Deepfake Image Detection Using Deep Learning

Harshada Kiran Patke

Submitted for the Degree of Master of Science in

Machine Learning

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

August 31, 2022

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 11718

Student Name: Harshada Kiran Patke

Date of Submission: 07 September 2022

Signature:

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Chris Watkins who has shaped, guided and refined my work through this experiment. His subject expertise, his intuition on the areas to explore and his patience as a teacher played a major part in making this project.

Abstract

The rapid progress in Generative Adversarial Networks (GANs) and other deep learning-based techniques of deepfake creation, results into a loss of trust in digital media. Deepfake applications can create fake images, videos, audio or even fake news to intentionally spread misleading information, this can cause harm to a person's privacy, and identity and further can cause a threat to national security. Therefore, this is an immediate challenge we are facing as a community to find the real truth of online material. The main motivation behind this project is to detect the deepfake images which are generated by GANs. More specifically, the proposed project focuses on constructing deep learning based Convolutional Neural Networks (CNN) for classifying real and fake images. Convolutional Neural Network is capable of extracting statistical information from high dimensional image data and allowing to perform tasks on images like image classification, segmentation and object detection. In this project, we are using NVIDIA's Style-GAN generated fake images and real images from Flicker available online on Kaggle to effectively train the deep neural network constructed using CNN in order to detect deepfake images accurately. Furthermore, we are using transfer learning architectures called DenseNet, and InceptionNet for the same task of deepfake detection and comparing performance with a Custom deep neural network.

Table of Contents

1. Introduction

With the help of development in digital image processing techniques, Generative Adversarial Network (GAN) and various deepfake techniques like alteration to images can generate fake images or videos that are too difficult for the human eye to detect. This advanced technique examines the face style, hairstyle, eye color, facial expression, and movements of a person's body and applies it to the image of another person. Deepfake techniques require images or videos in a large amount for training deep learning models so it can result in realistic images or videos. Celebrities or politicians have a lot of images available on websites so that's why they become the first target of the deepfakes. In 2017, first, deepfake video was created by using celebrity images and was replaced with face of porn actor. It is frightening and dangerous to someone's security if deepfake models are used to spread fake images or fake news for misleading information. Deepfakes can be used to bring tensions between politics, religion, countries, or in elections to fool the public and affect the results of the voting. Deepfake techniques can be used to generate fake satellite images of the earth to show objects which don't really exist in that place to the confused military, such as showing a fake bridge over the river this can cause a battle between the countries [31,32].

There can also be a positive influence of deep fakes such as digital avatars, Snapchat filters, or generating voice for those who have lost or can be making of episodes without reshooting them [33]. Deepfakes also have creative and productive use in the media industry such as creating dubbing videos for foreign movies, and virtually trying on cloths. But number of negative uses of deepfakes is more than the positive uses. Recent developments in deep learning and the software industry made it possible for a non-technical person to manipulate and create a fake image with less and fewer efforts, for example, DeepNude software is the dangerous threat it can transform a person's image into a porn image [32]. Therefore, any ordinary person can become a victim of this. These methods of creating a misrepresentation of images violate someone's privacy and identity.

Therefore, finding real truth in the digital world has become very important and also challenging when dealing with deepfake images or videos, because deep fakes are created using deep learning methods. So, for detection also we can take advantage of deep learning methods like neural network or convolutional neural network, or discriminator network which is part of GANs network used for deepfake generation. So, the main objective of this project is to build a strong convolutional neural network that is computationally efficient, robust, and precisely detects deepfake images and also another objective of this project is to build better discriminator which can help GANs to differentiate real vs fake images in order to make the fake image more realistic.

In this project, we proposed a Custom convolutional neural network architecture capable of extracting statistical information from high-dimensional input data, which is trained on GAN generated deepfake images and real images of the human face so it can classify real and deepfake image accurately. And furthermore, we aimed to compare Custom CNN architecture with already existing and robust CNN architectures which are already trained on large amounts of images like DenseNet and InceptionNet by using the transfer learning approach.

2. Background Research

2.1 Deepfake

Deepfake is a combination of two words "deep learning" and "fake". Deep learning is a subset of machine learning based on artificial neural networks and it is getting used to make fake or forged images or videos. It's a kind of media manipulation approach such as changing facial expression, hairstyle, eye color, or speech in order to spread misleading information. Deepfake uses Convolutional Neural Networks and some advanced deep learning techniques like Generative Adversarial networks (GAN), Style GANs, Deep Convolutional GANs (DCGAN), and encoders and decoders to create fake images and videos called synthetic media which are very similar to original images and videos and also difficult for the human eye to differentiate between original and fake. Some traditional approaches like visual effects and modifications use computer graphics to create deepfake images or videos.

2.2 Convolutional Neural Networks

A Convolutional Neural network or CNN is one of the methods of deep learning neural networks. The main intention behind this design is to process the matrix or arrays of data such as image pixel data. Its widely get utilized in computer vision applications that deal with images. CNN is able to capture the pattern between the image data like lines, circles, shapes, and even faces. CNN directly deals with raw input image data without any Preprocessing. As shown in the below image, CNN is a multilayer feedforward neural network that is made up of many hidden layers and is followed by an activation function that's how it becomes capable of memorizing shapes.

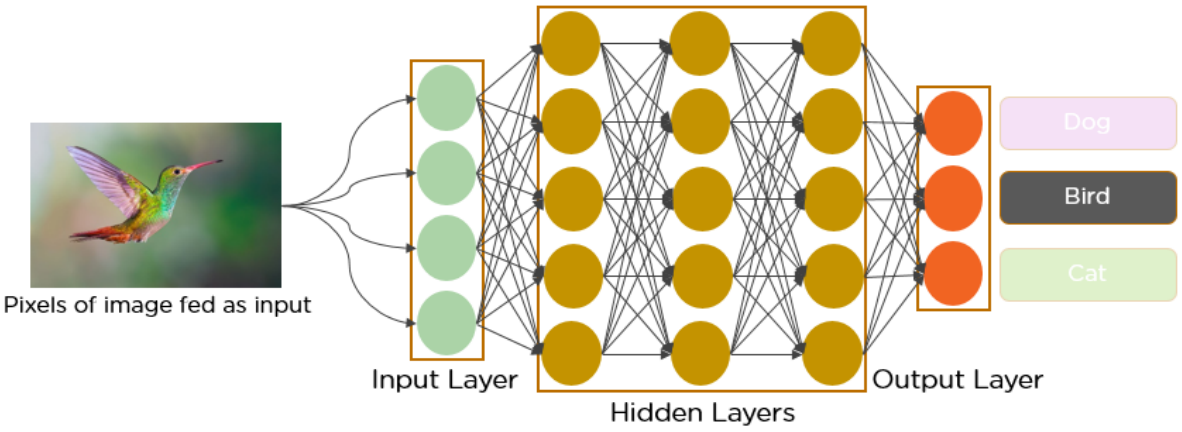


Figure 1. Convolutional Neural Network [29]

The above-shown structure of CNN is similar to the human eye's visual cortex which processes the images and gradually identifies complex features. And this CNN is used for image classification and text classification in Natural language processing; hence, we are using it for classifying images as real or fake in the proposed methodology.

2.2.1 BatchNormalization

This is the training technique of a deep neural network. While training a neural network with multiple layers can cause a change in the distribution of input to the layers and it gets changed after every mini-batch when the weights are getting updated. This type of change in the distribution of input on a neural network is called "Internal covariate shift" in technical terms[22]. In the batch normalization process, it performs the normalization to inputs after execution of every mini-batch, which gives the effect of stabilizing the learning process and also reduces the number of epochs required to train the neural network.

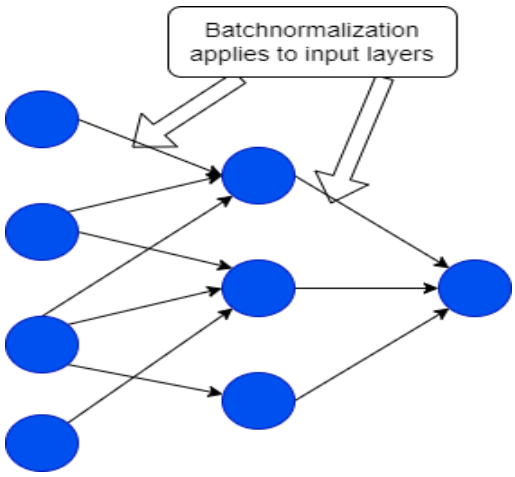


Figure 2. BatchNormalization

Batch normalization is implemented by calculating the mean and standard deviation of input neurons to layers every mini-batch and using the value of this statistic to apply the standardization process, which in turn stabilizes the input values[22].In my custom CNN architecture, I have applied it before each convolutional layer so it can feed the stable distribution of input to the neural network.

2.2.2 Convolutional Layer

The Convolutional layer is the core building block of CNN, most of the computations is performed in this layer. It needs input data, filter, and output which is called a feature map. The typical filter size is (3 x 3) used at the practical level but can vary in size. This filter is applied to input image data which is the calculation of the dot product between the input image data and filter and the result of this dot product is called a feature map.

3	4	2	7	9
5	6	1	5	3
8	7	4	6	2
3	9	8	7	4
2	4	6	6	2

input (5 X 5)

*

0	1	2
3	5	0
8	4	2

filter
(3 X 3)

=

0	4	4
15	30	0
64	28	8

featuremap
(3 x 3)

The above calculation is shown a 1-dimensional matrix, but the real input image matrix is having three channels(RGB), so the input image 3-dimensional. The values In the filter matrix is called weights it remains constant, and it moves across the input matrix by shifting the stride value.

In Custom CNN architecture we have applied the ReLU activation function on top of the feature map to achieve nonlinearity. In the convolution, the number of filter channels must be the same as the number of input image channels, the same process is shown in below image, and on top of this convolution operation, ReLU activation is added with bias value and store it as output.

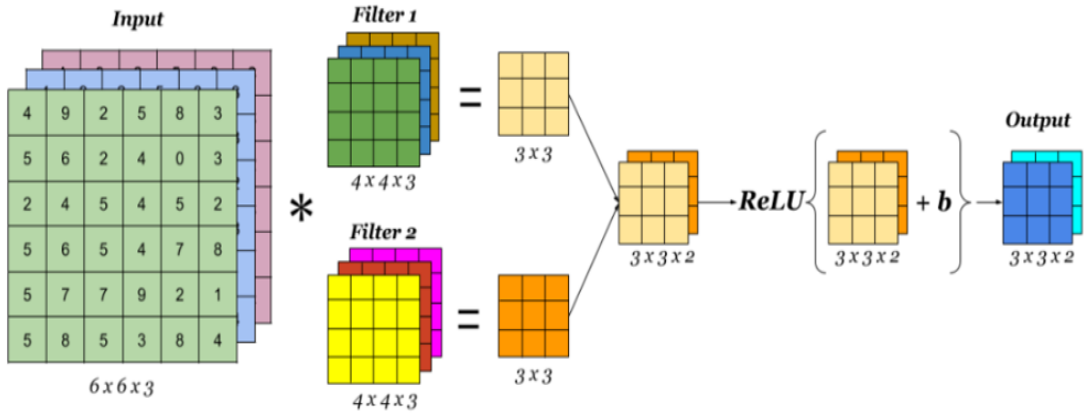


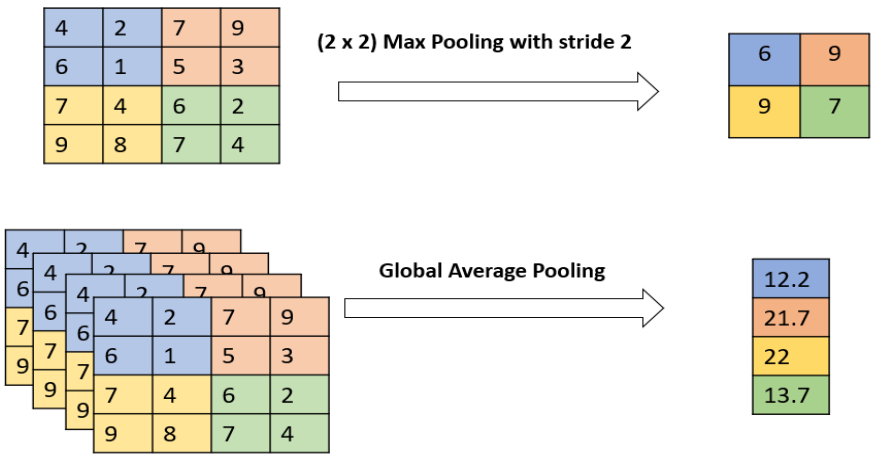
Figure 3. Convolutional layer with activation function[23]

2.2.3 Maxpooling and Average Pooling

This is the downsampling technique, the height and width of the image will get reduced to half of the original size. There are four types of pooling :

- Maxpooling
- Global Maxpooling
- Average pooling
- Global Average pooling

In custom neural network architecture, I have utilized the benefits of max-pooling and global average pooling. As the word says, Maxpooling, It takes out the max value of their region. For implementation, we need pool size and a stride by which we shift the pool.



As shown above, It performs the compression operation on every 4 pixels and maps it to a new single pixel in a (2 x 2) grid. And for global average pooling, it calculates the average of each channel or feature map(if the output is forwarded from the last layer), the advantage of global average pooling is, instead of adding a fully connected layer to the feature map, it takes avearge of each feature map and resulting into single dimensional vector. In general, pooling is the process of reducing the size of the input image by half, and also it able to capture the variance of the image.

2.2.4 Activation function

The Activation function is a mathematical equation, which is responsible for nonlinearity in neural networks and makes them to learn the complex pattern in data. As the name says activation, it activates the neuron by setting some threshold value on output. Basically, the activation function switched the neuron ON or OFF. In our proposed CNN architecture we have two types of activation functions:

2.2.4.1 ReLU activation

It was proposed by Nair and Hinton 2010 and is the most widely used activation function for the deep learning area[25]. ReLU stands for rectified linear activation function, its non-linear in nature, the function will output the input directly if the input value is positive otherwise it returns zero as output because of this property, we can say that, ReLU is a linear function for all positive input value and non-linear function for all negative input value. As shown in below graph, z is the value of the neuron on top of it will apply ReLU.

$$a = \max(0 , z)$$

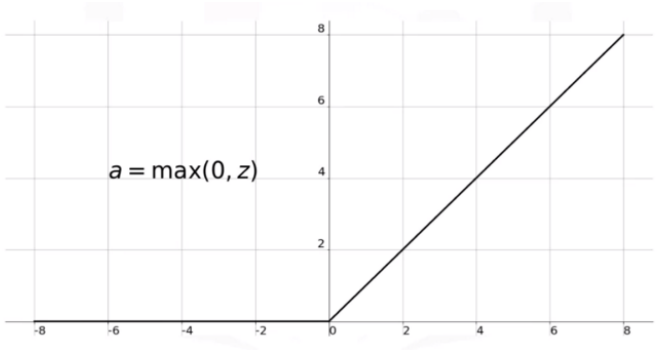


Figure 4. ReLU activation function.

