Performance Comparison of CNN Models for Detecting GAN Generated DeepFake Images

by

Nawshin Islam
16201003
Zahid Mahbub Akash
16301146
Toriqul Islam Dipro
16201106
Mehedi Hasan Munna
16301195

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

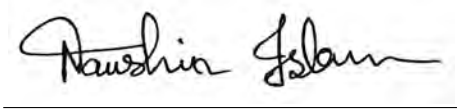
Department of Computer Science and Engineering
Brac University
October 2020

Declaration

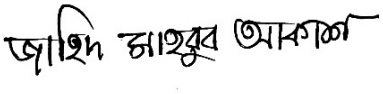
It is hereby declared that

1. The thesis submitted is my our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

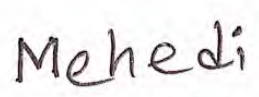


Nawshin Islam
16201003



Zahid Mahbub Akash
16301146

Toriqul Islam Dipro
16201106



Mehedi Hasan Munna
16301195

Approval

The thesis/project titled "Performance Comparison of CNN Models for Detecting GAN Generated DeepFake Images" submitted by

1. Nawshin Islam (16201003)
2. Zahid Mahbub Akash (16301146)
3. Toriqul Islam Dipro (16201106)
4. Mehedi Hasan Munna (16301195)

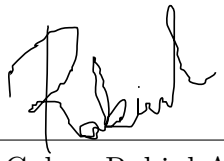
Of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 17, 2020.

Examining Committee:

Supervisor:
(Member)

Md. Ashraful Alam, PhD
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Co-Superviser and Program Coordinator:
(Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Mahbub Alam Majumdar, PhD
Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

Deep fake images are content created by humans using the most advanced technologies. The technology imitates human expression to create fake images that are very similar to the real ones. Fake image creation has become very frequent due to rapid technological advancement. Again, with that advanced technology, it is possible to detect an image being fake or real. Therefore, we came up with the idea of giving a solution that which would be the most suitable network to rely on and why. Also which approach would be able to detect whether an image is real or fake most accurately. Using approaches of Neural Networks and Convolutional Neural Network, it becomes easier to decide and believe that which one's are real for sure. These deepfakes often creates misunderstandings among people. An image will be given to the system and it can verify whether this is a finely created fake image or not.

Keywords: DeepFakes, CNN Architectures, Deep Learning

Dedication

We would like to dedicate this research work to our parents. Without their constant support, we could not achieve what we achieved so far.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Md. Ashraful Alam sir and co-advisor Md. Golam Rabiul Alam sir for their immense support and advice in our work. They helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Recently, there had been a lot of advancement in the sector of technology. All these allow people to do various types of work. These have been developed to be advantageous to human. Such technologies are blessings for the future. But everything comes with advantage and disadvantage both. Human Civilization is now facing different kinds of forgeries due to the advancement of technology. DeepFake images are are one of them.Deep fake images are content created by humans using the most advanced technologies, Artificial Intelligence and Deep Learning. The technology imitates human expression and creates fine fake images that are very similar to the real ones. This often creates misunderstanding among people. Sometimes it can be life-threatening. It is possible to use these fakes for different purposes where all are not good. People now-a-days are using technology as a way of harassment, security breaking and other unethical purposes. An example to this is creating fake images.DeepFakes are being created that seems so realistic that it becomes hard to differentiate by human eyes. It has been started very recently. It can be easily predicted that, in the upcoming days, these fake creating can be more accurate and deceiving.

1.2 Motivation

As a result of such activity, in near future a time will come when people will no longer be able to believe what they see which can easily create unrest to the society. Instead of experiencing the wonders and new inventions of technology, people will be more concerned about not falling under the trap of advancement. This situation is definitely not acceptable to people and for those who are continuously working hard to move to a better future.
Advancement of technology will keep coming with it's own advantages and disadvantages. But because of the disadvantages, we can not refrain ourselves from using those. From that perspective, we came up with means by which we can detect such DeepFakes and embrace the positive sides of technology.

1.3 Research Objective

Knowing the area of work and work that has been done till now.Knowing the method and models that others used in related area and compare among them. Finding the best possible method.

1.4 Research Problems

While going in depth of this, we have seen that many approaches are used to detect fakes. But there are some approaches which only classifies image. Detects if an image has a face or not. So we had to go through the vast area of Fake image detection algorithms, learn which ones are classifier and which ones are detector and generator. Availbale resources of gathering knowledge about these were not so much. Another problem we faced in collecting images. The testing needed a very large amount of data. This included securitu issue. Not a lot of people were eager to share their image. Thus, we had to use dataset of celebrity face.

1.5 Workplan

We collected Data of face images that contained both real and fake face. By indicating to fake face, we mean that those faces are generated by machine. In most cases, the persons of those image do not even exist in real life. After that, we processed the images and extracted features from the dataset, fed the same dataset in five different models which are MobileNet v2, VGG-19 Binary, Inception v4, DenseNet121 and MesoInception. Finally observed and compared the result.

Chapter 2

Literature Review

There had been solutions for detecting fake images. Elaborately saying, those are detection of spoof attack, not a direct image of a person or things. Spoof attack is when fake content is created to break a security system or to pretend to be someone else. In such cases, there are specified areas of spoof attack. Such as fingerprint, face, iris spoof. Systems had been developed to detect whether the bio content really belongs to a valid person or not.

Detecting iris spoof, Galbally et aI. [1] includes a feature of eye movement, such as focus, motion, occlusion and pupil dilation. Whereas, another detection depends on the texture of the pupil. In the case of face spoofing, Schwartz et aI. [2]have taken color, texture, and shape of face region into consideration and applied the Partial Least Square classifier to decide real or fake.

Menotti et aI.[3] mentioned about Architectural Optimization and Filter Optimization approach. Architectural Optimization uses layer that is fixed by certain parameters. In this method, a layered network is first obtained. Then architecture is evaluated according to SVM results. The SVM score is kept obtaining until the termination criteria is met. Then according to the result, the best convolutional architecture is found. For that, filter weights are generated randomly. Whereas, Filter Optimization calculates the weight of image keeping a fixed Architectural Optimization (cf10-11) using the back-propagation method. Then again, a new architecture is derived in this method which decreases the size of the weight and allows it to increase the size of the input image from 32*32 to 128*128. The new architecture is called spoofnet. This spoofnet is again trained as the modified architectures decreased the learning capacity of the network. Finally, from 32*32 images, five 24*24 image patches are cropped. The standard evaluation protocol of all benchmark is followed. Methods being evaluated in terms of detection accuracy (ACC) and half total error rate (HTER). And when all these were done, they retrieved prediction scores from the testing samples, calculated threshold value of above which samples are predicted as attacks.

According to [4], In the forensic area, different feature measuring trait-specific quality properties have already been used for iris and fingerprint applications. So, measuring the ridge and valley frequency may be a good parameter to detect fingerprint spoof but not for iris. On the other hand, the amount of occlusion of the eye is valid as an iris anti-spoofing mechanism but will have little use of detecting fake fingerprints. Measure like the Mean Squared Error respond more to additive noise, whereas other such as the spectral face error are more sensitive to blurriness, while gradient-related features react to distortions concentrated around edges and texture. At first CNN (Convolutional Neural Network) is used for the detection of fake images. In 2014 GAN (Generative Adversarial Network) was introduced which has a generator and discriminator. The generator can generate the fake image and discriminator target is to distinguish between real and fake images. There are many algorithms that were proposed. But in this paper, the researcher used a different process that can detect the fake image and also GAN generated images with a high macro average of f1 score of 0.9865. They proposed a structure with depth-wise separable convolution as the main component which with a smaller number of parameters compared with the one consists of traditional convolution.

In other researches, they also proposed some techniques. Like Wang et aI. [2] proposed that splicing traces easy to find in yCrCb than RGB. [5] Agarwal and Chand proposed that a method under wavelet transform and texture descriptor. Bayer proposed a new version of CNN that adaptively learn manipulation detection features. The whole process divided into three parts, which are Feature Extraction, Image Scaling and Classification. As per, Feature extraction in YCrCb color Y is the main component of image and Cr and Cb is the purity component. Tampering traces is more pronounced and had a smoother edge and also there is edge information between background and foreground . In feature extraction they convert the RGB image into YCrCb color format. Y contains the main component details of the image where CrCb contains less information. That's why this model focus on analyzing image edge. for this SCHAR operator is important for detecting image edge detection .it is the variant of SOBEL operator. It has two operators one is for detecting vertical edge another for detecting. horizontal edge. In Image scaling, they convert the image edge information matrix to the GLCM. GLCM can resize the edge information matrix to a model size without losing data. If the edge is rough than it is an untampered area if it is smooth it indicates it is a tampered area. We calculate GLCM in four direction (0,45,90,135) degree. Guo et aI. [6] proposed Histogram-based fake colorized image detection (FCID-HIST) and Feature encoding-based fake colorized image detection (FCID-FE). This paper shows decent result for both proposed methods against multiple state-of-the-art colorization approaches. They evaluate the performances of the proposed methods by selecting parameters for FCID-HIST and FCID-FE and detecting different fake images generated by state-of-the-art colorization approaches. The results demonstrate that both FCID-HIST and FCID-FE perform decently against different colorization approaches and FCID-FE gives more consistent and superior performances compared to FCID-HIST in most of the tests.

For large size of image convolutional networks works great recently. krizhevsky [7] achieved better accuracy. Karen Simonyan and Andrew Zisserman [8] works on depth of the image. They add more convolutional layer because of increasing the depth of the network. They measure the improvement based on the principle of (krizhevsky,2012); (ciresan et al.2011). For training the data they fixed the size of the data which is 224x224. They are using a filter with a field of 3x3 that's why they capture the small changes of the image. On the other hand, they also utilize 1x1 filters. Which is the linear transformation for input. They also fixed the convolution stride to 1 pixel. Because of using 3x3 field they can manage the image at every pixel with stride 1. They evaluate in many scales like single scale evaluation, multi scale evaluation, multi-crop evaluation. Their work evaluated deep convolutional networks. For large scale of image, they are going up to 19 layers.

Chapter 3

Data Analysis

3.1 Dataset Collection

A data set is a collection of information. In Machine Learning, we require a training data set. It is the real information set utilized to prepare the model for performing different activities. In deep learning collecting a huge and authentic dataset is the main focus to conduct the research properly. Also gathering the data from a reliable source and understanding the data is very essential. One of the most important procedure to study in this field is data preparation since deep learning algorithms depend on data collection. Data preparation includes characterizing the fitting strategy for gathering information.

Deepfakes are manufactured media in which an individual in an existing picture or video is supplanted with somebody else's resemblance [9]. Whereas the act of faking substance isn't modern, deepfakes use superior approach of machine learning and counterfeit insights to control or create ocular and sound substance with a tall dynamic to hoodwink. The most machine learning strategies utilized form deepfakes are based on deep learning and including preparing generative neural network architectures, such as auto encoders [10] or generative adversarial networks (GANs) [11]. However most deep fake innovation is based on generative adversarial networks (GANs). GANs empower calculations to move past classifying information into producing or making pictures. The AI advances that control deepfakes and other altered media are quickly advancing, making deepfakes so difficult to identify that, indeed human can't dependably state the distinction. Deepfake Detection is outlined to incentivize fast advance in that region by welcoming members to compete to make better approaches of recognizing and avoiding controlled media..

To form a deepfake video of somebody, a maker would to begin with prepare a neural network on numerous hours of genuine video film of the individual to donate it a practical "understanding" of what he or she looks like from numerous points and beneath distinctive lighting [G]. Deekfakes can be image, video or even audio. To make a deepfake image or video the creator needs to work with huge information at the same time to spot the deepfake we also need broad dataset. In a dataset different expression of a same person are being recorded as images to train the model more accurately Usually Deepfake dataset generated by several Deepfake, GAN-based, StyleGAN and non-learned methods. However, GAN generated deepfakes are quite

impossible to define whether it is real or fake..

In our study we have used three different datasets to train and test our models and spot a deepfake effectively. We have collected the dataset from a reliable source Kaggle which is the number one halt for information science devotees all around the world. Even we have used one of the largest deepfake dataset in our study to train models.

3.2 Data Processing

3.2.1 FFHQ Faces Data Set Processing:

Flickr-Faces-HQ (FFHQ) dataset contains very intense quality image dataset of human faces which is produced by generative adversarial systems (GAN). The FFHQ dataset comprises of 70,000 high-quality PNG images. The quality of the image is very high which is at 10241024 resolution. It contains significant variety based on age, class and image foundation. It too has great scope of extras such as shades, caps, eyeglasses etc. The images were taken from Flickr and and automatically aligned and cropped through dlib. Different programmed channels were utilized to prune the set, and at long last Amazon Mechanical Turk was utilized to evacuate the periodic statues, works of art, or photographs of photographs.

Figure 3.1: Sample Images of FFHQ Dataset

The images have been resized to 128*128 pixel. All 70,000 images are in PNG format. The dataset makes sure that it doesn't contain any duplicates images. For utilize cases that require separate training and validation sets, we have designated the primary 60,000 pictures to be utilized for training and the remaining 10,000 for validation.

