DEEPFAKE DETECTION

by

AHMED SHERIF AHMED ELBAKRY MOHAMED
URN: 6497705

A dissertation submitted in partial fulfilment of the
requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2020

Department of Computer Science
University of Surrey
Guildford GU2 7XH

Supervised by: Lilian H. Tang

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Ahmed Sherif Ahmed Elbakry Mohamed
May 2020

Abstract

Deepfake is a technique used to superimpose images of a target person onto source person. Recently, with the rapid development of Machine Learning techniques it has become easier to create fake videos of a target person doing or saying things the source person does. This technique is powered through the use of Generative Adversarial Networks and deep learning. With such rapid development this have become very difficult to detect with the naked eye, causing a huge threat to society and privacy. In this paper the main aim is to create a counter technique that can detect deepfakes and report them with high accuracy.

Convolutional Neural Networks have recently demonstrated very high accuracy in image recognition and classification tasks. This problem of classifying videos as real or fake is addressed in this paper through a variety of techniques.

First, the replication Mesonet and XceptionNet are considered in this project, 2 popular models in deepfake detection are replicated and finetuned to the given data set to further understand their potential in detecting deepfakes. Both models yielded very high accuracy and recall scores which confirmed that they can be relied on in detecting deepfakes.

Then a combination of a CNN plus Long Short Term Memory model which is able to learn sequence patterns is considered. Unfortunately, the LSTM model yields lower accuracy scores than both CNN models as standalone.

Finally, it can be concluded that traditional CNN models that perform well on image classification tasks such as MesoNet and XceptionNet are suited to this task as they yielded the best results with very high recall rates of 1, and with the availability of larger datasets LSTM models might be able to generalize effectively with careful feature engineering on which sequences being input to.

Acknowledgements

I would like to thank Dr. Lilian Tang for the extensive feedback provided on this project, and guidance on the technical implementations and write ups of the report. Without her help this would have been very difficult to pursue.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Background

Deepfake (a portmanteau of "deep learning" and "fake") is a technique that can superimpose face images of a target person to a video of source person in order to create a new video where the target person appears to be doing or saying things that the source person does. Although the act of manipulating video content is not particularly new; recently it has witnessed major improvements supported by the rapidly developing domain of Machine Learning and Computer Vision as well as the availability of more powerful hardware components such as GPUs.

Machine learning is one of the most popular Artificial Intelligence techniques used for processing and detecting patterns in big data, a self-adaptive algorithm that improves with experience/ iterations or newly added data; to produce accurate human-like predictions on newly unseen datasets. Deep learning is a branch of Machine Learning that utilizes several hierarchal Neural Networks to carry out the process of Machine Learning mimicking the Human Brain. Deep learning models have been used by Deepfake algorithms to analyze facial expressions and movements of a person and produce facial images of another person making similar expressions and movements.

Recent advances in GPUs have allowed the development of new Deep learning models such as generative adversarial networks and autoencoders and made training possible in shorter/ realistic timeframes. Such models have been used extensively in many domains including the medical community where fake data could be generated to aid in the learning process of a model when real data is too small to be sufficient for the model to learn from.

As a result, nowadays we have very powerful Deepfake generation tools, resulting in realistic tampered videos which have tremendously improved the deception rate over the past few years. With the availability of such tools online, anyone that has access to a sufficient number of facial shots of a person could easily create a Deepfake video of them doing or saying something.

The problem of Deepfake mainly affects high profile demographics including Celebrities and Politicians as their images are available extensively on the internet. As tools improve this problem will consequently start affecting more demographics where parties could benefit from such scams. In September 2019, the AI firm Deeptrace found 15,000 Deepfake videos online which almost doubled in 9 months (Sample 2020), indicating the magnitude of the problem Deepfake could have now and in the future.

Recently, Deepfake detection have improved due to the availability of new Deep learning techniques, but the main issue underlies here is that such new techniques could also be utilized in the opposing direction for creating DeepFakes. Therefore the exploration and development of new independent Deepfake detection methods is crucial to surpass current generation techniques.

1.2 Project Description

This project attempts to provide a Deepfake detection tool that could be used by individuals affected by Deepfake to provide a sense of reliability for the general public so they can confirm the originality of the videos they see on a daily basis.

This project will explain the development of a Deepfake detection technique which is robust to various generation methods. Starting with data acquisition and exploration then the project will address the problem of face detection and extraction from video frames, then the alignment of faces and the analysis on a frame to frame basis. Then the analysis of different feature vectors extracted from frames in order to create a combination of features that can assist in the creation of a model that can identify real from forged videos.

In order to create a method that is robust to various generation techniques, this project is going to consider the development of a Machine Learning algorithm that is capable of detecting multiple frame progression sequences in real videos and fake ones, unlike most current detection techniques which rely mainly on a single frame to detect artifacts.

Furthermore, in this project experimentation with different Deep learning techniques is going to be carried out to consider the possibility of combining several techniques together in a model that is robust to the visual artifacts produced in the generation process.

1.3 Project Aims

This section details the projects main aims and objectives which is reflected in evaluating the achievements of the project as a whole.

Step 1

Acquire Data which to base Deepfake detection on: Various datasets are going to be obtained from third parties in order to train the Machine Learning on so it can make predictions on data in the wild identifying whether the input image is a real image or fake.

Step 2

Explore the latest state-of-the-art techniques used to Deepfakes in videos: Currently there are some SOTA techniques that is being used to detect forgery in videos. The replication of such techniques and testing their performance to create a benchmark threshold in which the project will aim to improve upon.

Step 3

Face Detection, Cropping and Alignment: The core section of data exploration is detecting faces in videos then cropping and aligning them together so that the model is restricted to extract and understand only relevant areas in videos where it could be used to train the model as to later enable the model on deciding the originality of the face image input.

Step 4

Manipulate the Data to contain all key elements in modelling human signals: By removing irrelevant features and combining datasets where necessary; the final dataset is going to be ready for training and making predictions upon. This ensures that the models prediction is more based on relevant features which is most likely going to produce better results.

Step 5

Create model which improves upon the latest SOTA Deepfake detection: Post data acquisition and exploration, a model must be developed to detect tampering/ forgery in video content which produces better results than the latest SOTA methods.

1.4 Structure of Report

Section 1 - Introduction

This section covers the problem history and magnitude. Relating to the project's expected outcomes and aims attempting to address the problems issues.

Section 2 - Literature Review

This section covers a comprehensive overview of the problem, as well as explaining the latest techniques used to generate and detect Deepfake videos, with their respective core methodology and SoTAs.

Section 3 - The Dataset

This section focuses on exploring and manipulating the dataset. Preparing the dataset for training and testing in a form that is usable by the Machine Learning models.

Section 4 - MesoNet

This section details and explores the architecture of MesoNet, one of the best performing models in this domain in detecting forgery within images, as well as replication of the model and training it on the dataset used in this project.

Section 5 - XceptionNet

This section details and explores the architecture of XceptionNet, one of the best performing models in this domain, as well as replication of the model and training it on the dataset used in this project.

Section 6 - CNN + LSTM

This section explores the possibility of utilizing one of the SOTA CNN models examined previously in a more robust pipeline that is able to detect intra-frame inconsistencies in videos, through the use of a variant of RNN LSTM to detect and understand frames sequences.

Section 7 - Results

This section details the accuracy and performance of the various methodologies adopted in this paper, as well as comparing the results on the unseen test set prepared.

Section 8 - Evaluation

In this section a comprehensive review and evaluation of the project is carried out and measures against the projects initial aims and objectives accordingly. Additionally, details any shortcoming as well as the future possible research directions.

Section 9 - Ethics

This section reflects on the ethical, social and legal issues surrounding the projects development and completion. An in depth study of ethical issues in Deepfake creation and detection is carried out respectively.

Section 10 - Final Conclusions

This section provides final remarks and thoughts on the project as a whole, and reflects upon the projects completion.

Chapter 2

Literature Review

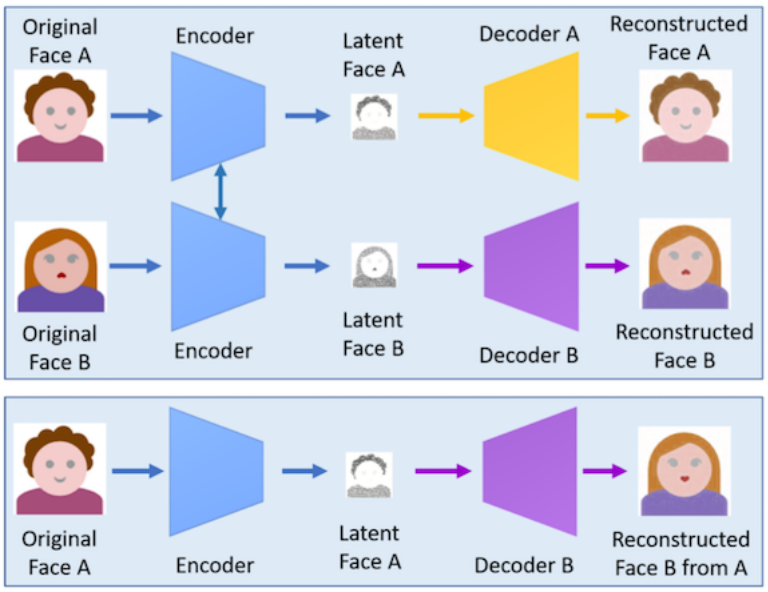
2.1 Deepfake Video Generation

Deepfake generation have recently become very popular due to the availability of new Deep learning techniques and its improved ability to alter/ manipulate higher quality videos. Additionally, Deepfake creation web applications have become widely available on the internet allowing many users ranging from expert to novice to easily access to such tools and create Deepfake content.

Deep learning has always been known for its ability to represent complex and high dimensional data. Early attempts at Deepfake creation utilized a variant of Deep learning is deep autoencoders which have been also used extensively in dimensionality reduction to represent the data in more concise manner.

In 2017 the first Deepfake creation tool Fakeapp was developed by a Reddit user using autoencoder-decoder pairing structure to swap faces in videos (*Faceswap: Deepfakes software for all* n.d.). In figure 2.1 the autoencoder extracts latent features from both face images where the weights are shared in the two encoders, then each face image uses a different decoder to reconstruct the facial images again. This network strategy enables common encoder to find common features and similarity between the two faces and their components locations such as mouth, nose, eyes and eyebrows.

