GANs [6] recently. GANs are generative models that estimate the aforementioned real data distribution. More on GANs can be found in the previous chapter (Section 2.2). We decided to use the most recent general-purpose GAN published at the time of starting our research, i.e., Self-Attention GAN [16]. We used this particular GAN for the following reasons,

1. The performance of this GAN was unprecedented on the ImageNet [21] dataset.

2. After empirical evaluation of our datasets, we hypothesized that,

    - Structure around one minutia cannot be significantly different from that around another.
    - Iris texture around the circumference of the pupil correlates even at distant locations.

    As Self-Attention GAN models long-range dependencies in images, it could work better in our case.

3.1.2 Matching

After we estimate the real data distribution using our SAGAN model and generate new unseen images, we need to match them with real images using some matching algorithm. Standardness and availability determined the matching algorithms used in our research.

- **Fingerprint**: We selected the *Bozorth3* fingerprint matching system, taken from the NBIS suite maintained by NIST[1]. This particular system was selected because it is open-source, and it is standardly used by researchers that have done similar work.
- **Iris**: No freely available Iris matching system is available online. Luckily, Chaturvedi *et al.* recently worked on an Iris matching Siamese network[2]. They were able to get commendable results (CRR 97.4, EER 1.53).

3.1.3 Finding Masterprints

Given a space filled with biometric images and a matching network, the only task that remains is finding masterprints from that space. Each sample from the distribution that we estimated using SAGAN would have a matching result with all poses from all subjects from the real dataset. If masterprints exist, the number of matches of those would be high. Our objective, then, is to find the point in this space with maximum such matches. As we saw

[1] `http://www.nist.gov/itl/iad/ig/nbis.cfm`
[2] Parinaya Chaturvedi worked on the network in his MTP in 2018-19.

in Section 1.2.1, CMA-ES outperformed other optimization techniques in previous works to generate masterprints. Therefore we decided to use it for our optimization task.

3.2 Biometric Specific Procedures

The general framework covers the high-level step-wise goals that need to be accomplished for the proposed solution to work. However, some biometric specific tasks need to be taken care of, for supplying the general framework with appropriate inputs. The following sub-sections will discuss them for Fingerprint and Iris separately.

3.2.1 Fingerprint

Due to practical constraints and for better user experience, smartphone fingerprint sensors are typically small and only partially use a fingerprint. This makes them more susceptible to attacks. We tried to focus on this loophole of modern systems. We used the FVC2002 DB1-A dataset[3] consisting of 100 subjects with 8 images each of the 8 fingers (excluding thumbs). This makes the total number of classes equal to 800. To make partial fingerprints from the real dataset, the following steps are taken. The procedure is visualized in Figure 3.1.

1. Binarize the image using the first three steps from the procedure mentioned in 2.6.1.
2. Remove the unused white portions on the side.
3. Cut the image into overlapping regions of suitable size (150×150).

The images would now be ready to be seen by our GAN. After the successful generation of new fake images from our GAN, minutiae need to be extracted from the images to feed into the matching algorithm. The NBIS suite provides software to do that called *Mindtct*. This is used to extract minutiae from generated as well as real fingerprints. After that, they are matched using the *Bozorth3* software. Optimization for masterprints is done using CMA-ES.

[3] `http://bias.csr.unibo.it/fvc2002/default.asp`

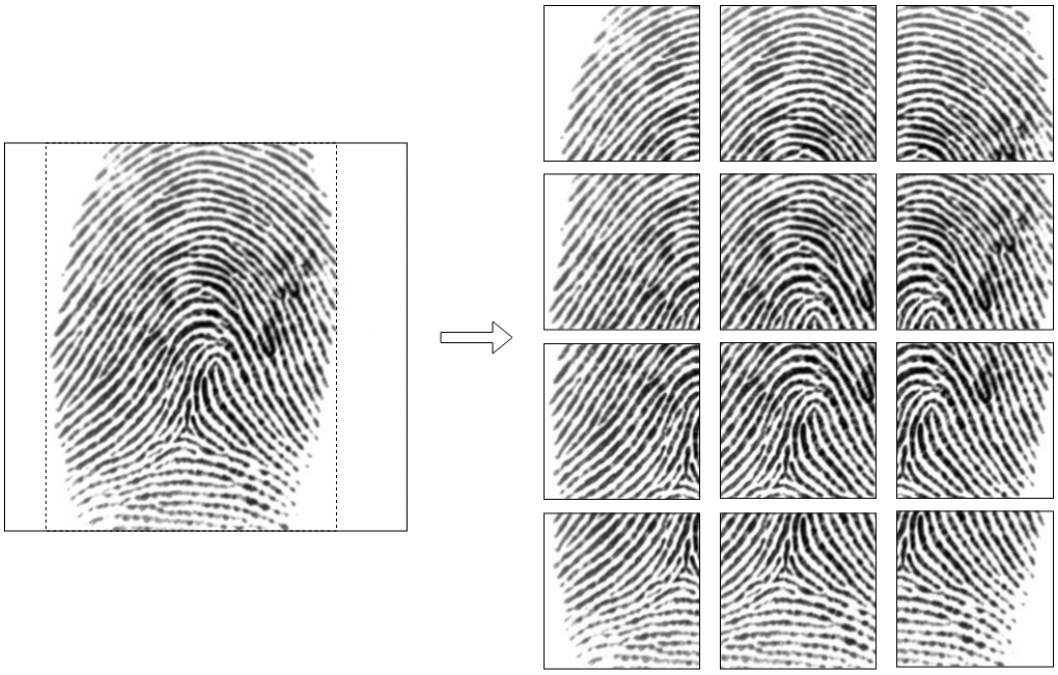


Fig. 3.1: Segmenting and cutting the actual fingerprint into partial fingerprints.