ReLU is purposely used in hidden layers of the neural network and resolves the problem of vanishing gradient descent. Also, ReLU can produce sparsity in the complex model by returning zero value.

2.2.4.2 Sigmoid activation

Sigmoid is a logistic function, which is used in logistic regression or binary classification, Since we are trying to classify an image as real or fake, we are using it in the last dense layer of the proposed CNN architecture. Sigmoid returns the output value between 0 and 1 in S-shaped as shown in the below image[26]. Here z is the value the of neuron and calculating the sigmoid value by applying below sigmoid formula.

$$a = \frac{1}{1 + exp(-z)}$$

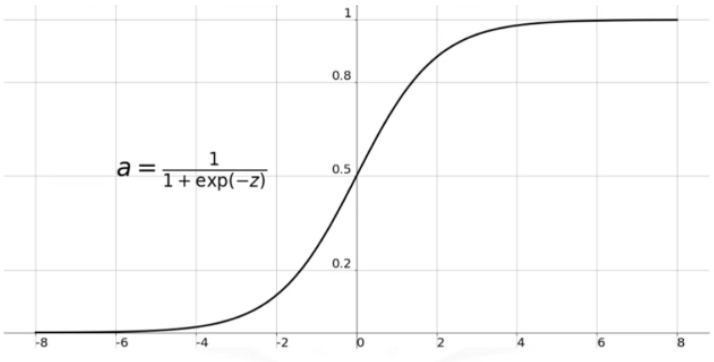


Figure 5. Sigmoid activation function.

Its mostly used in machine learning models where they want to predict the probability of the output class, as sigmoid returns output as 0 to 1it considers them as probability or likelihood that's why sigmoid is became the best option[26].

2.3 Generative Adversarial Networks

GANs were introduced by Ian Goodfellow and other researchers, in 2014. It consists of 2 neural network generators and a discriminator. Generators are responsible for generating fake samples of data by using some noise vector and trying to fool the discriminator network. The Discriminator network on the opposite side tries to differentiate between the real and fake data samples. The Generator and Discriminator both are neural networks and execute to compete with each other in the training process. These steps are repeated multiple times and in this, the generator and discriminator improve their performance after each repetition.

Different types of GANs:

2.3.1 Vanilla GAN

Its simplest form of GANs, consists of two simple multi-layer neural networks, as shown in the below diagram called Generator G and Discriminator D which use noise vector to generate the fake image and discriminator to differentiate between real and fake image and try to optimize the performance using optimizers.

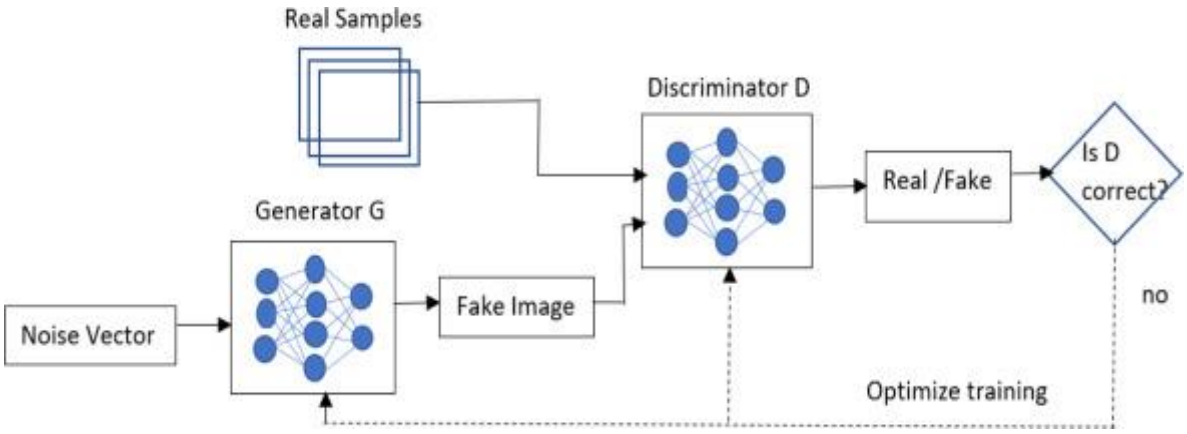


Figure 6. Vanilla GAN Architecture[30]

So GANs are trained in two parts:

In the first part, the discriminator network is trained when the generator network is inactive. The Discriminator is trained on real data for a few epochs and checks if it correctly predicts them as real data and is also trained on the generator's output to correctly predict them as fake. In the second part, the generator network gets trained while the discriminator network is inactive.

And after getting predictions from discriminators, it uses the results to get better performance than the previous epoch and tries to fool the discriminator network. It uses the below loss function in order to improve the performance of generating fake images. The original GAN proposed by Ian Goodfellow can be defined as a contest between two networks G and D.

The min-max objective is formally defined as follows:

$$\min_{G} \max_{D} V(D,G)$$

$$V(D,G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p(z)}[\log (1 - D(G(z)))]$$

Where,

- *G* = Generator
- *D* = Discriminator
- *Pdata(x)* = distribution of real data
- *P(z)* = distribution of generator
- *D(x)* = Discriminator Network
- *G(z)* = Generator Network
- *X* = sample from Pdata(x)
- *Z* = sample from P(z)

Where x stands for the input i.e noise vector, p_{data} and $p_{z(z)}$ are the distribution of real and generated fake data respectively. The above loss function follows the objective of binary cross entropy loss. The output of the discriminator D(x) in between [0,1] because it goes through the sigmoid activation unit. The above min-max optimization equation is famous in deep generative models however this model suffers from an imbalanced training issue of two neural networks.

2.3.2 DCGAN – Deep Convolutional GAN

This is the most famous neural network architecture of GAN, proposed by Radford et al. in 2016, this method is more stable and is called Deep Convolutional GAN(DCGAN)[1]. It is the same as Vanilla GAN, just replacing the two neural networks G and D with some changes. It is built using convolutional layers without the use of a max-pooling layer and a fully connected layer.

Its main focus is on down-sampling and up-sampling techniques by replacing the pooling layer with a strided convolutional layer and transposed convolutional, using Batch normalization and removing fully connected layer with deeper architectures. It uses "ReLU" activation in a generator for all layers except the output layer, which uses the "Tanh" activation function. The below figure shows the Generator network architecture.

As shown in below figure, the architecture of Generator(a) and Discriminator(b) network. The Generator network is responsible for generating images using a d-dimensional vector and using some deconvolutional layers, producing 64 x 64 x 3 images. This image then give it to the discriminator network as input for classification.

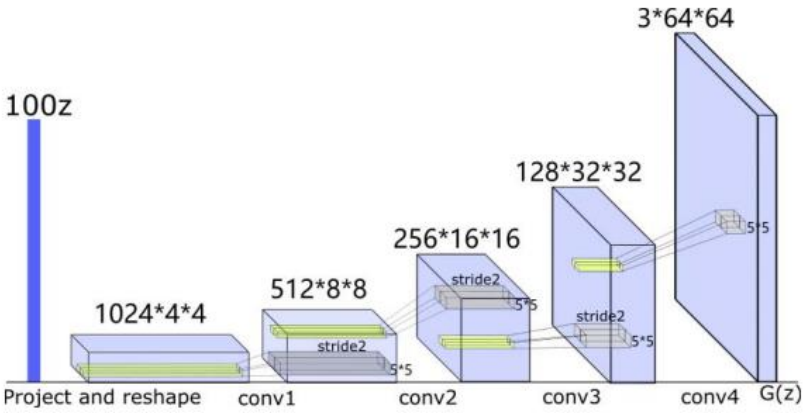


Figure 7. Generator architecture of DCGAN[3]

And on the other side, the Discriminator network follows a similar structure using traditional CNN layers which is responsible for differentiating whether the image is real from a given dataset or fake generated from the Generator network. DCGAN follows the same loss function while training both networks.

$$\min_{G} \max_{D} V(D, G)$$

$$V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p(z)}[\log (1 - D(G(z)))]$$

Here x is the image and z is the noise vector of d-dimension and $p_{data}(x)$ and $p_z(z)$ are probability distributions of x and z. D(x) is the probability of input being generated image from $p_{data}(x)$ and (1- D(G(z))) is the probability of being generated from $p_z(z)$ In GANs, the Discriminator is trained to increase the correct answer rate and the Generator is trained to decrease $log(1- D(G(z)))$ to deceive the Discriminator [2].

In the training process, while optimizing discriminator output, we get maximum $V(D, G)$ and when we try to optimize generator output, we get minimum $V(D, G)$. This optimization problem is displayed in the below formulas:

$$D^*_G = \arg\max V(G,D) \quad (1)$$

$$G^* = \arg\min V(G, D^*_G) \quad (2)$$

Here Generator G understands the data distribution of the sample and generates the sample similar to real training data with noise z which follows distribution like uniform distribution or gaussian distribution and achieves a result as good as the actual sample.The Discriminator calculates the probability of the sample being taken from training instead of data generated. If the sample is real from training data, the discriminator returns an equal probability otherwise discriminator returns a small probability[3].

2.3.3 Style GAN

In traditional GAN, we don't have any control over the styles of the image which is being generated. We only input the noise vector to the generator network and get the image as output from the discriminator network. The Style GAN paper [4] proposed by NVIDIA, the generator network architecture which follows the idea of Style transfer networks. It proposed the generator network in such a way that we can control the image synthesis process. It easily tunes the parameters for the high-level attributes of the image like gender, age, hair color, hair length, glasses, pose, eyes, and facial features.

Style GAN enables the network to generate images by doing scale-specific and interpolation operations. Style GAN has done changes only to the generator network, discriminator network is not modified at any level. In the traditional GAN Network, the noise vector is directly forwarded into the generator module whereas in Style GAN architecture noise vector is forwarded to the mapping network which is 8 fully connected networks before it is going to the generator module.

The main components of Style GAN architecture are:

1) Mapping Network:

As shown in the above diagram, the mapping network has 8 fully connected layers who is responsible for encoding the noise vector Z into latent space vector W which is the combination of a linear transformation and translation. This latent space vector W is then transferred to the Affine transformation module called "A" the output of this module is represented as the letter y in the below formulas. Later this output y along with features of convolution layer (x) are passed as input to the Adaptive Instance Normalization layer.

2) Adaptive Instance Normalization (AdaIN)

AdaIN is a normalization technique that follows the idea of Batch Normalization which is basically used in GAN to improve the performance of discriminator training. And recently getting used in the generator module.

$$BN(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

Batch Normalization computes the mean and standard deviation of x. It has 2 hyperparameters called gamma and beta which perform the role of scale and translation factor. Here in, Adaptive Instance Normalization uses the batch normalization technique for instance normalization which is done for every instance in a batch.

AdaIN Formula:

$$AdaIN(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

Its receives input x from convolutional layers and y from the Affine transformation module and simply assigns channel-wise mean and variance of x to

match those of y[5]. It simply scales the normalized content input with σ(y) and shifts with μ(x). In Short, AdaIN formula applies style transfer techniques on feature space by manipulating features with help of input mean and variance [5].

3) Noise Vector (B)

Noise Vector B is added at different layers of the Synthesis network not just at beginning of the network as we have seen in traditional GANs. Noise vector B is the representation of a single channel image of uncorrelated gaussian noise. It is first added to the output of 3 x 3 convolutional network and further added in AdaIN Instance Normalization.

The aim of this noise vector is to induce stochastic details into generated sample data of the image and using this we can control the variations of the image like combed hair, skin color, eyes color, hairstyle, and beard. This stochastic variation is added in such a way that, it won't affect the main perception of the image.

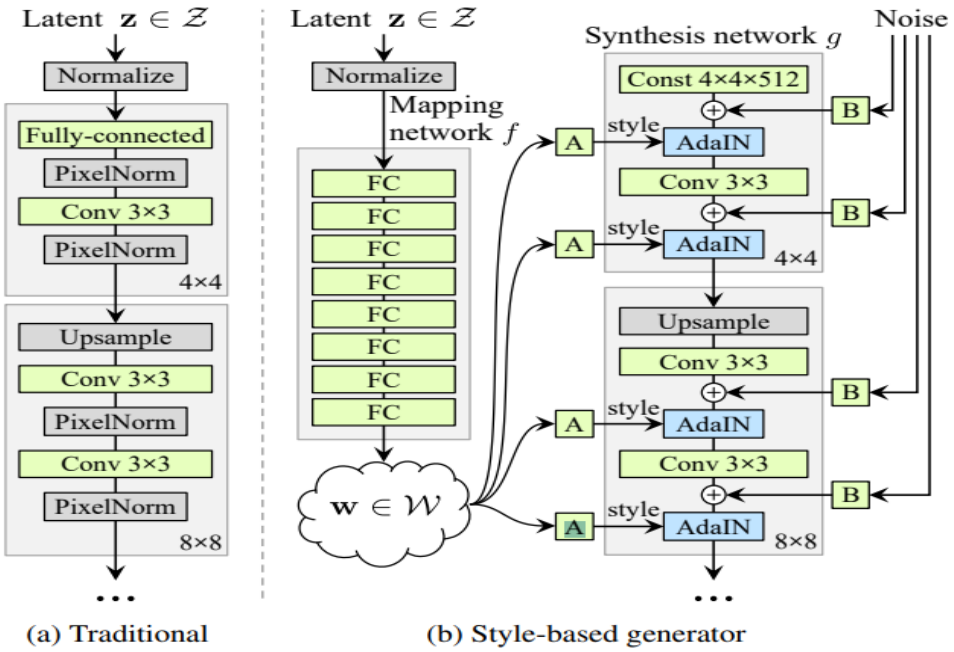


Figure 8. StyleGAN architecture [22]

2.4 Transfer Learning

As the name says, Transfer, It transfers the knowledge of an already trained machine learning model, in order to develop a new model which is different but solves a related problem. In technical terms, we transfer the weights of the trained model, learned at task 1 to new task 2. It uses the knowledge of a model that is learned with a

lot of training data, Instead of starting the learning process from scratch. In our experimental study, we have used a few transfer learning models for classifying the image as real or fake.