An improved version of Deepfake creation GAN (Generative Adversarial Networks) utilizes the same principle of autoencoder-decoder structure but additionally it includes



(a) Deepfake Generation

Figure 2.1: A deepfake creation model using two encoder-decoder pairs (Thi, M., Tien, Nguyen & Nahavandi 2019)

adversarial loss and perceptual loss which leads to both networks competing against each other until there is no further improvement is noticed in performance. This addition enables the faces and eye movements to look more realistic in videos as their loss metrics which have a huge contribution to detection are now significantly taken into consideration.

GANs are considered an ongoing threat to Deepfake detection. Whenever Deepfake detection can improve, this information could be incorporated into a new GAN model where it would beat current/ latest Deepfake detection methods through exploiting such improvements in detection methods. Improvements in Deepfake detection reveals weaknesses/ vulnerabilities of generation techniques, therefore this is an ongoing threat where detection and generation are going to be continually improving over time and affected by the rise of new Machine Learning techniques.

2.1.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks are commonly used to generated new data that have not been created before. They learn about data and their characteristics fed as an input in order to create new data similar to the one being trained on. It consist of two model a

generator, which generates new data based on random input/ noise, and a discriminator which assesses the data and judges its authenticity, whether this is a real or fake input as produced by the generator model. When training the network of the generator and discriminator both models learn in parallel and from scratch.

In the context of deepfakes, GANs are used to generate fake faces of people, when given the necessary time and computational power they can create almost real like face enactments of people. In the training process the generator gets better at creating more real like images of people and the discriminator gets better at distinguishing the difference between both. The learning process continues until the discriminator is no longer able to distinguish between real and fake images, as the generator becomes really good at creating fake images that is very similar to real data.

2.1.1.1 The Discriminator

The discriminator in a GAN is a classifier, that aims to distinguish between real and fake image inputs. The discriminator can take any possible architecture that is suitable to the problem it is being trained on. The discriminator's training data comes from two sources, real data instances which the model uses as a positive example and fake data instances created by the generator and labeled as negative examples.

When training the discriminator, it is connected to two loss functions which are associated to the generator's loss and discriminator's loss. During the training process generator's loss is ignored and it is only connected to the discriminator's loss function. The discriminator's loss function is calculated by measuring the output error on classifying the data as real or fake. The discriminator loss penalizes the discriminator for wrong classification instances and back propagates the loss back through the network updating its weights through the discriminator's network.

2.1.1.2 The Generator

The generator in GAN is responsible for creating fake data based on random input noise. This is performed through incorporating the feedback from the discriminator's loss into the generation network. This feedback assists the generator in updating its weights also

through backpropagation, and eventually getting better at creating fake data that is very similar to real input data fed to the discriminator.

The input to the generation netwrok is random noise, this helps in creating a variety of input for the generator to transorm into a meaningful data instance in the problem domain. In deepfakes generator's learn to create images or videos of people that is based on source pictures of another individual using a variant of GANs.

The generator's learning process is mainly affected by the discriminator network in an indirect way. In the training process the generator inputs fake data to the discriminator's network and the discriminator produces an output that we are trying to affect. The ultimate goal is to make the generator produce fake data that the discriminator fails to distinguish between, so If the discriminator classifies an input from the generator network as fake, this penalizes the generator network.

Typically we don't want the discriminator's weights to change at the training of the generator as this would make it more difficult for the training procedure, but later on both networks are trained alternatively for one or more epochs. So when the generator is training the discriminator stays the same and when the discriminator is learning to identify real from fake instances the generator stays the same.

2.1.1.3 Convergence

As the generator is training it gets better at creating fake outputs, if the generator's training is successful, this happens until the discriminator is no longer able to tell the difference, having almost a 50 percent chance at classifying the input as real or fake from the generator. This progression of the generator causes a convergence problem as a whole, as the discriminator's feedback gets less meaningful over time, and training might reach a point where the generator is getting useless feedback from the discriminator due to the improvement of the generator. If training continues past this point the generator's quality might collapse affecting the fake output it is producing.

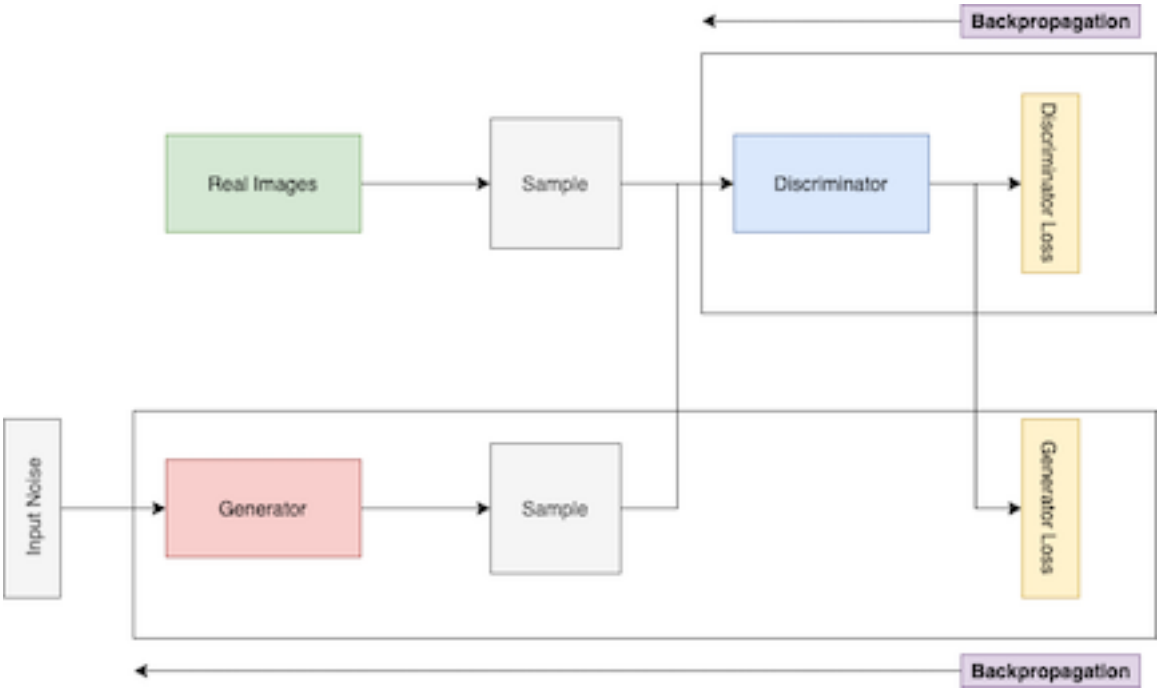


Figure 2.2: Training Generator and Discriminator Backpropagation

(Thi et al. 2019)

2.1.1.4 Minimax Loss

In GANs there are two losses affecting the model's process, one for the generator and the other for the discriminator. In order to combine both losses in a meaningful manner they both use the same loss function but differently. In the original paper that introduced GAN's they drive a loss function which enables the generator and discriminator to use the same function in a min max game. The generator aims to minimize the loss function while the discriminator aims to maximize the loss function.

This value function is given by :

$$min_g max_d V(G, D) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \tag{2.1}$$

Where $D(x)$ is the discriminator's estimate of the probability that real data instance x is real, E_x is the expected value over all real data instances, $G(z)$ is the output provided by the generator given input noise z, $D(G(z))$ is the discriminator's estimate of the probability that a fake instance is real, E_z is the expected value over all random inputs

to the generator. In effect the generator can' affect E_x thus minimizing the generator's loss is equivalent to minimizing E_z, which is provided the generator is providing more realistic values this loss will decrease.

2.2 Deepfake Video Detection

Deepfakes pose a great challenge for democracy, privacy and national security. As soon the threat was introduced many methods have been introduced to detect Deepfakes. Early attempts in detecting Deepfakes were based on handcrafted features obtained from visual artifacts produced by the generation process. More recently, Deep learning have been used to automatically extract salient and discriminative features, also in detecting inconsistencies between images/ video frames.

Deepfake video detection is considered a binary classification problem where classifiers are used to classify between real and tampered videos. This methods requires a huge number of data including real and fake videos. The number of fake videos is becoming increasingly available through the development of new generation techniques, but is still limited in terms of validating emerging detection techniques.

One of the first benchmarks created was a notable dataset consisting of 620 videos created using the open source available Face-swap GAN (*Faceswap: Deepfakes software for all* n.d.) and the VidTIMIT videos database. These videos were then used to test various Deepfake detection methods. Preliminary test results indicated that popular face recognition models such as VGG and Facenet are unable to detect DeepFakes effectively.

2.2.1 Temporal Features across Video Frames

In the process of Deepfake generation temporal coherence between video frames is not enforced effectively. Based on that observation (Sabir, Cheng, Jaiswal, AbdAlmageed, Masi & Natarajan 2019) proposed the use of spatio-temporal features in video streams to distinguish between original and tampered videos. Video frames were passed accordingly to DenseNet then integrated with a Recurrent Convolutional Model (RCN) exploiting temporal inconsistencies across frames.

Similarly, (Guera & Delp 2018) proposed another methodology exploiting intra-frame inconsistences. A pipeline was developed that utilizes CNN and LSTM to detect Deepfake videos. First, the CNN is used to extract high level feature vectors from video frames then fed into LSTM to create a temporal sequence descriptor of video frames. Finally, a fully connected neural network is used to classify between Deepfakes and Original videos.

2.2.2 Visual Artifacts within Video Frames

Unlike previous methods highlighting inconsistencies between intra video frames; which mainly uses deep recurrent networks, in this section I am going to investigate the other approaches currently emerging based on inter-frame inspection. In the generation process there are also artifacts produced within single frames, where could be detected using classifiers.

2.2.2.1 Deep Classifiers

In the process of Deepfake creation; resolution poses a great obstacle. Typically, it is easier to manipulate videos with low resolution as less artifacts are generated. Such artifacts are produced when using an affine face warping approach (i.e. scaling, rotation and shearing), these artifacts can be detected by CNN models such as VGG16, ResNet50, ResNet101 and ResNet152.

Based on the observation that CNN models can automatically extract salient features and detect inconsistencies within a frame, Darius Afchar and Vincent Nozick proposed MesoNet which encapsulates Meso-4 and MesoInception-4, where the model performs a mesoscopic analysis on the dataset enabling it to reveal artifacts produced in the generation process.

Recently, (Nguyen, Yamagishi & Echizen 2019) proposed the use of capsule networks as the architecture for detecting tampering in videos. The capsule network utilizes a dynamic routing algorithm, highlighting the relation between different hierarchal pose objects after extracting features obtained from VGG-19 network. The capsule network comprises of three primary capsules and two output capsules one for real and one for fake images.