3.2.2 Iris

The Iris dataset used in our research is CASIA-Iris-Lamp[4]. This dataset consists of 411 subjects with 819 classes corresponding to the two eyes of each subject (some classes are missing). Images (as shown in Figure 2.6a) from the dataset are large (640×480) and maximum area of the image is irrelevant for matching. To segment out Iris portion from this big image, we used the procedure explained in Sections 2.6.1 and 2.6.2. The procedure results in a transformation as shown in Figure 3.2.

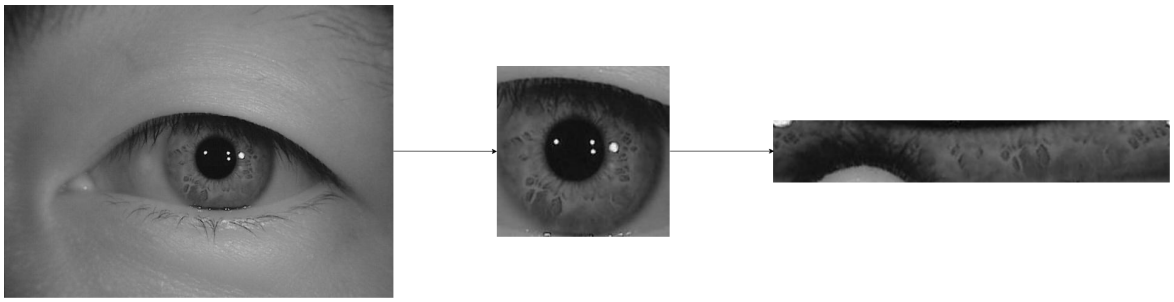


Fig. 3.2: Segmenting and Normalizing the original Iris image.

[4] http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp

Chapter 4

Experiments, Results and Analysis

4.1 Introduction

Given the problem, data, and methodology, all that remains is to conduct experiments, gather results, analyze them to find problems and re-iterate until they agree with the theory. As our work focussed on two different biometric traits, we discuss experiments, results, and analyses for the same in two different sections. We also discuss some failed experiments and discuss why they failed.

