**BE. PROJECT-2**

**ON**

**MITIGATION OF SPOOFING ATTACKS IN BIOMETRIC FINGERPRINT SYSTEM USING GENERATIVE ADVERSARIAL AND CONVOLUTION NETWORKS**

**Submitted by**

**Sahil Bhola (2016UIC3003)**

**Archit Talegaonkar (2016UIC3013)**

**Shashank Sinha (2016UIC3017)**

**Nipun Gupta (2016UIC3045)**

In partial fulfilment of B.E. (Instrumentation and Control Engineering) degree

of University of Delhi

**Under the Guidance of**

Dr. Prerna Gaur

****

**DIVISION OF INSTRUMENTATION AND CONTROL ENGINEERING**

**NETAJI SUBHAS INSTITUTE OF TECHNOLOGY**

**UNIVERSITY OF DELHI, DELHI**

DECLARATION

We hereby declare that this thesis entitled “**Mitigation of spoofing attacks in biometric fingerprint system using generative adversarial network and convolutional network**” was carried out by us for the degree of Bachelor of Engineering in Instrumentation and Control Engineering under the guidance and supervision of **Prof.** **Prerna Gaur**, Netaji Subhas Institute of Technology, University of Delhi, India.

We certify that, to the best of our knowledge, this thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material form the work of other people included in our thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

Place: Netaji Subhas Institute of Technology, University of Delhi

Date: 29th July 2020

**Sahil Bhola Archit Talegaonkar Shashank Sinha Nipun Gupta**

**(2016UIC3003) (2016UIC3013) (2016UIC3017) (2016UIC3045)**

ACKNOWLEDGEMENT

First and foremost, we would like to sincerely thank our research supervisor and mentor, Prof. Prerna Gaur (Professor, ICE Division, NSIT) for guiding us in the right direction and providing us with valuable inputs and feedback regularly. Without her assistance and dedicated involvement in every step throughout the process, this project would have never been steered to the path it is on today. In spite of her busy schedule, she helped us realize the long term goal, guided us to reading material, and so on and so forth. She surely has left an indelible imprint on the project.

We would also like to thank our Institution and all the faculty members and staff of ICE division without whom this project would have been a distant reality. We also extend our heartfelt thanks to those who directly or indirectly helped us complete this project.

**Sahil Bhola Archit Talegaonkar Shashank Sinha Nipun Gupta**

**(2016UIC3003) (2016UIC3013) (2016UIC3017) (2016UIC3045)**

THESIS CERTIFICATE(SUPERVISOR)

This is to certify that the report entitled “**Mitigation of spoofing attacks in biometric fingerprint system using generative adversarial network and convolutional network**” being submitted by Sahil Bhola, Archit Talegaonkar, Shashank Sinnha and Nipun Gupta to the Division of Instrumentation and Control Engineering, NSIT, for the award of Bachelor’s degree in Engineering, is the record of the bonafide work carried out by them under our supervision and guidance. The results contained in this report have not been submitted either in part or in full to any other university or institute for the award of any degree or diploma.

**Supervisors**

**Prof. PRERNA GAUR**

**ICE Division**

**NSIT, New Delhi, India**

**THESIS CERTIFICATE(HOD)**

This is to certify that the report entitled “**Mitigation of spoofing attacks in biometric fingerprint system using generative adversarial network and convolutional network**” being submitted by Sahil Bhola, Archit Talegaonkar, Shashank Sinha and Nipun Gupta to the Division of Instrumentation and Control Engineering, NSIT, for the award of Bachelor’s degree in Engineering, is the record of the bonafide work carried out by them under our supervision and guidance. The results contained in this report have not been submitted either in part or in full to any other university or institute for the award of any degree or diploma.

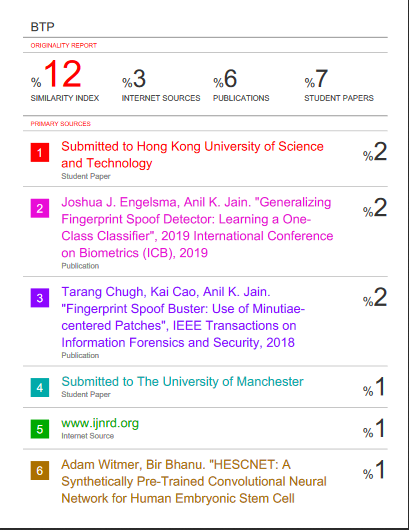
**Head Of Department**

**Prof. PRERNA GAUR**

**ICE Division**

**NSIT, New Delhi, India**

**SIMILARITY REPORT**



**ABSTRACT**

Today fingerprint detection system are being used widely , from a corporate

office to military camps . They are secure , have speed and accurate but they are vulnerable to spoof attacks. And the primary purpose of the fingerprint reader is to provide reliable and accurate user authentication but also to be secure and ensure user confidence.

The most prominent vulnerability in fingerprint spoof detection system were

poor generalization of spoof classes that means whenever a unknown spoof

material was given to the detection system and the error rate increases upto 3 folds .

To Improve the accuracy and performance of the fingerprint detection systems

when fabricated to a unknown number of spoof materials thus decreasing the

cross performance error rate. Hence improving the poor generalizing problem

of a fingerprint spoof detector using generative and other convolution networks.

We are using one class classification and Minutiae extraction approaches using

DCGANs and MobileNets Respectively and using these networks giving a spoof

score to given fingerprint and found out that our results had an accuracy of 5-

10% more than the previous binary spoof classifiers.

**LIST OF TABLE**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **DESCRIPTION** | **PAGE** |
| Table 1 | *Dataset Description of Livdet* |  |
| Table 2 | *Dataset Distribution and Accuracy Result* |  |

**LIST OF FIGURE**

|  |  |  |
| --- | --- | --- |
| **Figure** | **Description** | **Page** |
| **Figure 1.** | Fingerprint Scanning Steps |  |
| **Figure 2.** | Fingerprint cloning using fevicol |  |
| **Figure 3.** | Fingerprint Play Doh Cloning |  |
| **Figure 4.** | One Class Classification vs Binary Class Classification |  |
| **Figure 5.** | Illustration of GANs abilities by Ian Goodfellow |  |
| **Figure 6.** | GANs Workflow |  |
| **Figure 7.** | GAN Generator Model |  |
| **Figure 8.** | GAN Discriminator Model |  |
| **Figure 9.** | GAN Schema |  |
| **Figure 10.** | Batch Normalization |  |
| **Figure 11.** | Single Layer Perceptron Model |  |
| **Figure 12.** | Sigmoid function |  |
| **Figure 13.** | Tanh Function |  |
| **Figure 14.** | ReLU |  |
| **Figure 15.** | Leaky ReLU |  |
| **Figure 16.** | Generator architecture layerwise summary and parameters |  |
| **Figure 17.** | Discriminator architecture layerwise summary and parameters |  |
| **Figure 18.** | Generator and Discriminator Training Model |  |
| **Figure 19.** | Minutiae Based Extraction |  |
| **Figure 20.** | Starting with Noise |  |
| **Figure 21.** | Intermediate images better with epochs |  |
| **Figure 22.** | Fingerprint final |  |
| **Figure 23.** | Testing Results |  |

**TABLE OF CONTENTS**

1. INTRODUCTION
2. LITERATURE REVIEW
3. METHODOLOGY
   1. ADVERSARIAL LIVENESS DETECTION ONE CLASS CLASSIFIER
      1. ADVANTAGE OF ONE CLASS CLASSIFIER OVER BINARY CLASS CLASSIFIER
   2. INTRODUCTION TO GAN
      1. WORKING OF GANS
   3. GENERATIVE PARADIGM LOGICAL ANALYSIS
      1. WHAT IS THE CONCEPT OF GENERATIVITY
      2. GAN AND OTHER GENERATIVE MODEL GAP
         1. ROBUST VISIBLE CONVICTION NETWORKS
         2. SELF ENCODER VARIATION
      3. HOW DOES THE GAN STAND
         1. HOW DOES THE GAN FUNCTION
         2. WHAT IS DCGAN
      4. IMPLEMENTATION OF DCGAN IN CHAINER
         1. DEFINE GENERATOR MODEL
         2. PREPARE DATASET AND ITERATOR
         3. PREPARE MODEL AND OPTIMIZER
         4. PREPARE UPDATER
         5. PREPARE TRAINER AND RUN
         6. START TRAINING
   4. GENERATOR MODEL
   5. DISCRIMINATOR MODEL
   6. TRAINING OF GAN
   7. LOSS FUNCTION AND OPTIMIZER
      1. LOSS FUNCTION
      2. TYPES OF LOSS FUNCTION
         1. HINGE LOSS
         2. CROSS ENTROPY LOSS
         3. REGRESSION LOSS FUNCTION
      3. OPTIMIZER AND ITS TYPE
         1. GRADIENT DESCENT
         2. MOMENTUM
         3. ADAM
         4. BATCH NORMALIZATION
   8. ACTIVATION FUNCTION
      1. SIGMOID ACTIVATION FUNCTION
      2. TANH ACTIVATION FUNCTION
      3. ReLU ACTIVATION FUNCTION
      4. LEAKY ReLU ACTIVATION FUNCTION
   9. DCGAN ARCHITECTURE AND TRAINING
   10. MINUTIAE EXTRACTION CENTRED PATCHES TECHNIQUE
       1. IMPLEMENTATION
          1. IMAGE AUGMENTATION
          2. MODEL IMPLEMENTATION
4. DATASET AND LIBRARY USED
5. RESULTS
   1. COMPARISON BETWEEN TWO APPROACHES
6. FUTURE WORK AND CONCLUSION
7. REFERENCES
8. ABOUT AUTHORS

**1.  INTRODUCTION**

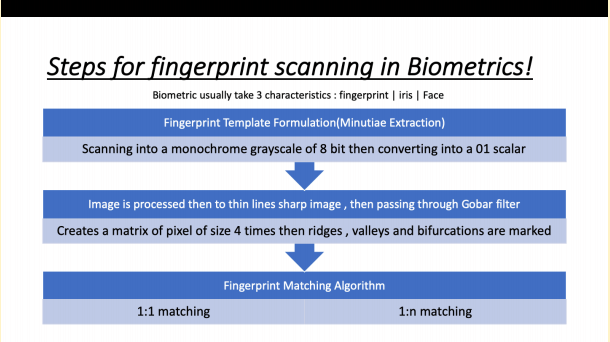
Today fingerprint detection system are being used widely , from a corporate office to military camps . They are secure, have speed and accurate but they are vulnerable to spoof attacks.