	Raw			
	DF	**F2F**	**FS**	**NT**
Steg. Features + SVM [27]	99.03	99.13	98.27	**99.88**
Cozzolino *et al.* [17]	98.83	98.56	98.89	**99.88**
Bayar and Stamm [10]	99.28	98.79	98.98	98.78
Rahmouni *et al.* [51]	98.03	98.96	98.94	96.06
MesoNet [5]	98.41	97.96	96.07	97.05
XceptionNet [14]	**99.59**	**99.61**	**99.14**	99.36

(Afchar, Nozick, Yamagishi & Echizen 2018)

Figure 2.3: Comparison between latest methods for Deepfake detection on Faceforensics++

2.2.2.2 Shallow Classifiers

Shallow classifiers also rely on the artifacts or inconsistencies of features between real and fake images. Unlike Deep classifiers, Shallow classifiers are based on examine the difference in frame progression as expected in real images then using an unsupervised learning technique to determine the class of the image/ frame.

(Yang, Li & Lyu 2019) proposed a method which observes the difference between 3D head poses and expectation for poses, bases on 68 facial landmarks for the central region of the face. Then the extracted features are fed to an SVM to obtain detection results.

Similarly, another method exploits the shortcomings in the generation process when examining the eyes, teeth and facial contours. The visual objects emerge from lack of regional accuracy, inaccurate or imprecise estimate of the lighting of the image, or imprecise estimate of the geometry underlying it. Missing reflections and incomplete information in the eye and teeth regions are used for deep-fakes identification, as well as texture characteristics derived from the facial field dependent on facial landmarks.

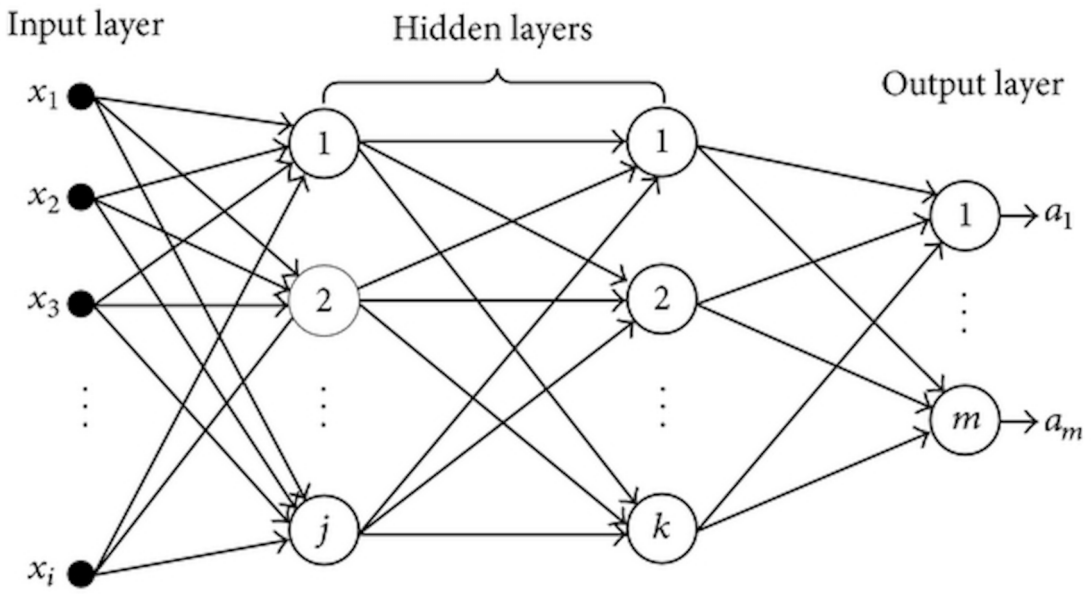


Figure 2.4: Artificial Neural Network

(Hao & Wang 2014)

2.3 Potential Models to Use

2.3.1 Artificial Neural Network

Artificial Neural Networks development have been inspired by the human brain and its key components. An artificial neural network is a Machine Learning algorithm, that have proven to outperform other generic algorithms in solving complex tasks and provide accurate reliable results in some problem domains. It attempts to mimic brain components and transmission of signals through artificial neurons; these are the basic component of an artificial neural network. Although inspiration came from human brain, they gradually have became quite different from biological neurons

Artificial Neural Networks consist of nodes/ neurons connected to one another in an arranged manner. They can consist of many layers where each layer has several neurons inside, each layer is connected to the one after starting from the input up till the output layer. When a network consists of more than 2 layers it sis called a deep neural network, middle layers are called hidden layers and may be fully connected to one another.

2.3.1.1 The Neuron

Each neuron in an ANN performs an operation and yields the result to the next connected neuron in the next layer. The neuron is an essential component of an ANN as it is responsible for performing operation calculations then updating in order to get better results.

The neuron or the Threshold Logical Unit is one of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt. The inputs and outputs have real numerical values and each input connection is associated with weight. The neuron computes a weighted sum of its inputs

$$Z = W^T \times X$$

, then applies a step function to that sum

$$step(Z)$$

and outputs the result where output is

$$h_w(x) = step(W^T \times X)$$

2.3.1.2 Activation Functions

In order for the ANN to work properly a key component is the activation function that is applied in the neuron before producing the output of the weighted sum of inputs. The activation function maps the output of the neuron to a set of logical values, helping normalize the output between 0 and 1 or -1 and 1. A key characteristic of an activation function is that it has to be differentiated over its output range as to avoid getting stuck where the gradient is 0, which will cause the network to fail updating weights and progressing when propagating the error gradient back the network.

2.6 shows the most popular activation functions used and their derivatives, as for the very basic step function we can see that its derivative is 0 everywhere which causes a problem when updating the gradients, therefore it have been replaced with other activations such as Relu where the deriative is only 0 at the extreme edge cases.

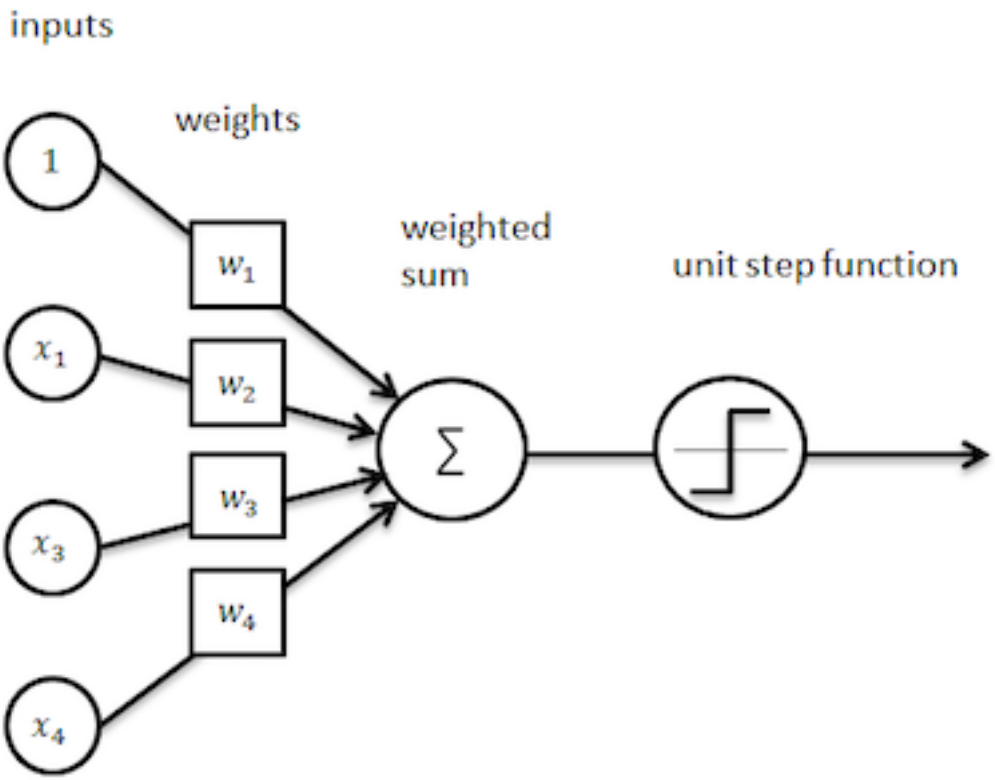


Figure 2.5: Threshold Logical Unit (Neuron)

(Hao & Wang 2014)

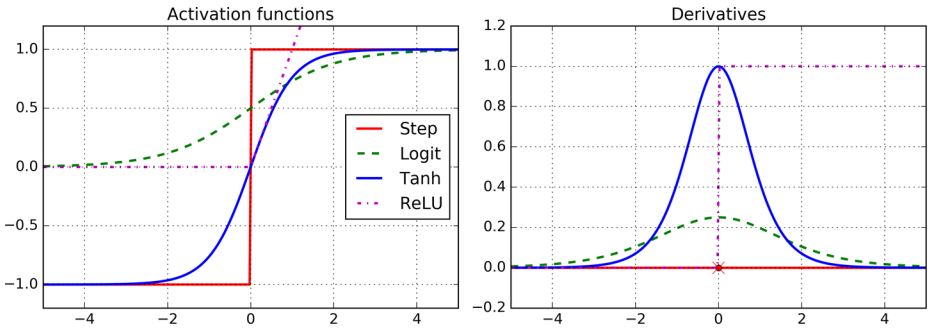
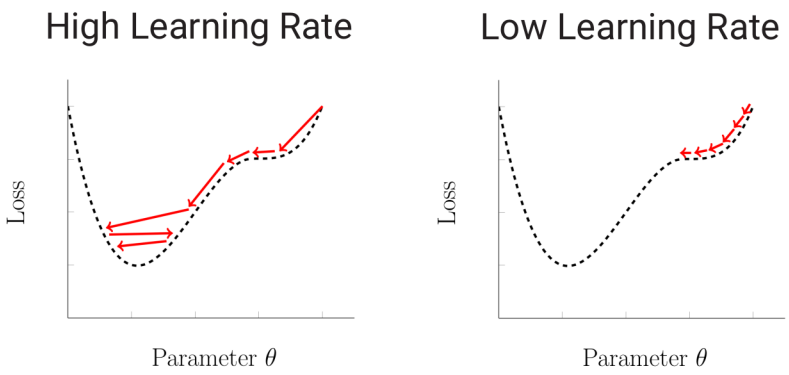


Figure 2.6: Activation functions and their derivatives

(Gron 2017)



(a) Gradient Descent

Figure 2.7: Gradient Descent and optimal Learning Rate

(617 2018)

2.3.1.3 Gradient Descent Optimization

Gradient Descent is a generic optimization algorithm that can be applied effectively on a range of problems. The main objective of Gradient descent is to alter parameters iteratively to minimize a cost function (e.g. RMSE).