3.2.2 Deepfake 150*150 Dataset Processing

DeepFake(150*150) is a massive dataset having 112 thousand files. Every image is enlarged by 80 pixels and resized to 150*150. Each and every images of this dataset

are generated by MTCNN crop. The dataset is divided into 50 directories having different range of images and 50 .json files. The images mostly contain different expressions of several people to train our models more accurately. We have used this dataset (MTCNN crop generated fake images) to train our models so that they can understand how a deepfake looks like and the models can easily distinguish between real and fake image.

Figure 3.2: MTCNN Generated Fake Images

3.2.3 Challenges with Dataset

Deepfake Detection Challenge is a very large video dataset. The size of the dataset is 470GB which contains 800 videos as .mp4 format. It has a bunch of .mp4 records, part into compressed sets of 10GB a piece. A metadata.json goes with each set of .mp4 records, and contains filename, label (REAL/FAKE), unique and part columns, recorded underneath beneath Columns.

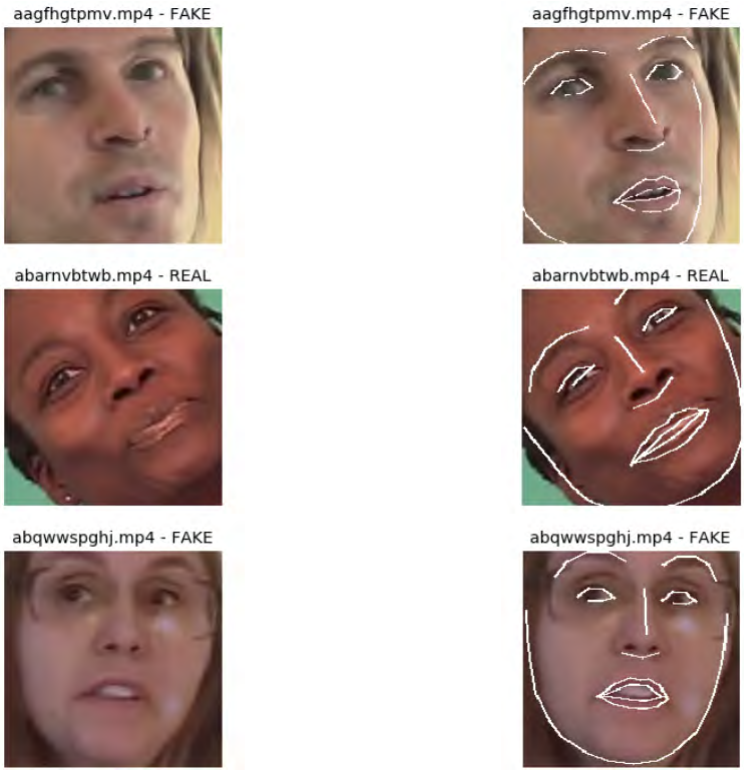


Figure 3.3: Sample REAL and FAKE frames Extracted from the Dataset

The dataset contains train-sample-videos.zip, test-videos.zip and sample-submission.csv. the train-sample-videos.zip is a ZIP record involving a sample set of training videos and a metadata.json with names. It includes 400 files in mp4 format and one .json file. test-videos.zip is a zip record containing a little set of videos to be utilized as an open approval set. To get it the datasets accessible for this competition, audit the Getting Started data. It also contains 400 files in mp4 format. sample-submission.csv is a test accommodation record within the redress organize.

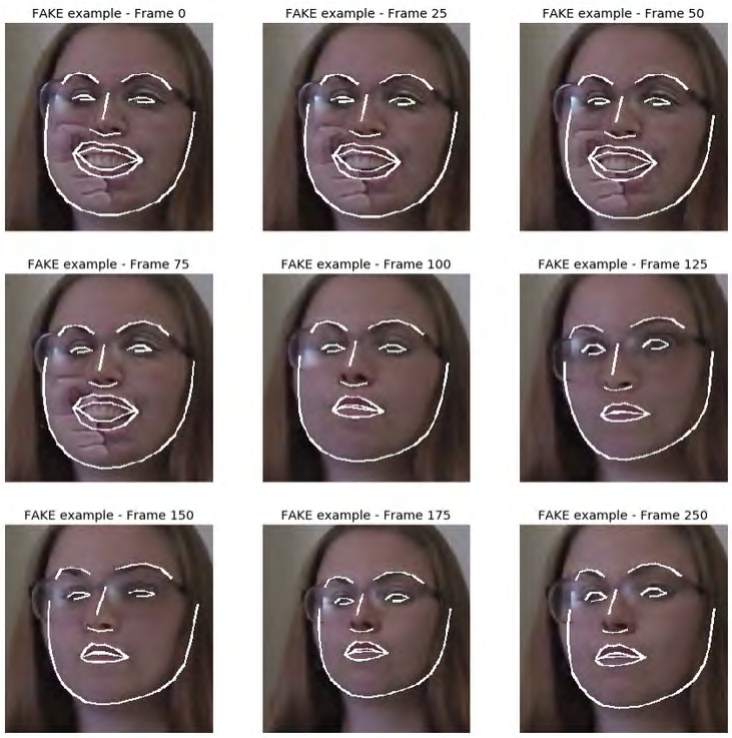


Figure 3.4: Extracting Fake Frames from the Video Dataset

Figure 3.5: Distribution Labels of Deepfake Detection Challenge Dataset in the Training Set

3.3 Feature Extraction

Feature extraction may be a handle of dimensionality reduction by which a starting set of crude information is decreased to more reasonable bunches for technology. A function of these large information sets is a huge number of variables which require a great deal of computational resources to process. Extraction of features is the term for strategies that select and /or assemble factors into features, successfully lessening the sum of information that must be prepared, whereas still precisely and totally depicting the first information set. Feature Extraction is very useful when we need

to minimize the number of resources without losing any important information or data. It can also reduce the redundant information and machine's effort to process the data. It also increases the accuracy of trained models by extracting feature from the input data. In our study, we have used three methods for extracting the feature which are PCA, tSNE and DBSCAN.

3.3.1 Principal Component Analysis (PCA)

PCA is a strategy to bring out solid designs in a dataset by suppressing varieties. It is utilized to clean information sets to form it simple to investigate and examine. In our research we used PCA in such a way that all the images of a person or similar to that person stay together so that we can identify if a deepfake image is created by this image or not.

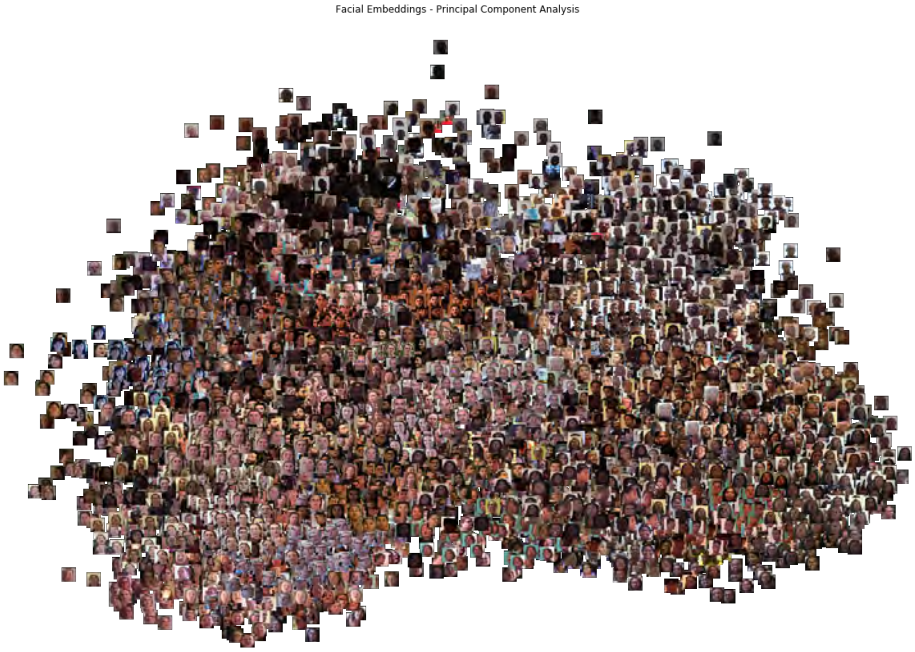


Figure 3.6: Clustering by Principle Component Analysis(PCA)

3.3.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN alludes to unsupervised learning strategies that distinguish particular groups/clusters within the information, based on the thought that a cluster in data space could be a coterminous locale of tall point thickness, isolated from other such clusters by touching districts of moo point thickness. DBSCAN uses two key parameters which are

1. eps: The distance that indicates the neighborhoods. Two focuses are considered to be neighbors on the off chance that the separate between them are less than or rise to to eps.

2. minPts: Minimum number of data points to define a cluster.

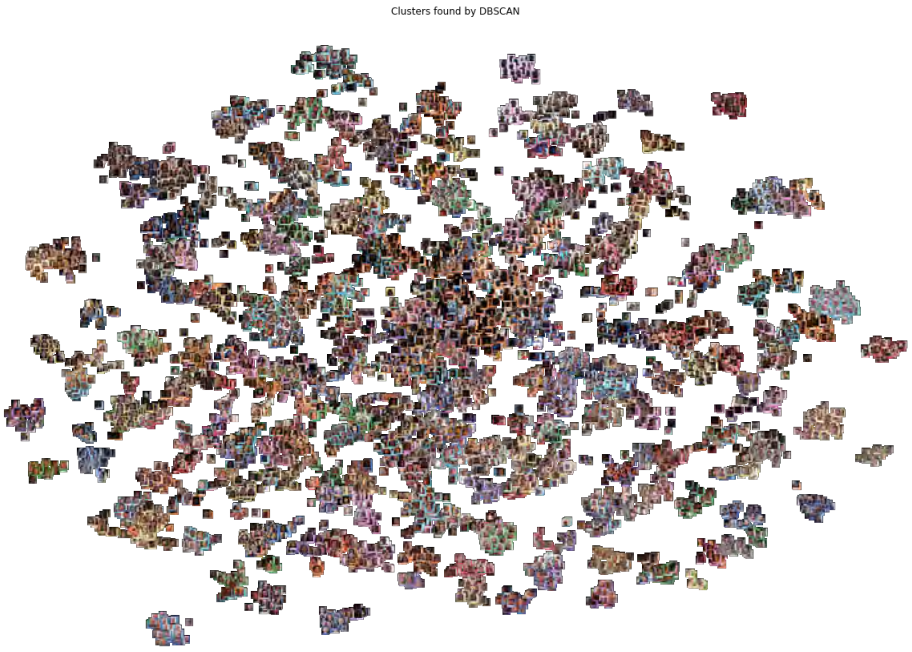


Figure 3.7: Clustering by Density-Based Spatial Clustering (DBSCAN)

We used this unsupervised method to find the underlying structure of our data. DBSCAN clustering algorithm looks for similarities and dissimilarities among the data and similar data points are grouped together.The figure 03 above shows that, each and every face mask are colorized and packed in a way that all the similar images are together and by which we can easily understand what is the origin of a picture and from where it was created.

3.3.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a machine learning approach which takes a high dimensional data and reduce in a low dimensional graph. We have used this technique to cluster our data. tSNE clusters the input data such a way that all the similar face stays in a group or selected area and shows if any picture is being created from the source image.

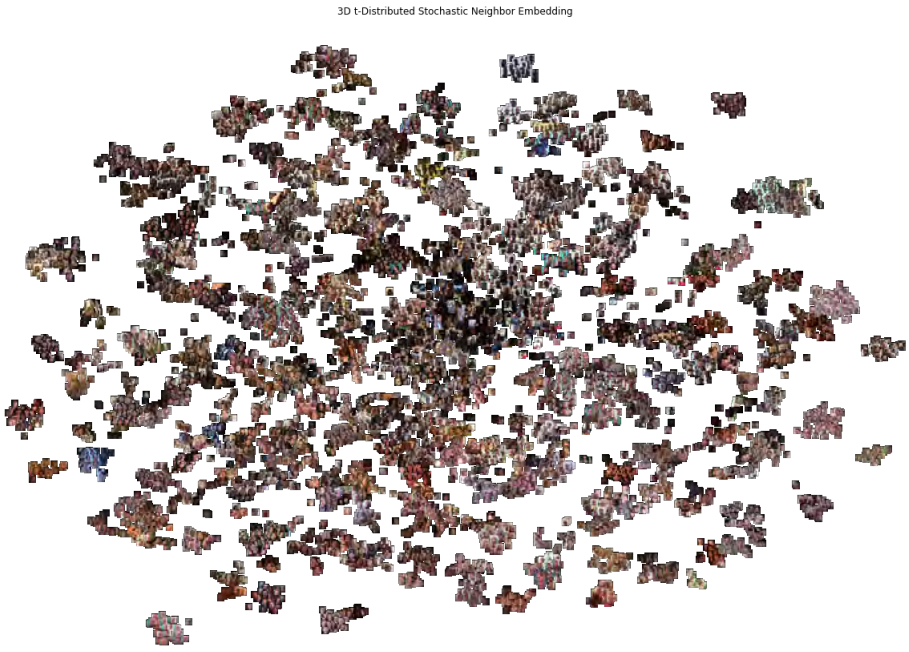


Figure 3.8: Clustering by (t-SNE)

3.3.4 AnnoyIndex

Approximate Nearest Neighbors with Annoy is a small library written to provide fast and memory-efficient nearest neighbor lookup from a possibly static index which can be shared across processes. To find out the similarities between two images we need to know the annoy index of these images. To conduct our study, we also figured out the annoy index for our dataset. Figure 04 shows annoy index and duster of a particular video.