Today, fingerprints biometrics are taking place of traditional IDs, used in forensics, border crossing security, mobile authentications, payment transactions, ATM machines , laptops and places where user authentication is required. Locks can be stolen , safes can be broken, passwords can be guessed sooner or later . So how do we protect the things that we value?

Further, the costs of maintaining password and token based systems are very high and inefficient. Resetting lost or forgotten passwords takes up IT support

time and reduces employee productivity.

Fingerprint recognition looks for the unique patterns of ridges and valleys that are present in an individual’s fingerprint. Here then we use biometrics say fingerprint scan , Retinal Scan , iris scan , Face scan as they cannot be forged. In specific fingerprints have,—the tiny friction ridges on the ends of our fingers and thumbs make it easier to grip things. What makes fingerprints such a brilliant way of telling people apart is that they are virtually unique. These examples are extraordinary to each person and subsequently help to recognize people from a whole populace. Fingerprints are inalienable to people and can neither be lost nor taken which makes it exceptionally exact and dependable Moreover, the accessibility of ease unique mark perusers combined with simple mix capacities has prompted the wide spread arrangement of unique mark biometrics in an assortment of associations.

****

*Fig 1: Fingerprint Scanning Steps*

To avoid spoof detection automated fingerprint detectors were trained to

distinguish between live and bonafide fingerprint from known spoof

materials. But they were still vulnerable to spoofs made with materials not

given in training. To solve this many Deep convolutional networks using whole image and minutiae based local patches are used.

Different Spoof Attacks :

• In general , spoof attack is providing false data to gain illegitimate access to

the system

• Spoof artifacts are provided to sensor to fool the system

• These artificial objects imitates biological and behavioral characteristics

• There are number of unknown & known spoof materials & techniques for

the forgery of data or other resources.

Example of Spoof Attacks:

• In smartphones unlocking and accessing with fingerprint has become very

common , hackers are gaining access by scanning and printing fingerprints

by using conductive inks and printing on paper cut accessing mobile

phones.

• In MSU , they have developed wearable finger that mimics human skin in

optical , mechanical and electrical Properties

• Similarly various other cloning materials like playdoh, dental molding, 3D

Fingerprinting

****

*Fig 2.Fingerprint cloning using fevicol*

****

*Fig 3 Fingerprint Play Doh Cloning*

**2. LITERATURE REVIEW**

Conventional biometric fingerprint reader is reliable, speed, security, and spoof attack vulnerability. To prevent the problem, the traditional device used hardware and software before authentication to detect spoof attacks used other sensors, capturing features such as breathing, thermal performance, and blood flow, odor.

Just using a monitor but using textural, autonomic, neurological functions. Technology does not. This is a binary problem with classification, where an unknown number of spoof classes should not be given as the unknown number of spoof materials which are not feasible. Therefore, errors are decreased as "unsighted" fake materials are observed.

Spoofing also creates vibration and loss of points such as frictional arms, forks, islands, etc. In this way they use this insight to perform a 2-class CNN by using a nearby pad. The technique is better than previous methods using the whole image or randomly chosen patches for novel packaging materials.

The downside of this strategy is

1. Is the use of local 96x96 patches Downsize whole CNN teaching fingerprint images.
2. Offers a considerable amount of data per fingerprint on average of 48 patches.
3. Learns strong patch textural characteristics to distinguish live fingerprints and spoof.
4. Furnish a digital layer of fingerprint images
5. Able to locate CNN spoof partial fingerprint output.

**3. METHODOLOGY**

So there are two approaches:  One we use GANs with considering spoof detection as a one class classification problem. Another approach is to  Extract Minutiae based patches and give them a spoof score using MobileNets. Hence our aim will be to implement these approaches and get better results as compared to the given binary CNN classifiers and CNN models.

**3.1 Adversarial Liveness Detection : One Class Classifier**

To avoid spoof detection automated fingerprint detectors were trained to distinguish between live and bonafide fingerprints from known spoof materials. But they were still vulnerable to spoofs made with materials not given in training. To solve this single classification class was proposed. Goal is to train the detector only to detect live fingerprints then spoof of any other

material will be rejected. We accomplish this by training over our dataset

with GANs network.

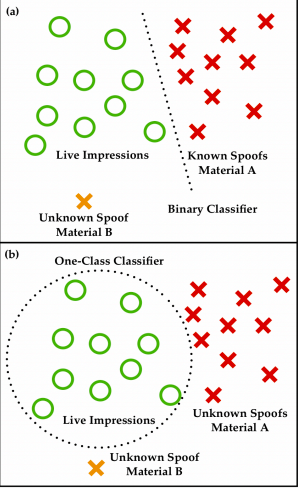
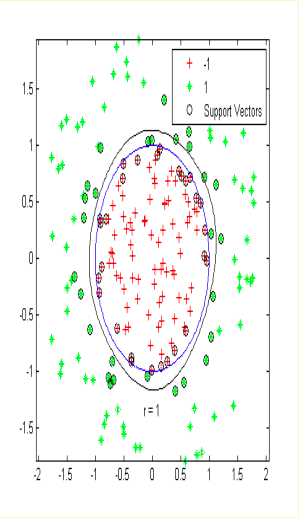
***3.1.1 Advantages of one class classifier over binary classifier?***

• As it were live tests are required for preparing hence dispensing with task of manufacturing huge number of spoof impressions from different materials

• One class classifier do not overfit the data while binary does hence cross performance decreases

• It only learns what constitutes a live fingerprint & does not use spoof material of any specific material during training.

• So they have a tight decision boundary around one class say live samples and all other class samples are unknown (i.e. spoof)

****

*Fig 4 : One Class Classification vs Binary Class Classification*

**3.2  Introduction to GAN**

Generative Adversarial Networks or GANs were introduced and developed by Iann J. Goodfellow in 2014. It was described as “the most interesting idea in the last 10 years in Machine Learning”. Generative Adversarial Networks are classified as generative models. They are capable of generating or producing new data by learning from existing data.After sufficient training, they can generate data that look at least superficially authentic to human observers, having many realistic characteristics.GAN has led to thousands of research papers in recent years. The scope of GA is not just limited to generating data. In Reinforcement Learning, it enhances the learning speed of a robot.



*Fig 5: Illustration of GANs abilities by Ian Goodfellow.*

***3.2.1 Working of GANs***

As the name suggests, Generative Adversarial Networks is made up of 3 parts- Generative, Adversarial and Networks.

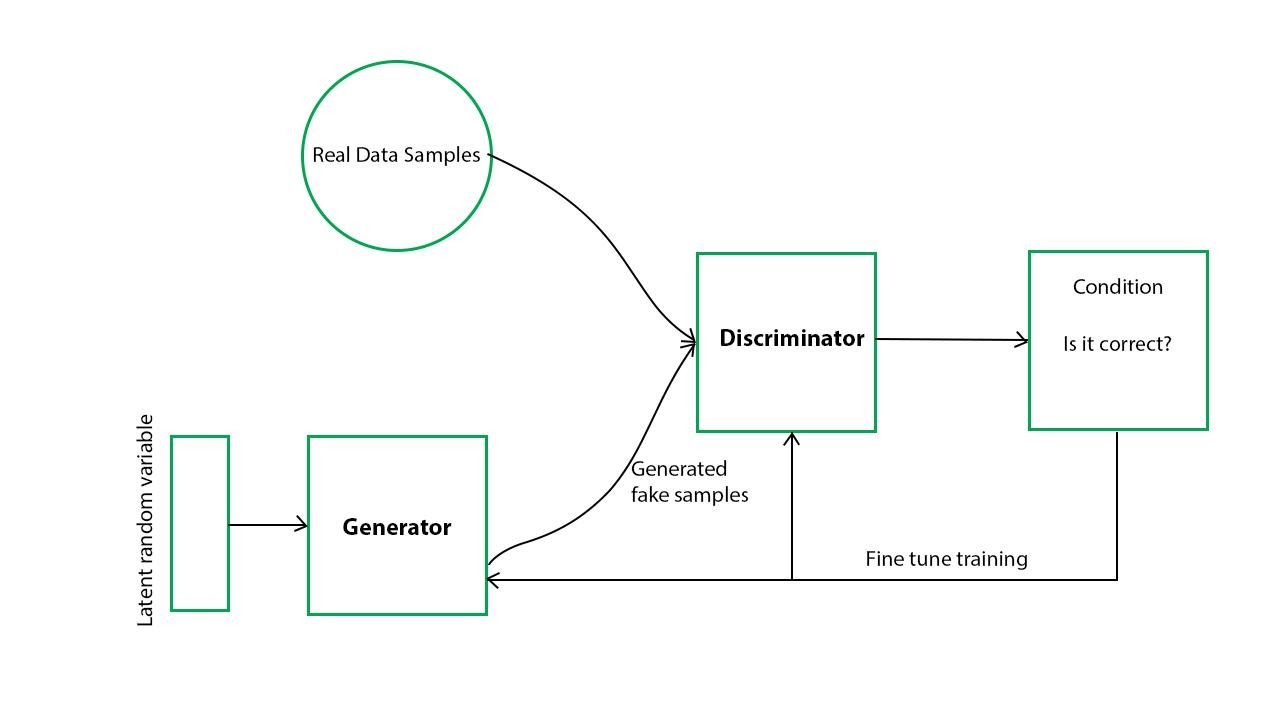
**Generative-** Learning a generative model, relating to a data generation with respect to a model for probability

**Adversarial-** It describes the adversarial setting in which the model is trained. Adversarial is related to conflicting or opposing goals.