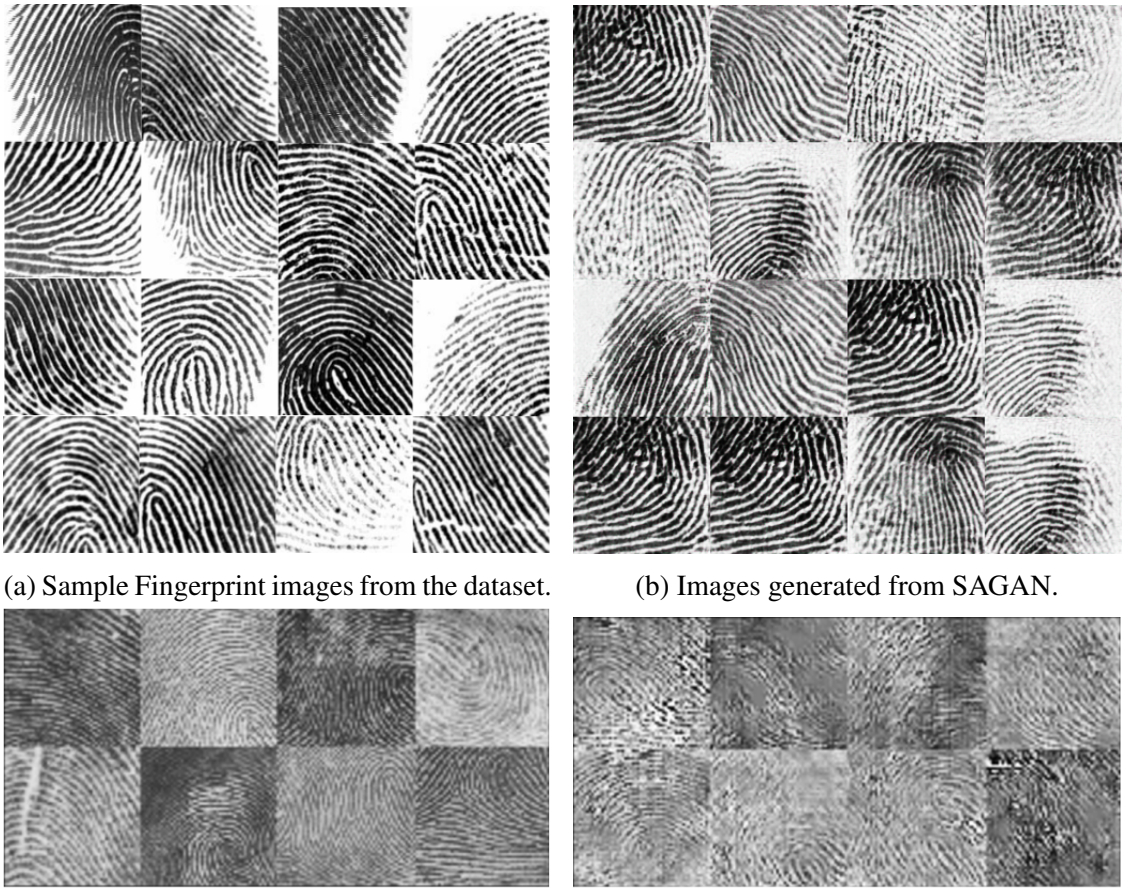
4.2 Fingerprint

4.2.1 Image Generation

The first task in our process to find masterprints is to approximate the real data distribution of our fingerprint dataset. The partial fingerprint dataset obtained after pre-processing as discussed in Section 3.2.1, contains 8,220 samples. We trained the SAGAN model as described in the original paper with a slight modification. We resized the 150×150 image to 128×128. It was found empirically that square image sizes of shape $(2^n \times 2^n)$ performed better in our model than others. This is because the down-sampling (through *Conv* layers) and up-sampling in discriminator and generator respectively, remain consistent across dimensions in each layer. If the image size changes, at least one of the layer architecture has

to change, resulting in poorer performance. The baseline code for SAGAN was taken from the already implemented version on Github from Junho Kim[1].

The model quickly converged and was able to generate real-looking images. General metrics for the quantitative performance of GANs are not yet developed as it is very subjective and still an open area of research. The results generated by our model can be seen in Figure 4.1. In contrast, corresponding images by [5], can also be seen, although on a different dataset (which is not publicly available).



(a) Sample Fingerprint images from the dataset.

(b) Images generated from SAGAN.

(c) Sample Fingerprint images from the dataset.

(d) Images generated by [5].

Fig. 4.1: Fingerprint image generation results and comparison.

[1]Unofficial Implementation: `https://github.com/taki0112/Self-Attention-GAN-Tensorflow`

4.2.2 Fingerprint Matching

To generate masterprints, a benchmark matching system is required. As discussed in Section 3.1.2, we decided to use the *Bozorth3* open-source fingerprint matching system in the NBIS suite from NIST. The matching procedure of two fingerprints is simply two-fold,

1. Use *mindtct* to extract minutiae from the two images.
2. Pass that data to *bozorth3* for matching to get a score.

As no matching system is perfect (not even humans), there is always a certain minimum acceptable FMR for each matching system. The FMR can theoretically be set to zero, allowing no false matches to be accepted but this decreases the FRR as well. As this trade-off exists, an optimal threshold is set such that both of these values are acceptable, keeping in mind the importance of user-experience or security in particular test-cases. Usually, an FMR of 0.01 or lower is used in fingerprint matching systems. Therefore, for setting the threshold score for matching, we did the following.

1. Extracted the minutiae of all images in the dataset using *mindtct*.
2. Matched each pair of these images using *bozorth3* and got a matching score for those pairs.
3. Found the threshold score for which the FMR falls below 0.01.

The threshold score found was **10** where scores ranged in $(0, 170)$. This would be used as a matching threshold when masterprints are tested against the dataset.

4.2.3 Finding Master Finger Prints

We tested random images generated from SAGAN on the matching system, and some of them matched with up to 7 classes from the dataset. This was a clear indication of the existence of masterprints. We then decided to optimize on the number of matches that a particular masterprint gets with the dataset. As discussed in Section 3.1.3, we used the CMA-ES optimization technique to do the same. The steps followed were,

1. Sampled a population of possible masterprints from the latent space used by the generator of SAGAN. We took 256 samples in each generation.

2. Extracted the minutiae and matched them with all the images from the dataset. Used the threshold score found earlier to consider the number of class matches. Assigned the number of class matches as the fitness score of each sample.

3. Using these fitness scores, the mean and covariance matrix used in sampling the new population was updated (see Section 2.5).

4. These steps were repeated for 250 generations when the algorithm converged to almost a single point.

We found that a single masterprint found using this simple fitness function and optimization, was able to match with **184** classes out of the 800 in a dataset ($\sim$ 23%). This is a remarkable result that suggests that standard minutiae matching algorithms can be fooled easily using masterprints. The masterprint found is shown in Figure 4.2.