2.4.1 ImageNet

ImageNet is dataset which containing 14 million annotated images. In deep learning and computer vision, model needs to be trained on various image dataset in large amount. Deep learning model needs to learn useful features from images while training. After leaning it for one single time, we can use it in task like image classification or image detection. So ImageNet gives pretrained database of images so we don't need collect large collection of images and no need to train it. We can take advantage of readily available resources for training the model.

2.5 Confusion Matrix

It was invented by Karl Pearson in 1904, by the term "Contingency table"[27]. The Confusion Matrix is a representation of the number of correct and incorrect predictions v/s. an actual number of correct and incorrect records. It's used for the performance evaluation of classification models. The confusion matrix shows how many data points are classified correctly and misclassified for all the classes and that's why it is better than calculating only the model's accuracy. The confusion matrix has 4 basic terminologies as shown in below figure:

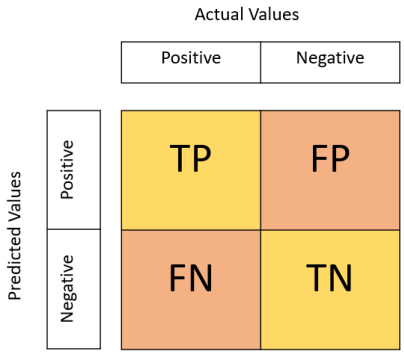


Figure 9. Confusion matrix

- **True Positive (TP):**
  The Classifier model has predicted it as Positive and the true label is Positive.

- **True Negative (TN):**
  The Classifier model has predicted it as Negative and the true label is Negative.

- **False Positive (FP):**
  The Classifier model has predicted it as Positive and the true label is Negative.

- **False Negative (FN):**
  The Classifier model has predicted it as Negative and the true label is Positive.

2.6 Classification Measure

These are the classification measures calculated using confusion matrix.

2.6.1 Accuracy

It measures the correctly predicted records. In short, it calculates the sum of truly predicted classes divided by total predicted values.

$$\text{Accuracy} = \frac{TP + TN}{TP+TN+FP+FN}$$

2.6.2 Precision

It measures the ratio of how many actual positive classes are predicted correctly out of all positive predicted classes. True positive values are divided by positively predicted values.

$$\text{Precision} = \frac{TP}{TP+ FP}$$

2.6.3 Recall

It measures the ratio of how many observations of positive class are actually predicted as positive. Here True positive is divided by a total number of actual positive classes.

$$\text{Recall} = \frac{TP}{TP+ FN}$$

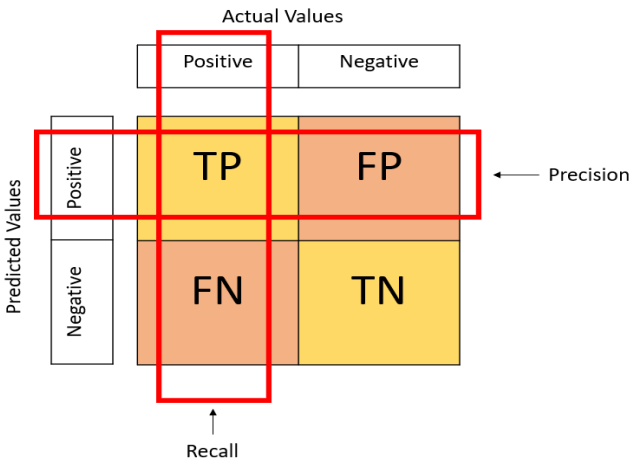


Figure 10. Precision and Recall

3. Literature Review

3.1 Deepfake Creation technique

Deepfakes becoming popular day by day as it creates fake videos with good quality also in few applications provides an option for an edit that is easy to understand for a nontechnical person, and these applications are developed using deep learning techniques. Deep learning is known for playing with complex and high-dimensional data.

The deepfake creation technique is divided into four main categories:

3.1.1 Entire face synthesis

This method is proposed by NVIDIA which implemented powerful Style GAN, able to generate face images of non–existent persons. This method achieved great results by generating high-resolution and high-quality images that look exactly like a real person image. They started with a low-resolution image with less number of layers in the generator and gradually increases the layer and resolution in order to get a high-quality image. This technique helps the video game and 3D modeling industry but is also harmful to creating fake profiles on social media.

3.1.2 Identity swap

This method replaces the face of one person in an image or video with the face of another person. One of the architectures of deep learning that's autoencoders, which are utilized for dimensionality reduction and image compression [19]. The first deepfake application was developed by a Reddit user by using deep learning's autoencoder–decoder pairing structure [19]. In this application, the autoencoder captures the features of the face image and on the other hand, the decoder is used to paint or generate the face image, and used another encoder–decoder pair to swap the faces. This kind of application is harmful because it could be used for pornographic videos and financial fraud and many others.

3.1.3 Attribute Manipulation

This method is used for modification of some attributes of the face like changing the color of hair, skin, gender or adding sunglasses. The majority of this modification process is implemented in StarGAN. In this paper [21] in 2019, proposed a method to generate a new face image with desired facial attribute values without editing other parts of the generated image in simple words only change what you want. Attributes stand for hairstyle, hair color, hairstyle, color skin, age gender, and mustache. In this paper, they introduced a new method called attribute classification constraint and applied it on image generated data to make sure it making correct modifications on desired attributes and worked on attribute excluding details which helps to preserve the original generated image data by this way we can change what we want.

3.1.4 Lip syncing

This category belongs to video manipulation, which synthesizes the lips region of the target object and makes it consistent with any audio clips. In this paper [20] in 2019, Proposed talking face generation. At a high level, face generation and synchronized those generated

image frames with the clip of audio and implemented it using a framework called disentangled Audio-Visual System (DAVS) which was able to generate high-resolution quality video frames by integrating it with an audio clip. They have used deep learning components called GAN, encoder, and decoder. But the limitation of this project is lacking the synthesis of facial expression.

3.2 Deepfake Detection Techniques

Below are the published methods used for forensics evaluation of deep fakes images or videos.

In “Exposing AI Generated Fake Face Videos by Detecting Eye Blinking”, published by Yuezun Li, Ming-Ching Chang and Siwei Lyu in 2018. They introduced method to detect fake face videos generated by the neural network. There method is based on the detection of eye blinking patterns in videos, its one of the features which not well represented in the synthesized fake videos. Using this pattern, they differentiate between real and fake faces. Later in time, this method was overcome by advanced models of deepfake generation which included eye blinking patterns in the training data itself.[6]

In “Exposing Deepfake Videos By detecting face wrapping artifacts ” published by same authors Yuezun Li and Siwei Lyu in 2018. They introduced methods that focus on observations that recent deepfake models generate limited-resolution images which is modified to match the original face in the source video. This kind of transformation shows distinctive attributes in the deepfake-generated video.The authors developed CNN network which can effectively capture the distinctive attributes and is able to differentiate real and fake video[7].

In "Exposing deep fakes using inconsistent head poses" published by Xin Yang, Yuezun Li, and Siwei Lyu in 2018. The authors have revealed another area of focus by using head poses which were the main features to detect the inconsistencies in deepfake images. They have built the system based on the observations that deepfakes are generated using the splicing synthesized face region into the original image and in doing so, introduce errors that can be identified when 3D head poses are calculated from face images. And using these features they have trained SVM classifier to detect real and fake images[8].

In “Identification of Deep Network Generated Images Using Disparities in Color Components” published by Haodong Lia, Bin Lia, Shunquan Tana, Jiwu Huanga. The author found the disparities between the deep network generated image and camera real image in different color components and also they have observed differences in chrominance components, especially in the residual domain. Based on this, proposed feature set to capture color image statistics for identifying fake generated images. This Model have good accuracy even when training and testing data is mismatched moreover When the GAN model is unknown, their methods achieve good performance[9].

In “Deep Fake Image Detection Based on Pairwise Learning” published by Chih-Chung Hsu Yi-Xiu Zhuang and Chia-Yen Lee. They have proposed deep learning-based model to detect fake images using contrastive loss. Initially, they used state-of-the-art GANs to generate fake-real image pairs. Next, the reduced Densenet is developed to two stream network structure to allow pairwise information as input. Then proposed model is trained using pairwise learning to differentiate features between real and fake images[10].

In “Deep fake detection using a sparse autoencoder with a graph capsule dual graph CNN” published by Kandasamy V1, Hubálovský Š1, Trojovský P2 in 2022, This method is able to detect diverse types of spoofs made in images or videos which are generated using deep learning models like GANs, Long short term memory and Convolutional neural networks. In the first part of the training phase, extracted the feature frames from fake videos or images using sparse autoencoder with graph long short-term memory (SAE- GLSTM) method and in the second part used capsule dual graph CNN model for classifying sequence of frame or images as real or fake[11].

Below table shows the survey of deepfake detection methods:

Authors	Method of Detection
Hsu, Zhuang & Lee (2020) [12]	CNN concatenated to CFNN
Chintha et al. (2020) [13]	Convolutional bidirectional recurrent LSTM
Fernandes et al. (2020) [14]	Resnet50Model [102], pretrained on VGGFace2
Xuan et al. (2019) [15]	DCGAN(Deep Convolutional GAN) WGAN-GP(Wasserstein GAN - Gradient Penalty) PGGAN (Progressive Growing GAN)
Nguyen, Yamagishi & Echizen (2019)[16]	Capsule Networks

4. Dataset

For performing deepfake analysis on images, we obtained a dataset from Kaggle[28], which consisted of 140k images of human faces split into a train set, validation set and test set. There are 100,00 images in the training data set, 20,000 images validation dataset, and 20,000 images in the test data set which again split into an equal ratio of two classes called real and fake. Below graph shows the distribution of images into 3 datasets.

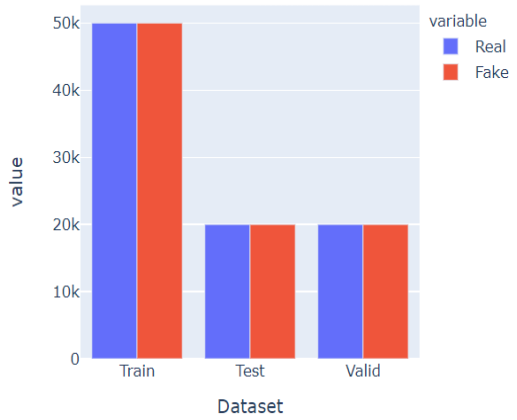


Figure 11. Dataset counts

Real human face images were collected from an image and hosting service called Flickr. And for generating deepfake images, NVIDIA's implementation's Style GAN [22] is used.

The above are some of the image samples of real and fake images from train dataset with label on top of it. Before taking this images for training or even for testing , we have scaled the values of image pixel from 1 to 255 using scale operation in image data generator.

5. Proposed method

In the proposed method, We have used deep learning techniques, Convolutional Neural network architecture(ConVets), and transfer learning architectures(DenseNet and InceptionNet) for training it on real and fake image data, and classify it. And later peform the comparison of performance.

5.1 Custom Neural Network Architecture

In the Custom Neural network, we have used various building blocks of the Convolutional Neural network which are, Batch normalization, Convolutional layer(feature map), max-pooling, global average pooling, dense layer, and Relu and sigmoid activation function. Now will see the custom neural network architecture which is successfully able to detect the GAN-generated images and real images using all the above-mentioned building components of Convolutional Neural Network(CNN). These components are popularly used in computer vision applications for image processing, image segmentation, image masking, and object detection in images or videos. CNN architecture is the winner of its first image recognition content In 2011, and in 2012 also. And even nowadays, CNN is still the queen of computer vision.

CNN is very good at capturing the design of the input image like lines, circles, any shapes, even the eyes or face wrinkles. CNN contains multiple convolutional layers on top of each other. Each convolutional layer makes the whole architecture to recognize the sophisticated shapes of the image. In the proposed model, we are using a combination of multiple-layer convolution or in simple words multi-layered feed-forward neural network along with upsampling and downsampling techniques discussed above.

In the proposed method, We have applied a total of 6 convolutional layers with multiple filters of size (3 x 3), and each CNN layer is paired with a batch-normalization layer which is applied before sending input to the convolutional layer and also paired with the max pooling layer, which is applied after the convolutional layer so it can reduce the size of the image data pixel and able to preserve meaningful information. The activation function ReLU is used with each convolution block which performs a non-linear operation and also reduces the risk of vanishing gradient descent and introduces sparsity in a neural network.

In the very last part of the proposed model, we performed Global Average pooling, which calculates the global average of the feature map got from the last convolutional block, After performing global average pooling, we are applying the flattening operation so we can get a 1-dimensional vector which again fully connected with one last neuron called dense layer with a sigmoid activation function. The output of the sigmoid activation function is a probability that lies between zero to one(0 - 1). By keeping the threshold value as 0.5, any output value smaller than 0.5 is classified as a fake image, and any output value greater than 0.5 is classified as a real image.