Figure 3.9: Similar Images Annoy Extracted from the Dataset

Based on annoy index we can understand how much chunk has created. In our study, 131 clusters are spread across more than one data chunk. Figure 3.9 to 3.13 shows some of the examples of chunks.

Figure 3.10: Cluster 77 with 2 chunks

Figure 3.11: Cluster 186 with 3 chunks

Figure 3.12: Cluster 52 with 4 chunks

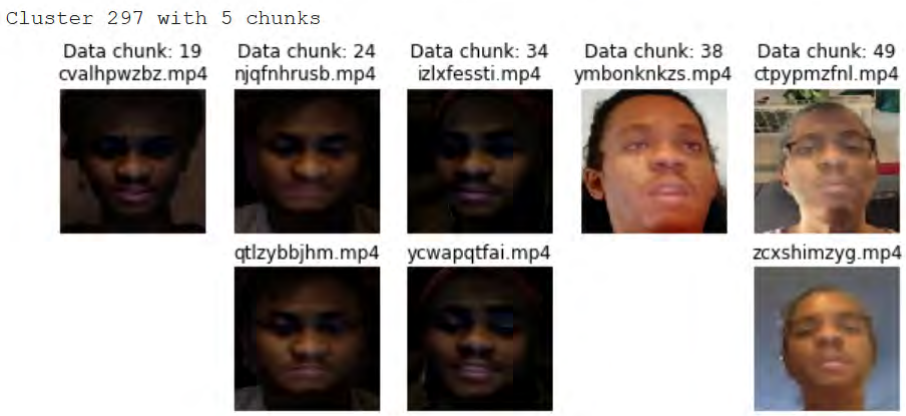


Figure 3.13: Cluster 297 with 5 chunks

Chapter 4

Algorithms

4.1 MobileNet V2

MobileNet is an architecture of CNN for Image Classification, Mobile and Embedded Vision Applications. It is based totally on Depthwise Separable Convolution which diminishes the model size and complexity and mainly precious for versatile and embedded vision applications [13]. It is very much visible that massive models like VGG16, ResNet-50 aren't very smart choice for mobile system because of their vast architecture and computational power. It is always a wise thought to utilize a model which is efficient and fast while considering a neural network in our mobile applications. For instance, a real time video where we want to form forecasts regularly. A neural network like ResNet-50 will consume a lot of power and is unacceptable to use in real- time.

There is a trade-off as in most things engineering. In common, the bigger model is, the superior comes about it gives. But the slower it runs and the more vitality it employments up, too. Expansive models rapidly make the phone hot and deplete the battery which is not a suitable person experience. In contrast, a simple model run faster, consume less energy and typically gives the less accurate outputs.

Mobile ML practitioner recommends to use a model architecture which fast at the same time gives very good results. Such as MobileNet V1, MobileNet V2 which offers good enough outputs. There are other models like VGGNet, ResNet but what makes MobileNet exceptional that it takes very much less computation electricity to run or follow transfer learning to. This makes it a great match for mobile devices, GPU-free embedded systems and computers or low processing performance, greatly sacrificing the accuracy of the outputs. It is additionally pleasant suitable for web browsers as browsers have drawback over computation, graphic processing and storage.

Modern image detection system which requires millions of parameters costs a lot of training data set and high-performance computing power and energy thus it becomes very inconvenience in daily use. Machine Learning has changed the idea and introduced some environmentally friendly technologies that can be very much efficient in everyday use. MobileNet is such an architecture which uses Convolution Neural Networks (CNN) method, limited resources and decent number of training data

for image finding on mobile devices and embedded systems. MobileNet supplants costly convolution layers by a cheaper depthwise seperable convolution, which could be a 33 depthwise convolution layer taken after by a 11 convolution layer [12]. This requires a parcel less learned parameters than a standard convolution but roughly does the same stuff.

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
Conv dw / s1 5 × Conv / s1	3 × 3 × 512 dw 1 × 1 × 512 × 512	14 × 14 × 512 14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Figure 4.1: Full Network Architecture of MobileNet

In MobileNet V2 the depthwise convolution layer is located in the middle of the architecture. There is another 11 convolution known as bottleneck layer right after the depthwise layer which diminishes the number of channels. The number of channels entering the block is as same as the number of channels going out that is 64, a residual relation is also present. This helps increase the flow of gradients during the backward transfer, much as in ResNet. The activation function is ReLU6 and there is no activation present behind the bottleneck layer. The complete architecture of MobileNet v2 consists of 17 of those building blocks in one row. A standard 11 convolution, a global average pooling layer, and a classification layer follow this.

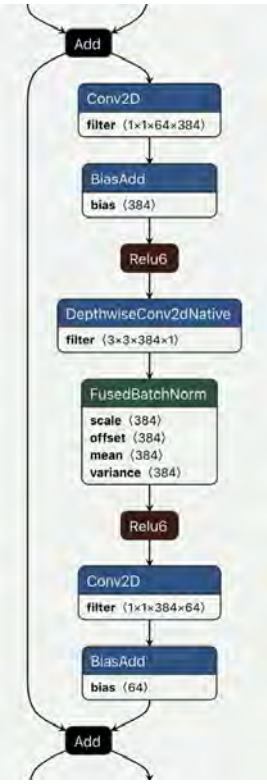


Figure 4.2: MobileNet v2 Architecture

Depthwise Separable Convolution

In MobileNet architecture DepthwiseSeparable Convolution is utilized to decrease the model size and complexity which is especially valuable for versatile and implanted vision applications.

- Model size: Less Number of Parameters
- Complexity: Less Multiplications and Additions(Multi-Adds)

Two straightforward worldwide hyper-parameters that productively exchange off between latency and accuracy are presented as Resolution Multiplier and Width Multiplier . Depthwise separable convolution is a depthwise convolution followed by way of a pointwise convolution as follows:

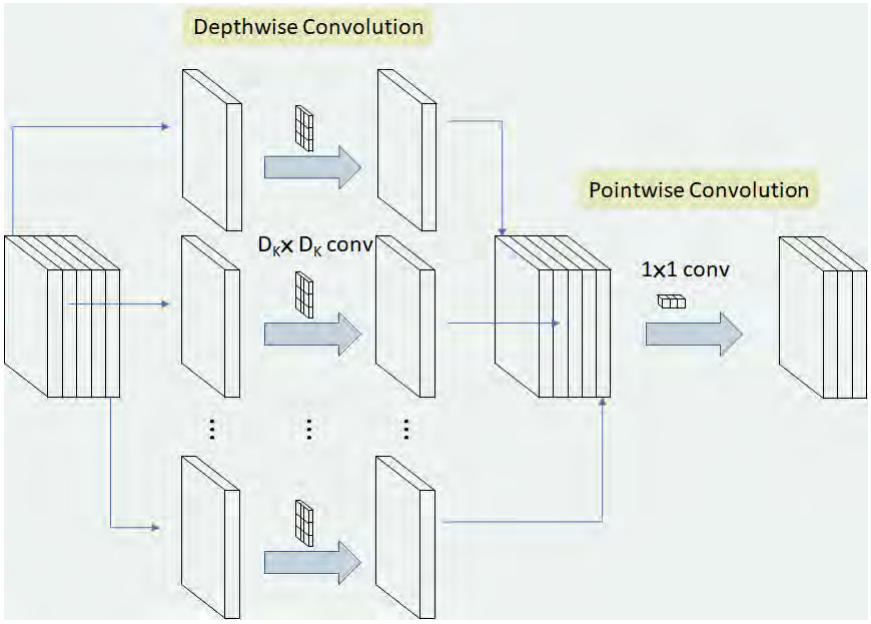


Figure 4.3: Depthwise Convolution followed by a Pointwise Convolution

1. Depthwise convolution is the spatial convolution of the channel-wise DKDK. Let's assume we have 5 channels in the figure above then we will have 5 DKDK spatial convolution.

2. Pointwise convolution actually is the 11 convolution to change the dimension.

The use of filters over all input channels and the combination of these values is carried out in a single step in quality convolution. On the other hand, Separable Depthwise Convolution divides it into two sections. The first is Depthwise Convolution, which is the stage of filtering, and then Pointwise Convolution, which is the stage of combining. A single input channel at a time is transformed by Depthwise Convolution. This is different from the usual convolution.

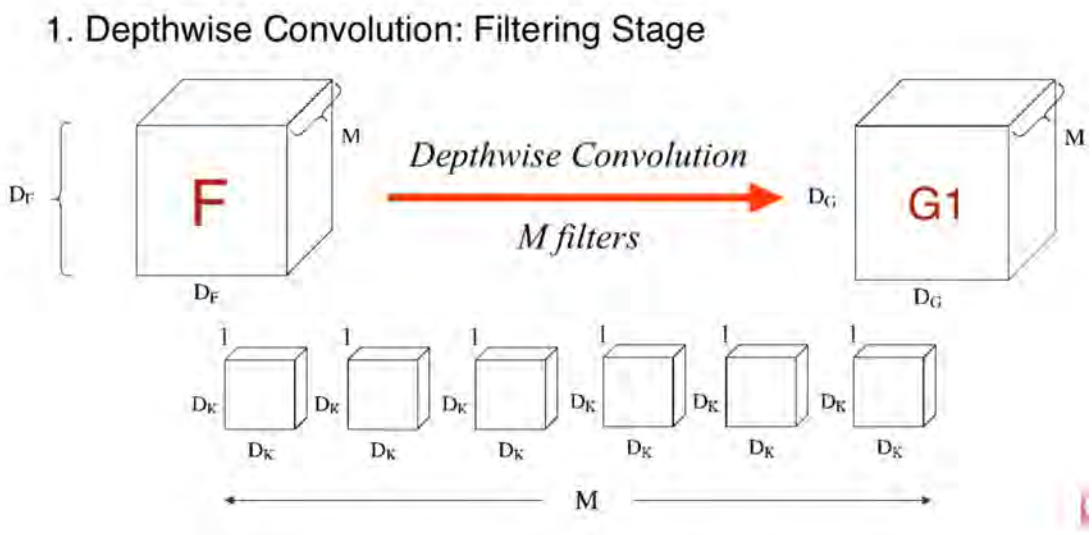


Figure 4.4: Depthwise Convolution

This is the end of the first phase that is the end of Depthwise convolution now this is succeeded by Pointwise convolution. Pointwise convolution involves performing the linear combination of each of these layers. Here the input is the volume of shape DGDGM. The filter K has a shape 11M. This is basically a 11 convolution operation over M layers. The output will thus have the same input width and height as the input DGDG for each filter. Pretending that we wish to use some N such filters then the output volume arrives DGDGN.