Gradient Descent measures the local gradient of the error function with respect to the parameters vectors (θ) , then it moves in the direction which would minimize the cost function/ descending gradient until the gradient reaches 0 indicating that it have reached a minimum.

An important parameter in Gradient Descent is the step size, which is determined by the learning rate hyperparameter. The learning rate needs to be set carefully because if the learning rate is too small, the algorithm will have to go through many iterations in order to converge to a minimum, which will take very long time.

On the contrary, if the learning rate is too high, it is possible for the algorithm to overshoot the minimum, which may even lead to divergence of the algorithm with larger values than it was settling on initially, which is not useful. Therefore, the learning rate hyperparameter must be tweaked carefully to ensure successful operation of the Gradient Descent optimization algorithm.

2.3.1.4 Backpropagation

The term "backpropagation" refers to the notion of propagating gradient calculations backward through the network, could be described as gradient descent using reverse mode autodiff. For each training instance, the algorithm feeds it to the network and computes the output of every neuron in each consecutive layer.

Then it measures the output error (i.e. the difference between expected output and actual output), then it computes how much the last neuron contributed to that error, then it measures the contribution of the layer before and so on until it reaches the input layer. This reverse pass efficiently measures the error gradient across all weights by propagating the error gradient backward in the network.

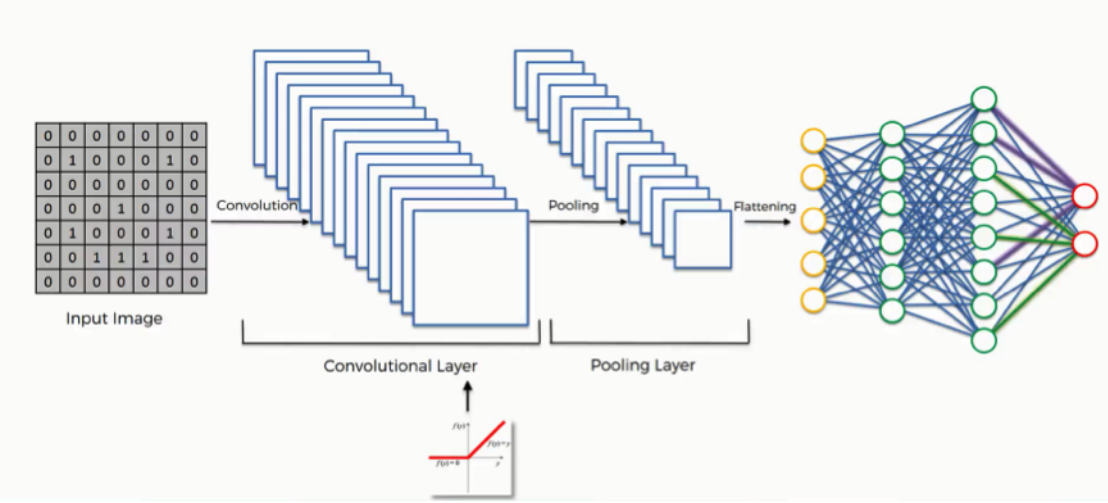
2.3.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) emerged from the study of the brain's visual cortex, and have been used extensively in image recognition tasks since the 1980s (Amor 2020). Recently, CNNs have been able to achieve superhuman performance on some complex visual tasks. CNNs have been particularly developed to accommodate image inputs and its architecture enables it to outperform other ANN in image recognition and detection tasks. Convolutional Neural Networks have assisted in automating feature extraction from images which were handcrafted earlier by computer vision engineers. The history of convolutional neural network design started with LeNet-style models , which were simple stacks of convolutions for feature extraction and max-pooling operations for spatial sub-sampling

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction (Saha 2018) ,which is accomplished through the convolutional layer.

2.3.2.1 Convolutional Layer

The Convolutional layer is considered the most important building block of CNNs. Neurons in the first Convolutional layer are only connected to their receptive fields in the



(a) Convolutional Neural Networks

Figure 2.8: A CNN Architecture

(Thi et al. 2019)

input image, unlike traditional ANN where layers are fully connected. Then neurons in the second layer are only connected only to neurons within a small rectangle in the first layer. This architecture allows the network to concentrate on low-level features such as color and gradient in the first hidden layer, then assemble them into higher level features such as edges in the next hidden layer up till the output layer 2.8.

2.3.3 GoogLeNet

In CNNs it has been around for quite a while that the most straightforward way of improving the performance of deep neural networks is by increasing their depth and the number of convolutional layers as well as increasing the width; the number of neurons at each layer. However, this poses many problems such as making the network prone to overfitting and decreasing its ability in generalizing well to new data, also this has a dramatic effect on increasing the use of computational resources.

In 2014 GoogLeNet also called InceptionV1 was a milestone in the development of CNN classifiers, achieving first place in the ILSVRC 2014 Detection Challenge on ImageNet, where 7 GoogLeNet models were trained and an ensemble was calculated from their aggregate predictions.

GoogLeNet development was based on the observation that salient parts in the image can have extremely large variation in size. For instance, an image with an object could have that object placed differently and occupying a various portions of the image in general. Because of that, the decision of the kernel size becomes quite an issue, as larger kernel sizes are preferred when the object is more distributed globally, while a smaller kernel size is preferred when the object is distributed more locally.

GoogLeNet strikes just the right balance between those to kernel approaches succeeding in capturing both global and local information for the object. Through a different approach, GoogLeNet employs multiple kernel sizes on the same level, where each layer has more than one kernel size operating on and extracting features.

2.3.3.1 Depthwise Separable Convolution

The depthwise separable convolution is the depthwise convolution followed by a pointwise convolution.

Depthwise convolution is the channel-wise $(n \times n)$ spatial convolution. Suppose in the figure above, we have 5 channels, then we will have 5 $(n \times n)$ spatial convolution.

Pointwise convolution actually is the (1×1) convolution to change the dimension (Tsang 2018).

2.3.3.2 Inception Module

The basic component of the GoogLeNet is the Inception module which first reduces the dimension of the input first using a (1×1) kernel, then (3×3) and (5×5) convolutions are performed on the input layer, with an additional pooling layer which has the (1×1) kernel reversed after its output.

2.3.4 MesoNet

MesoNet is a CNN model which analyses the frames of a video individually at a mesoscopic level. Stacked of several convolutional layers, batch normalization and max pooling operations; this model have proven to yield promising results when identifying deepfake

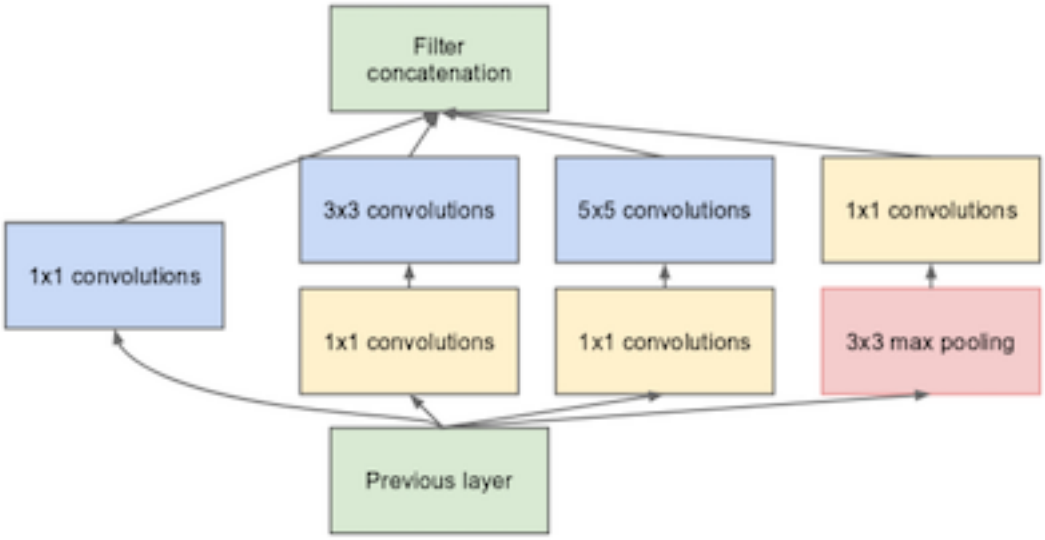


Figure 2.9: Inception Module with dimension reductions
(Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke & Rabinovich 2014)

from original images. Yet keeping the number of layers and parameters low at 27,977 parameters, thus computationally efficient to train.

MesoNet presents a method to automatically detect face tampering in videos, through a single frame prediction. This means that this method is regarded as a deep classifier searching for visual artifacts withing only a frame, unlike other methods searching for cross frame temporal inconsistencies.

The method particularly focus on two recent approaches used to generate hyper-realistic forged videos: deepfake and face2face. Traditional image forensics techniques are usually not well suited to videos due to their compression that strongly degrades the data (Afchar et al. 2018). Therefore, the use of a deep convolutional classifier is a promising approach when searching for artifacts.

2.3.4.1 Meso-4

The basic architecture of Meso starts consists of four layers of successive convolutions and pooling, and is followed by a dense network with one hidden layer. To improve generalization, the convolutional layers use ReLU activation functions to enable the model

deal with non-linearities and Batch Normalization to regularize the output and avoid the vanishing gradient effect, and the fully-connected layers use Dropout to regularize the network.

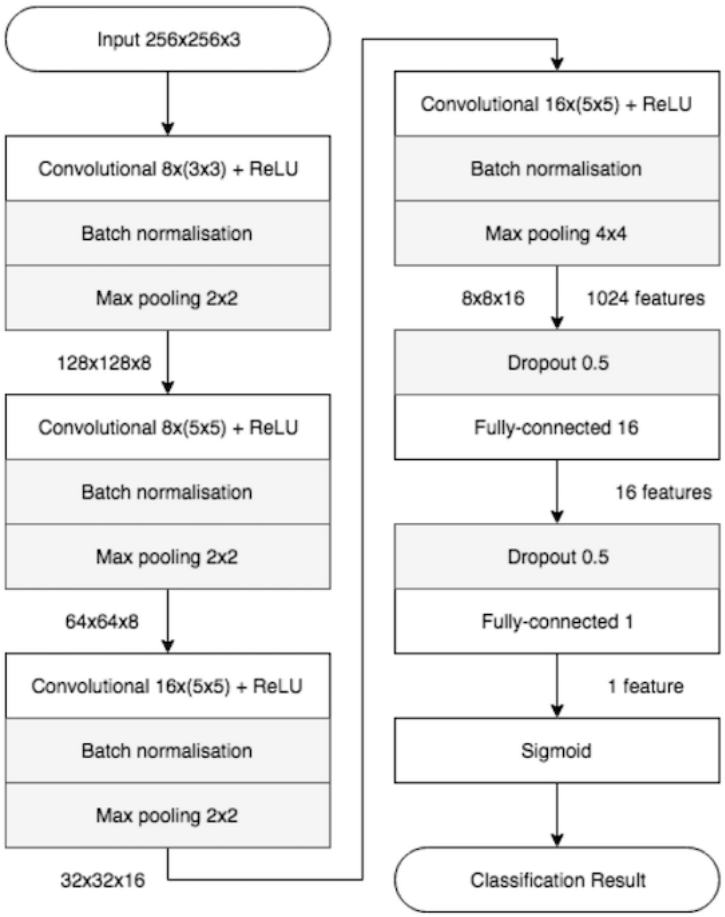


Figure 2.10: The network architecture of Meso-4.