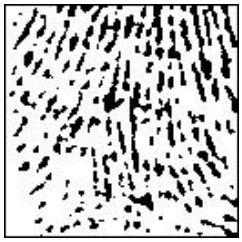


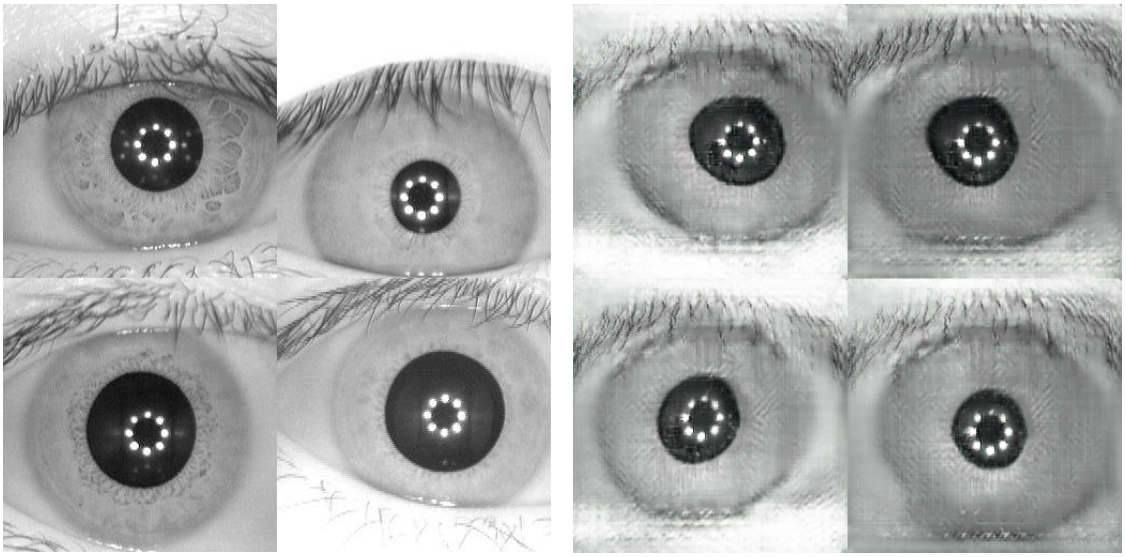
Fig. 4.2: Master fingerprint found using our methodology on the FVC dataset.

As is evident from Figure 4.2, the masterprint is a very light impression of a fingerprint containing many breakage points in ridge patterns. A minutia is the location of breakage points in the ridge pattern of fingerprints like ridge ending or bifurcation, among other less frequent characteristics. Minutiae matching works on the principle of relative position and orientation of minutiae in a particular fingerprint, which is regarded as unique for each subject. The masterprint has a lot of these minutiae because of its unusually grainy nature of ridges. This makes its features (minutiae) overlap with many classes in the dataset, resulting in a fingerprint which can match with many classes on a minutiae matching system. This justifies our hypothesis of the existence of masterprints for the case of fingerprints on a particular minutiae matching system and concludes our work on fingerprints.

4.3 Iris

4.3.1 Image Generation

To train SAGAN on Iris images, we first segmented them using the technique discussed in Section 2.6.1. This was done because in trying to generate the whole image, SAGAN was not able to learn minute details in the Iris region of images, which was the only important region in the image. When trained to generate the whole image, the model learned macro details. Even when trained on smaller full images containing less of the irrelevant details, the training did not proceed well. See sample generated images on the significantly zoomed in images from CASIA-Iris-Interval[2] dataset in Figure 4.3.



(a) Sample Iris images from CASIA-Iris-Interval dataset.

(b) Images generated from SAGAN using the CASIA-Iris-Interval dataset.

Fig. 4.3: Iris image generation using full images.

Therefore we decided to train the model on accurately segmented Iris images. In relevance to the matching network (Section 4.3.2), generating normalized Iris would have been the optimal choice, but this choice was dropped after rigorous experiments as discussed in Section 4.4. From the original images in the CASIA-Iris-Lamp Dataset, we got 16,211 segmented images. The training process was similar to what we did with the Fingerprint dataset, with some new modifications.

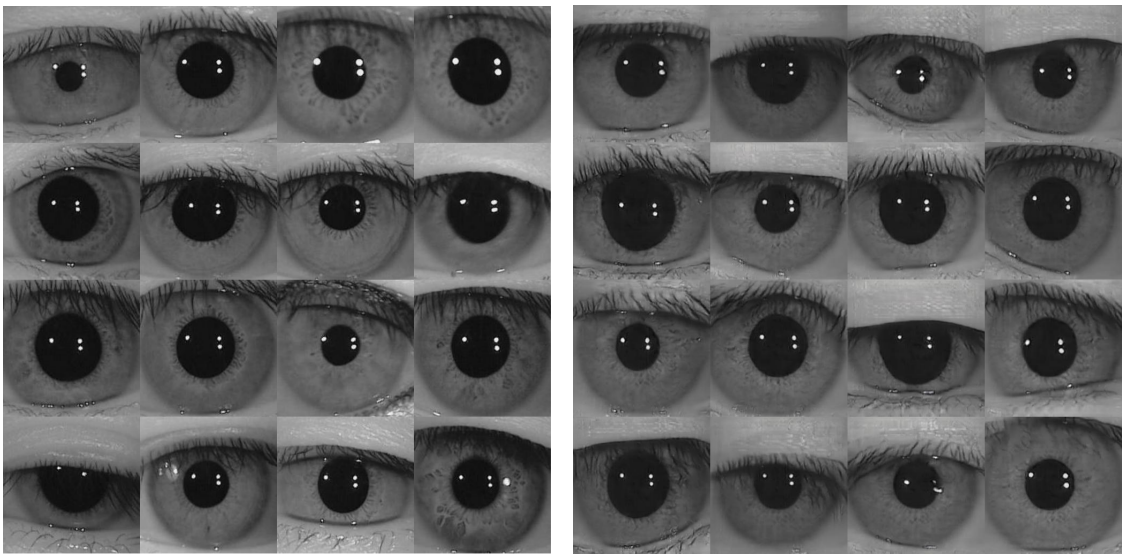
[2] http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp

1. Resized the Iris images to (256×256) due to its larger average size.

2. Learning rate of generator **increased** to 0.0002 from 0.0001 of the model used in fingerprint dataset training. Learning rate of discriminator **decreased** to 0.0001 from 0.0004.