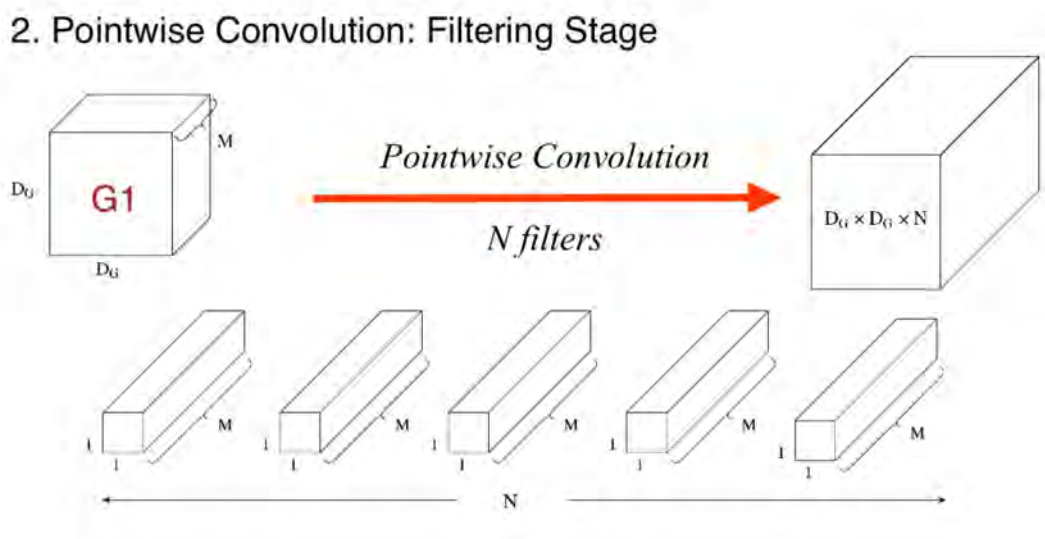


Figure 4.5: Pointwise Convolution

Lets take a look at the complexity of this convolution we can split this into two parts as we have two phases. First, we calculate the number of multiplications in depthwise convolution. So here the kernels have a shape DKDK1. So the number of multiplications of one convolution operation is all DK times DK, DK square where applied over the entire input channel. Now such multiplication are applied over all M input channels. For the each channel we have a different kernel and hence the total number of multiplications in the first phase that is Depthwise convolution is M times DG square time DK square. Next we compute the number of multiplications in the second phase that is Pointwise convolution. So the number of multiplications for this kernel is DG time DG times M. So for some N kernels will have N times DG times DG times M such multiplications and thus the total number of multiplications is the sum of multiplications in the Depthwise convolution stage plus the number of multiplications in the Pointwise convolution stage. We can take M times DG square common now we compare the standard convolution with Depthwise convolution we get the ratio as the sum of reciprocal of the squared dimensions of the kernel DK to put this into perspective of now effective Depthwise convolution. Let us take an example, so consider the output feature volume N of 1024 and a kernel of size 3. That's DK is equal to 3 plugging these values into the relation. We get 0.112 in other words, standard convolution has 9 times more the number of multiplications as that of Depthwise Separable convolution this is a lot of computing power we can also quickly compare the number of parameters in both convolutions. In standard convolution each kernel has K times DK times M. Learn about parameters since there are N such kernels there are N times M times DK squared parameters in Depthwise

Separable Convolutions will split this once again into two parts. In the Depthwise convolution phase we use M kernels of shape DKDK. In Pointwise Convolution we use N kernels of shape 11M. So the total is M times DK square plus M times N. Taking the ratio we get the same ratio as we did for computational power required. The MobileNet architecture is built on depthwise separable Convolutions. In MobileNet architecture layers are taken after by a batchnorm [13] and ReLU nonlinearity with the exemption of the ultimate fully connected layer which has no nonlinearity and nourishes into a softmax layer for classification. Figure 2 contrasts a layer with regular convolutions, batchnorm and ReLU nonlinearity to the factorized layer with depthwise convolution, 11 pointwise convolution as well as batchnorm and ReLU after each convolutional layer. Down sampling is handled in both the depthwise and first-layer convolutions with strided convolution. A final average pooling in front of the fully connected layer reduces the spatial resolution to 1 MobileNet has 28 layers, counting depthwise and pointwise convolutions as separate layers.

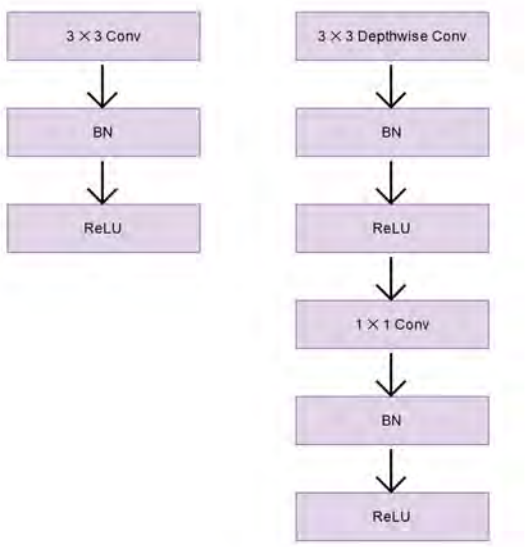


Figure 4.6: Standard Convolution (Left), Depthwise separable convolution (Right) With BN and ReLU

Applying Dataset on Model

MobileNet's last (trainable) convolution layers are completely optimized to classify whether the image is fake or real. In this model we trained the dataset with 10 epochs to find the validation loss, training loss and accuracy. Adam optimizer has been used with the Learning Rate 0.001 and Batch Size 128. After feeding the dataset into the model we got a decent accuracy which is
89.4%. The figure given below shows some sample real and fake picture from our datasets. The pictures labelling by 0 are the real image and 1 are the fake image detected by our model.

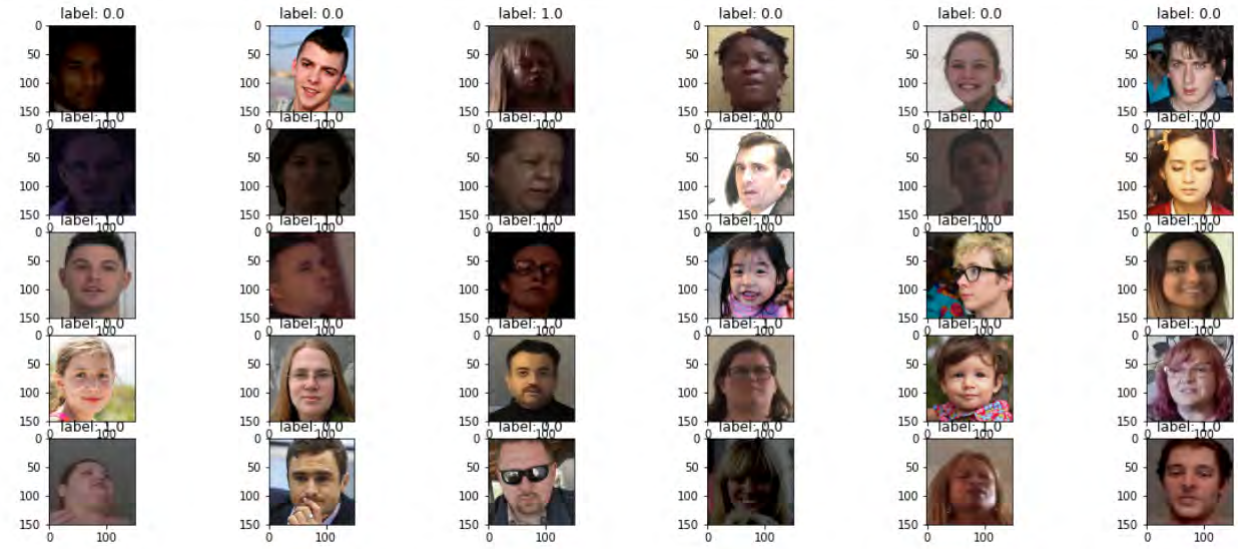


Figure 4.7: Fake image labeling by 1 and Real image labeling by 0

Result

The model accuracy of MobileNet V2 is 89.4%

Iteration	Training Loss	Validation Loss	Accuracy
1	0.3733	0.2793	0.893828
2	0.3524	0.2739	0.893664
3	0.3457	0.2714	0.894073
4	0.3431	0.2713	0.893582
5	0.3401	0.2701	0.894155
6	0.3389	0.2737	0.894073
7	0.3347	0.2680	0.89399
8	0.3365	0.2708	0.894483
9	0.3342	0.2686	0.894237
10	0.3313	0.2681	0.894646

Table 4.1: Training loss, Validation Loss and Accuracy of the model, MobileNet v2

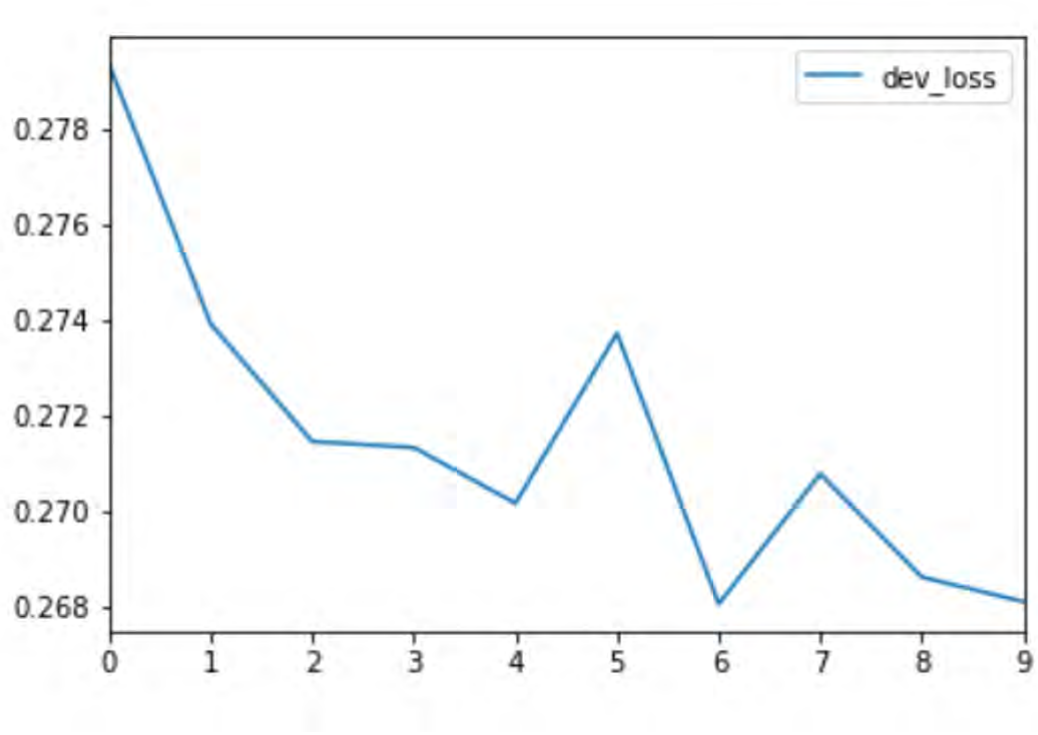


Figure 4.8: Validation Loss Plot

The figure given above shows the validation loss plot where validation loss is given on Y-axis and epoch's number is given on X-axis. For each case we used validation to measure the accuracy and if an epoch gives a better accuracy we saved that epoch.

Figure 4.9: Confusion Matrix of MobileNet V2

From the figure 4.9 we can see that,

- There are 5274 real images and they are predicted as "REAL"
- There are 834 real images and they are predicted as "FAKE"
- There are 453 fake images and they are predicted as "REAL"
- There are 5655 fake images and they are predicted as "FAKE"

Meanwhile, we got the value of F1 score for MobileNet V2 which is 0.8388465467117153

4.2 Inception v4

Inception is a module of Convolutional Neural Network. To explain Inception, defining CNN is must. Convolutional Neural Network is widely known as CNN. It is a densely connected network containing multiple layers. There are 3 layers in a Convolutional Network. Those are, Convolutional layer, Max-pooling Layer and Fully Connected Layer. The task of Convolutional layer is to obtain the data of the image whereas the Max-pooling layer does the downsampling operation. The resultants of max pool layer are fed to the fully connected layer.

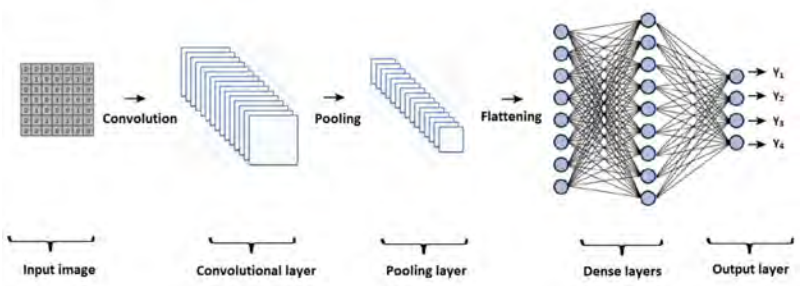


Figure 4.10: Architecture of CNN

Description

Inception module was introduced to reduce the expense of computation. The basic version of inception module works by performing a convolution on the input by applying three filters of different sizes. Those are (1x1, 3x3, 5x5). After that, max pooling is performed. Then, the resulting outputs are concatenated together, where one max-poling is directly concatenated. After that,it is sent to the next layer. By structuring the CNN to perform its convolutions on the same level, the network gets wider gradually, not deeper [14].

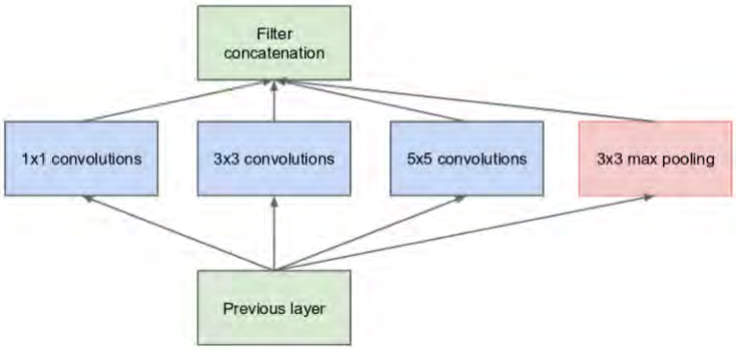


Figure 4.11: Inception Module

The module has been further designed to reduce the computational cost even more.And an extra 1x1 convolution is added before the 3x3 and 5x5 layers.1x1 convolutions are cheaper than those of the 5x5 convolutions.