(Afchar et al. 2018)

2.3.4.2 MesoInception-4

An alternative structure comprises of replacing the first two convolutional layers of Meso-4 by a variant of the inception module. The idea behind that is to increase the kernel sizes operating on one layer thus increasing the functional space of optimizing the model. Instead of the original (5×5), (3×3) dilated convolutions are used as to deal with various scale information. (1×1) dimension reduction kernels are used before the dilated

convolutions and an additional (1×1) convolution in parallel that acts as skip-connection between successive modules (Afchar et al. 2018).

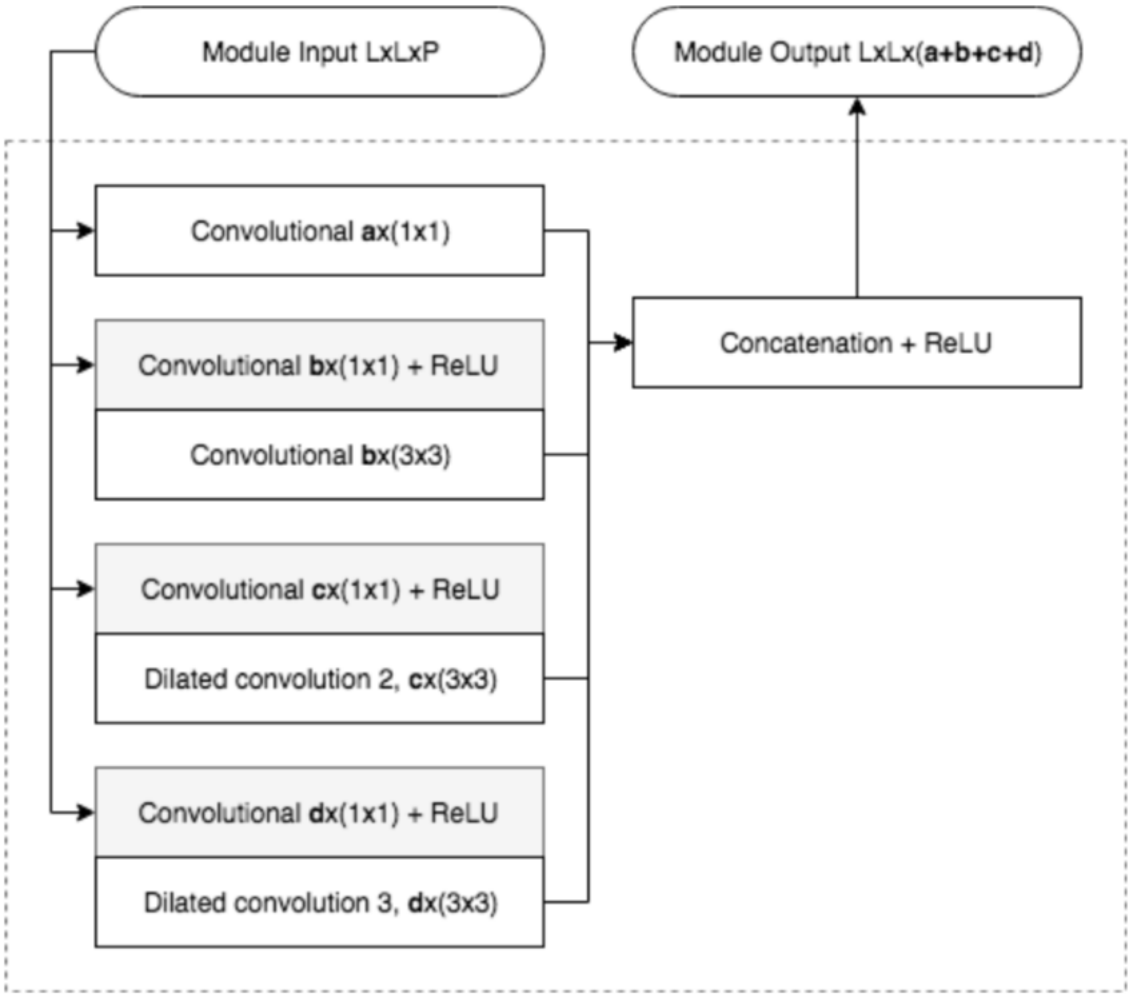


Figure 2.11: The network architecture of MesoInception-4.
(Afchar et al. 2018)

2.3.5 XceptionNet

Xception stands for "extreme inception, is one of the best performing architectures in the domain of image classification. It utilizes Inception modules concept but taken to an extreme in a way that keeps the number of parameters almost the same while achieving better accuracy than InceptionV3 network.

In Inception, first separating the image region and the color channels at the same time then it uses 1×1 convolutions to project the original input into several separate, smaller input spaces, and from each of those input spaces a different type of filter is used to transform those smaller 3D blocks of data.

On the other hand, XceptionNet takes this one step further. Instead of partitioning input data into several compressed chunks, it maps the spatial correlations for each output channel separately, and then performs a 1×1 depthwise convolution to capture cross-channel correlation. XceptionNet is based upon depthwise separable convolutions, which gives it the edge of performance over Inception based on the hypothesis that the mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled (Chollet 2016) .

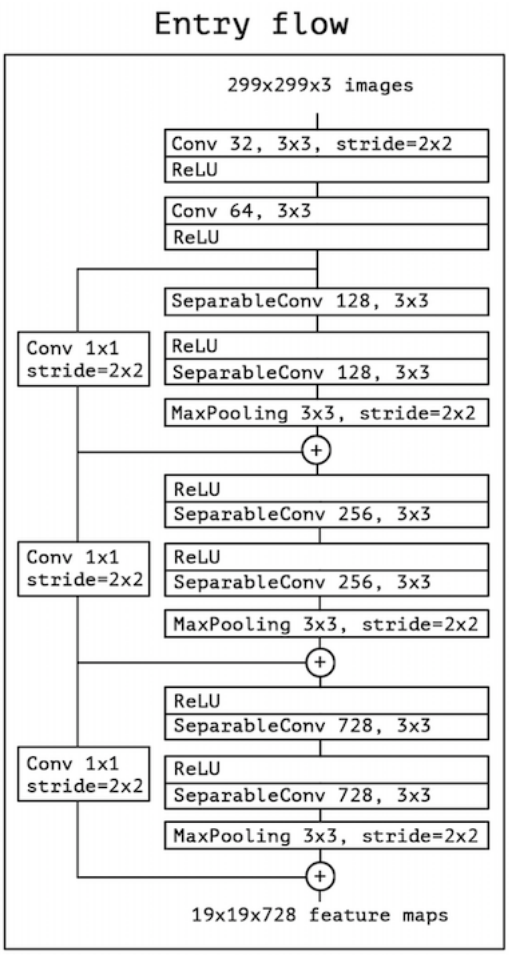
2.3.5.1 Modified Depthwise Separable Convolution in Xception

2.3.5.2 Architecture

The architecture of XceptionNet is schematically inspired by the VGG-16 architecture, the Inception architecture family of convolutional neural networks , which first demonstrated the advantages of factoring convolutions into multiple branches operating successively on channels and then on space and finally the concept of depthwise separable convolutions. The architecture is split into three stages:

2.3.6 Recurrent Neural Network

Unlike traditional neural networks, Recurrent Neural Networks have directed cycles in them. RNNs have loops which allows information to persist. ANNs consist of several layers where each layer is connected to the consecutive layer, from the input layer up till the output layer. In RNNs neurons in one layer are connected to the next one in the same layer which gives it that property of persistence.



(Afchar et al. 2018)

Figure 2.12: Entry Flow for XceptioNet

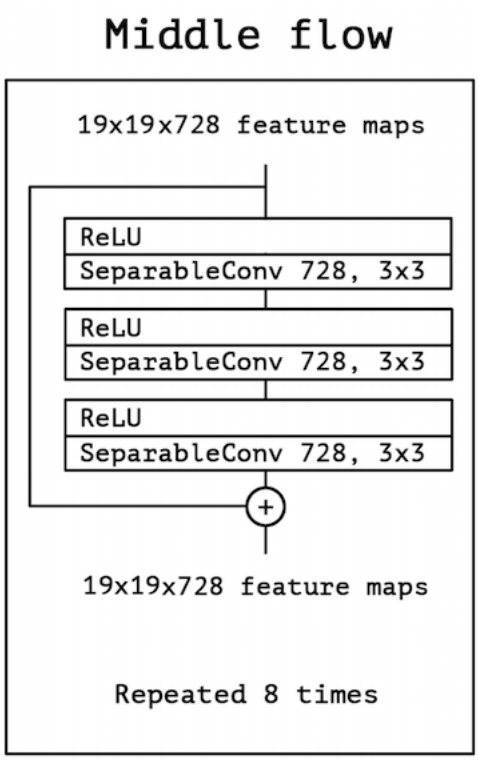
2.3.6.1 Recurrent Neurons

Recurrent neurons differ from traditional neurons in the sense that they receive multiple inputs, from neurons on the same layer. They operate on timesteps which is the sequence order input to the network.