3. Generator was trained twice for each training step of the discriminator.

These steps were necessary because several training experiments failed with other learning rates as discriminator losses dropped to zero. The discriminator was quickly able to differentiate between real and fake images, not allowing the generator to rise in performance. To tackle this, more focus was laid on the generator training, as is evident from the modifications. Even after this hyperparameter tweaking, only once were we able to generate good Iris images. Even with the same architecture and hyperparameters, the training did not stabilize for future experiments. Luckily, we saved the model that generated the best images, and the results can be seen in Figure 4.4. The thing to note here is that the training of SAGAN and GANs, in general, is very sensitive not only to hyperparameters but initializations and even the training sequence of inputs.



(a) Sample segmented Iris images from CASIA-Iris-Lamp dataset.

(b) Images generated from SAGAN using the CASIA-Iris-Lamp dataset.

Fig. 4.4: Iris image generation using segmented data.

4.3.2 Iris Matching

As discussed in Section 3.1.2, due to unavailability of standard open-source Iris matching systems, we decided to use a neural network trained by Chaturvedi *et al.*, which was trained to match on 783 classes out of the 819 in the CASIA-Iris-Lamp dataset. It takes normalized Iris images as input and brings them down to a 128-dimensional embedding space. In this space, the Euclidean distance between each embedding determines whether they matched or not. The threshold distance for an optimum match was found to be **0.2586**. At this threshold, the EER was 1.53, which is pretty standard for matching systems and good enough for us to try finding masterprints with respect to it. For the task of matching, we needed to normalize the segmented images generated by our model first. The algorithm used for that task is discussed in detail in Section 2.6.2.

4.3.3 Finding Master Iris Prints

Following the same strategy as Fingerprints, we used the CMA-ES optimization technique to find masterprints for Iris. However, there were several interesting findings in our experiments to do the same, which are summarized below.

- Random generated samples did not match with any of the classes, in contrast to fingerprints, which matched with many.
- We tried to find optimum samples for masterprints by using several fitness functions like,
    1. **Number of matches**: Similar to Fingerprint but this approach only lead to masterprints matching with at most 2 classes.
    2. **Minimum Distance**: Intuitively, this approach only found the Iris sample from the space which had the most similar features to any single image from the dataset.
    3. **Weighted Minimum Distance and Number of Matches**: Did not result in anything better than previous approaches.

4. **Minimum Mean Distance with all classes**: To find a generic masterprint that would be similar to many classes, minimizing the mean distance with all classes was another approach to be tried. However, this did not work either.

- Also, uniform sampling instead of random normal sampling was tried as well, but with no results.

These experiments led us to investigate why our technique was not able to find masterprints. To visualize the higher dimensional embeddings and see how distant they are from each other, we used a technique called t-SNE [22]. t-SNE is typically used to visualize high dimensional data in 2 or 3-dimensional space by estimating lower dimensional distributions that behave similarly to the original data. We used this technique to plot the embeddings of real Iris images in 2-dimensional space. To avoid over-crowding, the data for 100 classes is shown in Figure 4.5.