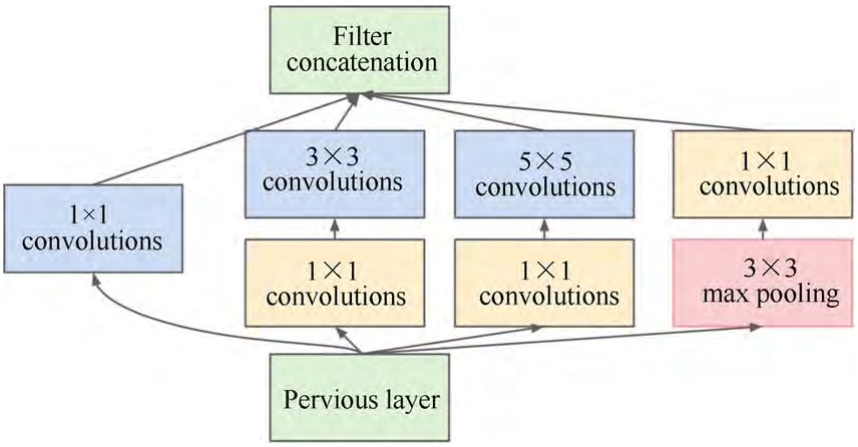


Figure 4.12: Inception Module with Dimension Reduction

We applied our dataset on Inception v4. It is possible to train this without partitioning the replicas, with memory optimization to backpropagation along with the previously used techniques of the earlier versions. The figure 4.14 shows the complete architecture of Inception v4, whereas 4.15 and 4.16 indicates the expansion of stem and Inception-A, Inception-B, Inception-C.

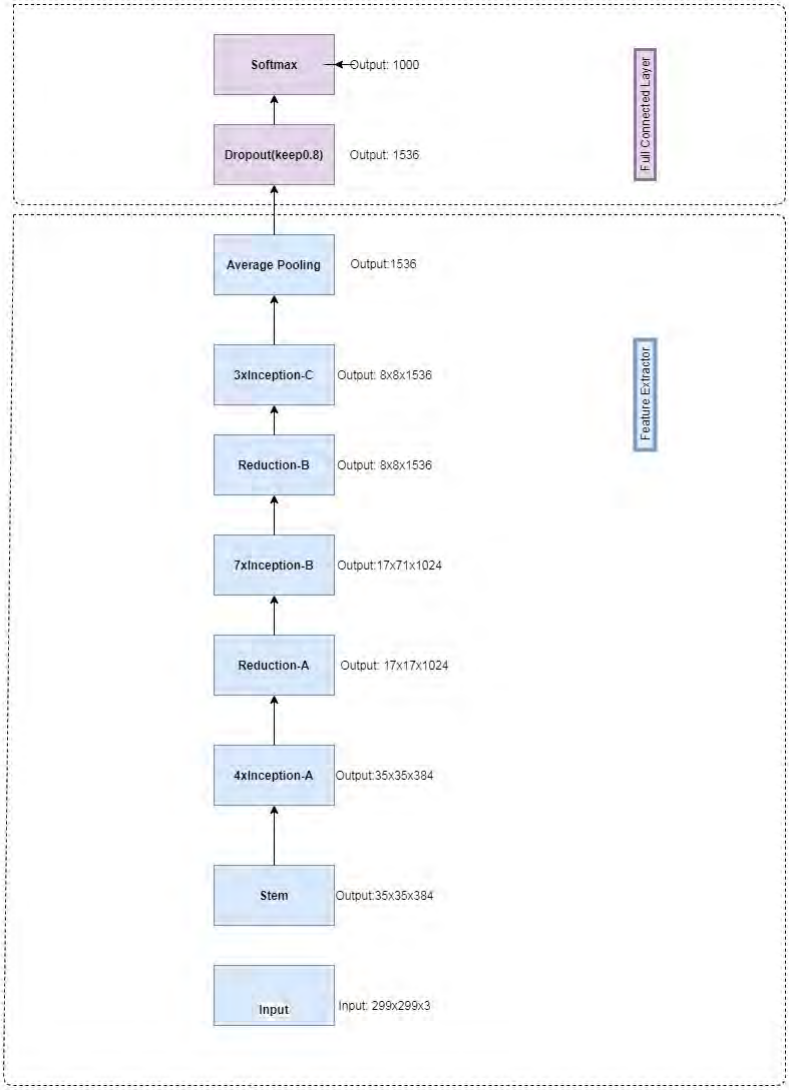


Figure 4.13: Complete Network Schema of Inception v4

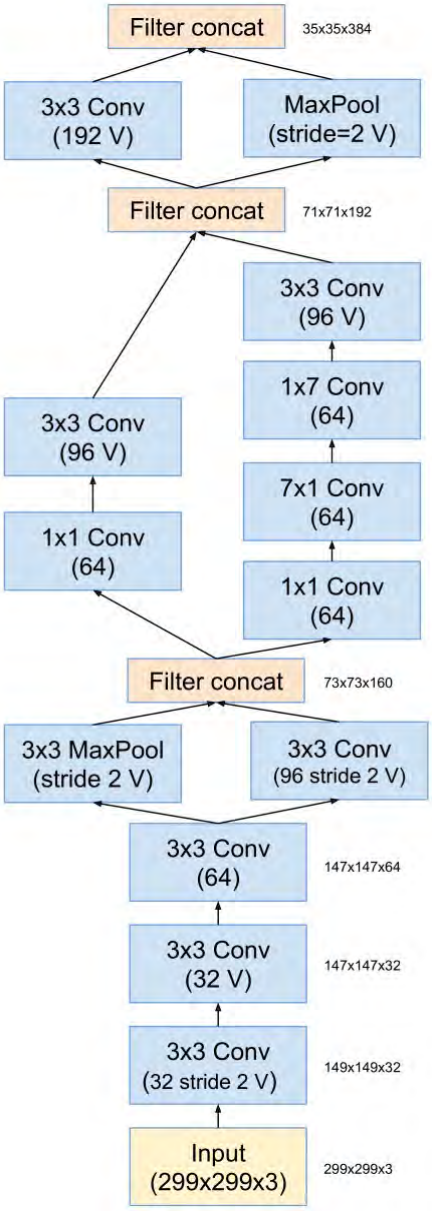


Figure 4.14: Expansion of Stem

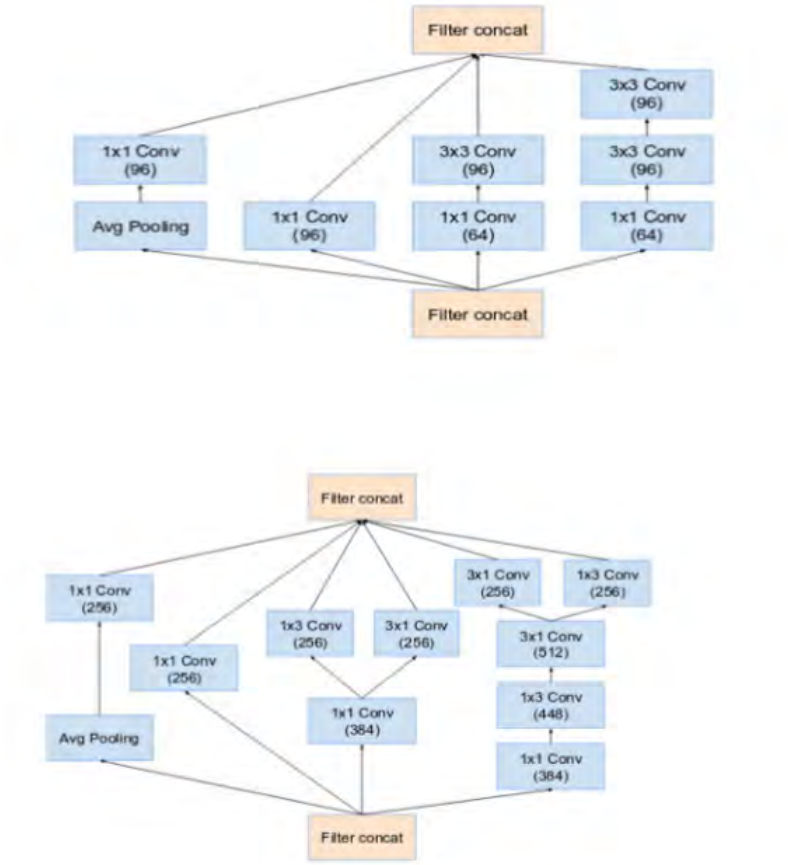


Figure 4.15: Expansion of Inception-A, Inception-B, Inception-C

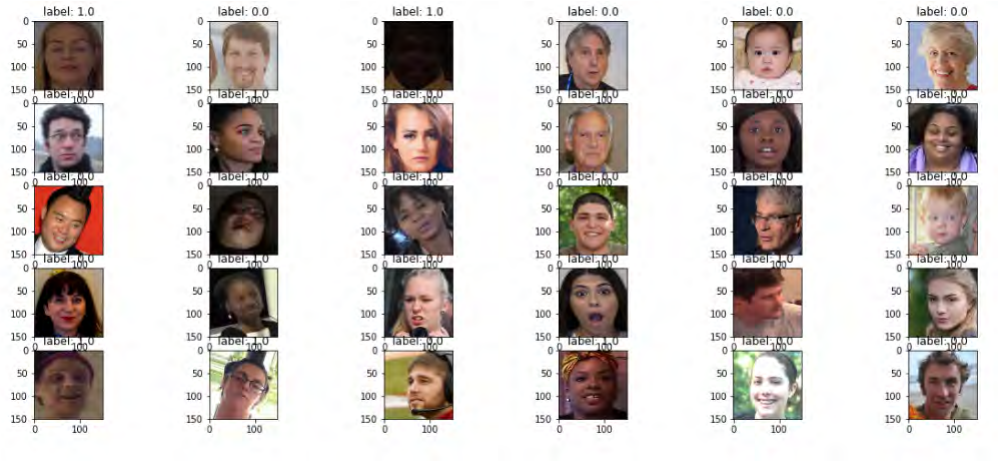


Figure 4.16: Fake Image Labeling by 1 and Real Image labeling by 0

Applying Dataset on Model

Inception v4's convolution layers are completely optimized to classify whether the image is fake or real. In this model we trained the dataset with 10 epochs to find the validation loss, training loss and accuracy. Adam optimizer has been used where our Learning Rate is 0.001 and the Batch Size is 128. After feeding the dataset into the model we got a decent accuracy which is 84.32%. The figure given below shows some sample real and fake picture from our datasets. The pictures labeling by 0 are the real image and 1 are the fake image detected by our model.

Result

Iteration	Training Loss	Validation Loss	Accuracy
1	0.5132	0.4271	0.804355
2	0.4922	0.3991	0.828831
3	0.4905	0.3992	0.830141
4	0.4874	0.3902	0.841110
5	0.4861	0.3905	0.838982
6	0.4858	0.34103	0.11395
7	0.4814	0.3809	0.843238
8	0.4810	0.4015	0.821709
9	0.4782	0.3853	0.839473
10	0.4766	0.3931	0.832105

Table 4.2: Training loss, Validation Loss and Accuracy of the model, Inception v4

The accuracy of the model is 84.32%.And the graphical representation of this is as follows:

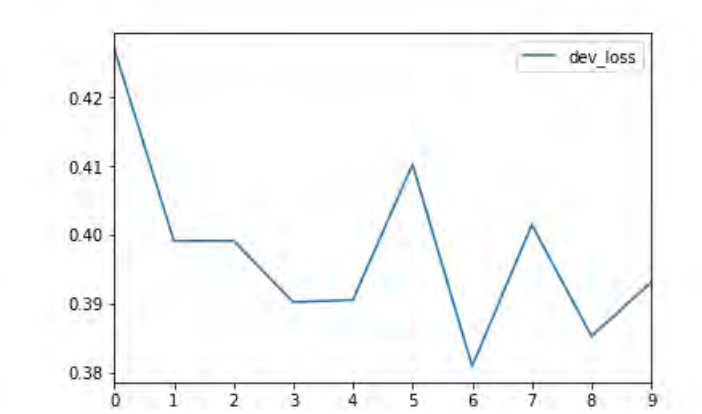


Figure 4.17: Validation Loss Plot

Confusion Matrix of Inception v4

Figure 4.18: Confusion Matrix

And the value of F1 score for Inception v4 is 0.8978328173374612.

- 4827 real images and they are predicted as "REAL".
- 1281 real images and they are predicted as "FAKE".
- 770 fake images and they are predicted as "REAL".
- 5338 fake images and they are predicted as "FAKE".