For each timestep t the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$



(Afchar et al. 2018)

Figure 2.13: Middle Flow for XceptioNet

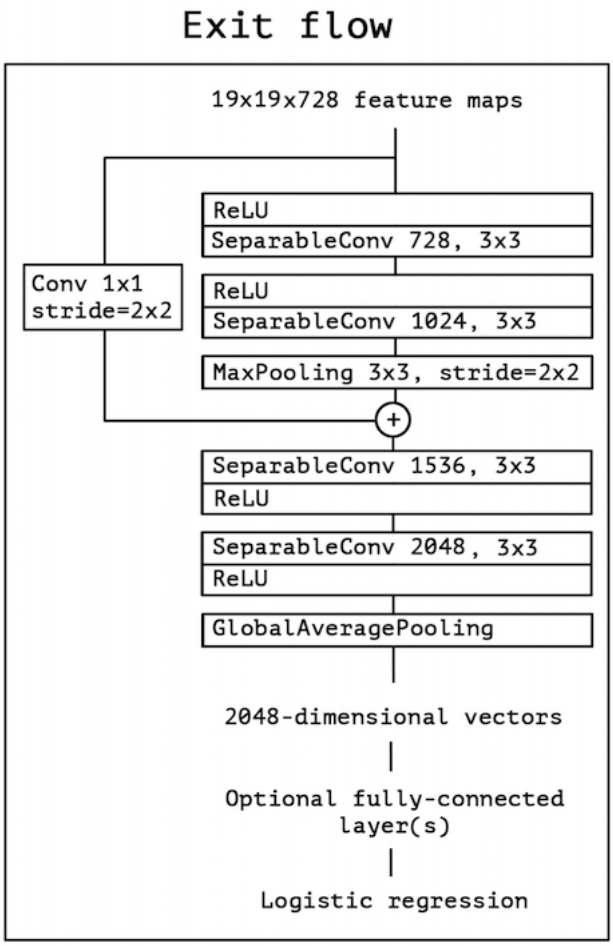
where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally by the neurons in the previous timestep, and g_1, g_2 are activation functions.

RNNs become very useful when the data is dependent on a sequence where previous inputs have an effect on next output. As the gap grows RNNs face difficulties storing information for a long period of time.That's were LSTMs tackle this problem.

2.3.6.2 LSTM

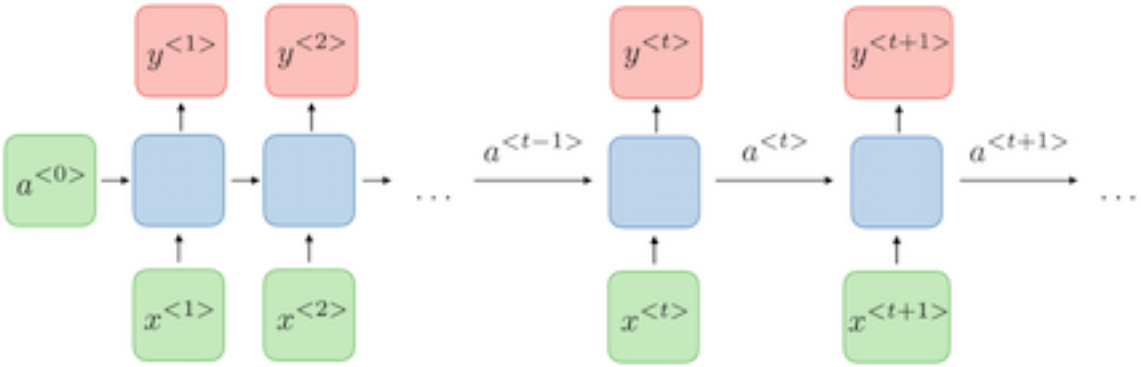
Long Short Term Memory LSTMs is a variant of RNN where they have special memory cells built in allowing it to store information for longer periods of time. Normal RNNs suffer from exploding and vanishing gradient problems where LSTMs do not. A set of cells is incorporated into LSTMs allowing it to decide when to store information and when to forget it, thsis et of cells is controlled by gates.

Gates are a way to optionally let information through. They are composed out of a



(Afchar et al. 2018)

Figure 2.14: Exit Flow for XceptioNet



(Amidi & Amidi 2017)

Figure 2.15: Architecture of a Recurrent Neural Network

sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs a value between 0 and 1, where 0 indicates to let nothing in and 1 let all. A combination of 3 gates is used in LSTMs to support this process of remembering information and deciding on which information to keep.

Chapter 3

Data Analysis

3.1 Description of Datasets

3.1.1 Deepfake Detection Challenge (DFDC)

The main dataset that is going to be used in this project is Deepfake Detection Challenge Dataset (DFDC), this dataset consists of a large number of deepfake videos gathered under different environments e.g. Indoors, Outdoors and with various lighting in background to ensure it is representative of the challenge posed in real life.

DFDC dataset is the most recent and yet the largest available dataset in this domain of deepfake detection utlising different deepfake generation techniques making it representable of the problem in hand. However, the dataset mainly focuses on including a large amount of fake videos and few real videos which poses a challenge of unbalanced data. Therefore, the possibility of applying balancing techniques is going to be considered in the project.

3.2 Dataset Construction

The Deepfake Detection Challenge dataset consists of 5,000 videos featuring two facial modification algorithms. The videos included different lighting conditions and head poses, participants were given the option to record their videos with any background, which yielded visually diverse backgrounds. The rough approximation of the general distribution

of gender and race across this dataset is 74% female and 26% male; and 68% Caucasian, 20% African-American, 9% east-Asian, and 3% south-Asian.

The Deepfake Detection Challenge dataset is a collaboration between Facebook, the Partnership on AI, Microsoft, Amazon and academics from Cornell Tech, MIT, University of Oxford, UC Berkeley, University of Maryland, College Park, and University of Albany-SUNY (*Data Policyl* n.d.). For this first version of the DFDC dataset, a small set of 66 individuals where chosen from the pool of crowdsourced actors, and split into a training and a testing set. This was done to avoid cross-set face swaps. (Dolhansky, Howes, Pflaum, Baram & Ferrer 2019)

Two methods used for generating this dataset (Method A and Method B), a number of face swaps were created across subjects with similar appearances inferring (skin tone, facial hair, glasses, etc.). After a pairwise model was trained on two identities, each was swapped using Methods A and B.

The first algorithm in the dataset used to produce this dataset method A did not produce sharp or believable face swaps if the subject's face is too close to the camera, indeed it required faces to be small in margin in the overall video taken from far away. Therefore, it resulted in easy-to spot fakes in rear camera conditions. All swaps were performed on videos where the average face size ratio was less than 0.25 (Dolhansky et al. 2019). The second method Method B produces lower quality video swaps but is similar to off-the shelf face swap algorithms as discussed in section 2. Further augmentations were performed on the dataset in order to make it more robus and difficult for deepfake detection including:

1. reduce the FPS of the video to 15
2. reduce the resolution of the video to 1/4 of its original size
3. reduce the overall encoding quality

The DFDC dataset contains a quite large number of tampered videos when compared to their original pairs. Almost for every real video there are 3.5 fake videos, unlike other deepfake datasets where some the ratio was 1:1 as in the earliest version of FaceForensics 2.5. The dataset contains 5214 original source videos that were created by actors to later

be used by Method A and Method B to alter the video and superimpose an image to the source actor.

Dataset	Ratio (tampered:original)	Total Videos	Source
Celeb-DF	1:0.51	1203	YouTube
FaceForensics	1:1.00	2008	YouTube
FaceForensics++	1:0.25	5000	YouTube
DeepFakeDetection	1:0.12	3363	Actors
DFDC Preview Dataset	1:0.28	5214	Actors

Table 3.1: Specs of the most relevant deepfake datasets

3.3 Dataset Exploration

The data is comprised of .mp4 files, split into compressed sets of 10GB per folder. The total size for the dataset is almost 470 GB. Each of the 50 sub-folders of the dataset contains a subset of the .mp4 videos accompanied with a metadata.json.

Metadata Columns:

1. **filename** - the filename of the video
2. **label** - whether the video is REAL or FAKE
3. **original** - in the case that a train set video is FAKE, the original video is listed here
4. **split** - this is always equal to "train".

When examining the metadata.json head the top 5 results are displayed in 3.1.

In order to better understand the metadata.json and corresponding video files I have created a function to display the first frame of few fake videos from the 1st folder for the dataset out of 50. Shown in figure 3.1. As well as also displaying a sample of 3 real videos, to further understand the variations and inconsistencies in the dataset 3.3. Through inspection at first it can be noted that some fake videos are really difficult to tell from real, that is due to many reasons being the high quality tampering tool used

	label	split	original
aagfhgtpmv.mp4	FAKE	train	vudstovrck.mp4
aapnvogymq.mp4	FAKE	train	jdubbvfswz.mp4
abarnvbtwb.mp4	REAL	train	None
abofeumbvv.mp4	FAKE	train	atvmxvwyns.mp4
abqwwspghj.mp4	FAKE	train	qzimuostzz.mp4