**Networks-** Using deep neural networks for the model architecture and training.

Let’s take an example of a police man and a counterfeiter to understand the working of GANs. Job of a counterfeiter is to make fake currency which looks like real notes. Job of police man is to identify fake currency from real currency, i.e- discriminate between fake and real currency. Let’s say initially police is expert at detecting currency. Once, it detects fake currency, it sends a feedback to counterfeiter to make necessary changes in order to make the fake currency looks more real. Now, counterfeiter makes better fake currency and policeman has tough time finding the difference between fake and real currency.

GANs consists of two deep networks**-** generator and discriminator. Generator is similar to counterfeiter and discriminator is similar to policeman. Generator aims to generate forfeit data samples and give it to discriminator. Discriminator is responsible for differentiating between actual and generated data. Generator and discriminator are made up of neural networks. During training phase, they both run a min-max game. They repeat these steps for multiple epochs and both of them gets better during training after each repetition.



*Fig 6: GANs Workflow*

Generator gets trained due to feedback from discriminator and thus produces better fake data. Discriminator is getting both real data and fake data from generator which are getting better. As a result, discriminator network also learns how to make better distinction between real data and fake data.

# **3.3 Generative paradigm logical analysis**

We offer a system that uses empirical theories and terminology in the area of research and construction. The description is called a graphical model, and the path forward is science.

Modelling. Modelling. They describe an empirical concept in particular with respect to AI through a reference from an input x to a result y

f: x expects

This is why the guideline f is obtained from the preparation of knowledge. We use the data collection of knowledge sources s(n)=d1, d2 as well as dN as the preparatory information and we make a standard f because of solo instruction. The data collection for knowledge sources and their outputs s(n)= (d1, c1), fun (dN, cN) is used in the applied research. How should we treat a controlled learning issue as a simple pattern, such as characterizing images of canines and felines. The data collection consists of the detail’s images d1, d2, along, dN and its c1 = cat tags, c2 = dog, along, cN = cat.

## *3.3.1 What is the concept of generativity?*

The conveyance p: s hastap(s) which provides the information s for the preparation is demonstrated at the time when we consider the generative model. The core principle is that the generative model forms the dispersion of probability p with the direction f. Both x and y of f: x are distributed as follows.

x: details about planning s

y: the likelihood that knowledge s will be generated

Because we unambiguously model the likelihood distribution p, we will find the probability p(s). In this case. In these conditions, we will also update the limits of p to improve the likelihood and train the model with the maximum probability estimate. There is some space for the planning process in this sense because you can quickly maximize the probability by streamlining the boundaries. The preparation process can be easy. However, there is a drawback to the fact that we have only the form of assessing the propensity to render a program of inspection.

Throughout all cases, we also only have to check it according to the distribution in. The probability p(s) is used for sample testing separately. We may not display the possibility of diffusion p(s) explicitly once in a while, but specific concentrating variables promote research.

The probability theft p(s) can be modelled in two ways. The primary case is to show the likelihood of p(z) and p(s) conveyance by displaying the vector z inactive. This distinction is provided later on by the VAE. Secondly, we present the idle variable z and models the generator example] s = G(z)] that matches the p(s) conveyor. This section has a location for the GAN. Such models can produce information for the preparation of the inert variable z according to arbitrary numbers.

For following purposes these generative structures may be used:

• Creative activities help (e.g. line drawing shading)

• Availability of individual interfaces (e.g. typical sentences)

• Increasing the expense of generating information (as a research program, for example)

As we speak about generative models, we often describe them in classification problems against biased models. However, it is common to describe it as a probability distribution model producing training data while thinking about GAN generative models.

## *3.3.2 GAN and other generative models Gap*

The generative models can be listed as showné in the following diagram, as explained in the GAN instructional exercise at NIPS 2016.

Some well-known generative structures, in combination with GAN are Fully observable perception networks (FVBNs).

#### 3.3.2.1 Robust Visible Conviction Networks (FVBNs)

The FVBNs deteriorate the probability dispersion p(s) into one-dimensional transports of probability using the following condition of the Bayes hypothesis:

pmodel(s)= file (si file), file(s)= file(s) file(s).

FVBNs may be expected to be an autoregressive model without primary measures provided based on the measurements recently made. The one-dimensional appropriation capacities of Recurrent Neural Networks ( RNN) and Convolutional Neural Networks ( CNN) was modelled independently by the PixelRNN and PixelCNN known as FVBNs. The benefit of FVBNs is that the model will learn with an expressly calculable probability. The challenge is that testing costs can be costly as each metric needs to be successively made.

#### 3.3.2.2 Self-Encoder Variation (SEV)

The VAE model p(s) produces planning information as follows: pmodel(s)= alternatively, alternatively pmodel(s) f) pmodel(s) dz. The two probability dispersions pmodel(s) and pmodel(s) using inert vector z are shown in both the potential dispersion z component(s).

When a model is to be generated using the highest likelihood approximation of s(n), it should be expressly described by pmodel(s) or pmodel(s). Nevertheless it is difficult to compute pmodel(s) and pmodel(s) because pmodel(s) recall the critical administrator as seen in the above condition

In this way, VAE approximates pmodel(z) with qmodel(z) with qmodel(z). Therefore, we can measure the lower probability limit and train the model by increasing the lower probability boundary. Inert variables can be measured by qmodel(s). The advantage of VAE is that it's really easy to check. The problem is that the amount of pmodels of probability acquisition is problematic, and incorrect values are used for model planning.

### *3.3.3 How does the GAN stand?*

GAN does not indicate explicitly, like FVBN's and VAE, the probability appropriation p(s) which provides the details for the planning. We build a G: z generator instead. The generator G checks the latent vector z accordingly. We also generate a discriminator D(x), aside from generator G. It identifies the instances from generator G and the actual ones from knowledge planning. Throughout the preparation of the disk D, generator G is also prepared for the reason not to differentiate between the generated tests and the discrimination human. The benefits of GAN are low cost testing and cutting-edge picture generation. The downside is that the probability transport models can not be determined because we do not show a probability circle and the inert variable z can not be deduced from an example. the inert variable z.

#### 3.3.3.1 How does the GAN function?

GAN uses the generator and the discriminator, as explained earlier. That is, we have developed two GAN neural networks.

If the networks are being equipped, we will fit the distribution of samples generated by the actual distribution s = G / z generated by the generator.

 Generator G knows how to disseminate the potential for Nash game equilibrium. In depth, the generator G is often prepared during the preparation of the discriminator D to avoid the discriminator D from discriminating between samples.

The relation between banknote forgers and police is used as a common pattern time and time again. Fake notes, including real banknotes, are sought by the forgers. The police are seeking to distinguish genuine verified receipts from false documents. The

efficiency of the police is expected to expand marginally, so that actual banknotes and counterfeit bills are well understood. The forgers will not be able to use fake banknotes, and they can turn more and more equivalent fake banknotes into real ones. If police increasingly develop the ability to identify authentic and counterfeit bills, they will eventually recognise the forger 's decision to offer a comparable bill if legitimate.

Numerical articulations explain the planning process as follows. In the first case, as the prejudice against D(s) is likely to generate the example s from the true transport, it is generally expressed as follows:

D(s)(s)=p(s)p(s)

When we are organizing the dispersions of examples from true circuitry and of examples from generator G, it means that we need to restrict the variations between the two allocations. While we are organizing the dispersion of examples from real circulation s Pmodel(s) and p(s) DJS can be built using D(s) as follows:

2DJS=== DKL(p(x) the total amount of the logo(x))+DKL(p(x) the total number of loads to be charged in each event.

The discriminator D will be increased to the DJS, and the generator G will be confined by the standard. In comparison, a G(x) dissemination pmodel(s) will organize true p(s) dissemination.

The epmodellog(1−D(x)) minGmaxDEp(x)logD(x)

In preparation of the above min-max problem, the discriminator D(s) and generator G(x) can be refreshed again.

#### 3.3.3.2 What is DCGAN?

An extension of one or more type of DCGAN are all of the above processes. It uses a racist engine and an engine

While Dcgan is a challenging program to introduce, its ability to execute the specified function is made simpler by a number of variables, that's what separates Dcgan from other algorithms that have been in operation for a very long time but only recently became classified as real, usable. The Dcgan model is based not on heuristics but on the generative operation of the generator and on the capacity of the discriminations in tough circumstances to tell the correct from the wrong.