4.3 DenseNet121

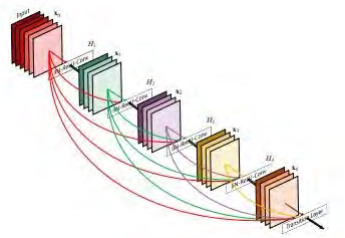


Figure 4.19: 5-layer Dense Block with a Growth Rate

In the past years of experience, we know that Convolutional network has applied in image and video recognition. It can be enough accurate and also worth applying for training if we can shortly connect the input and output layers.It has already proven it's prosperity in identifying traffic sign, faces, objects. A seemingly novice research problem arises with the information regarding the input passes through several layers. It gives rise to the possibility of information being 'washed out' or even vanished when it reaches the end that is the start of the network. This is a result of CNN's becoming increasingly deep.
Signals from one layer to its next are bypassed through identity connections by ResNets and Highway Networks. ResNets are shortened by dropping layers at a random whilst training to allow for information to flow ina an improved way and gradient by the means of Stochastic depth. FractalNets periodically compiles several parallel layers sequenced with various convolutional blocks to obtain a large nominal depth. Meanwhile also maintaining the short paths within the network. They create short paths from preliminary layers to the next layers by taking these different approaches in network topology and training procedure.

This is primarily to ensure optimally maximized information flow within the allying layers within the network. Every layers are engrossed in a connection (with complementing feature-map sizes) with each other in a direct manner. With the purpose of preserving the feed-forward nature, each of the included layers an gulf inputs in addition to the previous ones from all preceding layers. Then they pass on their own feature-maps towards all the subsequent layers.

It is illustrated in a schematic manner. The features are concatenated. To all L1 subsequent layers, there very own feature-maps are passed on. This precludes the introduction of L (L+1) 2 connections in an L-layer network. Due to the pattern of the dense connectivity , this approach is known as Dense Convolutional Network (DenseNet).

Description

The requirement for fewer parameters compared to the conventional convolutional networks because there is no necessity to relearn feature maps that could be reduced, a possible counterintuitive outcome of this dense connectivity pattern is obtained. This is passed on layer after layer. The preceding layer reads state from its previous and writes to the subsequent layer. The information which must be preserved, is

passed on and the state is altered. The information preserved is made understandable by additive identity transformations by the means of ResNets. During training many layers protrude to little giving and can be dropped at random. Due to this the state of ResNets is comparably deemed similar to recurrent neural networks. On the contrary to this concept, the numeral count of parameters of ResNets is significantly more because each of the layers have its own weight. DenseNet architecture on the other hand explicitly differs between information which is added to the network and information that is privy to preservation.

DenseNet [15] layers add only a small portion of feature-maps to the "collective knowledge" of the network as they are very narrow. This helps to keep the rest of the feature-maps unaltered. Thus, the classifier makes a decision based on all feature-maps in the network.
Advantage of DenseNets is the efficient and better flow of information and gradients throughout the network making for an easier training process. Also, better parameter efficiency is another added benefit. A deep supervision is maintained by each of the layers having uninterrupted access to gradients from the loss function and the original input signal. This again contributes to the convenience of training of deeper network architectures. DenseNets [16] are basically assessed on four competitive benchmark datasets which are CIFAR-10, SVHN, CIFAR-100 and ImageNet. Our models requireS much less parameters than already existing algorithms with better accuracy. This contributes to the substantial outperforming of the current trailblazing results on the aforementioned benchmark tasks.
Neural networks encounters a few drawbacks in the process which can be effectively overcome by the usage of Capsules. Capsule is a basically a pack of neurones whose Vector gives us the parameter instance of some entity of an object or anything connected to it. They make predictions with the help of transformation matrices and other necessary algorithms, and judging the outcome of the predictions they are activated. The size of the activity vector means the probability that an entity is present and the orientation is used as the instantiation parameters. The main problem regarding the system at present is that the inverted images and its noises are overlooked. But by the introduction of Capsules this can be solved. The main motive of the presented system is to come across the sidelines of convolution Neural Networks by the help of Capsules.
Moreover, many layers are seldomly contributing and can be dropped which have been proven by some variations of RestNets. Actually the reason behind the big number of RestNets is that the weights to learn of every layer. As an alternative, layers of DenseNets are narrow and just a only small set of new feature maps are added by them.

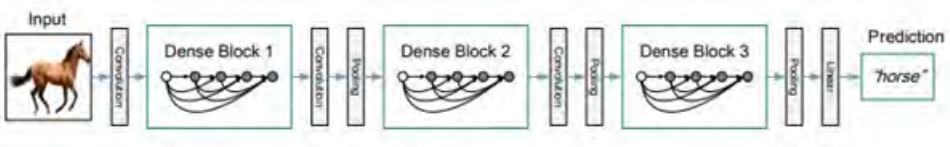


Figure 4.20: A deep DenseNet with three Dense Blocks

Applying Dataset on Model

We imported all the dataset images into the model. Then we proceeded to resize all the .png file images in 224x224 pixels. This was done because all the images were not the same size. Due to this we resize the images in the same resolutions so that the images can be trained in the same way. Additionally, the previous version of the DenseNet supports 224*224 pixel. We took 29962 images for test size and random state =2019, 258110 images were taken for training and 64527 images for validation. Afterwards we leveled the fake image folders and real image folder and set the class as 'FAKE' and 'REAL' respectively. Then we proceeded to make a validation set and for this we set the partition =2. For the image data generator we rescaled 1 divided by 255 and validated split=0.2, through that we can see the accuracy after every segment of epoch and can save for the individual epoch. When we observe better accuracy– we save the model. We kept the batch size=64 and class mode=binary (we require 2 results either fake (1) or real(0)). We altered the model by removing linear layers and added some additional layers. The built model is densenet. The activation function is 'relu' and 'sigmoid'. After running the process we obtained 1 output that is either real or fake. The optimizer=Adam and learning rate =0.0005.

Result

Iteration	Training Loss	Validation Loss	Accuracy
1	0.0016	0.0010	0.0.9997
2	0.0014	0.0026	0.99935
3	0.0012	0.0061	0.9983

Table 4.3: Training loss, Validation Loss and Accuracy of the model, DenseNet121

DenseNet121is a face detection algorithm. Thus we got an accuracy of 99%

4.4 MesoInception

As fake image detection is a challenging task because it has some data complication. Data can be a large file,mall file, the position of fake image can be tilt , can be blurry. For that detecting fake image we use various algorithm, MesoInception is one of them. The reason for choosing MesoInception is it do batch normalization better. Mesoinception use the inception layer but add some extra layer. The layers of mesoinception are conv2D, stride, padding,dilation rate. In conv2d layer there is activation layer which activate the model which I send first .and defined as x.

Description

In MesoInception there are three levels of layers

- Inception layer

- Batchnormalization Max pooling

Stride and Padding is used. Taking 2d grid ones size is 20x20 and another is 10. So if we decreases the size, there comes the stride which allows us to convert the data in small segment without losing the information. Stride can reduce the size of our outcome by lowering the resolution of outcome. Padding is used to give output with same height and width as input have.

Stochastic gradient descent [17] is used where it is updating the model parameter by one sample per iteration is called stochastic descent gradient.

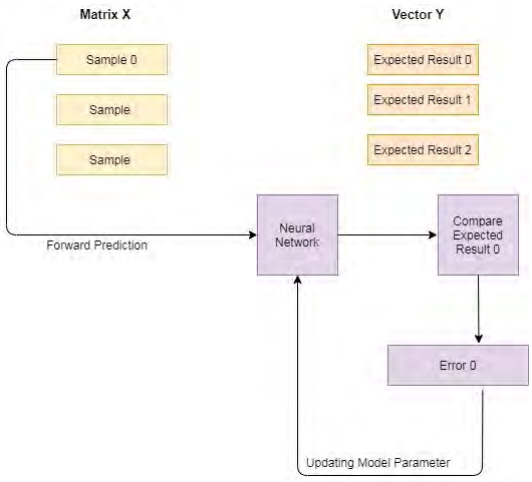


Figure 4.21: Figure 36: Stochastic Gradient Descent

Mini batch gradient is also used in MesoInception where this is Updating model parameter by more than one sample per iteration but less than all the sample is called mini batch gradient descent.

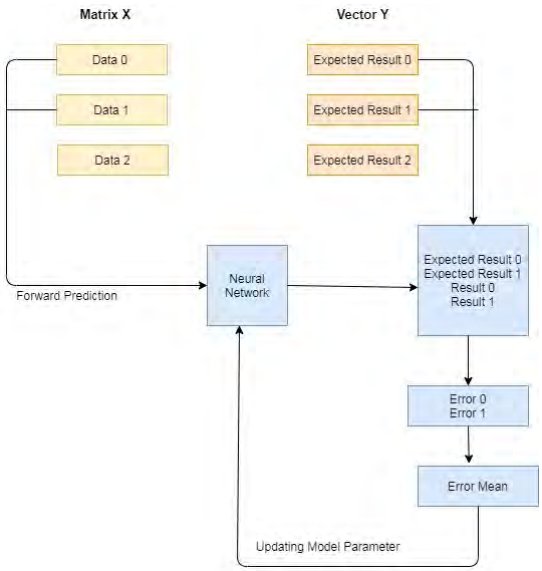


Figure 4.22: Figure 37: Mini Batch Gradient Parameter

Batch gradient descent is when all the sample are used to update the model parameter then it is called back gradient descent.

Normalization- in our work we have to use normalized method. In machine learning if the value is in geometrical then for computer to understand we have to use normalized method.

Batchnormalization [13]: It is the technique for training deep neural network that standardized the input to a layer. For each mini batch when we give input as mini batch then every layer takes a normalized input. That means value will be in same range. That's why the process has stabilizing effects and it also reduce iteration. During back propagation each layer is trying to correct itself separately for the every error. To overcome from this problem batchnormalization adds normalization layer between each layer.In neuron network there is one transform equation and one activation function . That's why for z linear transformation will be z(1). When it is used as input for batchnormalization[14] then it produced z(1) normalized output. The value of $z^n isnormalized$.
And then the activation function ReLu , Sigmoid can be performed.

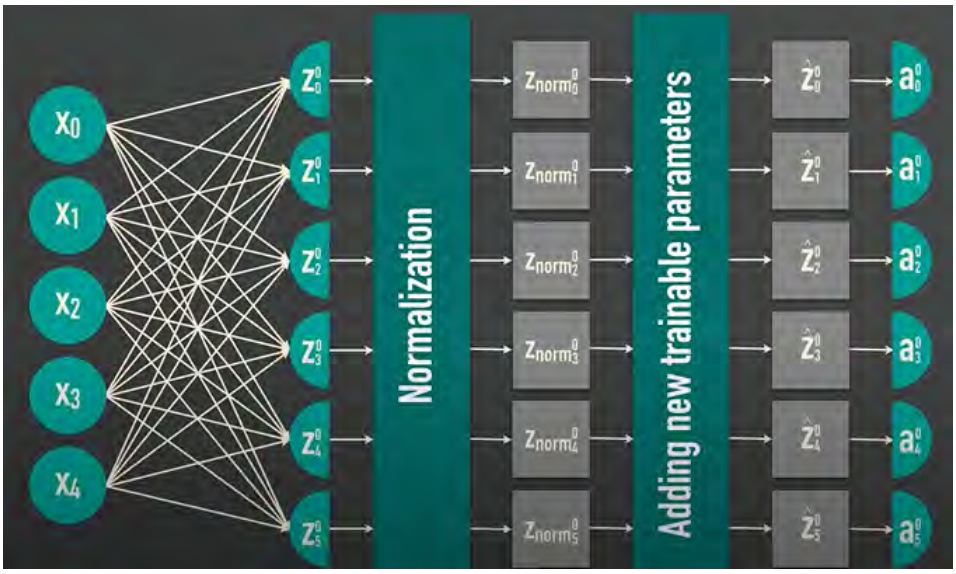


Figure 4.23: Batchnormalization

The complete model of MesoInception is as follows

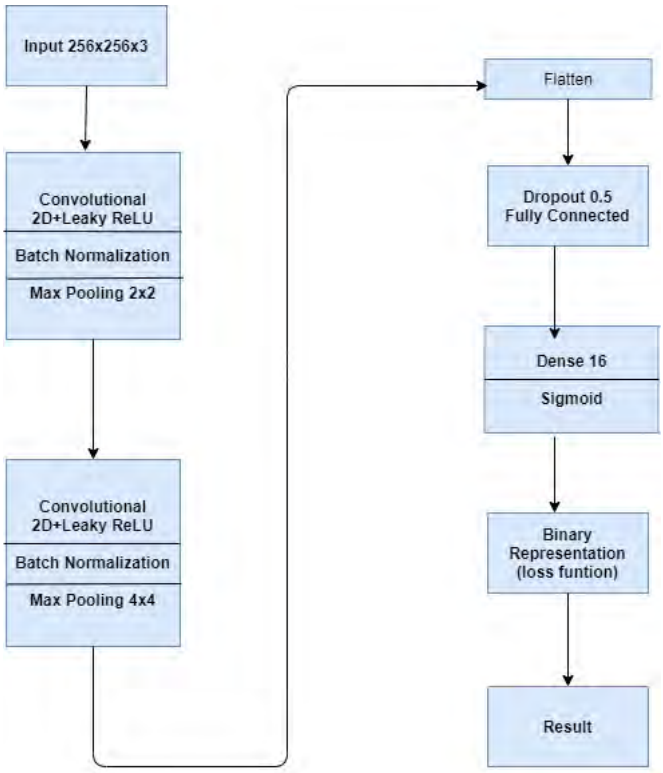


Figure 4.24: Complete Architecture of MesoInception

Applying Dataset on Model

We used 6 epochs with 0.5 Dropout and Activation function LeakyReLu (alpha=0.1)

Result

We obtained an accuracy of 63.05% in MesoInception.