Please find the model architecture of the proposed method below:

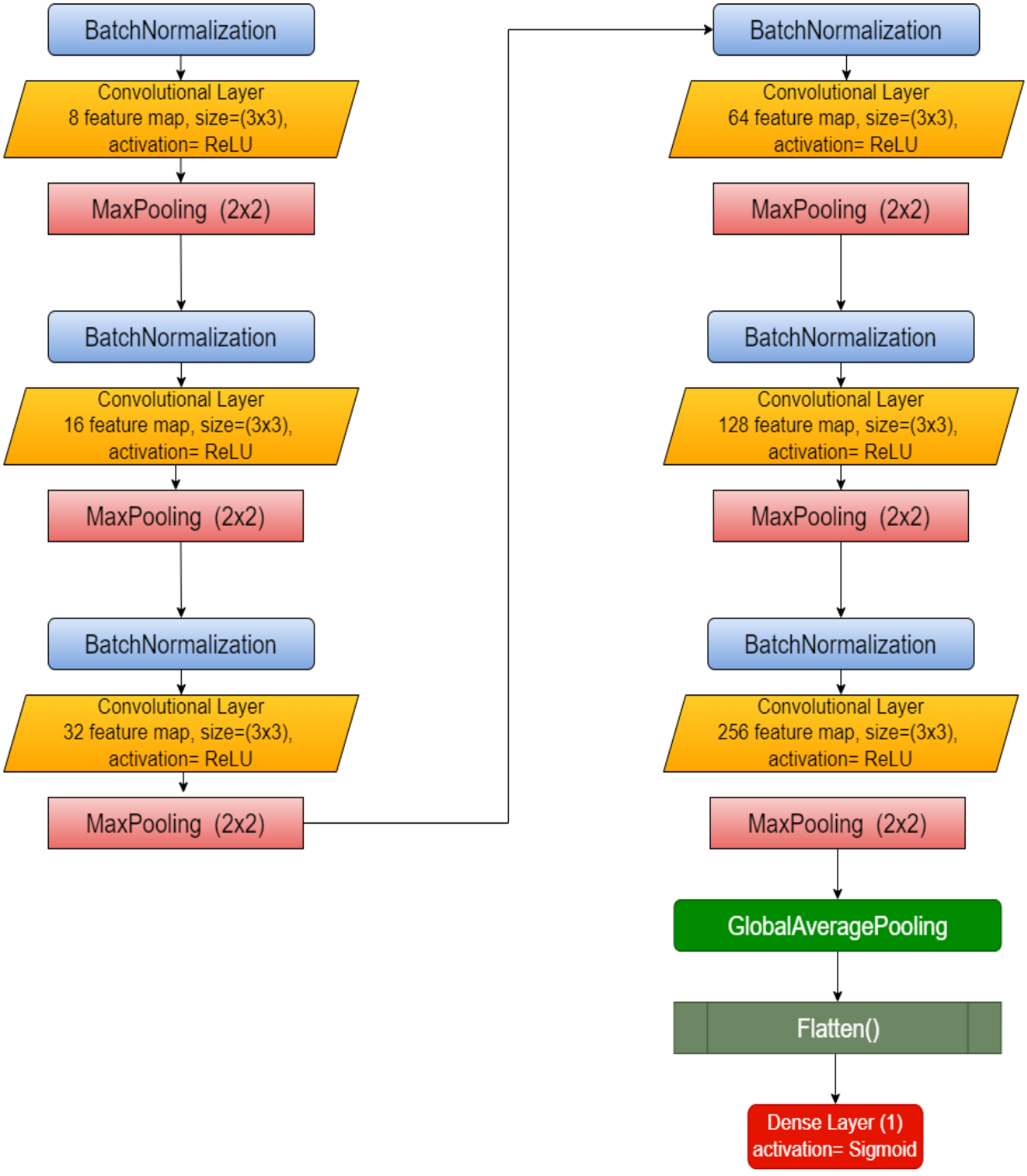


Figure 12. Custom Model Architecture

5.1.1 Results

The Custom CNN model is trained on 100,000 images with 20,000 images in a validation set with 10 epochs. Below is the accuracy and loss graph while training on the train dataset and validation data set.

1) **Accuracy :**

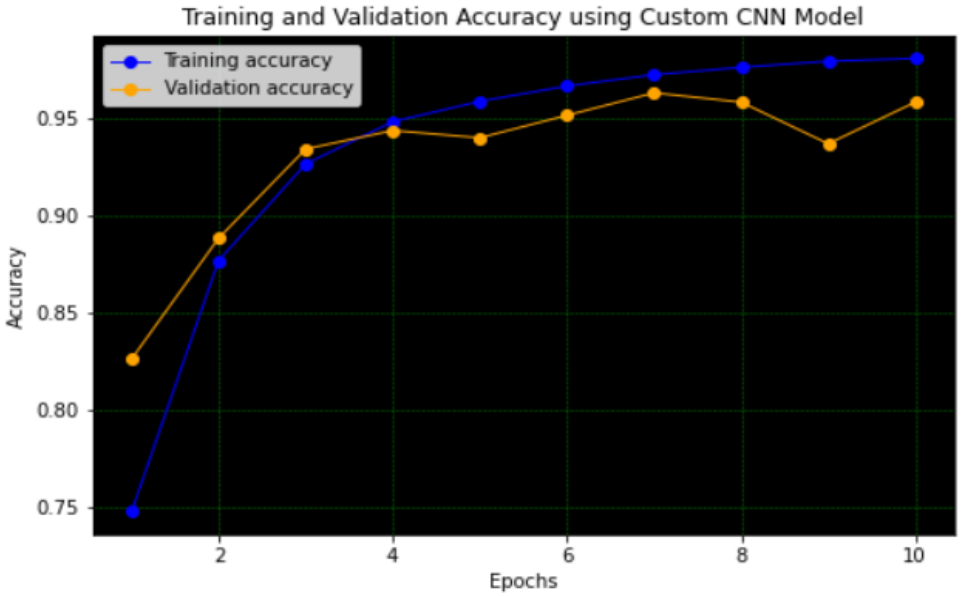


Figure 13: Training and Validation accuracy using Custom CNN model

2) **Loss:**

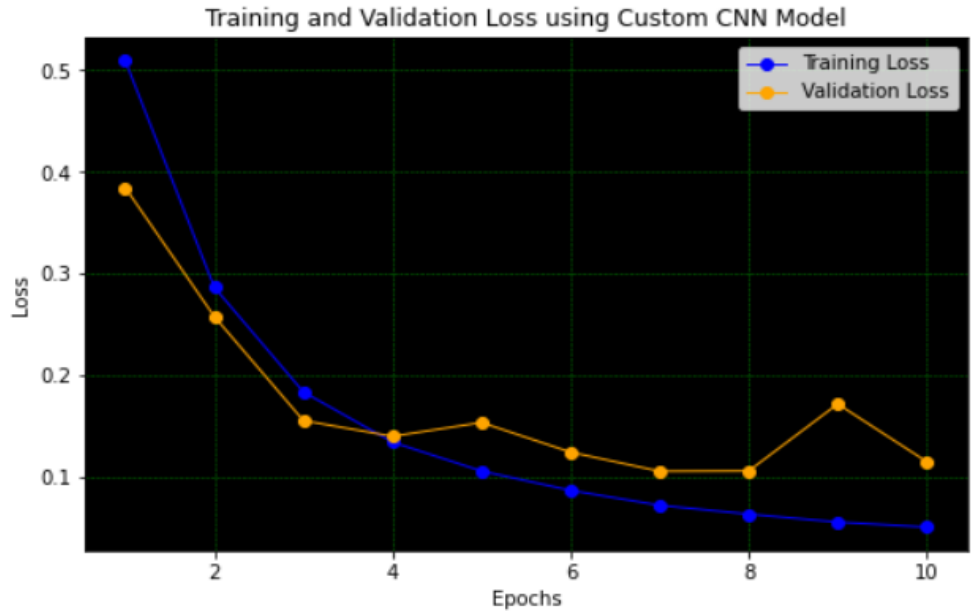


Figure 14: Training and Validation loss using Custom CNN model

As We can see in the above graphs, with each epoch, training, as well as validation accuracy, is increasing and the loss is decreasing. And for the test data set accuracy below is the confusion matrix.

5.1.2 Results on Test dataset

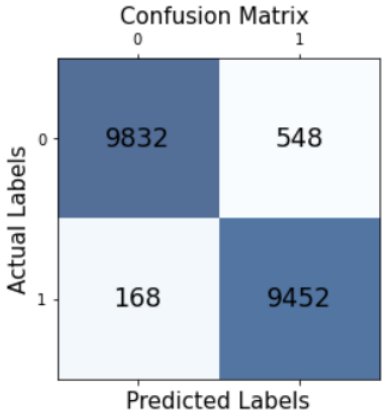


Figure 15. Confusion Matrix Result of Custom CNN Model

As we can see in the above Confusion matrix, out of 20,000 images, 19,284 images are classified correctly and 716 images are misclassified. So overall classification measures are displayed below:

- Accuracy: 96%
- Precision: 96%
- Recall: 96%

As per the results mentioned above, Custom CNN architecture is giving good results by using less number of convolutional layers as compare to other architectures and also we can say that this architecture can be used in GAN as strong discriminator network who is able to classify real and fake image generted by Genrator Network. In general, When we train GANs, that means we trained Both descriminator and generator together. Generator improves the performace based on discrimanator's accuracy. At practical level, In GANs, when discriminator gives 70% accuracy then only its stops training. But In our case, If we consider this model as discriminator, then its giving 90% accuracy. Hence we can prove that, Our Custom CNN architecture can be a strong discriminator as well stroong and efiicient deepfake detector.

5.2 DenseNet

In the second proposed method, We are using the transfer learning approach for detecting the deepfake image. We are using Densenet because it provides multiple advantages by overcoming the problem of vanishing gradient descent, improves the feature propagation, and also every layer contains only a limited number of parameters by keeping only the required number of filters as mentioned in Table 1[17], so this might improve the accuracy of deepfake image detection model. With DenseNet structure, I have trained architecture on our own image dataset, and predicted the output as Real vs Fake image.

5.2.1 What is DenseNet?

It is densely connected convolutional, it's way to increasing the depth of deep convolutional networks. The problem with a deeper CNN network is, that when its calculates gradient in the opposite direction they get vanished before reaching another end. It makes the training more efficient by placing shorter connections between the layers.

Its connected with all other layers in the network, for example, as shown in the above figure, the first layer is connected to the 3rd, 4th and 5th layer and the Second layer is connected the to 3rd, 4th and 5th layer, by this strategy it will pass maximum information between layers and contain the feature map of all preceding convolutional blocks.

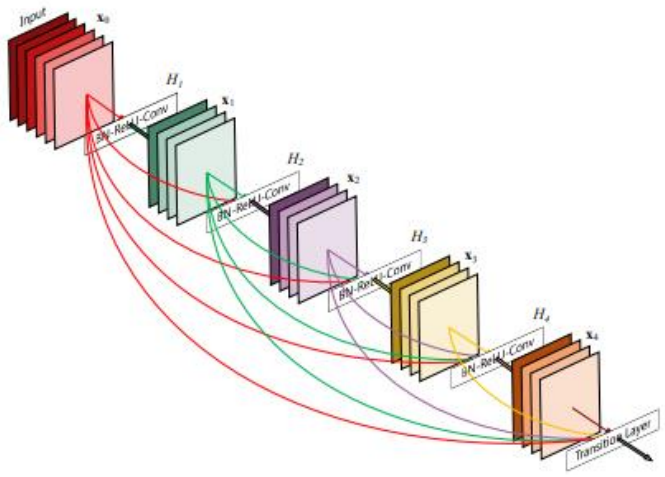


Figure 16 . Dense Net Architecture[17]

The main blocks of Dense net are dense blocks and transition layer as displayed in below table presented in paper[17].As shown in the above table, Dense Net starts with the Convolution layer followed by the pooling layer then continuously followed by Dense Block and Transition layer for four times in a row.

1) Dense Block:

In the column of DenseNet121, every dense block has two convolutional layers with the size (1 x 1), and (3 x 3). In dense block1, repeated 6 times, dense block2 repeated 12 times, dense block 3 repeated 24, and dense block 4 repeated 16 times.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. Dense Net architectures [17]

2) Transition layers:

The main aim of the transition layer is to remove channels to half of the existing channel. It contains a (1 x 1) convolutional layer and (2 x 2) average pooling layer with stride 2.So in short, This model contains, 4 dense blocks connected with Batchnormalization, ReLU activation function and convolution of size (3 x 3), and transition layers, after these dense blocks we added global average pooling and dense layer with one neuron and sigmoid activation function for classification real or fake.

5.2.2 Results

This DenseNet Model is trained with the same train image dataset with 10 epochs.

1) Accuracy:

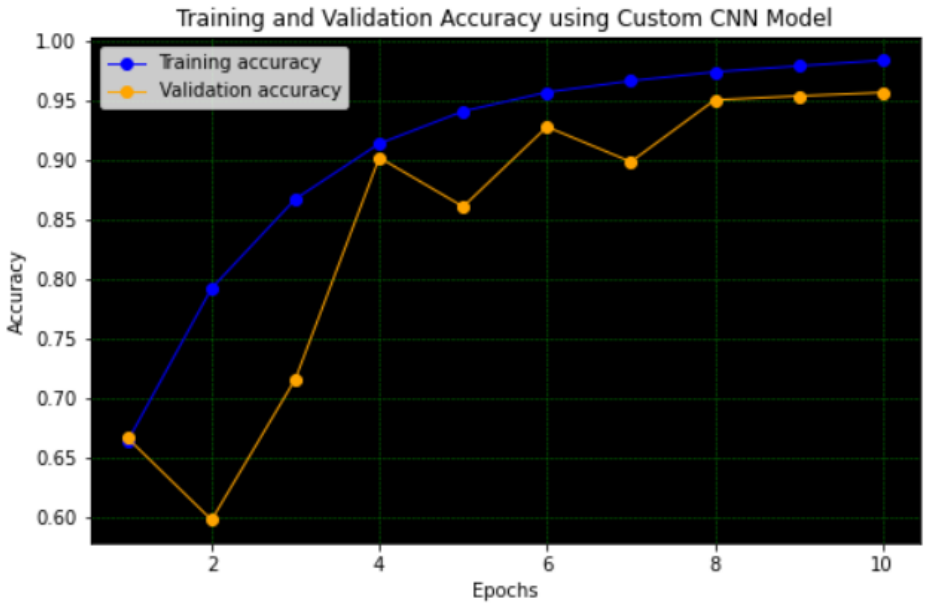


Figure 17: Training and Validation accuracy using DenseNet Model

2) Loss:

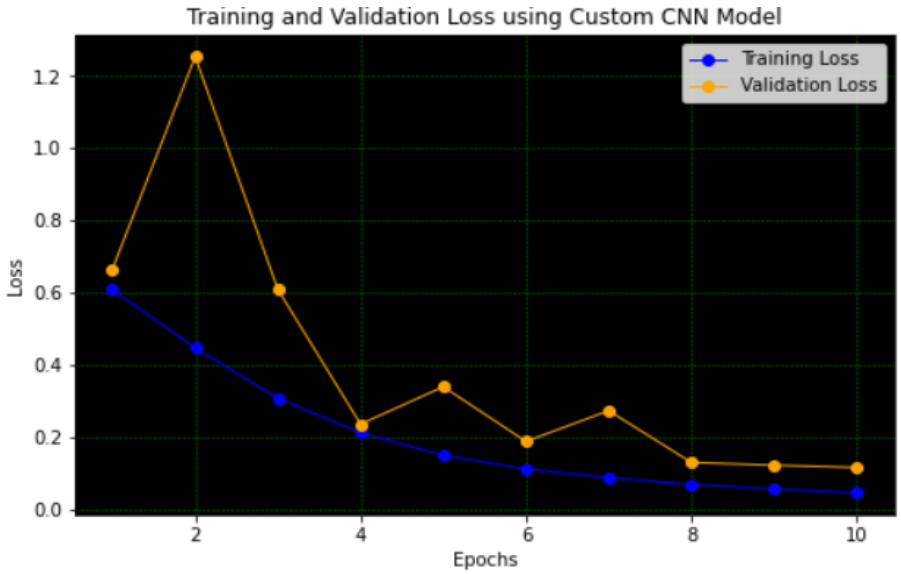


Figure 18: Training and Validation loss using DenseNet Model

By using the DenseNet model, train dataset accuracy ie reached up to 98.81% and on the validation set , accuracy reached up to 96%, and also loss is reduced gradually.