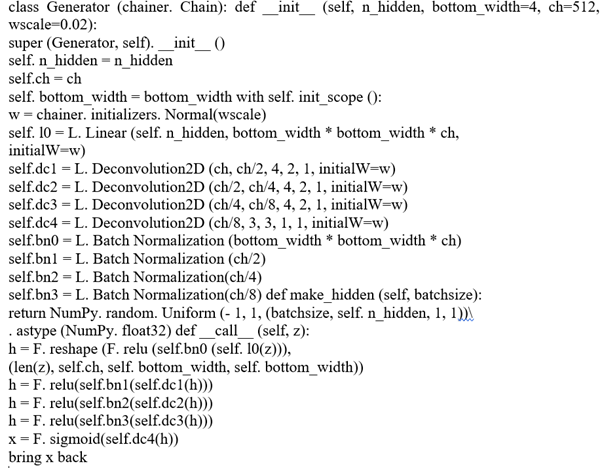
1. Convert completely linked layers of discriminator into regular, regional grouping layers

2. The generator and discriminator using bunch standardization layers

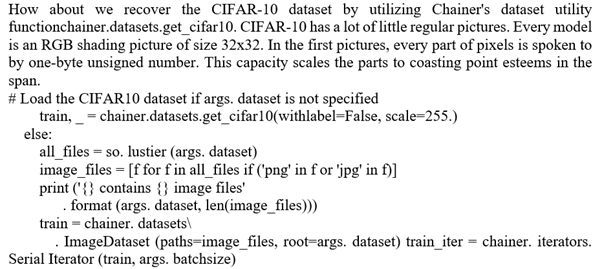
3. Using the capacity of the discriminator to launch the ReLU

### *3.3.4 Implementation of DCGAN in Chainer*

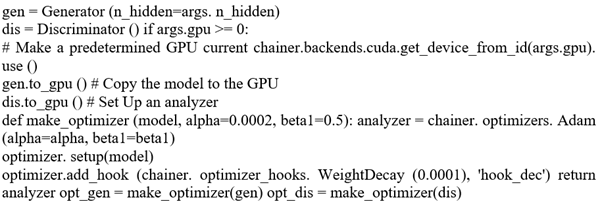
#### 3.3.4.1 Define the generator model



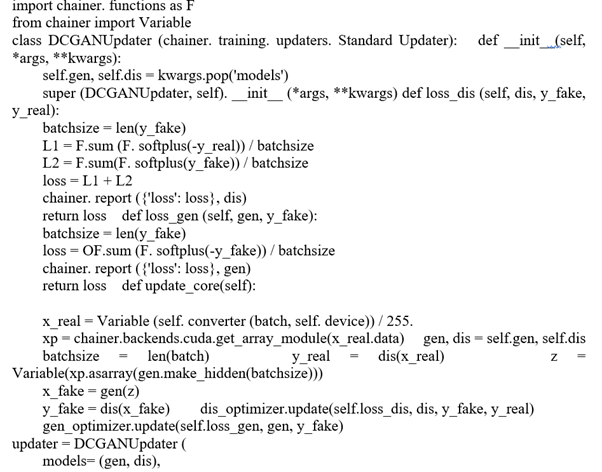
#### 3.3.4.2 Prepare dataset and iterator



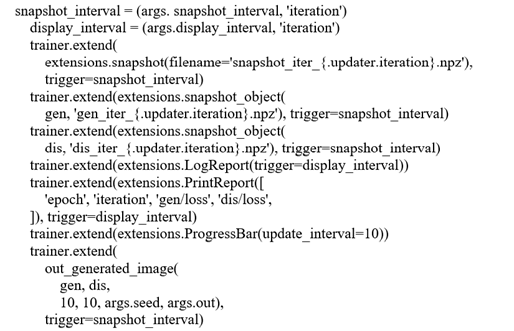
#### 3.3.4.3 Prepare model and optimizer



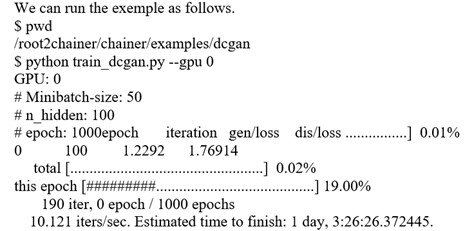
#### 3.3.4.4 Prepare updater



#### 3.3.4.5 Prepare trainer and run



#### 3.3.4.6 Start training



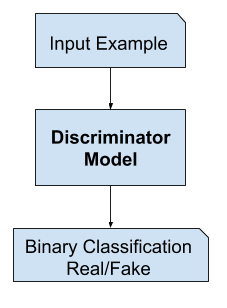
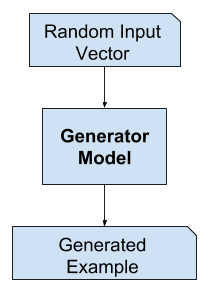
# **3.4 Generator Model**

That's , a inactive space gives a compression or high-level concepts of the watched crude data such as the input information conveyance. Within the case of GANs, the generator show applies meaning to focuses in a chosen idle space, such that modern focuses drawn from the inactive space can be given to the generator demonstrate as input and utilized to create unused and distinctive yield examples.

After training, the generator model is freezed i.e- no weight updates take place in the generator model and it is only used to generate new samples belonging to the output domain.

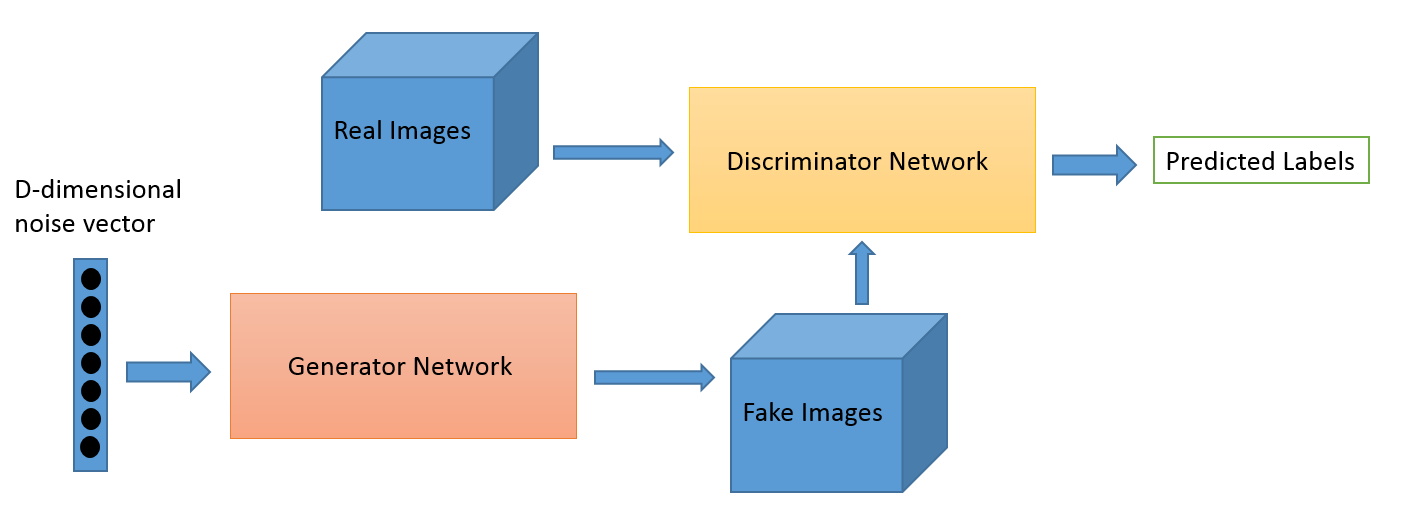
# **3.5  Discriminator Model**

The discriminator model uses a model from the space as info and predicts a binary class label of genuine or forfeit. The genuine model originates from the preparation dataset. The created models are the yields given by the generator model. The discriminator is a conventional model which has been utilized for binaryaclassiﬁcation. Now and then, the generator can be repurposed as it has ﬁgured out how to sufficiently separate features from models in the difficult space. A few or the entirety of the component extraction layers can be utilized in transfer learning applications utilizing the equivalent or comparative input information.



*Fig 7: GAN Generator Model                         Fig 8: GAN Discriminator Model*

# **3.6 Training of GAN**

****

*Fig 9: GAN Schema*

Training of GANs takes place in two phases:

**Phase 1:** In the first phase, Generator is froze while the discriminator is trained. Only forward propagation takes place during this time and no back-propagation is done across the network. Actual data from dataset is used for training of discriminator for n epochs and it’s accuracy of predicting them as real is observed. Batch of generated images is also given to discriminator along with actual data so that it can learn to distinguish between them and improve it’s prediction.

**Phase 2:** Now the discriminator is frozen and only the generator is trained.

These 2 phases are repeated for multiple epochs and then generated data is checked manually to check if it seems genuine to the human eye. If the generator's output are acceptable, then we stop the training, else we continue doing it until we get better results.

# **3.7 Loss function and Optimizers**

## *3.7.1  Loss Function*

Loss function is also known as error or cost functions. Each ML algo or model learns by the way toward optimizing loss function. Loss Function are the fn in ML that help with the assessment of exactness of a given forecast. Loss function work gives high numeric worth, if the expectation is made far away from the real or genuine value. For model to deliver great expectation, it must have low deviation from genuine worth for example low loss. We use some optimization techniques like gradient descent algorithm, to reduce the loss in our prediction.