Iteration	Data Loss
1	0.633397108791929
2	0.6466134329858001
3	0.6494618990815383
4	0.664477240870187
5	0.6599882225491571
6	0.662093652113206

Table 4.4: Data Loss, MesoInception

There is another function log loss. Log loss function is better when there is more than a thousand epoch.

Random Loss	0.7256620055750759
1 loss	17.26978799617044
0 loss	17.269388197455342
0.5 loss	0.6931471805599454

Table 4.5: Data Loss of Log Loss Function

4.5 VGG-19

VGG-19 is a trained Convolutional Neural Network from Visual Geometry Group, Department of Engineering Science, University of Oxford. K. Simonyan and A. Zisserman proposed the Convolutional Neural Network model in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"[8]. VGG-19 known as deeper Convolutional Neural Network with 19 greater layers in distinction to AlexNet with more than one 33 kernel-sized filters one after an-other. The VGG-19 model used to be not the victor of ILSVRC30 2014, in any case, the VGG Net is one of the fundamental persuasive papers due to the fact it fortified the idea that CNNs ought to have a profound arrange of layers in prepare for this a range of leveled representation of visible facts to work. The VGG-19 model, a add up to of 138M parameters, was put 2nd in classification and 1st in localization in ILSVRC 2014. This model is prepared on a subset of the ImageNet27 database [8].

Description

VGG-19 model, a 19-layers CNN that absolutely used 33 filters with stride and pad of 1, collectively with 22 max-pooling layers with stride 2 is a huge network, and contains a add up to of around 138 million parameters. VGG-19 might also be a more profound CNN with extra layers compared to AlexNet. To minimize the range of parameters in such profound systems, it employments little 33 filters in all convolutional layers and great utilized with its 7.3% error rate [8]. The VGG-19 is prepared on greater than a million images and it can categorize images into numerous category, for example in face, bus, house etc.

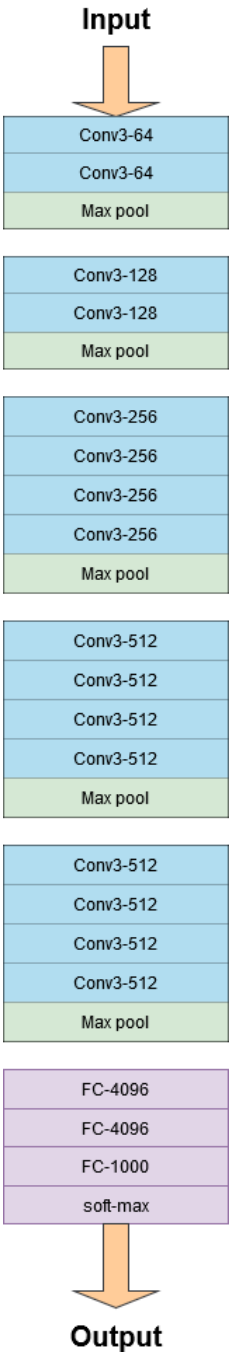


Figure 4.25: VGG-19 Model with 16 Convolutional Layers

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
Input (224 × 224 RGB image)					
Conv3-64	Conv3-64 **LRN**	Conv3-64 **Conv3-64**	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64
maxpool					
Conv3-128	Conv3-128	Conv3-128 **Conv3-128**	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128
maxpool					
Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 **Conv1-256**	Conv3-256 Conv3-256 **Conv3-256**	Conv3-256 Conv3-256 Conv3-256 **Conv3-256**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
Soft-max					

Figure 4.26: ConvNet Configurations

Network	Number of Parameters
A,A-LRN v2	133
B	133
C	134
D	138
E	144

Table 4.6: Number of Parameters(in million)

The number of parameters shown above the numerous columns A, A-LRN, B, C, D and E appears the numerous designs attempted with the aid of the VGG group [8]. The column E alludes to VGG- 19 architecture. The VGG-19 was trained on the task of classification 1000 in the ImageNet challenge (ILSVRC). The network takes a (224, 224, 3) RBG image as the input. The layers are signified by "conv(size of the filter)-(number of such filters)". In VGG-19 there are convolution layers are of 3X3 filters and stride is 1 and for padding and maxpool layer are of 2X2 filters and stride is 2. It has 2 fully connected layers which lead to the output. VGG-19 has 19 layers which holds the 19 in VGG-19. VGG-19 has a network with 138 parameters. In this 19 layers there are 16 convolution layers and 3 fully connected layers, this are the total 19 layers of VGG-19 architecture. The output goes to three Fully-Connected layers, in this three layers first two layers have 4096 neurons each and third one has 1000 channels. Among these layers, softmax is the last layer and all the layers are using ReLU. The layers with trainable weights are as it were the convolution layers and the Fully Connected layers. Maxpool layer is utilized to decrease the measure of the input picture where softmax is utilized to form the ultimate choice.

VGG Neural Networks VGG based not only on smaller window sizes and gradients, It also mentions another very significant feature of the first convolutionary layer CNNs: profund. VGG19 architecture approaches convolutionary layers use a very small receptive field (3x3, the smallest scale often catching up/down/left). There is also a 1X1 filters which act as linear input transformation. Composition of multiple operators obtains CNNs. The multiple operators are individually called layers. The operator of the Rectified Linear Unit (ReLU) is probably the simplest and quite useful example

$$Yijk = max0, xijk \tag{4.1}$$

In a way, ReLU acts as a tracker, with the static rule that every pattern is observed if there is a sufficiently high filter response.. At convolution, the bias is implemented, effectively subtracting 0.2 from the subsequent reactions.

Fully-connected layers: there are three fully-connected layers in VGG-19, In these three layers, first two layers, each contain 4096 channels and the third one contains 1000 channels. Among these three layers, the last layer is Softmax.

Hidden layers: ReLU(a big breakthrough) is used in both open layers of VGG from Alexnet that decreases workout time). Local Response Normalization (LRN), typically not used by VGG, as LRN increases memory use and training time without any obvious accuracy improvements.

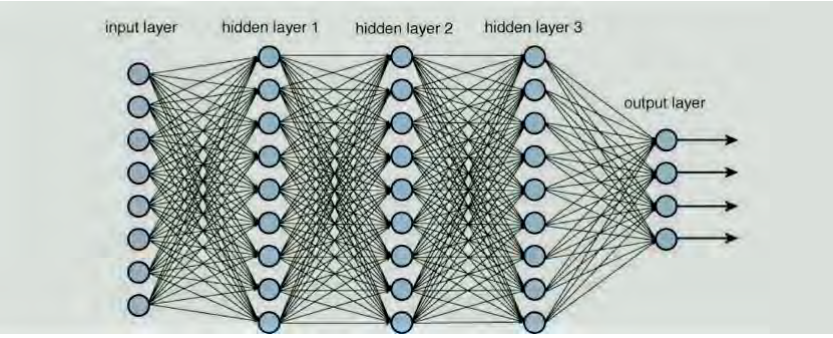


Figure 4.27: Convolutional Neural Network with Three Hidden Layers

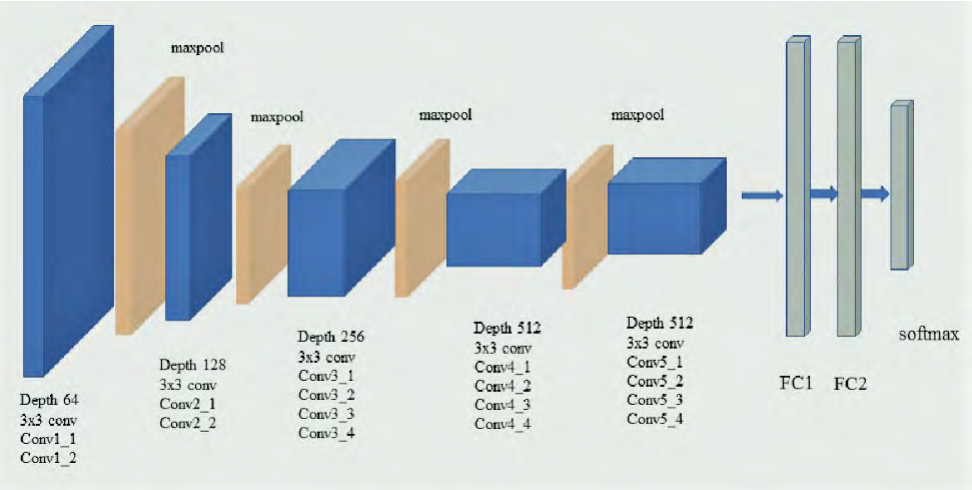


Figure 4.28: VGG-19 Architecture

4.5.1 Binary Classification

The goal of a binary classification problem is to create a deep model Learning that allows for an option in cases where only two possible values are taken from the item to be expected in all. Binary cation problems suggest assigning an individual to one of two groups by assessing a collection of attributes. Binary image segmentation of a categorized image in which each pixel is marked as pixel Sign component (1) valid or not (0). The binary classification on the segmentation of images is implemented in VGG-19 Binary. The VGG-19 was trained on the ImageNet challenging a challenge of 1000-class classification, and the input is an RBG image (224, 224, 3). The VGG-19 Binary architecture works on the parallel division of activity signs (from RGB pictures of estimate 224x224 and going for grayscale covers) where each pixel must be marked as either portion of an activity sign (1) indicates that the image is fake or (0) indicates that the image is real. In binary classification, VGG-19 pre-trained on ImageNet and freezes all layers, then pops up the last Dense layer of 1000 units (one for each of the initial 1000 picture classes), then installs 50176 Dense layer (one for each of the 224 * 224=50176 pixels) and reshapes it. This will train this model on tens of thousands of yes / no binary GAN generated images to ensure that the VGG-1939;s last (trainable) convolution layers are fully trained to distinguish GAN generated fake or real image.

Applying Dataset on model

VGG19's last (trainable) convolution layers are completely optimized to classify whether the image is fake or real. In this model we trained the dataset with 10 epochs to find the validation loss, training loss and accuracy. Adam optimizer has been used with the Learning Rate 0.001 and Batch Size 128. After feeding the dataset into the model we got a decent accuracy which is 87.3%. The figure given below shows some sample real and fake picture from our datasets. The pictures labelling by 0 are the real image and 1 are the fake image detected by our model.

Result

Figure 4.29: Fake image labeling by 1 and Real image labeling by 0

The model accuracy of VGG19 Binary is 87.3%

- There are 64773 fake train samples
- There are 64773 real train samples
- There are 6108 fake validation samples
- There are 6108 real validation samples

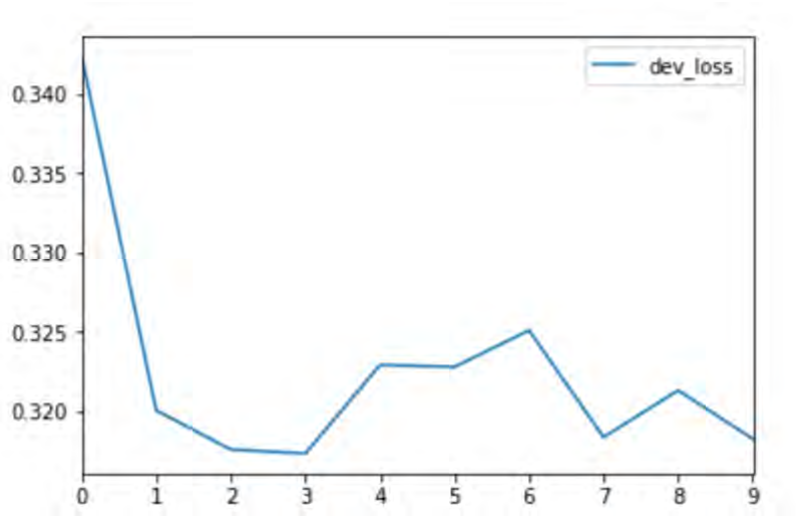


Figure 4.30: Validation Loss Point

The figure given above shows the validation loss plot where validation loss is given on Y-axis and epoch's number is given on X-axis. For each case we used validation to measure the accuracy and if an epoch gives a better accuracy we saved that epoch.