5.2.3 Results on Test Dataset

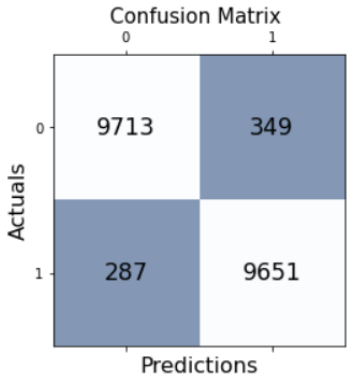


Figure 19. Confusion Matrix Result of DenseNet Model

As we can see in the above Confusion matrix, out of 20,000 images, 19,364 images are classified correctly and 636 are misclassified. So overall classification measures displayed below:

- Accuracy: 97%
- Precision: 97%
- Recall: 97%

DenseNet is better at resue the information by providing the interconnection between the convolutional blocks, so that's why, In our comparative study we got higher accuracy using it. And it was able to detect highest number of fake images.

5.3 InceptionNet

Inceptionnet is a masterpiece in the world of computer vision. When another previous model just increased the depth of the neural network to improve the accuracy but along with it computational cost also increased. In InceptionNet, not only focus on the depth of the neural network but also increase the width of the neural network. It has many more advantages which are discussed below. Here we are using InceptionNet which is again a transfer approach, we have trained it with 100,000 real and fake images and 20,000 images in the validation set. Further, we will compare it with a custom neural network model.

5.3.1 What is InceptionNet?

Its also known as GoogleNet, its 22 layer deep convolutional network, winner of the 2014 ILSVRC Championship. Its perfectly designed and has applied lots of methods to improve performance in terms of accuracy and speed. The popular versions of InceptionNet is as follows:

- Inception V1
- Inception V2 and Inception V3
- Inception V4 and Inception Resnet

There is a large variation in the size of the main object in an image, so its become tougher to choose the correct kernel size of convolution, Large kernel size is preferable for object which is distributed over image globally and small kernel size is preferable for small size of object which is distributed over image locally.

So authors have introduced multiple sizes of the kernel at the same level, they are making network wider rather than making it a deeper CNN.

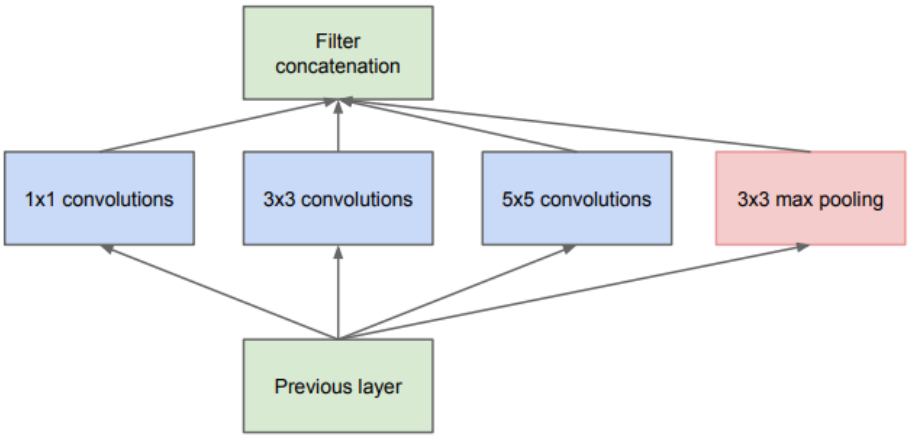


Figure 20. Inception module [18]

The above image is a naïve inception module published in a research paper [18], it applies a convolution filter on input with different sizes of filters (1x1, 3x3, 5x5) along with-it max pooling is applied and, in the end, outputs are concatenated and sent to next layer.

In the below figure, the Inception module with dimension reductions, the authors reduced the input channels by adding extra (1 x 1) convolution before (1x1, 3x3, 5x5) kernel size filters the idea behind is instead of passing (5x5) convolution to next layer , apply (1x1) convolution on top of it that's how its reduced the number of input channel.

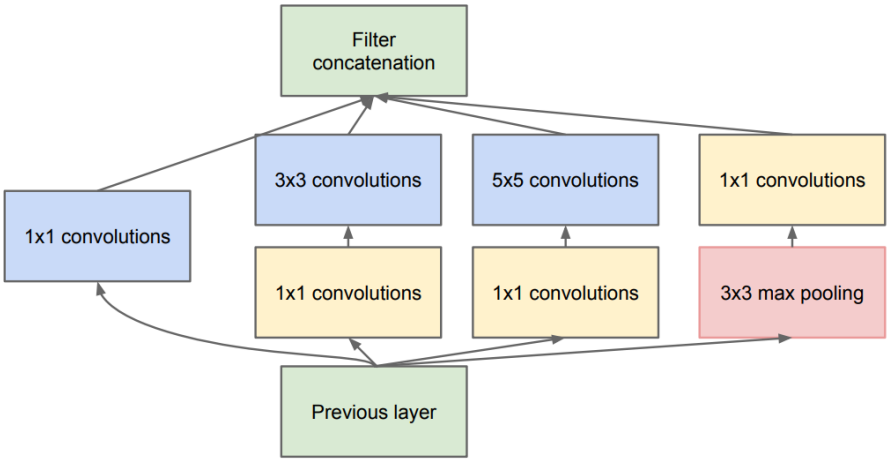


Figure 21. Inception module with dimension reduction [18]

Using the dimension reduction, the neural network architecture is built and popularly known as Google Net or Inception V1, the main architecture is shown below:

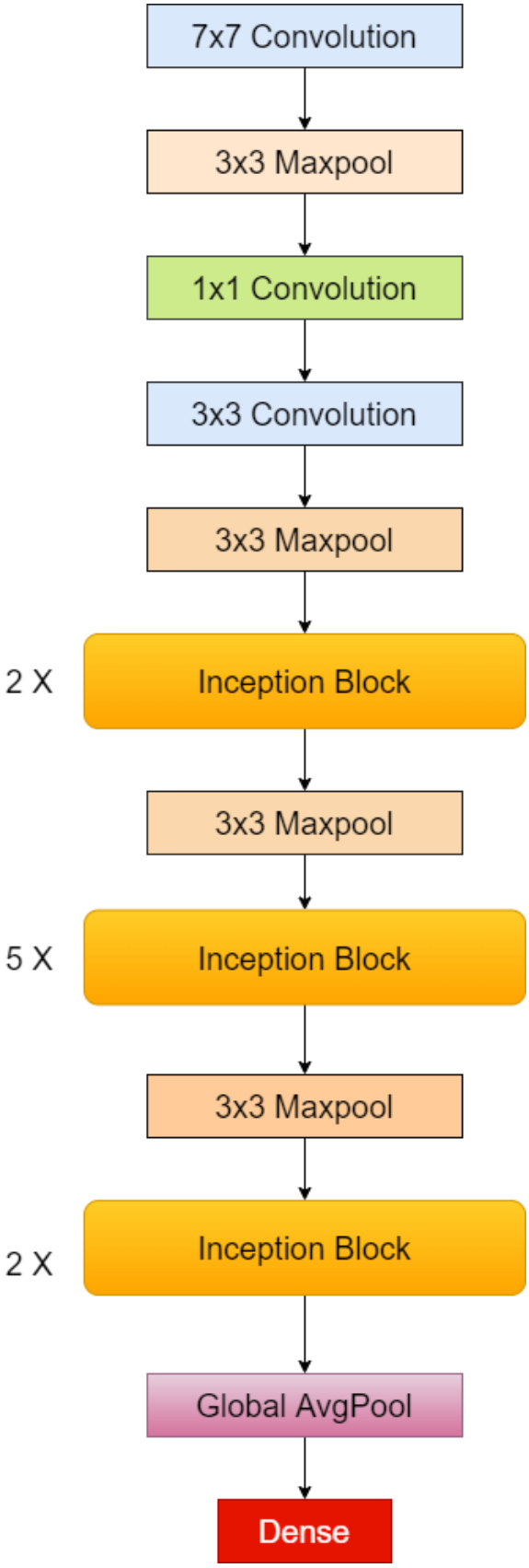


Figure 22. simplified InceptionNet architecture

5.3.2 Results

InceptionNet is also trained on the same dataset of 100,000 images and 20,000 validation set with the same number of epochs that is 10.

1) **Accuracy :**

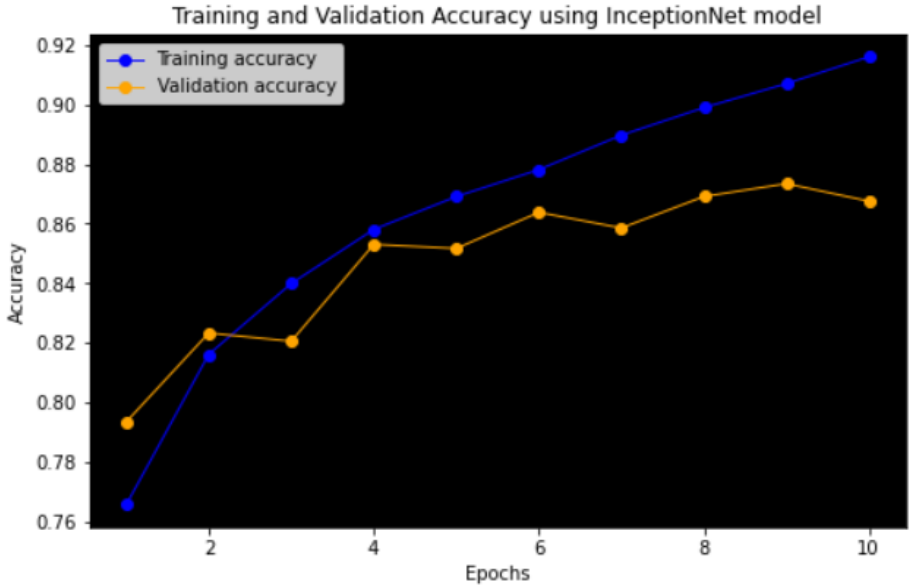


Figure 23: Training and Validation accuracy using InceptionNet Model

2) **Loss:**

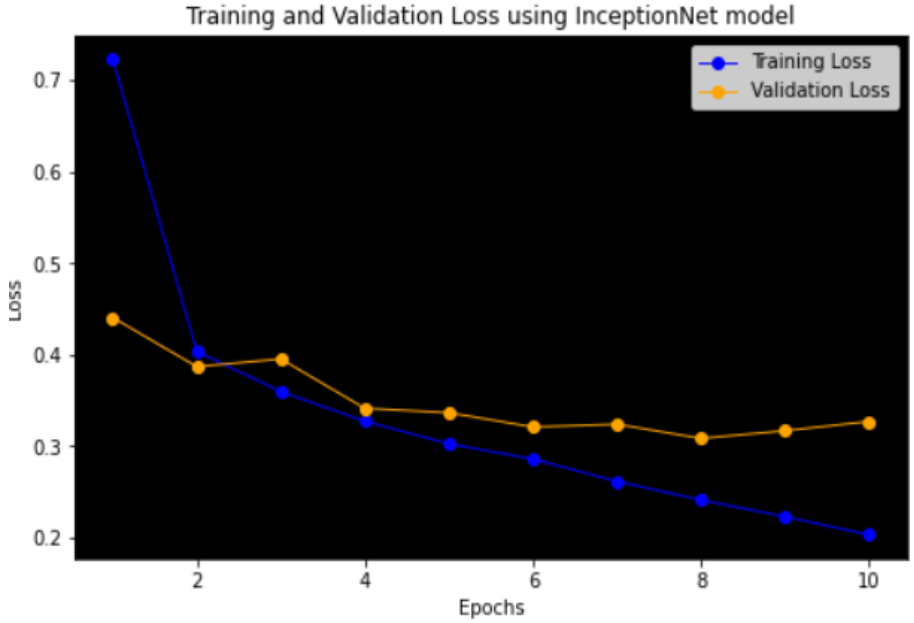


Figure 24: Training and Validation loss using InceptionNet Model

Using the InceptionNet model, maximum accuracy on train data set reached upto 91.53% and on the validation data set 86.62% which is quite less than custom model and DenseNet model.

5.2.3 Results on Test Dataset:

Based on the below confusion matrix, will calculate Accuracy, Precision, and Recall.

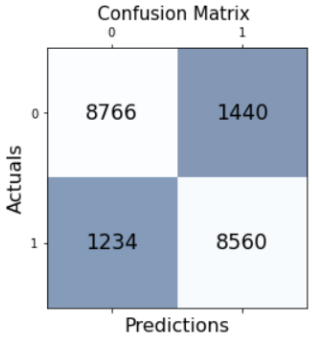


Figure 25. Confusion Matrix Result of InceptionNet Model

As we can see in the above Confusion matrix, out of 20,000 images, 17,326 are classified correctly and 2674 are misclassified. So overall classification measures are displayed below:

- Accuracy: 87%
- Precision: 87%
- Recall: 87%

5.4 Comparisions of Model Performance

Model	Accuracy	Precision	Recall	F1- Score
Custom CNN Model	0.96	0.96	0.95	0.95
DenseNet 121	0.97	0.97	0.97	0.97
Inception Net V3	0.87	0.87	0.87	0.87

Table 2. Comparision of Model performance

As we can see in the above table shows comparision among all the models which have been implemented in this work. All the models were tested on the test Dataset which consists of 10,000 real and 10,000 fake images. Among all the models, DenseNet has achevied the greatest accuracy 97% and after that Custom CNN model has achieved 96% accuracy and lastly Inceptionet model 87% accuracy, One of the reason behind Iess accuracy with InceptionNet is because of model complexity because its using 43 layers which is causing overfiiting on train datsetnot generalized will on unseen dataset.

Where DenseNet achieved the highest accuracy by using 4 dense blocks which are interconnected with each other. And our proposed CNN model achieved 96% by using 6 convolutional layers. so we can confirmly say that our small CNN deepfake detection model has achieved very good accuracy by efficiently utilizing the computational resources with lees number of layers and taking less time for training as well as for detections.

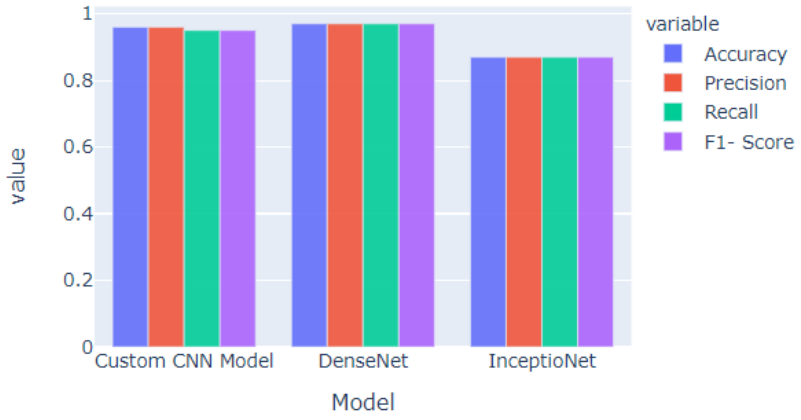


Figure 26. Comparision of Model performance

6. Experiments

6.1 Aim

The main goal of this experiment is to improve the accuracy of the custom CNN model on unseen data. To this end, we are focusing on the below areas for better performance.