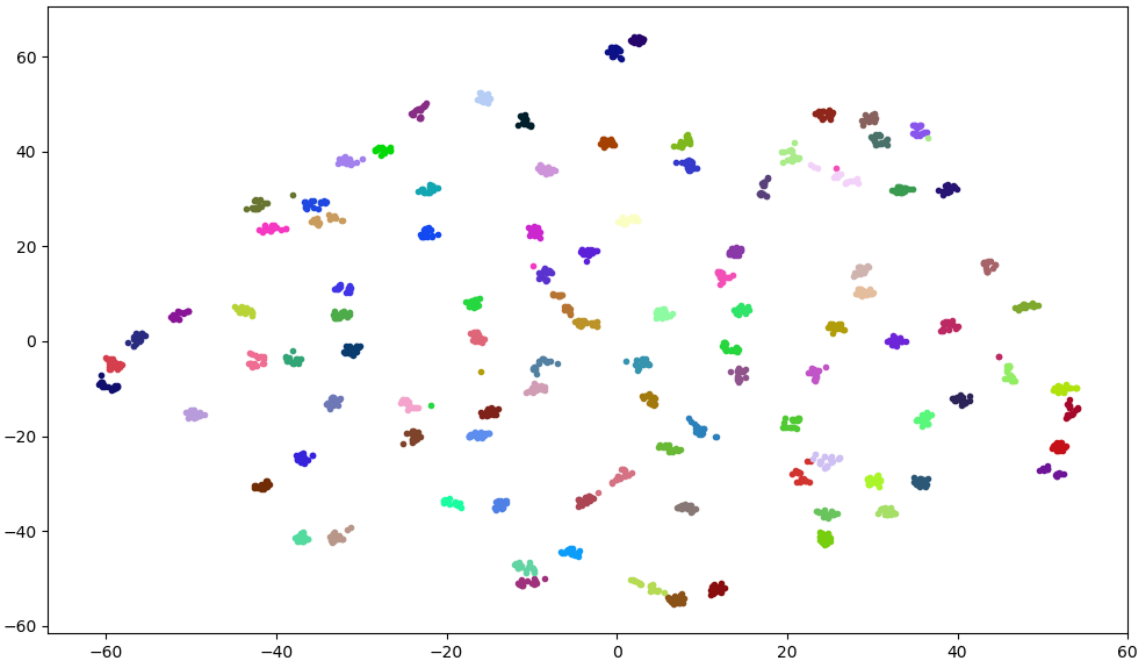


Fig. 4.5: t-SNE visualization of real Iris data embeddings.

It is evident that the real data is reasonably evenly spread out in the given space. It helps to understand why we are not able to find masterprints. Intra-class distances are low and inter-class distances are high, as they should be. Any image, when passed through this

matching network's convolutional layers, is brought down to the 128-dimensional embedding space and it becomes a point in the t-SNE representation. No matter what features an image has, it can never match with many classes as they are all distant. This is illustrated by an experiment that we next tried.

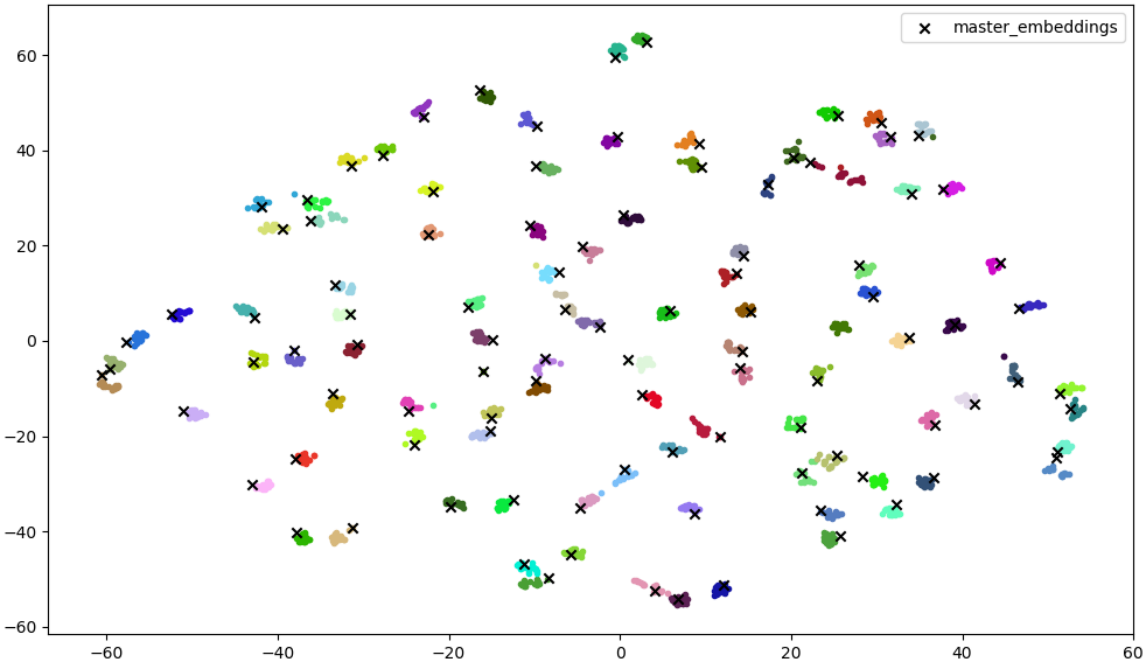


Fig. 4.6: t-SNE visualization of real Iris data embeddings alongwith class-specific matches found in the fake space.

We tried to find samples in the latent space, from which our generator generates images, which match with particular classes in the real dataset. We were able to find matches for all the classes. See Figure 4.6 which shows the t-SNE representation of the data from the previously used 100 classes along with 100 *master* Iris images found for each of these classes. The embeddings of those found images are shown as crosses. This image illustrates that,

1. The generator of our SAGAN model was able to estimate the real distribution very well as it could find samples in the space which match with individual classes.

2. The inter-class distances between classes does not allow masterprints to exist, as it is mathematically impossible for a point in this space to overlap with many class regions.

There were however a few classes which were significantly close in the embedding space and our optimization technique was able to find masterprints which could match with at

most 3 classes at once. With these experiments, we were able to successfully reject the hypothesis of the existence of masterprints for Iris with respect to the matching network we used. Conclusion and future work are discussed in the next chapter.

4.4 Failed Experiments

We started our experiments with the Iris dataset by using normalized images from the initial stage of Image Generation. This approach never resulted in any fruitful results, and we never even crossed the image generation stage satisfactorily, but there were many learnings from these experiments, which are briefly discussed, for future references.