Figure 3.1: metadata.json Head

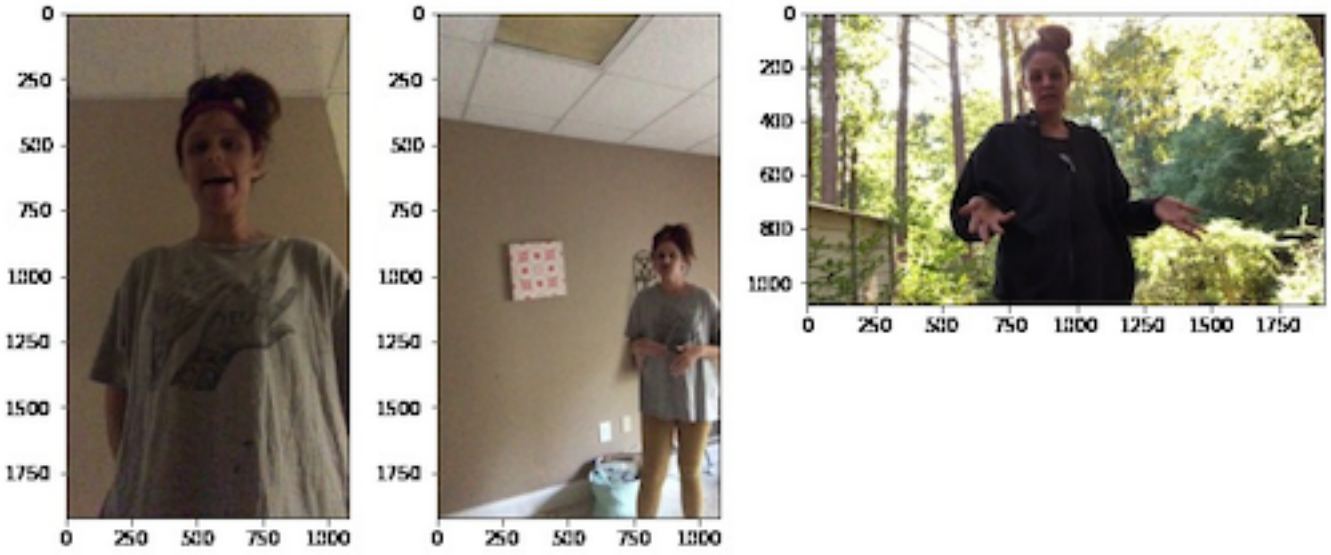


Figure 3.2: First frame of 3 sample Fake Videos

Figure 3.3: First frame of 3 sample Real Videos

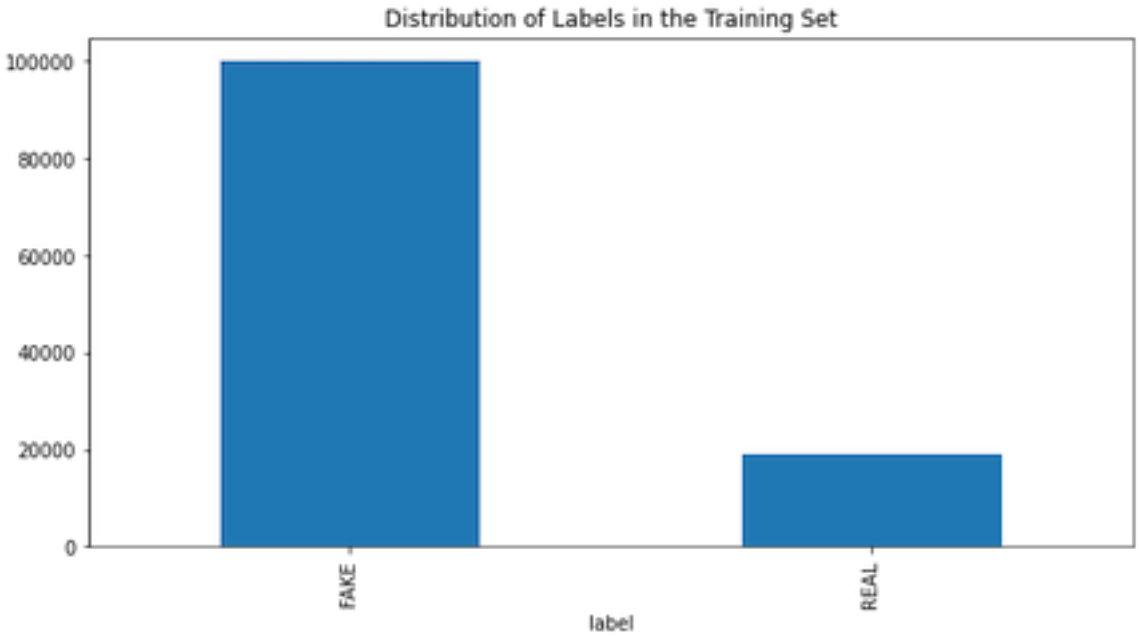


Figure 3.4: Distribution of Labels

or the degradation of frame resolution to match with the tampering tool used as lower quality images/ videos are easier to resemble a real when it is a fake one.

This dataset consists of videos of people with various poses, illuminations as well as different backgrounds and lighting. Therefore the next issue in developing the DeepFake detection pipeline is using a face detection package that has a high recall rate as well as its speed and its ability to be utilised on the GPU for boosting up the frames per second detetcted.

When combining all of the 50 metadata.json files and examining the distribution of real:fake videos we can see in 3.4 that the number of fake videos is way larger than the number of real videos in the dataset which poses a problem of an imbalanced dataset. Therefore the issue of balancing the dataset is going to be considered in the next section.

3.4 Face Detection, Cropping and Alignment

The first stage of the pipeline is detecting faces in video frames in order to be able to further manipulate it and inspect its originality using a Deep Learning technique. One of the most important characteristics of this dataset is its considerably large size of

470GB. Therefore the utilisation of a fast face detetction package is crucial and testing its performance in order to ensure that it is going to achieve good results on the dataset as a whole.

The three face detetcion packages are going to be tested on a sample video; which has a dark background and the face is not particularly visible on three different resolutions in order to ensure the robustness of the face detetction package chosen. The video used for testing consists of 300 frames.

facenet-pytorch

A PyTorch FaceNet (Schroff, Kalenichenko & Philbin 2015) paper to train a model of facial recognition using Triplet Loss and Cross Entropy Loss with Center Loss; Training is performed on the dataset VGGFace2 (Cao, Shen, Xie, Parkhi & Zisserman 2018) which contains 3.3 million face images based on more than 9000 human identities.

dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It contains various image processing algorithms and the dlib face detection and recognition is one of them. This tool extracts and maps an image of a human face to a 128 dimensional vector space where images of the same person are near to each other and images from different people are far apart.

mtcnn

MTCNN (Multi-task Cascaded Convolutional Networks) is a deep cascaded multi-task framework which exploits the inherent correlation between them to boost up their performance. In particular, the framework adopts a cascaded structure with three stages of carefully designed deep convolutional networks that predict face and landmark location in a coarse-to-fine manner (Zhang, Zhang, Li & Qiao 2016). MTCNN is availaible pretrained on two datasets (VGGFace2 and CASIA-Webface, and their corresponding facial recognition networks InceptionResnetV1 that can be used later to extract 512 dimension feature vectors for faces.

Summary of Results

Package	FPS (1080x1920)	FPS (720x1280)	FPS (540x960)
facenet-pytorch	12.97	20.32	25.50
dlib	3.80	8.39	14.53
mtcnn	3.04	5.70	8.23

Table 3.2: Summary of Results comparing various face detection packages on 300 frames

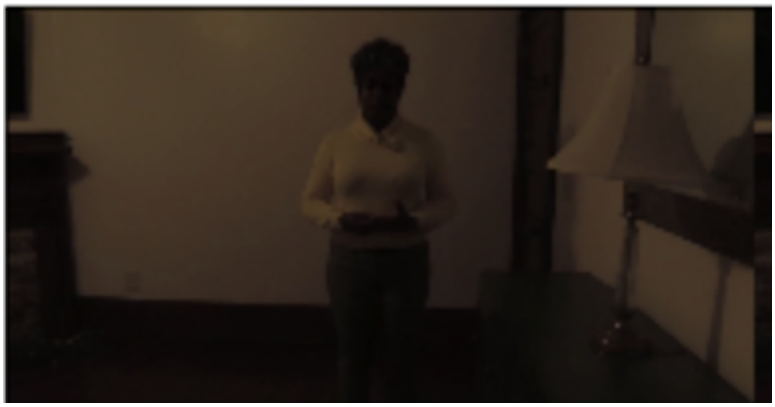


Figure 3.5: Video frames

Looking at the results in 3.2 we can see that MTCNN performs the lowest in terms of speed but in terms of recall MTCNN achieves the best results under various poses, illuminations and backgrounds. MTCNN produces really few false positives and fails to detect faces in really rare cases. Therefore the use of MTCNN in extracting faces is going to be considered while modifying the MTCNN algorithm in a way that can drastically improve speed through exploiting the similarities between coordinates of frames one after another.

3.4.1 MTCNN

The implementation of MTCNN that is going to be used is a variant of MTCNN; FastMTCNN this algorithm demonstrates how to achieve extremely efficient face detection specifically in videos, by taking advantage of similarities between adjacent frames, reaching 55 FPS, faster than the other implementations. Most importantly, MTCNN fails in very rare cases and has a higher recall rate than the other two algorithms. In 3.5 the illumination is poor with very dull lighting, DLIB and facenet-pytorch fail to detect frames in this case, while MTCNN produced a stable result with outstanding 3.6.

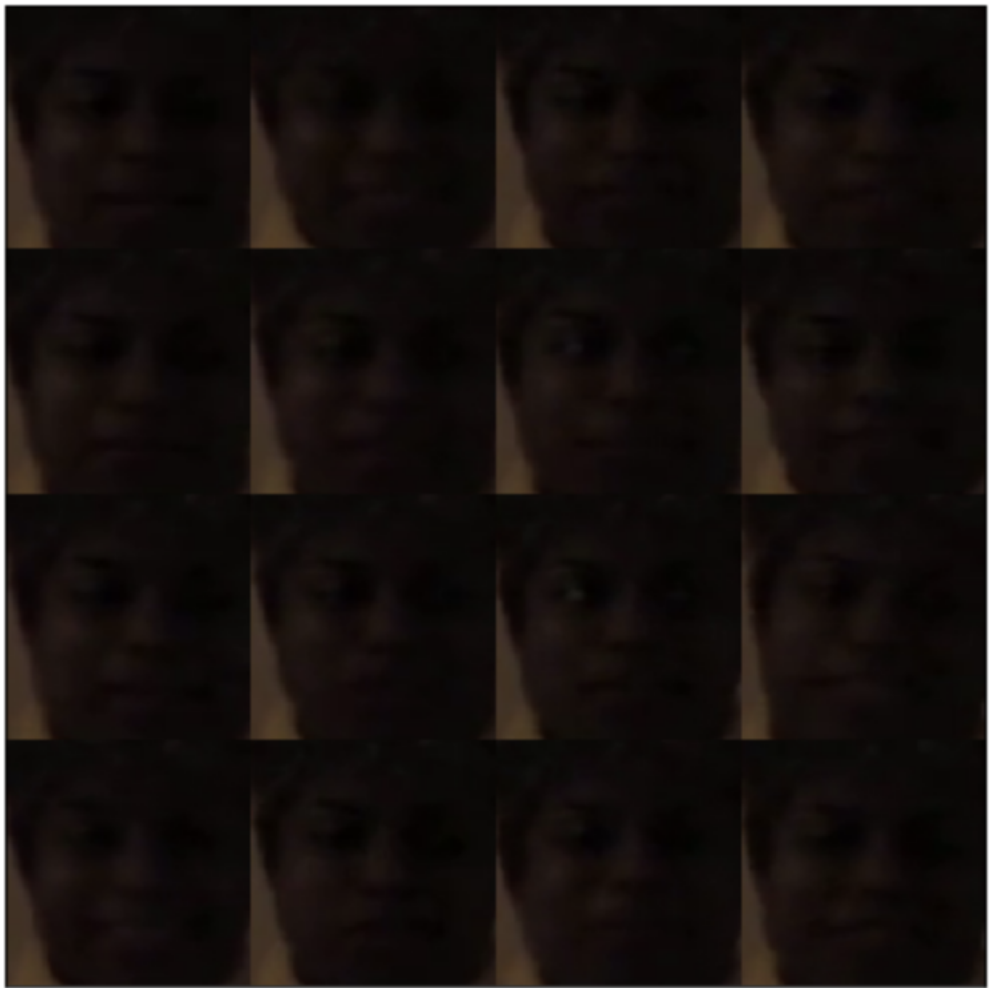


Figure 3.6: Cropped and aligned faces by MTCNN

Fast MTCNN

In the Fast MTCNN implementation several parameters need to be tweaked in order to achieve the desired output of faces cropped and have them saved in the correct directory.