- Reduce the Complexity of Neural Network
- Hyperparameter Tuning
- Operations on Augmented Data

6.2 Adding Dropout Layers

In this experiment, we will try to improve the generalization accuracy of the Custom CNN model by reducing the complexity of the neural network. And compare the performance.

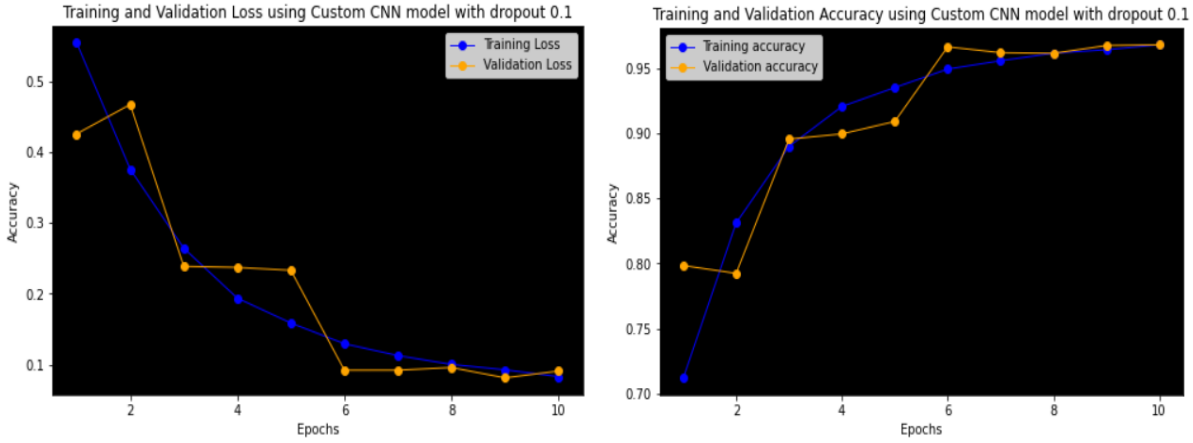
6.2.1 Method

For reducing the complexity, we have implemented dropout layers in between the multiple CNN layers. Dropout is one of the methods in deep learning for implementing regularization. It will drop out a few layers or deactivate some neurons randomly by the specified value of dropout rate during training time.

So, this way it will decrease the complexity of neural networks and also reduce overfitting on training data sets and result in good accuracy on test data sets. To perform this experiment added a dropout layer with dropout probability = 0.1 after the batch normalization layer and before the convolutional layer.

6.2.2 Results

In the below graphs, we can see the difference between the training set and validation set's accuracy and loss with and without dropout.



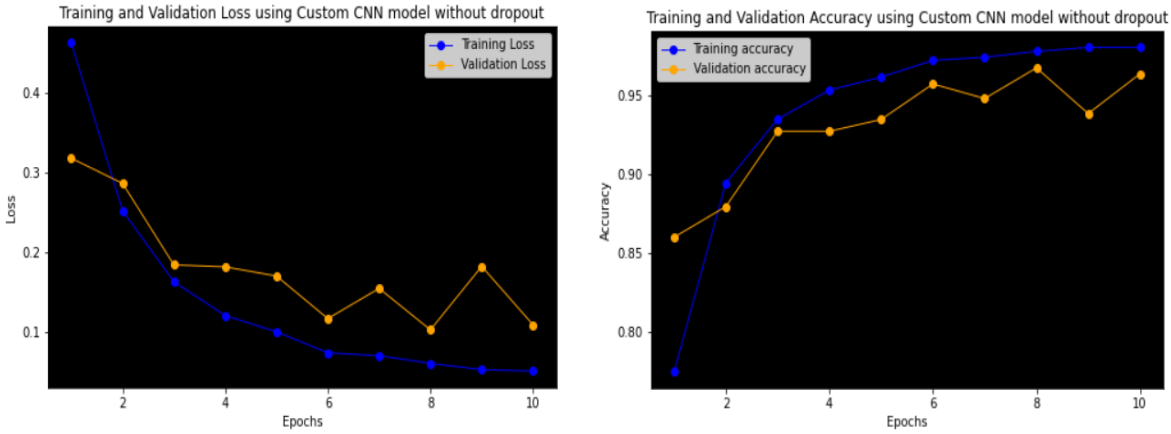


Figure 27. Model Performance with and without dropout

There is a negligible difference of accuracy and loss of Model with and without dropout rate, it's almost the same on the test dataset also. One of the possible reasons behind this is batch normalization which is already implemented in the custom CNN model. And the actual reason is- Batchnormalization removes internal covariate shift, addressing the problem by normalizing layer inputs. That mean even though we have deactivated few of the neurons, and reduced the calculations between the layers by applying dropout techniques.

Batchnormalization performs normalization on remaining activated neurons and stabilizes the values i.e. output values of previous layer, that's how it reduce the effect of dropouts as mentioned in paper [23] It also acts as a regularizer so that's why results with and without dropout is almost the same. So we can conclude it, we are not able achieve more accuracy with the help of dropout layer in existing CNN architecture but still it able to detect deepfake images in good amount.

6.3 Hyperparameter tuning

In this experiment, Trying to find the optimal parameter for the learning rate of the optimizer. Optimizer helps to minimize loss function by modifying learnable parameters like weights and bias of neural network. Therefore, will find the optimal parameter for the optimizer, which can help us to improve accuracy.

6.3.1 Method

To perform this experiment, I have trained a neural network with multiple values of learning rate – [0.001, 0.002, 0.003, 0.0001] separately on 100000 images training dataset and observed the performance as shown in results.

6.3.2 Results

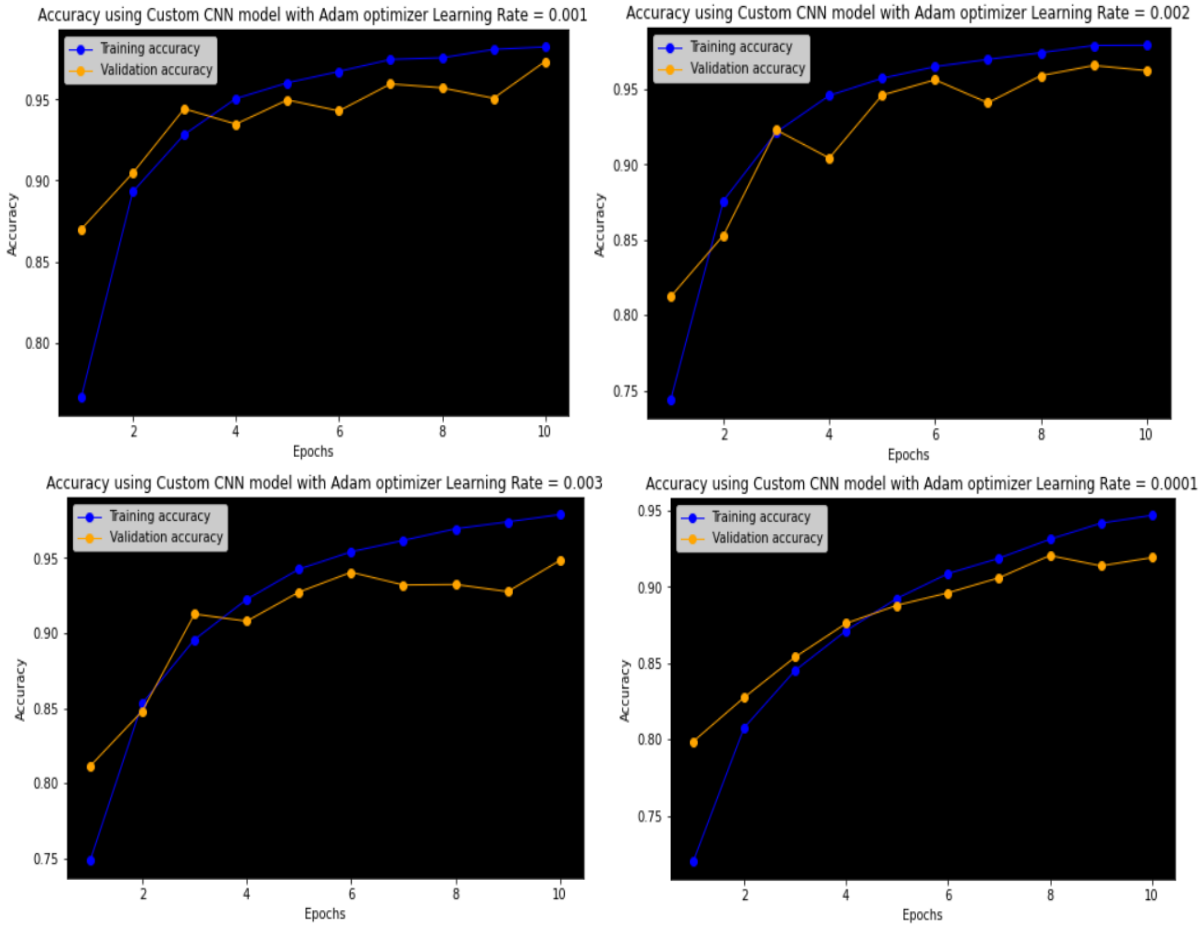


Figure 28. Model Performance with different learning rate

As we can see in the above graphs, every value of the learning rate achieved an accuracy of more than 95% but the learning rate of 0.0001 got 95% accuracy the reason behind this, a very small learning rate needs a long time for the training process to get the global optimal value, and it might be possible 95% accuracy is the value because of local optima.

The reason behind "0.001" learning rate works best is that, its smallest learning rate which makes the training of the model reliable and step towards minimization of loss are small whether with highest learning rate, changes in the weight values are big and optimizer makes the loss even more worse ulternativly accuracy got impacted.

Results on test dataset:

Below are the results on the test dataset, and as we can see "0.001" is the optimal learning rate who achieved 97% accuracy on unseen data.

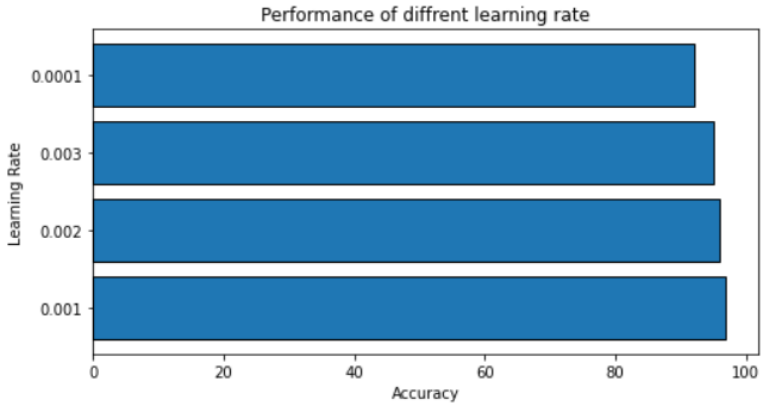


Figure 29. Performance of different learning rate

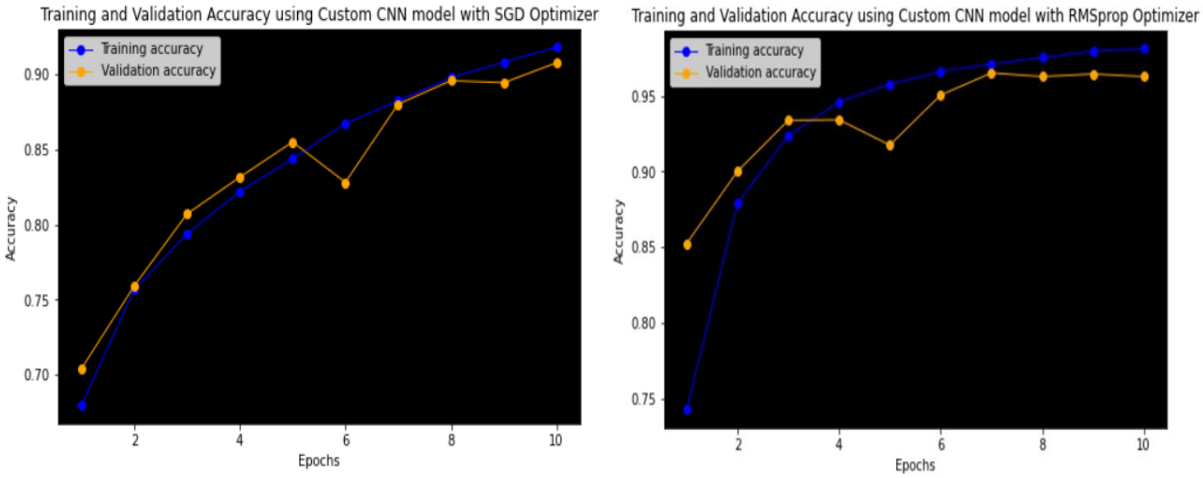
6.4 SGD vs RMSprop optimizers

In this experiment, we are trying to find the best optimizer for the proposed CNN architecture which will help to achieve the aim that is better accuracy on the test dataset. Which can be able to detect deepfake images more precisely with less computation power with less training time.

6.4.1 Method

To perform this experiment, trained Custom CNN model with RMSprop and SGD (Stochastic Gradient Descent) separately and observed the performance. Every optimizer has its own good advantages and disadvantages.