There are several types of loss function and one should be careful before selecting loss function for its machine learning algorithm. There are several factors that govern the selection of Loss function for your problem like, algorithm you are using, ease of evaluation of probability and derivative, presence of outlier etc.

## *3.7.2  Types of Loss Function*

On the basis of evaluation of machine learning models, i.e. classification or regression, the loss functions can also be divided into two types: classification or regression loss function.

#### 3.7.2.1 Hinge Loss

HingeaLossaisaaalossafunctionathataisausedaforatheatraining classifier models in machine learning. More precisely, it is used for maximum-margin classification algorithm i.e, SVM.

Ly=max⁡(0,1-t.y)

where ‘t’ is the intended output and ‘y’ is the classifier score.

Hinge loss is convex function but is not differentiable which reduces its options for minimising with few methods.

#### 3.7.2.2 Cross Entropy Loss

Cross entropy loss quantifies the likelihood forecast of a classification model whose yield is a likelihood esteem somewhere in the range of 0 and 1. Cross-entropy loss increments as the anticipated likelihood separates from the true value or realalabel. The log loss of a perfect model would be 0 and it’s high value suggests high error in our predictive model.

The general Mathematical expression for Cross-entropy Loss is,

L= -ac=1MYalog⁡(P)

Where, aM= total number of classes to be classified

Ya=abinaryaindicator (1aor 0)aif the classalabel ‘c’ is correctly classified for an observationa‘o’.

  Pa=aPredictedaprobability for an observationa‘o’ isaof classa‘c’.

For, binary classification, where M=2, cross-entropy can be calculated as:

CE= -(ylogp+(1-y)log⁡(1-p))

#### 3.7.2.3 Regression Loss Functions

##### ***Mean Square Error Loss***

Mean squareaerroraloss or MSE lossais defined asathe average value ofasquared differenceabetween the actual value and predicted value by learning model in regression.

Mathematically it is given as,

MSE= i=1n(Ti-Yi)2n

Where, T is true value i.e. actual value and Y predicted value.

The optimization of MSE is done by using gradient descent algorithm. It is more sensitive to outlier than MAE.

##### ***Mean Absolute Error Loss***

It can be defined as averageaofaabsoluteadifference between the actual valueaand apredictedavalueaby learning model in regression.

Mathematically it is given as,

MAE= i=1n|Ti-Yi|n

Where, T is true value i.e. actual value and Y predicted value.

The optimization of MAE is done by using gradient descent algorithm.

##### ***Mean Absolute Error Loss***

This Loss function is commonly used for regression problems.

Mathematically Huber loss is given as,

loss= {12 (T-Y)2  , if T-Y<δ T-Y-12 δ  , otherwise

Where, δ is set as specific percentile of the absolute residuals i.e. |T-Y|.

## *3.7.3 Optimizers\_and its types*

Optimizers are defined as algorithms which are used to change the parameters such as weights and learning rate of neural network model in order to minimize the losses. Optimization algorithms are responsible for minimizing the losses by optimally changing learning rates and weights of neural network. They help in getting most accurate results possible by making sure that losses are minimum. There are various types of optimizers available.

#### 3.7.3.1 Gradient\_Descent

Gradient descent is most common optimization algorithm. It is an iterative optimization algorithm responsible to reduce the cost function. It's utilized vigorously in classification algorithms and linear regression. It is also used in backpropagation in neural networks. Gradient indicates the direction of increase. In order to find the bottom most point in valley, one need to go other way of the inclination, so we change the learning parameters in the negative direction of gradient in order to minimize the loss. Through backpropagation, the loss is transferred from one layer to another and the model’s parameters also known as weights are modified depending on the losses so that the loss can be minimized.

θ= θ- αJ(θ)

It is easy to understand, compute and implement. However, this algorithm is prone to trap at local\_minima. In addition to it, weights are updated once gradient is calculated for the complete dataset. This will result in long time in convergence to minima if the dataset is large. It will also result in requirement of larger memory in order to compute gradient for the complete dataset. In order to overcome this problem, we use stochastic gradient descent or  mini\_batch1gradient2descent.

#### 3.7.3.2 Momentum\_

In order to tackle high SGD variance, momentum was used.It also smoothes the convergence process. It quickens the convergence leading to essential heading and decreases the variance to the unessential heading. Addition hyperparameter is consumed in this strategy known as momentum represented by ‘γ’. It results in high variance of the parameters and reduced oscillations. It is faster than gradient descent in terms of convergence. However, addition hyper-parameter is required whose selection is done manually and should be accurately choosen.

Vt= γVt-1+αJ(θ)

Weights are changed by- θ= θ-Vt.

Addition term is set usually as 0.9 or comparable value.

#### 3.7.3.3 Adam

Adaptive Moment Estimation or Adam chips away at momentum of first and second order. The nature behind the adam is that we would incline toward not to roll so brisk since we can jump over the minima, we have to decrease the speed a piece for a wary inquiry. Adam also maintains an exponentially\_decaying average of past\_gradients M(t), in addition to storing an exponentially\_decaying average of past\_squared\_gradients similar to Adadelta.

The strategy is excessively quick and converges quickly. It redresses vanishing learning rate and high variance. Nonetheless, it's computationally exorbitant.

M(t) = first moment i.e, Mean

V(t) = second moment i.e, uncentered-variance of the gradients.

mt= mt1-1t

vt= vt1-2t

We are finding mean of V(t) and M(t). This will resuilt inaL[m(t)] equalatoaL[g(t)] where, aL[f(x)]ais anaexpectedavalue of f(x).

To change the\_parameter-

t+1=t- nvt+ϵmt

Here,

β1 = 0.9

β2 = 0.999

ϵ = 10 x exp(-8)

#### 3.7.3.4 Batch Normalization

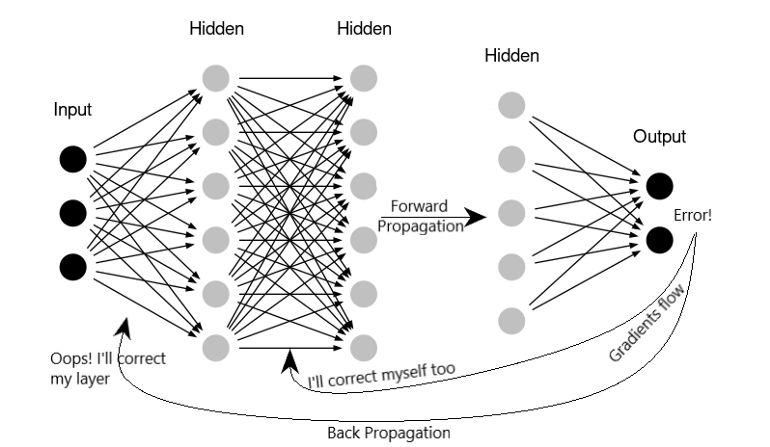
The dissemination of contributions to layers somewhere down in the neural network may change after every mini-batch when the weights are updated. This will lead to learning algoaperpetually pursue a moving objective. At the point when inputadistribution changes, concealed layers attempt to adjust to newadistribution which hinder the preparation procedure. This adjustment in the appropriation of contribution's to layers in the system is alluded as internalacovariateashift.

Batchanormalisation layer is utilized to standardize the yield's of past layers. Batch normalization is a procedure forapreparing deepaneuralanetwork that normalized the contribution's to a layer for every miniabatch. It impacts balancing out the learning procedure and significantly lessening the quantity of preparing ages required to prepare deepanetworks.

Batch Normalization layers has 4 parameters-

1. Gamma weights
2. Beta weights
3. Moving mean
4. Moving variance

Gamma and Beta weights are trainable parameters whereas Moving mean and variance are non-trainable.

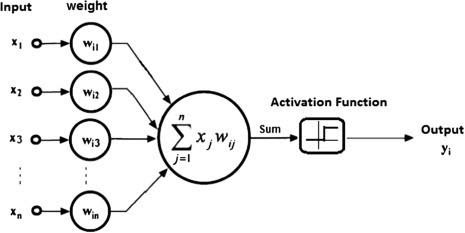


*Fig 10: Batch Normalization*

# **3.8 Activation Function**

Activation Functions are essential for an artificial neural system in learning and comprehending something entangled. They are used to add non-linearaproperties within ouranetwork. It’s primary target is to change over an input sign of a nodeain an artificial neuralasystem to a yieldasignal. That yield signal works as a contribution to the next layer in the stack.

Without activation power, a neuralanetwork would basically be a straightaregression model, which has restricted capabilities and doesn't perform well. We not only need our neural system to learn and estimate a straight fn but to perform something more confounded than that. Likewise without it, our neuralanetwork would not have the option to learn and display other entangled sorts of information, for example, pictures, recordings, sound, discourse, and so forth. This is the motivation behind why we utilize artificialaneuralanetwork procedures, for example, DeepaLearning to understand somethingacomplicated, highadimensional, non-direct, and large datasets, where the model has a few concealed layers in the middle of and has an exceptionally entangled architecture which causes us bode well and concentrate information from such complicatedly enormous datasets.