4.4.1 Direct Image Generation

The normalized Iris image (Figure 2.8), has the key structural difference of being rectangular instead of the square segmented Iris image, which was ultimately used by us. Image generation for this rectangular image (80×512) proved to be the trickiest of tasks. To start with, we changed the structure of our SAGAN model to generate images of this particular shape. We changed the dimension of latent space from 128 to 116, reshaped tensors and conv layer strides to generate an (80×512) image finally. The discriminator was also changed a lot, especially in the last layers, to find logits of an input image. Finally, we trained it on the normalized Lamp dataset containing 15,660 images. The best results can be seen in Figure 4.7.

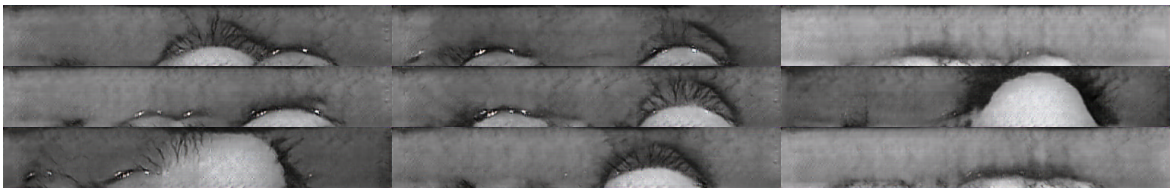


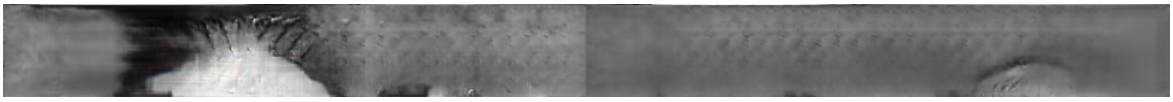
Fig. 4.7: Best results of training SAGAN on normalized Iris images.

These images look nothing like the real images from the dataset. Also, whatever texture they can generate is not relevant to our task. It tries to make patterns in the occluded part of the image while not filling the useful background patterns. This is because the location

of these occlusions keep changing in the real data, and there's no definite pattern for the generator to learn while making new images.

To see if this generator can make real-looking images, we used an autoencoder having the same bottleneck space as the latent space of generator used in SAGAN. We trained the encoder part of this autoencoder to learn representations of real images in the latent space while keeping its decoder part fixed and equal to the generator of the SAGAN. In this way, we wanted to see if the generator can make any real looking image and found that it was not able to do so. See Figure 4.8. This was because the AutoEncoder and generator were trained separately and could not produce dependent results.

(a) Sample real images from normalized Lamp dataset.



(b) Corresponding generated images using embeddings generated from the AutoEncoder passed to the SAGAN generator.

Fig. 4.8: Experiment with AutoEncoder, which evidently failed.

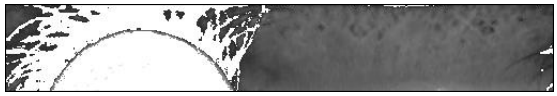
We also tried to train another architecture of SAGAN with 2 discriminators and a single generator. The generated normalized images were converted back to segmented Iris images using a reverse transform of what was discussed in Iris normalization (Section 2.6.2). This segmented image was then passed to the second discriminator along with real segmented images, and losses from both of these discriminators were used to train the model. This variant did not work any better than the vanilla model tried first probably because bad image production from the generator overshadowed the performance improvement provided by the second discriminator.

4.4.2 Indirect Image generation

Having tried all variants of direct image generation, we concluded that the generated images were of bad quality because the occlusion in the images was hindering the generator's ability to learn meaningful textures. To tackle this task, we tried to handle the occluded and back-

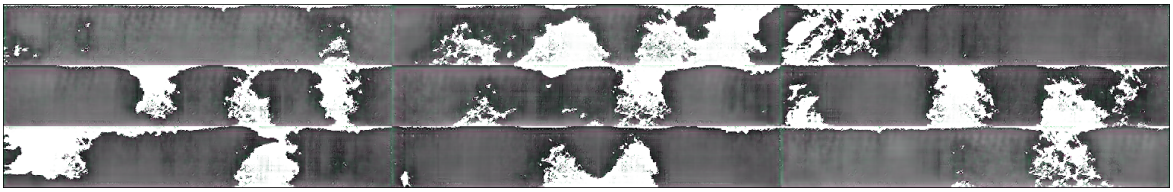
ground regions of the image separately. We found occlusion masks of the dataset images and trained two different networks to generate occluded and background regions separately. Empirically, we saw that painting the unwanted region white worked better than black, probably because the texture in the images is also dark in color. The result of independent training can be seen in Figure 4.9.

(a) Occluded region in a normalized Iris image.



(b) Background region of the same image.

(c) Samples of generated occluded region in a normalized Iris image.



(d) Samples of generated background region in a normalized Iris image.

Fig. 4.9: Training occlusion and background separately.

The two regions cannot be generated independently, however. Although the visual results were not appealing, we tried to train the two networks together, providing them a single input latent space vector. The combined added image from both generators is also fed in a third discriminator for an additional loss. The architecture is shown in Figure 4.11. This architecture did not work out as well, probably because of high independence between the two generators initially. Some training images can be seen in Figure 4.10.