6.4.2 Results



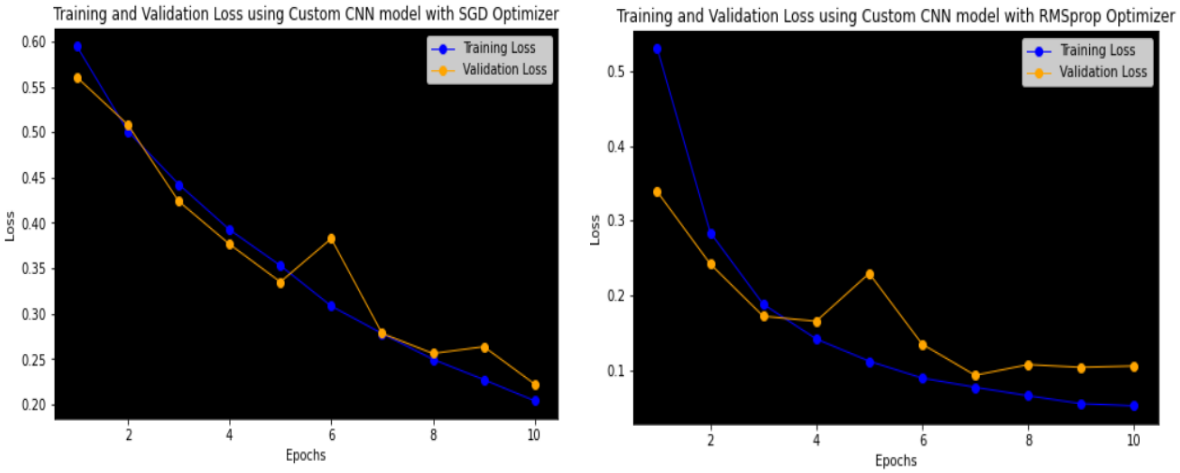


Figure 30. Model Performance with SGD vs RMSProp

As we can see in the above performance graph, Using SGD optimizer loss value on the training and validation set, started from 0.60 and at the maximum reduced to 0.20, and In the case of RMSprop loss value on the training set started from 0.55 and on validation set loss value started from 0.35 and both reduced to 0.1.

And also, RMSprop achieved very good accuracy on the test dataset. The reason behind this is, It adjusts the learning rate automatically as per the loss value and applies different learning rates with each parameter. While it's slow in computation but gives better results than SGD. And The reason behind less accuracy with SGD optimizer is frequent changes in the model parameter can lead to noise gradient values and hence results in less accuracy on the unseen dataset.

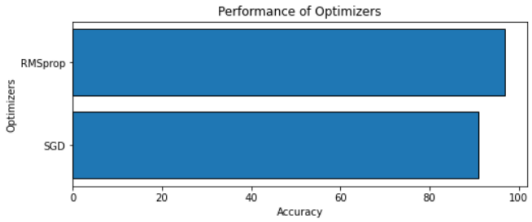


Figure 31. Performance of RMSProp and SGD

6.5 Custom Model with Data Augmentation

Data Augmentation is a method to artificially create a new dataset by using an existing dataset. It increases the variability in the dataset and alternatively helps to generalize the model on unseen data. On the other hand, it will impact the time duration of training but also it makes the neural network more robust. So that's why the correct amount of data augmentation can save lots of training time.

6.5.1 Method

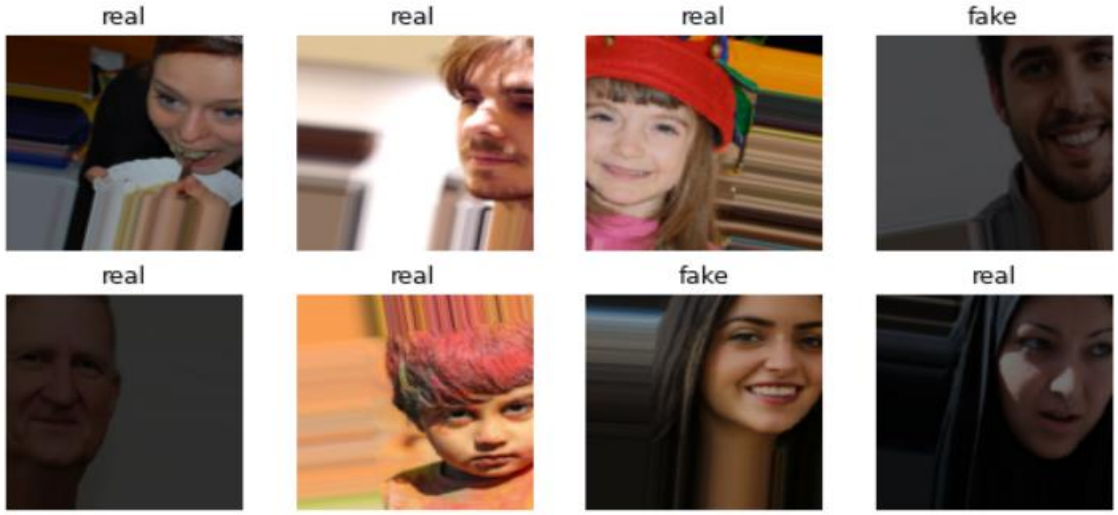
For Data augmentation, applied different operations on training data images using Image Data Generator, available in Keras library. These Data augmentation techniques

include rescaling, rotation flipping the image horizontally, and vertically, padding, zooming, cropping and shifting and some more transformation on images,

Below is the code snippet given:

```
image_gen = ImageDataGenerator (
    rescale=1. /255., #scaled pixel values from 1 - 255
    rotation_range=10, #rotation
    width_shift_range= [-0.3, 0.3], #horizontal shift
    height_shift_range=0.3, # vertical shift
    zoom_range=0.2, # zoom
    horizontal_flip=True, # horizontal flip
    brightness_range= [0.2,1.2] )    # brightness
```

After applying data augmentation code, got the below effects on training data image.



After performing data augmentation on train images As we can see in above some of the images got shifted by width and by height, and brightness has changed. In original dataset, we have only face images in some specific conditions which are limited. So our model might get less variety of images for training. So to increase variation in images, data augmentation techniques helped us to make minor modifications to existing images. And that's how augmented data improves the generalization of the model.

6.5.2 Results

While training on augmented data, the model has achieved more than 90% accuracy and loss decreased gradually up to 0.1 within 10 epochs only in order to get more accuracy on training and validation set we need to train the model for few more extra epochs, because model takes time to learn pattern on augmented data as it has more variance.

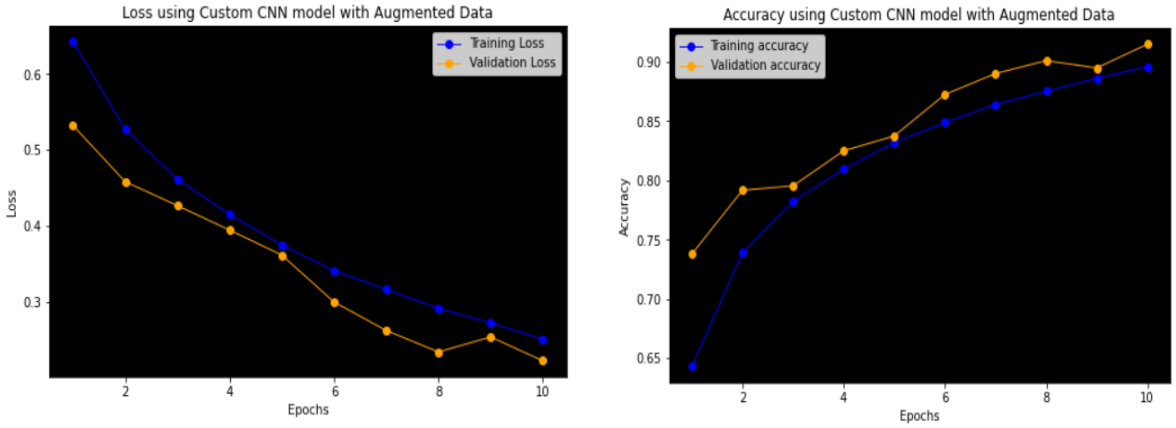


Figure 32. Custom CNN Model Performance

Results on test dataset:

On the test dataset, the proposed model has achieved 89% accuracy. As per the below confusion matrix, out of 20000 images, 18233 images are correctly classified and 1767 images are misclassified, so we can say that model has generalized well on unseen data but we can also improve it more by applying advanced tricks and techniques.

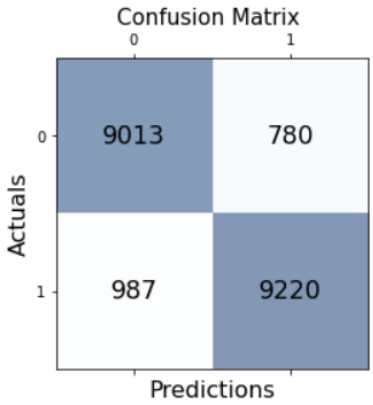


Figure 33. Confusion matrix on the augmented dataset.

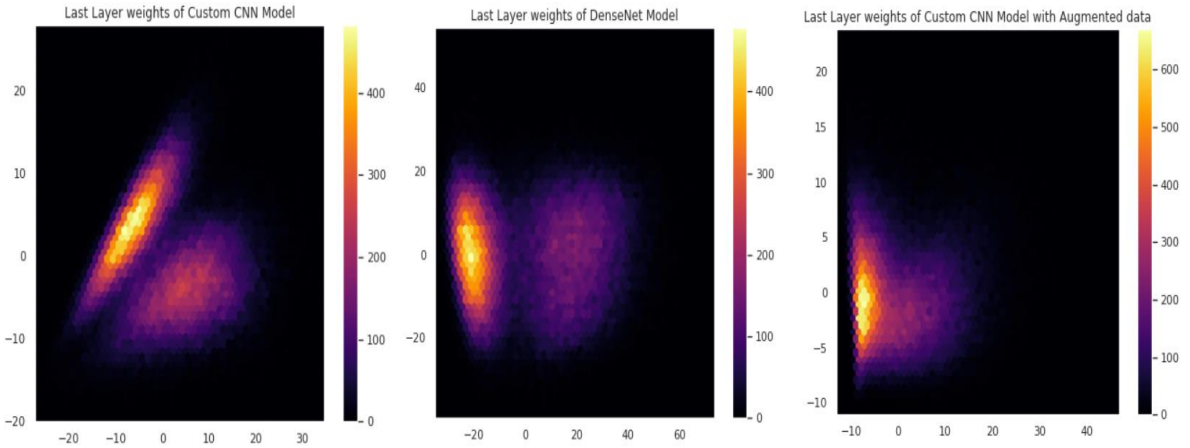
The reason behind getting less accuracy and high misclassification rate on test dataset is because model was trained on augmented dataset. Model has learned the pattern of the transformed images whose brightness has changed, image is rotated flipped and scaled. Training accuracy is high, that's means neural network model has learned it pretty well and tuned the coefficient values. but we are testing it on non-augmented dataset which normal image without transformation.

It's not best practice to perform augmentation on validation data and test data because the goal of this experiment is to make the Custom CNN model more robust and make it learn the variation of images And check the performance on test dataset which 89% accurate. Therefore, we can conclude, even we have increased the variation in training images our custom model performed well on test data set, so we have generalized the model and reduced the risk of overfitting.

7. Visualization of Last Layer output

In this section of the report, we are performing analysis on the output values get assigned to last layers before applying last sigmoid activation function which is responsible for classification of real or deepfake image. To perform this analysis, extracted the last layer output from the respective model and stored it in the data frames. After this, perform the scaling by applying the standard scaler function on it.

Since there are multiple columns available in the last extracted layer. For Custom CNN model and Custom CNN model with Augmented data calculated: (100000 x 256) parameters and DenseNet model calculated : (100000 x 1024) parameters. Its impossible to visualize high dimensional data so we have performed a dimensionality reduction algorithm called "Principal Component Analysis(PCA)" to get the top 2 PCA components that are able to capture variance in the data and perform visualization using hexabin plots. In Hexabin plots, It uses the technique of hexagonally binned density maps. In this, 2-dimensional plane(x,y), uniformly tiled with the hexagon, and the number of data points falling in respective tiled is counted and displayed with color range.



Custom CNN Model | DenseNet Model | Custom CNN - augmented data

Figure 34. PCA components

As we can see in above graphs, with Custom CNN model and DenseNet model weights lies between the range of -20 to 20 however majority values are nearby zero which means we can say that model has some sparse values which is beneficial for deep learning model, it makes the computation faster because some neuron has sparse value and also takes less memory for storage. Because of the sparsity, the model become less complex and reduced the risk of overfitting. In the case of Custom CNN model with augmented data, the last layer's weight values lie between -10 to 10. Majority of weight values are zero centric and sparse.

8. Future Research Direction

Deep learning is cutting edge technology always breaking new grounds. Same logic applies to deepfake generation. The methods and quality of deepfakes images or videos or audios has been improving day by day, so detection methods need to be improved accordingly. The main motivation behind this is, What AI has broken and is repaired by AI [33]. The proposed detection method is trained using human face images generated by StyleGAN and Flicker, which are having limited scope for images in the same position. For future studies, we need to train a model using various kinds of fake images which are captured in multiple positions having some body movements. And another future improvement is, that the model is trained using only Style GAN generated fake images, detection model needs to be train with latest fake generation techniques also for example images generated by DCGAN, WGAN-GP , PGGAN or such other techniques like identity swaps and attribute manipulations. Another improvement can be, detecting deepfake videos which consist of a sequence of image frames synced with fake speech, so it can be a new research direction to detect fake audio with fake image frames.

9. Conclusion

In this project, we have learned multiple deepfake creations and deepfake detection techniques, And proposed a state-of-the-art CNN deepfake detection model and also demonstrated this proposed deep learning model is highly accurate and computationally efficient as compared to other transfer learning approaches used in this work. The results of the proposed CNN architecture able to successfully detect real and deepfake images with good amount of accuracy which is 96% and less misclassification rate by using a small CNN network build using 6 convolutional layers as compared to other transfer learning approaches used in this work. DenseNet was able to detect images with 97% accuracy which is a combination of 58 layers and with the help of InceptionNet V3 got 87% accuracy by using a combination of 48 layers. That's mean our proposed CNN model is performed very well. But looking at the minimal misclassification rate, it can be an area of improvement. It is the same as the medical domain, even 4% of inaccuracy can make a large impact. We have also learned to make the detection model more generalized we need a large amount of image dataset which can make our model more robust.

10. Reference

[1] Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional GANs.

[2] Christine Dewi,; Rung-Ching Chen,; Yan-Ting Liu and Hui Yu, Various Generative Adversarial Networks Model for Synthetic Prohibitory Sign Image Generation.

[3] Du, Y.; Zhang, W.; Wang, J.; Wu, H. DCGAN based data generation for process monitoring.

[4] Tero Karras, Samuli Laine, Timo Aila, A Style-Based Generator Architecture for Generative Adversarial Networks

[5] Xun Huang Serge Belongie, Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization.

[6] Yuezun Li,; Ming-Ching Chang ,; Siwei Lyu , Exposing AI Generated Fake Face Videos by Detecting Eye Blinking.

[7] Yuezun Li,; Siwei Lyu, Exposing Deepfake Videos By detecting face wrapping artifacts ”

[8] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses, 2018.