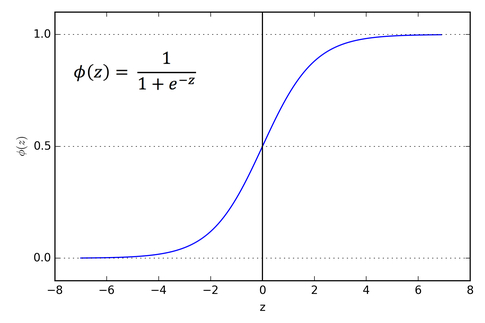
*Fig 11: Single Layer Perceptron Model*

A neuron(node) of an artificial neural network takes linear combination of input from neurons of previous layer, and then applies activation function to generate final output. These three functions are commonly used activation functions.

## *3.8.1 Sigmoid Activation Function*

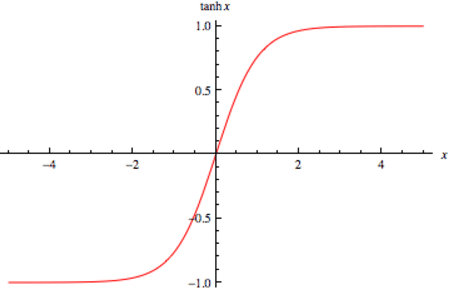
The sigmoid function will just give positive numbers somewhere in the range of 0 and 1. The sigmoid activation work is generally helpful for preparing information that is additionally somewhere in the range of 0 and 1. It is one of the most utilized activationaFn.

Sigmoidafunction is most popular and used widely.

******

*Fig 12: Sigmoid function*

## *3.8.2 Tanh Activation Function*

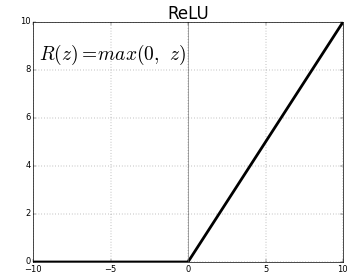
******

*Fig 13: Tanh Function*

Tanh function is similar to sigmoid. It is also nonlinear, thus we can add up multiple layers. It will undoubetly always remain in range (-1, 1). Althoughagradientaisastronger foratanh thanasigmoid asatheaderivatives areasteeper. Tanh Function also has the vanishing gradient problem similar to sigmoid function.

## *3.8.3 ReLu Activation Function*

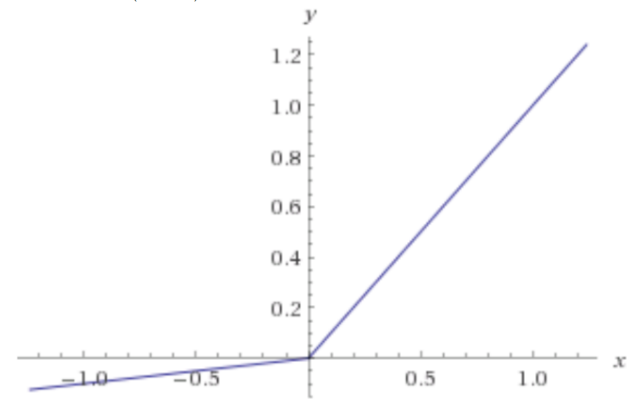
ReLU or the Rectified Linear Unit activation function is a very simple function that outputs 0 for any z<0, and is a simple linear function with slope of 1 ,R(z)=z for any z>=0. It still has the nice properties of a differential monotonic function, but at least for positive weight input values, it escapes the problem the sigmoid function has. Unfortunately, for negative input values - it still has the same issue, as all values are mapped to 0.



*Fig 14: ReLU Function*

## *3.8.4 Leaky ReLu Activation Function*

The disadvantage of using ReLU is that when input was less than zero, output was zero. This is referred to as dying ReLU. AaReLUaneuron is called deadaif it’sastuckainanegative range andaalways gives zero as output. Dying problem happens either due to large negative biases or due to large learning rate.



*Fig 15: Leaky ReLU*

Leaky ReLU has small slope for negative values. So the output is small value for negative side but never zero. Thus, it fixes the problem of dying ReLU.

# **3.9  DCGAN Architecture and Training**

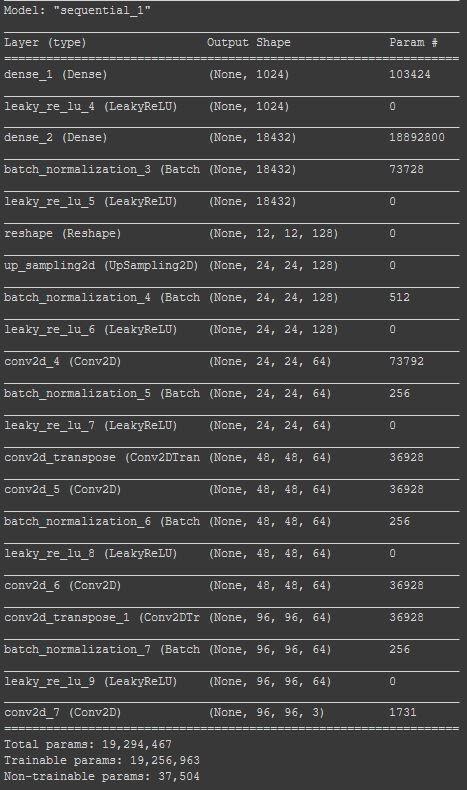
Let us  take a look at architecture of the Generator network:

* The 1st layer is a Dense layer with 1024 neurons.
* The 2nd layer is ‘Leaky ReLu’ as an activation function.
* The 3rd layer is a Dense layer with 18432 neurons.
* The 4th layer is Batch Normalization
* The 5th layer is ‘Leaky ReLu’ as an activation function.
* The 6th layer is Reshape and the resulting image is 12 x 12 with 128 kernels.
* The 7th layer is Up Sampling 2D and the output is 24 x 24 with 128 kernels.
* The 8th layer is Batch Normalization with momentum as 0.8.
* The 9th layer is ‘Leaky ReLu’ as an activation function.
* The 10th layer is Convolution 2D with 64 kernels each of size 3 x 3 and padding as ‘same’.
* The 11th layer is Batch Normalization with momentum as 0.8.
* The 12th layer is ‘Leaky ReLu’ as an activation function.
* The 13th layer is Convolution 2D Transpose with 64 kernels each of size 3 x 3, stride as 2 and padding as ‘same’.
* The 14th layer is Convolution 2D with 64 kernels each of size 3 x 3 and padding as ‘same’.
* The 15th layer is Batch Normalization with momentum as 0.8.
* The 16th layer is ‘Leaky ReLu’ as an activation function.
* The 17th layer is Convolution 2D with 64 kernels each of size 3 x 3 and padding as ‘same’.
* The 18th layer is Convolution 2D Transpose with 64 kernels each of size 3 x 3, stride as 2 and padding as ‘same’.
* The 19th layer is Batch Normalization with momentum as 0.8.
* The 20th layer is ‘Leaky ReLu’ as an activation function.
* The final layer is Convolution 2D with 3 kernels each of size 3 x 3, padding as ‘same’ and activation function as ‘tanh’.

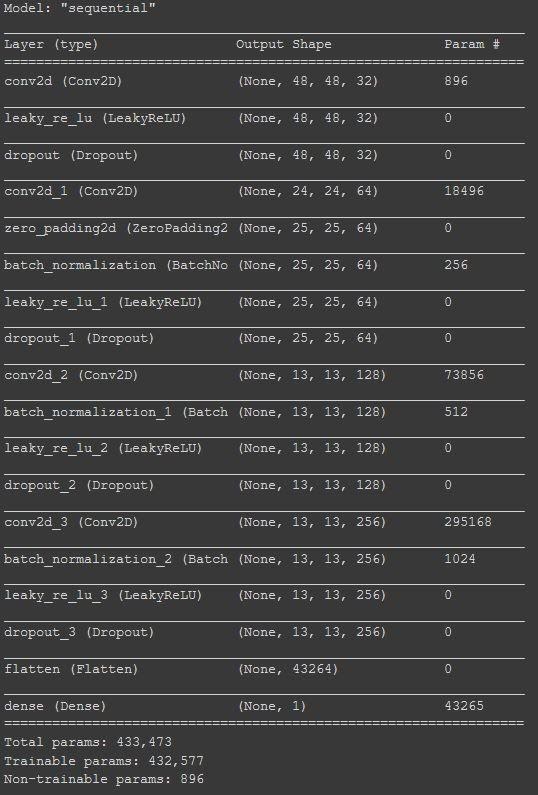
Now let us take a look at architecture of Discriminator network:

* The 1st layer is Convolution 2D with 32 kernels each of size 3 x 3 and with a stride of 2.
* The 2nd layer is ‘Leaky ReLu’ as activation with alpha=0.2.
* The 3rd layer is Dropout with a rate of 0.25.
* The 4th layer is Convolution 2D with 64 kernels each of size 3 x 3, stride 2 and padding as ‘same’.
* The 5th layer is Zero Padding2D with top and left padding as 0 and bottom and right padding as 1.
* The 6th layer is Batch Normalization with momentum as 0.8.
* The 7th layer is ‘Leaky ReLu’ as activation with alpha=0.2.
* The 8th layer is Dropout with a rate of 0.25.
* The 9th layer is Convolution 2D with 128 kernels each of size 3 x 3, stride 2 and padding as ‘same’.
* The 10th layer is Batch Normalization with momentum as 0.8.
* The 11th layer is Leaky ReLu’ as activation with alpha=0.2.
* The 12th layer is Dropout with a rate of 0.25.
* The 13th layer is Convolution 2D with 256 kernels each of size 3 x 3, stride 1 and padding as ‘same’.
* The 14th layer is Batch Normalization with momentum as 0.8.
* The 15th layer is Leaky ReLu’ as activation with alpha=0.2.
* The 16th layer is Dropout with a rate of 0.25.
* Now we flatten the above output and feed it to a fully connected network containing 1 hidden layer followed by an output layer. The hidden layer contains 43264 neurons with activation function as ‘sigmoid’. The last layer is the output layer which gives us only one output because our model is based on one class classification.