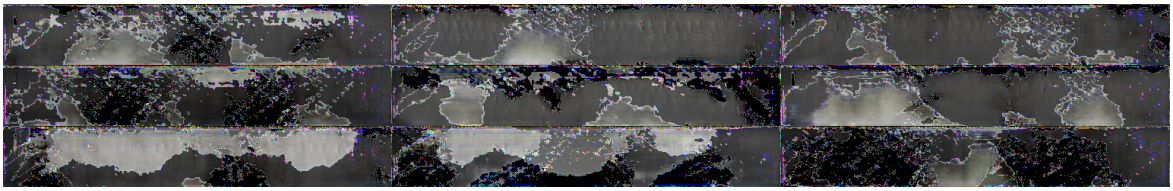


Fig. 4.10: Sample images from dual network architecture training.

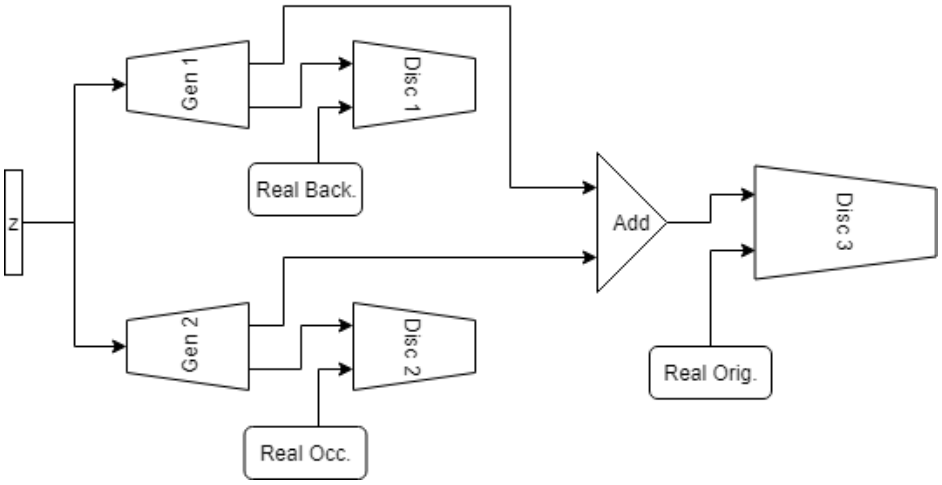


Fig. 4.11: Training both parts together in SAGAN.

Finally, after all these failed experiments we gave up working on the Normalized Iris dataset and moved on to the segmented Iris images, from which we were successfully able to get good results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This work presents new findings in the field of presentation attacks on biometric matching systems using masterprints. The proposed methodology was able to work on two different biometric modalities and generate the intended results. Our key contributions are,

1. Successful generation of masterprints with high match rates for the fingerprint dataset with respect to a widely used matching system.

2. Successful rejection of the hypothesis of masterprint existence in Iris with respect to a high-performance matching system, concluding that Iris is the better biometric among the two most popular biometric traits used for authentication.

Along with these results, we showed what quality of the trait or the matching system makes a fingerprint more prone to these attacks rather than Iris. We also conducted many experiments and documented our findings so that they can be used for future reference. Further testing with other matching systems should consolidate the effectiveness of our proposed approach.

5.2 Future Work

Our research has a few natural extensions like,

1. Using other matching systems on the same biometric modality.

2. Conducting experiments on other biometric modalities like Finger Knuckle prints, Palm prints, Ear or Face.

Comparing the effectiveness of our method on these different modalities and matching systems exhaustively, will provide ground to the discussion of the ideal biometric.

Another area of research is the improvement of matching systems. We saw that fingerprint minutiae matching systems are prone to such attacks because of their overlapping nature of features between classes. Improving the performance of the matching system by adversarial machine learning is another field of research, that can be extended from this work.

Bibliography

[1] A. Roy, N. Memon, and A. Ross, "Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 2013–2025, Sep. 2017.

[2] A. Roy, N. Memon, J. Togelius, and A. Ross, "Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition," in *2018 International Conference on Biometrics (ICB)*, pp. 39–46, Feb 2018.

[3] N. Hansen, S. D. Mller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, pp. 1–18, March 2003.

[4] K. Cao and A. K. Jain, "Learning fingerprint reconstruction: From minutiae to image," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 104–117, Jan 2015.

[5] P. Bontrager, A. Roy, J. Togelius, N. Memon, and A. Ross, "Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–9, IEEE, 2019.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, pp. 214–223, 2017.

[8] J. Daugman, "How iris recognition works," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 21–30, Jan 2004.

[9] N. Kohli, D. Yadav, M. Vatsa, R. Singh, and A. Noore, "Synthetic iris presentation attack using idcgan," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 674–680, Oct 2017.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[11] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Cartoongan: Generative adversarial networks for photo cartoonization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9465–9474, 2018.

[13] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," *arXiv preprint arXiv:1803.07422*, 2018.

[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[15] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[16] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[17] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," *arXiv preprint arXiv:1606.08921*, 2016.

[18] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[19] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of solid-state circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[20] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," tech. rep., SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1971.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[22] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.