[9] Haodong Lia, Bin Lia, Shunquan Tana, Jiwu Huanga Identification of Deep Network Generated Images Using Disparities in Color Components

[10] Chih-Chung Hsu,; Yi-Xiu Zhuang ,; Chia-Yen Lee Deep Fake Image Detection Based on Pairwise Learning.

[11] Kandasamy V1,; Hubálovský Š1,; Trojovský P2; Deep fake detection using a sparse autoencoder with a graph capsule dual graph CNN doi: 10.7717/peerj-cs.953 Published online 2022 May 31.

[12] Hsu, Zhuang & Lee (2020) Hsu C-C, Zhuang Y-X, Lee C-Y. Deep fake image detection based on pairwise learning. Applied Sciences. 2020;10(1):370. doi: 10.3390/app10010370.

[13] Chintha et al. (2020) Chintha A, Thai B, Sohrawardi SJ, Bhatt K, Hickerson A, Wright M, Ptucha R. Recurrent convolutional structures for audio spoof and video deepfake detection. IEEE Journal of Selected Topics in Signal Processing. 2020;14(5):1024–1037. doi: 10.1109/JSTSP.2020.2999185.

[14] Fernandes et al. (2020) Fernandes S, Raj S, Ewetz R, Pannu JS, Jha SK, Ortiz E, Vintila I, Salter M. Detecting deepfake videos using attribution-based confidence metric. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops; Piscataway: IEEE; 2020. pp. 308–309

[15] Xuan et al. (2019) Xuan X, Peng B, Wang W, Dong J. On the generalization of GAN image forensics. In: Sun Z, He R, Feng J, Shan S, Guo Z, editors. Biometric Recognition. CCBR 2019. Lecture Notes in Computer Science. Vol. 11818. Cham: Springer; 2019. pp. 134–141.

[16] Nguyen, Yamagishi & Echizen (2019) Nguyen HH, Yamagishi J, Echizen I. Capsule-forensics: using capsule networks to detect forged images and videos. ICASSP, 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); Piscataway: IEEE; 2019. pp. 2307–2311.

[17] Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, Densely Connected Convolutional Networks, arXiv 1608.06993 (2016)

[18] Christian Szegedy; Wei Liu; Yangqing Jia; Pierre Sermanet; Scott Reed; Dragomir Anguelov; Dumitru Erhan, Going deeper with convolutions,2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)DOI: 10.1109/CVPR.2015.7298594

[19] Faceswap: Deepfakes software for all. https://github. com/deepfakes/faceswap.

[20] Zhou H, Liu Y, Liu Z, Luo P, Wang X (2019) Talking face generation by adversarially disentangled audio-visual representation. In: AAAI Conference on Artificial Intelligence (AAAI-19), Honolulu

[21] He Z, Zuo W, Kan M, Shan S, Chen X (2019) AttGAN: facial attribute editing by only changing what you want. IEEE Trans Image Process 28(11):5464–5478

[22] NVIDIA Style GAN implementation: https://github.com/NVlabs/stylegan2

[23] Sergey Ioffe, Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift https://doi.org/10.48550/arXiv.1502.03167

[24] Multilayer Convolutional operation Student Notes: Convolutional Neural Networks (CNN) Introduction – Belajar Pembelajaran Mesin Indonesia (indoml.com)

[25] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall , pusblished in Activation Functions: Comparison of Trends in Practice and Research for Deep Learning, arXiv:1811.03378v1 [cs.LG] 8 Nov 2018

[26] Han, Jun; Morag, Claudio (1995). "The influence of the sigmoid function parameters on the speed of backpropagation learning".

[27] Karl Pearson 1904, Published in On the theory of contingency and its relation to association and normal correlation.

[28] Dataset source: https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces

[29] https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[30] https://www.sciencedirect.com/science/article/pii/B9780128235195000014

[31] Patrick Tucker. The newest AI-enabled weapon: ‘deep-faking’ photos of the earth. https://www.defenseone.com/technology/2019/03/next- phase- ai- deep- fakingwhole-world-and-china-ahead/155944/, March 2019.

[32] T Fish. Deep fakes: AI-manipulated media will be'weaponised' to trick military. https://www.express.co.uk /news /science /1109783 /deep - fakes -ai - artificial - intelligence - photos - video -weaponised-china, April 2019

[33] B Marr. The best (and scariest) examples of AI-enabled deepfakes. https://www.forbes.com/sites/bernardmarr/2019/07/22/the-best-and-scariest-examples-ofai-enabled-deepfakes/, July 2019.

[32] S Samuel. A guy made a deepfake app to turn photos of women into nudes. it didn't go well. https://www.vox.com/2019/6/27/18761639/ai- deepfake- deepnude- app-nude-women-porn, June 2019.

[33] Floridi, L. (2018). Artificial intelligence, deepfakes and a future of ectypes. Philosophy and Technology, 31(3), 317-321.

Code References:

1. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
2. https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization
3. https://www.tensorflow.org/api_docs/python/tf/keras/layers/AveragePooling2D
4. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
5. https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D
6. https://www.tensorflow.org/api_docs/python/tf/keras/layers/GlobalAveragePooling2D
7. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten
8. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
9. https://www.tensorflow.org/api_docs/python/tf/keras/models/save_model
10. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers
11. https://www.kaggle.com/code/zohaib30/fake-vs-real-tensorflow-keras
12. https://keras.io/api/applications/densenet/
13. https://keras.io/api/applications/inceptionv3/

A. Appendix

A.1 Professional issues

While working for this project I came across multiple technical issues and visualization issues which I would like to discuss further.

1. **Visualization with high dimensional data.**

In the section number "7: Visualization of Last Layer output", I wanted to plot the output values of last layer of respective model before applying last activation function "Sigmoid". As it was multidimensional so performed dimensionality reduction using PCA and selected 2 components for plotting. But model is using 100000 records for training, its too much data points for plotting which results into overplotting of data as shown in below graph.

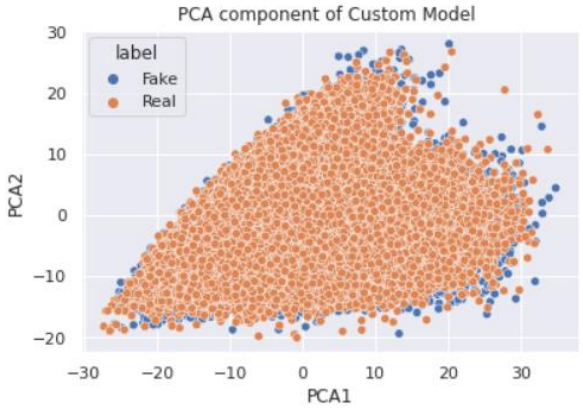


Figure. PCA components of Custom CNN model

So to overcome with this problem, "Professor. Chris Watkins", suggested me to use "hexagonally binned density maps" which is especially designed for plotting large amount of data. Hexabin plot available in matplotlib library Can help to visualize large amount of data on plane tiled with hexagon and represent the density with colormap as shown in the below graph.

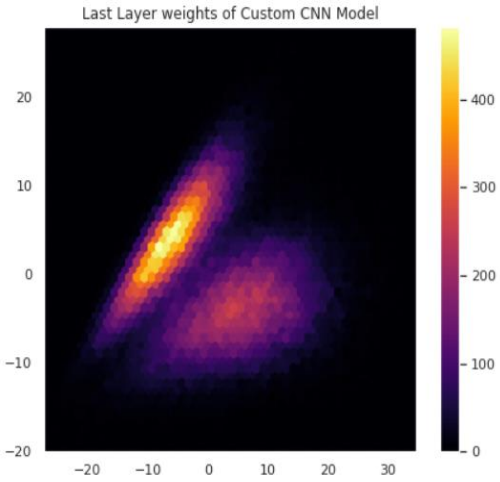


Figure. PCA component of Custom CNN model

So in the above graph, I can clearly see, how many number of data points lies near by zero or how many number of datapoints are actually zero.

2. Image Data Generators in Keras:

This is one of the existing function presents in Keras library for fetching image data into specified number of batches from given directory path. It will read image data into pixel values responsible for performing transformation on it. In our project We are using Image data generators for fetching image for training set, validation set and lastly for test set as well. Code snippet is given below:

```
image_gen_test = ImageDataGenerator(rescale=1./255.)

test_data = image_gen_test.flow_from_directory(
    base_path + 'test/',
    target_size=(224, 224),
    batch_size= 1,
    shuffle = False,
    class_mode = 'binary'
)
print("Test Dataset classes:",test_data.class_indices)
```

```
Found 20000 images belonging to 2 classes.
Test Dataset classes: {'fake': 0, 'real': 1}
```

Figure. Code snippet of Test data Generator

As shown in the above figure of code snippet, Image Data generator is defined for test dataset with name *"image_gen_test"* who is rescaling the pixel value in the range of 1 – 255. And later using function *flow_from_directory()* reading data from given *base_path* with batch size of 1. By default, the shuffle property of Image data generator is *"True"*, that means while reading image data from test data set, image was getting shuffle in random order and the original sequence of test data got changed while feeding input image to model for testing purpose, alternatively the predicted label from model also stored in shuffled order. However, the actual label of test data set was stored in original sequence which is later used to check whether classification done by model is correct or not. The result of this mistake leads to less accuracy even though accuracy of training and validation set was very high. this mistake happened because the order of predicted label is different than order of original label stored even though the predicted label was accurate.

After reading Keras documentation and comparing it with written code, the mistake got highlighted and then corrected the shuffle property to False for only test data generator which stopped the shuffling of data and preserve the original sequence of test data and predicted label also get stored in correct order, then while evaluating the results again, got very good accuracy on test data set.

A.2 Self Assessment

While working on this deepfake detection using deep learning, I have learned the importance of neural network architectures in deep learning, also learned how its powerful than machine learning algorithms for non-linear data processing. I started this project by looking deepfake images spread on online media, so thought how can we detect those deepfake images using Neural network. And then studied the GANs architectures, especially Discriminator Network, who is actually able to detect the real fake images and is the one who is great part of generating deepfake images.

I started this project with proper 12 weeks plan and that's how my organizational skills helped me to complete this project on time with proper research. My programming knowledge and knowledge which I gained through my modules helped me a lot in implementation of this project. I spent a time with my supervisor for discussing how can we leavarge the discriminator network for detecting deepfakes. To work on this project, we needed large amount of deepfake images and also same amount of real images. Therefore, I tried to develop the my own GAN network which can be able to generate fake images by using real images. I performed this experiment with small size image on MNIST dataset which are images of handwritten digits. I trained GAN network for 10000 epochs though the quality of images was very bad not even able the see digits in image. So, I decided to use someone' implementation of GAN like StyleGAN, which is developed by NVIDIA but due to lack of computational power not able generate large amount images so Online on Kaggle I found readymade generated deepfake images using StyleGAN only and also same amount of real images that too in good quantity 140K divided into 3 directories i.e. train, valid and test dataset.

After collecting this huge dataset, I started actual implementation of custom convolutional network architecture. While implementing this, I have gained the knowledge about different regularization techniques for images for example dropouts, L1 and L2 regularization and very strong effective regularization i.e. "Batchnormalization". I started with one layer of convolutional and checked the results by using confusion matrix. Sometimes I got bad accuracy as well as got very good accuracy. By using this trial and error basis, I have came to final 6 Convolutional Layer who is able to give 96% accuracy rate. After implementing my very own architecture, I compared this performance with transfer learning based model architectures, Like DenseNet121, Inception Net, this all model included in models directory. After this performed many experiments for hyperparameter tuning for example trying different learning rate for optimizer in custom CNN architecture. Also checked the performance of Custom CNN with different optimizers like SGD and RMSprop , this all experiments are included experiments directory. While implementing this, I have image preprocessing using Data augmentation, gained detailed knowledge about the GANs and

there exiting implementations how they are advanced from previous one. Learned transfer leaning architectures in depth.

For implementation of this project, I choose python language because its structured and simple programming language and also utilised karas and TensorFlow libraries to work on images. And the final part of this project is the report. I spent lots of efforts for writing this report in order to make others to understand my comparative work. Through this project journey, I gained lots knowledge about the deep learning, I learned about time management and also experienced about the how to solve technical and non-technical challenges. I believe my code, my work will be useful for future research.

A.3 How to use my project

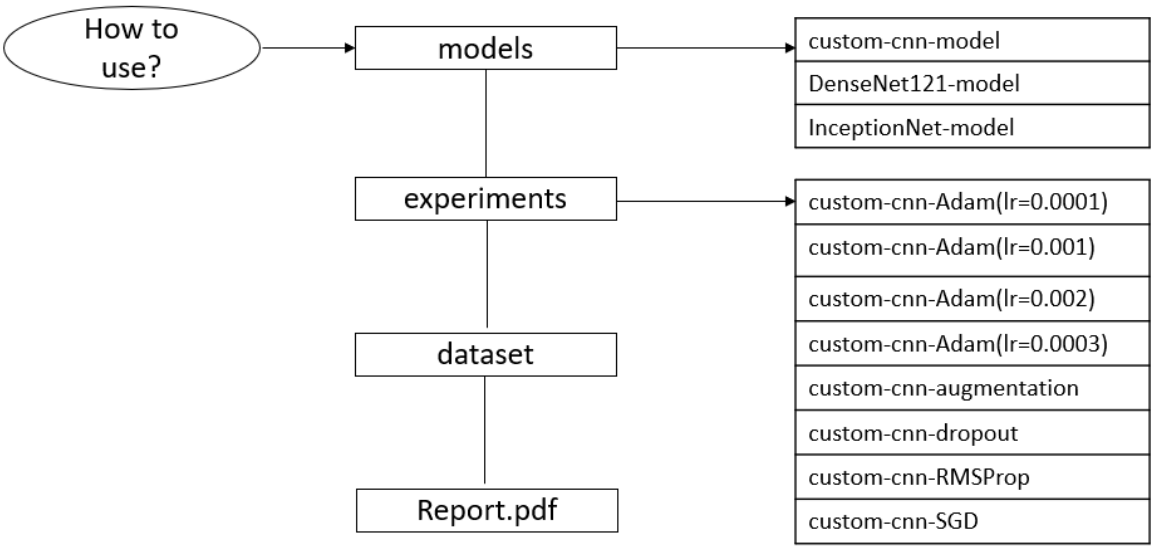


Figure. File structure of Project

As shown in above file structure, all the project files are saved in project.zip file. In models directory, main 3 models coded in Jupyter notebook file (*.ipynb*) is stored with model architecture name respectively. In second directory, all the experiments performed are stored with respective experiment name. In third folder, "dataset" as original image dataset is heavy in size around 4GB so not uploaded in folder instead given link to resource of image dataset in notepad so anyone can visit the link. Finally, report.pdf is stored.