Following is the architecture summary for Generator network.



*Fig 16: Generator architecture layerwise summary and parameters*

Following is the architecture summary of Discriminator network.

*Fig 17: Discriminator architecture layerwise summary and parameters*

The DCGAN training involved three steps:

1. Generating fake fingerprint images by passing gaussian noise into generator and updating binary entropy loss with Adam optimiser and learning rate = 0.0002 using stride over pooling

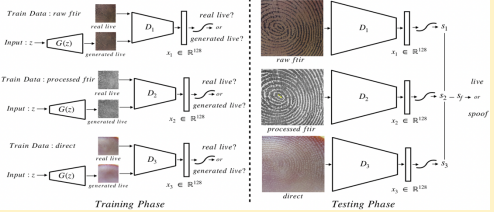
2. Then using the dataset having only real fingerprints and generated fingerprints as input to the discriminator and a sigmoid output giving value between 0 and 1.

3. In testing phase removing the generator and using only discriminator to give the sigmoid output.

4. Generator is using Deconvolution and having an output of image of 64\*64 and   Discriminator is just a mirror network of Generators.

5. Initially in DCGAN training we freeze the weight updating of Discriminator and only our generator get trained hence getting synthesised images and saved them.

6. Then using real and generated images to train the discriminator again



*Fig 18: Generator and Discriminator Training Model*

• Architecture discriminator had 5 convolution layers each having 5x5 filter and stride of 2, an average pooling layer, and two fully connected layers (128-dimensional for feature representation, followed by 1-dimensional for sigmoid classification layer). Every convolution layer is followed by Leaky Relu activation.

• Initially batch normalization brought some instability which was improved by group normalization

• We trained our GANS with a batch size of 64, a learning rate of 0.0002, and the Adam optimizer.

• Here to stop training we used spoof data for validation to determine when to stop training or tune hyper – parameters.

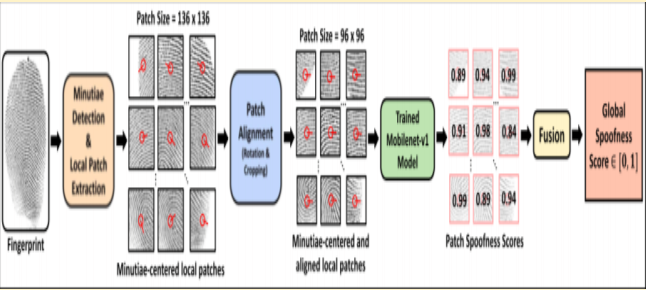
# **3.10  Minutiae Extraction Centered Patches Technique**

Few of the restrictions of numerous of the distributed spoof anti-strategies is their weak generalization execution over spoof material. when a spoof locator is assessed on parodies created utilizing materials that were not seen amid preparing, possibly there could be a 3-fold increment within the parody discovery blunder rates. To generalize an algorithm’s viability over spoof creation materials, called cross-material execution, a few ponders have drawn nearer spoof discovery as an open-set problem. Some Impediment for unique finger impression parody buster utilizing AlexNet and VGG is :

(i)Rectangular resizing picture of estimate, to a square say r\*t, suppose m × m, which results in different type of data held within the two spatial picture dimensions;

(ii) scaling back an picture, in common, leads to a significant loss of unfair information. It is important to consider different sources of commotion included within the spoof creation prepare itself that can present a few artifacts, such as lost friction ridge locales, splits, discuss bubbles, etc., within the spoof.

The essential result of such artifacts is the creation of spurious particulars within the unique mark pictures detected from spoofs. We utilize this perception to prepare a two-class CNN utilizing nearby patches around the extracted particulars, as contradicted to the total unique mark pictures or haphazardly chosen neighborhood patches,to plan a unique mark spoof detector.



*Fig 19: Minutiae Based Extraction*

## *3.10.1 Implementation :*

#### 3.10.1.1 Image Augmentation :

Image augmentation makes a bigger assortment and sum of preparing information which makes a difference in refining our classifiers and once more, battle over-fitting. We expanded the pictures by means of a two step handle:

(1) perform flat flip and

(2) trim five smaller covering pictures from both the first picture and its flipped duplicate. These five pictures come separately from the four corners and from the middle of the picture coming about in a add up to of ten modern pictures for each unique test. We chose this procedure since the perfect preprocessing arrange ought to incorporate trimming and dispense with picture rotation, hence canceling a few changes of the conventional augmentation techniques.

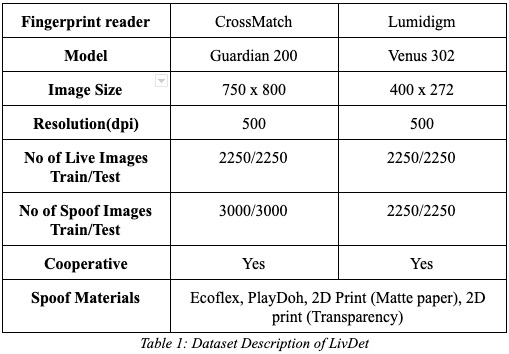
#### 3.10.1.2 Model Implementation

We utilized the TF-Slim library12 utilization of the MobileNet-v1 designing. The ultimate layer of the plan, a softmax layer of 1000 units (at first arranged to predict the 1, 000 classes of ImageNet dataset), was supplanted with a SOFTMAX LAYER with 2 units for the two-class issue, i.e. live vs. parody. The optimizer utilized to get ready the organize was RMSProp with strange angle plunge and a clump degree of 100. Data increment techniques, such as brightness change, sporadic altering, right angle flipping are utilized to ensure

 the prepared illustrate is solid to the conceivable assortments in interesting check pictures.

**4. DATASET AND LIBRARIES USED**

The dataset that we have utilized is MSU Unique mark Introduction Assault Dataset (MSU-FPAD). It was compiled by Michigan State University. Images were captured utilizing two readers: CrossMatch Gatekeeper 200 Lumidigm Venus 302 The determination of captured picture is 500 ppi Spoof creation materials utilized are: ecoflex, playdoh, 2D prints on matte paper and 2D prints on straightforward film It contains 9000 live pictures and 10500 fake images Also utilized Sokoto Coventry Unique finger impression dataset(SOCOFing) - made up of 6000 pictures from 600 African subjects. It contains data almost sex , hand , finger and have changed pictures for 3 diverse levels of change such as obliterations , z-cut , central revolutions . Utilized it for preparing and validation.



**LIBRARIES USED:**

Keras is used for implementing the one class classification dcgan spoof detection and minutiae based extraction. Pytorch has been used for synthesizing fake images in GAN experimented with MNIST data.

Data Analysis Strategies In SOCOFing dataset they have altered

fingerprint images with a strange toolbox over 500dbi resolution and

settings easy, medium, hard giving total images of 55734 altereds

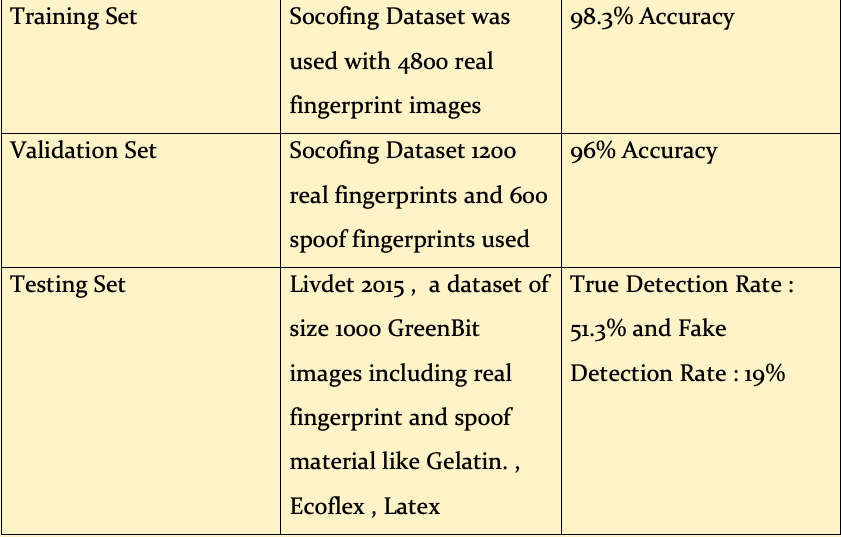
images of size 1x96x103. Then minutiae extraction algorithms are used to extract patches

1. **RESULTS**

After implementing one class classifier over the dataset mentioned. we got the following results:

The sigmoid output of the discriminator gave the spoof score for the input image and hence over the complete testing dataset the spoof score were measured and the average was taken. Gans work well for materials which are anomalous such as playdoh and gold fingers.