Iteration	Training Loss	Validation Loss	Accuracy
1	0.4711	0.3423	0.862885
2	0.4711	0.3423	0.862885
3	0.4450	0.3201	0.871726
4	0.4387	0.3176	0.874918
5	0.4314	0.3229	0.872626
6	0.4297	0.3228	0.872053
7	0.4255	0.3251	0.871316
8	0.4239	0.3184	0.870498
9	0.4222	0.3213	0.876801
10	0.4206	0.3183	0.874345

Table 4.7: Training loss, Validation Loss and Accuracy of the model, VGG-19 Binary

Figure 4.31: Confusion Matrix of VGG-19 Binary

- There are 5483 real images and they are predicted as "REAL"
- There are 625 real images and they are predicted as "FAKE"
- There are 910 fake images and they are predicted as "REAL"
- There are 5198 fake images and they are predicted as "FAKE"

Meanwhile, we got the value of F1 score for VGG19 Binary which is 0.8713435587964128.

Flowchart of Using the Algorithms

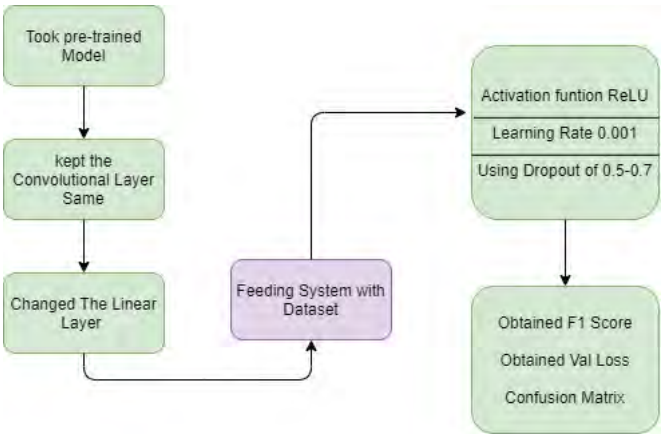


Figure 4.32: Implementing Algorithms

Chapter 5

Performance Evaluation

We have obtained graphical representation of Validation Loss, Confusion Matrix and F1 score.

5.1 Copmarison of Validation Loss

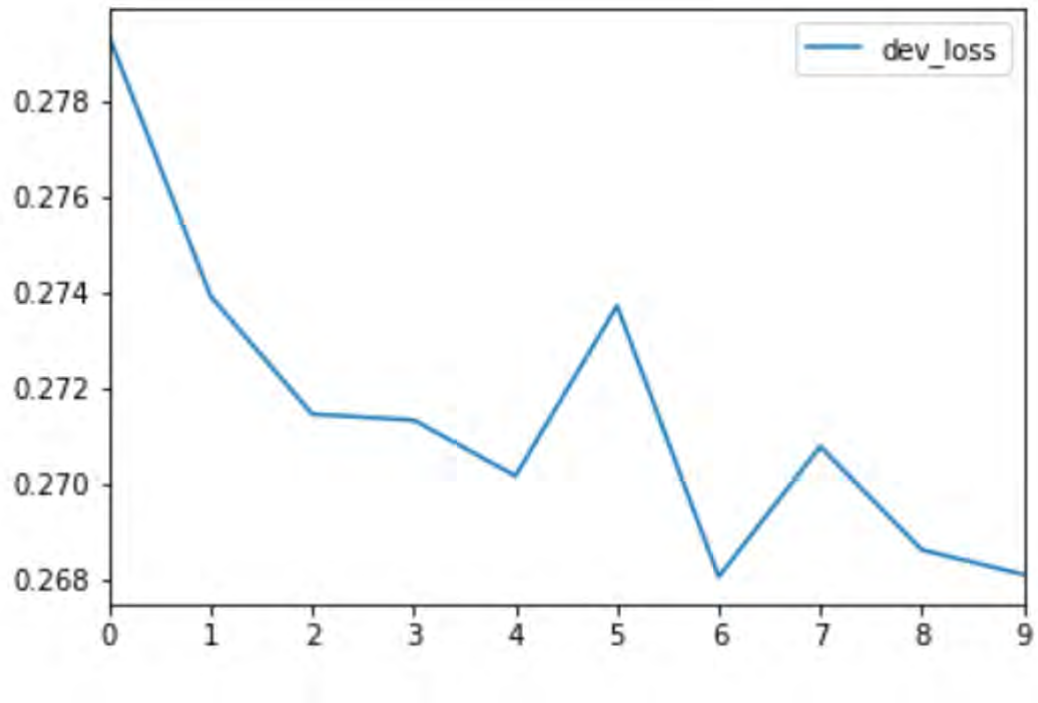


Figure 5.1: Validation Loss of MobileNet v2

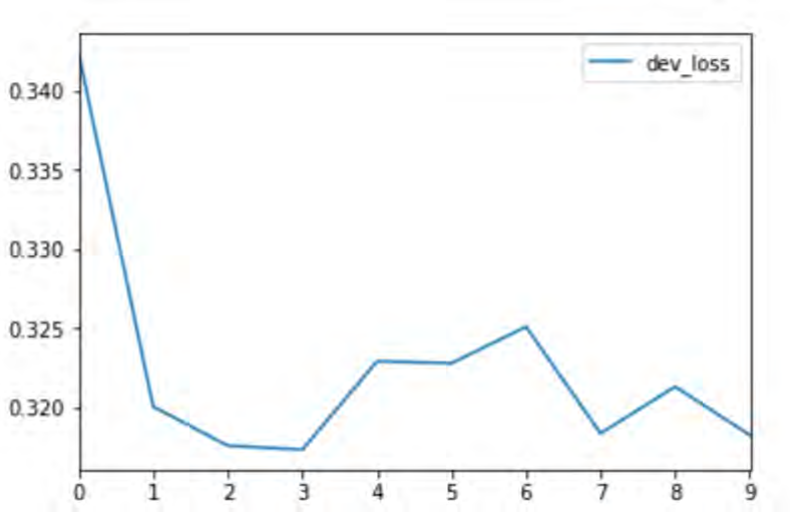


Figure 5.2: Validation Lossof VGG-19 Binary

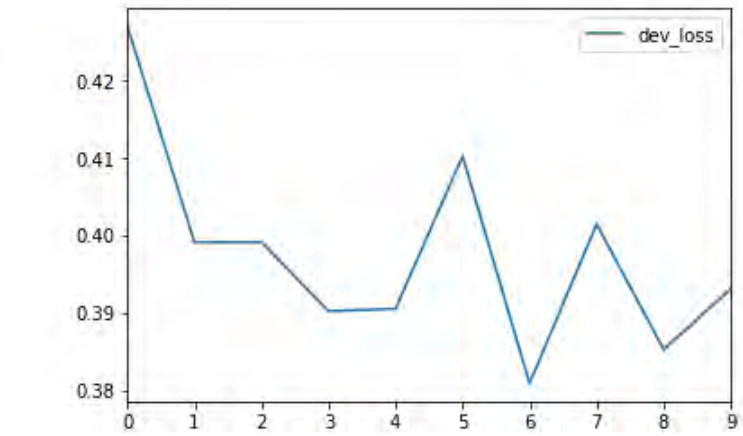


Figure 5.3: Validation Loss of Inception v4

5.2 Comparison of Confusion Matrix

Figure 5.4: Confusion Matrix of MobileNet

From the above figure 4.4 we can see that,

- There are 5274 real images and they are predicted as “REAL”
- There are 834 real images and they are predicted as “FAKE”
- There are 453 fake images and they are predicted as “REAL”
- There are 5655 fake images and they are predicted as “FAKE”

Figure 5.5: Confusion Matrix of VGG-19 Binary

From the above figure 5.5 we can see that,

- there are 5483 real images and they are predicted as “REAL”.
- There are 625 real images and they are predicted as “FAKE”.
- There are 910 fake images and they are predicted as “REAL”.
- There are 5198 fake images and they are predicted as “FAKE”

Figure 5.6: Confusion Matrix of Inception v4

The confusion matrix states that,

- there are 5274 real images and they are predicted as “REAL”.
- There are 834real images and they are predicted as “FAKE”.
- There are 453fake images and they are predicted as “REAL”.
- There are 5655fake images and they are predicted as “FAKE”.

5.3 Accuracy Result Analysis

There are total five models on which we have applied our dataset of fake and real images. The same dataset have been used in every of the model for genuine and uniform comparison. Among VGG-19, MobileNet, Inception v4, MesoInception, DenseNet121, the best accuracy has been obtained from MobileNet.MobileNet has given 89.78% Accuracy.

Algorithm	Model Accuracy
MobileNet v2	89.4%
Inception v4	84.32%
DenseNet121	99%
MesoInception	63.05%
VGG-19 Binary	87.3%

Table 5.1: Accuracy Results of Different Models

DenseNet121 is not considered to be the best here since it is recognizing face. In case of detecting image, whether it is fake or real, MobileNet can successfully detect machine generated 89.3% of the time. It is because of its architecture and simplistic optimization. Another reason is optimizing the Linear layer. From the Experimental evidence, we can say that, using linear layers is crucial as it prevents nonlinearities from destroying too much information. Using non-linear layers in bottlenecks hurts the performance by several percentage which we did not implied.

Chapter 6

Conclusion

We proposed a comparison among 5 different algorithms and showed the accuracy level of each algorithm to detect fake image. We have discussed about pros and cons and visualized the data of the algorithms. We have taken a massive dataset which contain a large amount of high resolution images. We have modified some of the model by removing and adding some additional layers. We saved our model by evaluating the validation set which showed the accuracy level while performing every epoch. Finally, we got the output as a binary number, either 0 or 1. We leveled 0 as 'REAL' and 1 as 'FAKE' respectively. Looking forward, we hope that this result of our work will be beneficial for everyone in this technology dependent world. And also will be helpful for going further to design a new model.

6.1 Future Work

As we have compared some algorithms for detecting fake image, we have a plan to work with more algorithms on this topic and also to see the performance of different models. Moreover, have a plan to work with different versions of Generative adversarial networks (GANs) model. The CycleGan will be our first priority. Also we will work more with reinforcement learning in the algorithms which we already worked with. We will try to increase epoch and the accuracy of the model to detect the fake images more precisely and shortly.

Bibliography

[1] J. Galbally, J. Ortiz-Lopez, J. Fierrez, and J. Ortega-Garcia, "Iris liveness detection based on quality related features", in *2012 5th IAPR International Conference on Biometrics (ICB)*, IEEE, Mar. 2012. DOI: 10.1109/icb.2012.6199819. [Online]. Available: https://doi.org/10.1109/icb.2012.6199819.

[2] W. R. Schwartz, A. Rocha, and H. Pedrini, "Face spoofing detection through partial least squares and low-level descriptors", in *2011 International Joint Conference on Biometrics (IJCB)*, IEEE, Oct. 2011. DOI: 10.1109/ijcb.2011.6117592. [Online]. Available: https://doi.org/10.1109/ijcb.2011.6117592.

[3] D. Menotti, G. Chiachia, A. Pinto, W. R. Schwartz, H. Pedrini, A. X. Falcao, and A. Rocha, "Deep representations for iris, face, and fingerprint spoofing detection", *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 864–879, Apr. 2015. DOI: 10.1109/tifs.2015.2398817. [Online]. Available: https://doi.org/10.1109/tifs.2015.2398817.

[4] J. Galbally, S. Marcel, and J. Fierrez, "Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition", *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 710–724, Feb. 2014. DOI: 10.1109/tip.2013.2292332. [Online]. Available: https://doi.org/10.1109/tip.2013.2292332.

[5] "Ai can now create fake porn, making revenge porn even more complicated", Aug. 2018.

[6] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, "Video forgery detection using correlation of noise residue", in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, IEEE, Oct. 2008. DOI: 10.1109/mmsp.2008.4665069. [Online]. Available: https://doi.org/10.1109/mmsp.2008.4665069.

[7] Krizhevsky, A. Sutskever, I, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks",

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *In: arXiv preprint arXiv:1409.1556*, 2014.

[9] Brandon and John, "Terrifying high-tech porn: Creepy 'deepfake' videos are on the rise", *Fox News*, Feb. 2018.

[10] J. Kietzmann, L. W. Lee, I. P. McCarthy, and T. C. Kietzmann, "Deepfakes: Trick or treat?", *Business Horizons*, vol. 63, no. 2, pp. 135–146, Mar. 2020. DOI: 10.1016/j.bushor.2019.11.006. [Online]. Available: https://doi.org/10.1016/j.bushor.2019.11.006.

[11] Schwartz and Oscar, "You thought fake news was bad? deep fakes are where truth goes to die", Nov. 2018.

[12] A. G. Howard, M. Zhu, and B. C. et al, "Mobilenets: Efficient convolutional neural networks for mobile vision applications",

[13] S. Ioffe and C. Szegedy, "Batchnormalization: Accelerating deep network training", *a. reducing internal covariate shift. In ICML*, 2015.

[14] T. Pandit, A. Kapoor, R. Shah, and R. Bhuva, "Understanding inception network architecture for image classification", Feb. 2020. DOI: 10.13140/RG.2.2.16212.35204.

[15] C. M., "Densenet", Oct. 2020.

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks", pp. 2261–2269, 2017.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradientbased learning applied to document recognition", *Proceedings of the IEEE*, 1998.