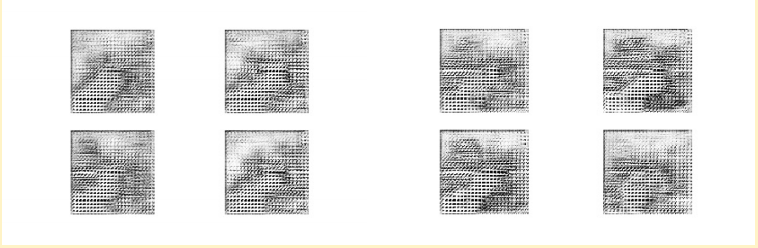
The sigmoid output of the discriminator gave the spoof score for the input. Gans work well for materials which are anomalous such as playdoh and gold fingers. And True Detection Rate was 51.2% as compared to the previous work where it has been 49% only.



*Table 2: Dataset Distribution and Accuracy Result*

Hence our model is discriminative enough to classify fake and real.

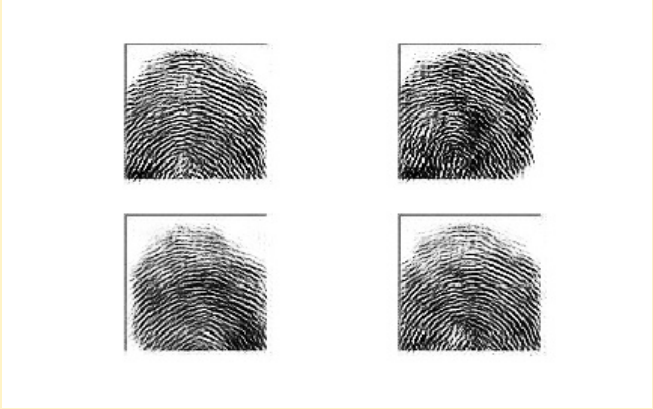
**Generated Images Output:**



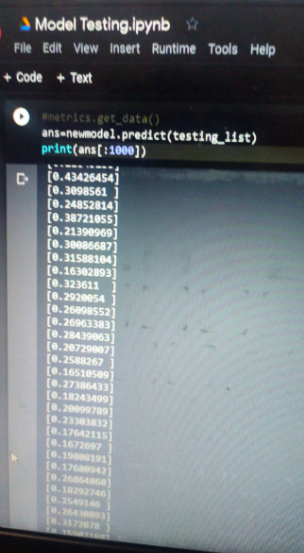
*Fig 20: Starting with Noise*



*Fig 21: Intermediate images better with epochs*



*Fig 22 : Fingerprint final*



*Fig 23 : Testing Results*

# **5.1 COMPARISON BETWEEN TWO APPROACHES**

* In minutiae based technique, using a spoof fabrication material canaintroduce someaartifacts such as missingafrictionaridges, cracks, airabubbles inathe spoofs.
* Thealocalaregionsaaroundathese minutiaeacanaprovide clues to distinguish between live and fake fingerprint. Hence we can use a two class CNN model to predict.
* In one class classification technique, we have used DCGAN’s to detect spoof fingerprints. It makes use of Generator which generates fake fingerprint images.
* These fake images along with real images are sent to Discriminator to classify them during training stage.
* In testing we only use Discriminator which outputs a probability of whether the image is fake or real

**6. FUTURE WORK AND CONCLUSION**

* We improved the problem of spoof detection through one class classification using DCGANs which require large data but at the same time eliminates poor generalization to unknown spoof materials.
* There is still some improvement that can be made. GANs generally struggle to distinguish live finger from spoofs made out of transparent material.
* This is because most of the live finger is visible through the spoofing material.

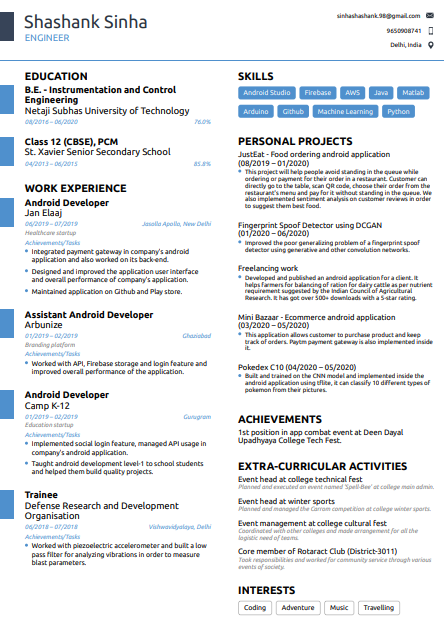
**7. REFERENCES**

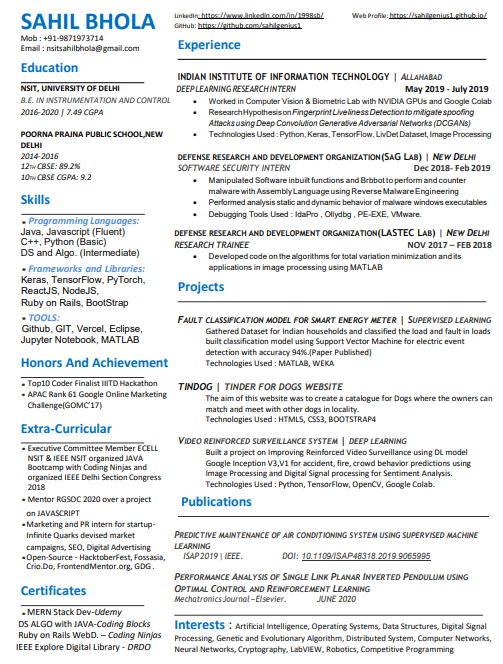
1. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

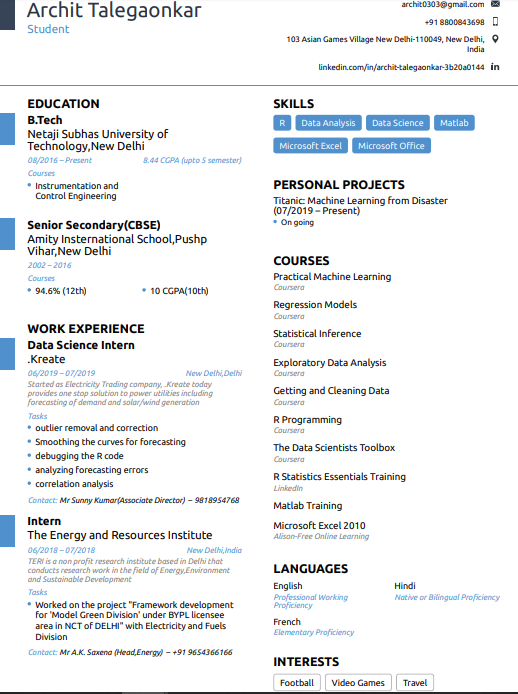
2. T. Chugh, K. Cao and A. K. Jain, "Fingerprint Spoof Buster: Use of Minutiae-Centered Patches," in *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2190-2202, Sept. 2018.

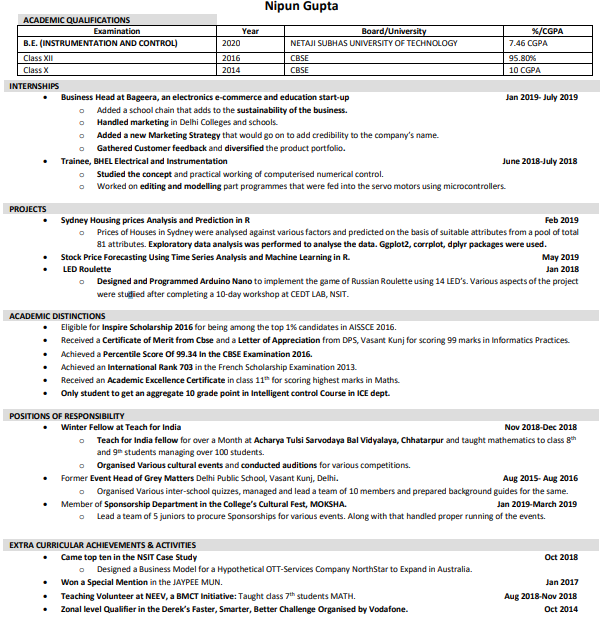
1. J. J. Engelsma and A. K. Jain, "Generalizing Fingerprint Spoof Detector: Learning a One-Class Classifier," *2019 International Conference on Biometrics (ICB)*, Crete, Greece, 2019, pp. 1-8.
2. YI Shehu , A Ruiz-Garcia, V Palade , A James . “Sokoto Coventry Fingerprint Dataset”. ArxiV. Preprint [arXiv:1807.10609](https://arxiv.org/abs/1807.10609), 2018.
3. Y. Ding and A. Ross, “An ensemble of one-class svms for fingerprint spoof detection across different fabrication materials,” in IEEE International Workshop on Information Forensics and Security, pp. 1–6, 2016
4. D. Sinha and M. El-Sharkawy, "Thin MobileNet: An Enhanced MobileNet Architecture," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York City, NY, USA, 2019, pp. 0280-0285, doi: 10.1109/UEMCON47517.2019.8993089.
5. D. Gragnaniello, G. Poggi, C. Sansone, and L. Verdoliva, “Local contrastphase descriptor for fingerprint liveness detection,” Pattern Recognit., vol. 48, no. 4, pp. 1050–1058, 2015.
6. A. G. Howard et al. (2017). “MobileNets: Efficient convolutional neural networks for mobile vision applications.”
7. V. Mura, L. Ghiani, G. L. Marcialis, F. Roli, D. A. Yambay, andS. A. C. Schuckers, “Livdet 2015-fingerprint liveness detection com-petition 2015.,” in BTAS, pp. 1–6, 2015.

**8. About Authors (CV)**

****

****

****

****