The first parameter is the stride which is a modification of MTCNN in which face detection is performed on only every N frames, and applied to all frames. In this I am going to set the stride to 1 as we are interested in saving all frames in a consecutive manner in order to enable the models being trained later on to spot inter-frame inconsistencies and for that to be achieved each frame needs to be cropped and saved. Then, the bounding boxes returned for frame 0 would be naively applied to frames 1 and 2. Similarly, the detections for frame 3 are applied to frames 4 and 5, and the detections for frames 6 are applied to frames 7 and 8. Assuming that faces do not move significantly between frames which is true for low stride numbers which is going to be used as 1 in this implementation.

If faces in frames move faster they are going t be blurry anyways so they wont be useful for the model.

The algorithm uses a smaller scaling factor (0.6 vs 0.709) than the original MTCNN algorithm to construct the scaling pyramid applied to input images. This modification makes the algorithm faster as a trade off with accuracy. Additionally, the margin is a parameter that adjusts the algorithms face cropping dictating the zoom out on the face when . Margin being set as 0 in this case in order to enable the models to focus on the facial tampering rather than being misguided by the surroundings context. Keep all is another parameter which when set to true all faces in the video for different actors is returned, so if a video has 2 actors faces of both actors are going to be returned.

3.5 Balancing the Dataset

In this section I am going to consider applying balancing techniques in order to ensure that the number of samples for each category (fake, real) is equal. This will significantly enable the model to train on equal proportions of data to make better regardless of the DeepLearning technique implemented. In this dataset the number of fake videos is considerably larger than the number of real videos as for each real video several techniques were utilized in order to create fake pairs from the same original real video. With ratio fake:real of 1:0.28 3.1 almost 72% of the data contains fake videos. 2.9.

3.5.0.1 Undersampling

Under-sampling balances the dataset by reducing the size of the abundant class. This method is used when quantity of data is sufficient. By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class, a balanced new dataset can be retrieved for further modelling (Wu & Radewagen 2017).

In this dataset the total number of video for the real videos is 6387 which is way less than the number of fake videos. Therefore under-sampling technique is going to reduce the number of fake videos from 35921 to the same as real videos so the overall when training all models they have a ratio of 1:1 instead of the overall aggregate ratio of 1:0.28.

Sampling	Fake	Real
Original	35921	6387
Undersampled	6387	6387

Table 3.3: Number of videos when considering the first 15 Frames of a Video (Training Data set)

3.5.0.2 Oversampling

Oversampling balances the dataset by adding further data for the unique or less occurring class samples in the dataset. Generally oversampling is regarded as a more robust technique that enables the model to see even more data, therefore leading generally to better accuracy. Unlike under sampling where some of the dataset is thrown away and not make use of, oversampling make efficient use of the dataset available, while also providing more samples for the rare class from another datasets.

In this project, I am going to experiment with both sampling techniques as to understand which provides better scores and enables the model to generalize better on an unseen test set.

In the case of single frame prediction, Flickr-Faces-HQ Dataset (FFHQ) is going to be used to balance the dataset. FFHQ consists of 70,000 high-quality PNG images and also contains considerable variation in terms of age, ethnicity and image background. Thumbnails subset of this dataset is going to be used in this project along with the main DFDC dataset.

Sampling	Fake	Real
Original	35921	6387
Oversampled	35921	35921

Table 3.4: Number of videos when considering the first 15 Frames of a Video (Training Data set)

3.6 Data Augmentations

Data Augmentation is technique used to create further more samples of the same data. Such techniques are used in order to allow the model to generalize better on the data and prevent the problem of overfitting; where the model learns too much about the data that it is not able to unseen test cases. This particularly could have an effect on the validation accuracy making it low when at the same time the training accuracy is getting higher. Therefore,techniques such as Rotation, Flipping, Saturation, Color, Hue and Contrast are altered to create a new instance of the image, providing a broader range of data for the model to train on and with variations such as the model could face in an unseen test set.

In this project I am going to experiment with various data augmentation techniques, not only limited to the ones mentioned above but also other where appropriate to the problem at hand. Alterations in resolution and frame degradation is going to be also considered as this is a particular vulnerability of deepfakes when the resolution is low it gets more difficult for the model to identify a real from a forged video as facial expressions are more generic leading to less inconsistencies being spotted.

3.7 Manipulate video sequence by Rolling Window

The rolling window is a popular technique utilized extensively in image recognition and object detection problems. In the current context, a rolling window plays an important role in dividing the input array of image sequences into smaller chunks sequentially, so that an LSTM model is able to observe and extract features local to the whole sequence between n consecutive frames.

First, a sequence of N frames is created, then a window of width W is picked as a parameter. The window starts from element

$$(N_1, N_2, ..., N_W)$$

and so on till the window has W frames in. Then the algorithm increments the starting index by one and iterates up till the window is reached again. So in total the number of

sub arrays created is going to be equal to the sequence length less window size + 1.

Let's consider an array of elements

$$(N_1, N_2, N_3, N_4, N_5)$$

, with window

$$W = 2$$

. The sub-arrays created are going to all have width of 2 elements and there will be

$$N/W + 1$$

which in this case is going to be

$$5/2 + 1$$

and the sub-arrays created are going to be

$$(N_1, N_2)$$

,

$$(N_2, N_3)$$

,

$$(N_3, N_4)$$

,

$$(N_4, N_5)$$

totalling of 4 sub-arrays created.

In our context, the same is applied to a sequence of images yielding the sub-arrays of image sequences desired for the LSTM to train on in section 6. This image sequence is going to be particularily useful to enable the LSTM to learn the sequence progression in between frames rather than just passing the whole sequence of 15 frames all at once.

3.8 Creating Training and Testing

After preprocessing the data and preparing it into various forms for the different models to be trained, by detetcing faces and cropping them using the MTCNN algorithm . Such

as creating the rolling window function for the LSTM model while other CNN models will only be based on single frame prediction, the next stage is to split the data accordingly into training and testing in order to be able to evaluate the performance of the models created.

The dataset is going to be split after balancing have been applied as to provide an unbiased sample test set for the models to make predictions on. The dataset is loaded into two variables holding the real and fake portions of the dataset, then loaded balanced into respective input data X and labels y.

The dataset is going to be split between training which is going to be used by the model to train upon, and validation which is going to validate the performance of the model and compute metrics such as accuracy and loss, where validation is an unseen set, different from the training set. Finally a test set which is going to be withheld until all experiments have been conducted on the models and validation scores have reached promising values and stop improving anymore.

In this project the split between training, validation and testing is going to be as depicted in 3.5.

Splt	Train	Validation	Test
Ratio	0.7	0.2	0.1

Table 3.5: Dataset split

K-fold Cross Validation

In the training process, concerning the training dataset and the validation dataset, K-fold Cross Validation is going to be performed as to better evaluate the model's performance. Cross validation is a sampling technique used to evaluate the model's performance on a limited data sample. The technique has a a parameter K which refers to the number of splits the data set is divided into. This is a very useful technique when the dataset is limited and there is great importance to evaluate the model's performance without bias.

The steps of the K-fold algorithm:

1. Shuffle the dataset randomly

2. Split the dataset into k groups

3. Choose a group to be test dataset

4. Keep the other k-1 groups for training

5. Fit model on training and evaluate it on validation dataset

6. Choose a group to be test data set

7. Evaluate the skill of the model using averaging its performance on folds

Loss Calculation

Due to the nature of the task being a binary classification problem, the loss metric used for these experiments is going to be binary cross entropy/ log loss, given by equation 3.1 :

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot log_e(\hat{y_i}) + (1 - y_i) \cdot log_e(1 - \hat{y_i})]. \tag{3.1}$$

where n is the number of samples, $\hat{y_i}$ is the predicted probability of deepfakes, y_i is 1 if the sample is fake and 0 if if the sample is real, the smaller the log loss the better the model is performing on the dataset.

Chapter 4

MesoNet

MesoNet family of models includes two pretrained model networks Meso-4 and MesoInception-4, where in MesoInception-4 the first two convolutional layers of Meso4 by a variant of the inception module are replaced; as to make it better extract features from input images on a wider scale per layer as discussed in literature review.

Looking at the benchmarks and accuracies for both models, it becomes clear that MesoInception-4 outperforms Meso-4. Therefore in this section I am going to experiment with MesoInception-4 and finetune it to the given DFDC dataset.

MesoInception-4 was trained on two different datasets, were the deepfake generation techniques were different. The first was trained on Deepfake generation technique and the second was trained by Face2Face GAN technique on FaceForensics dataset. Each of the models is going to be used in order to compare their performance on the DFDC dataset and understand which might yield better results whether through transfer learning or whether training the model from scratch is going to be more promising.

4.1 Transfer Learning

In transfer learning previous knowledge model on specific tasks is utilized through model weight initialization, particularily this can have a good effect on the learning process of the model were weights loaded for some layers and enabling some of the last layers to update weights and learn new information for this specific task of binary classification.

The MesoInception-4 model consists of 33 layers, starting with the input layer then 7 stacked convolutional layers concatenated together then followed by batch normalisation and pooling layers 4.1.

In this experiment the weights are going to be revoked for the input layer as well as the last 8 layers and only weights are going to be loaded for the remaining 24 middle layers consisting of all convolutional layers. As to enable the model learn the new inputs and modify the last 8 layers weights up to this specific dataset including dense layers. This might be particularly useful due to the similarity of the task and that the initial MesoInception-4 was originally trained on, which is the detection of deepfakes.

Initially, there are several model parameters to be chosen affecting the model's learning capability. The optimizer chosen to begin with is Adam which is an adaptive learning rate optimization algorithm, which can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. Leveraging the benefits of both; it is promising to yield good results.

4.1.1 MesoInception-4 - Deepfake

Deepfake is a dataset gathered by the authors of MesoNet comprising of 175 rushes of forged videos have been collected from different platforms. Running the model for 20 epochs was another decision as to look at when does the loss is stops improving. As well as exploring the weight changes of the convolutional layers as to understand the behavior of the network and the backpropagation algorithm on updating the weights of the pretrained the network.

In 4.2, the training loss is decreasing for all learning rates attempted (0.01, 0.0005, 0.0001) gradually. Specifically at learning rates of 0.0001 and 0.0005 they have the same training loss curve with final loss at epoch 20 of 0.1820. On the training set they all demonstrate normal decrease in loss behavior.

4.3 shows the validation loss for the learning rates tested. For learning rate of 0.0005 the validation loss decreases the most from epoch 1 till epoch 12, then starts fluctuating and increasing. The learning rate 0.0001 demonstrated similar behavior reaching the same loss of 0.0005 at later epoch interval of 16. Both reached the lowest validation loss

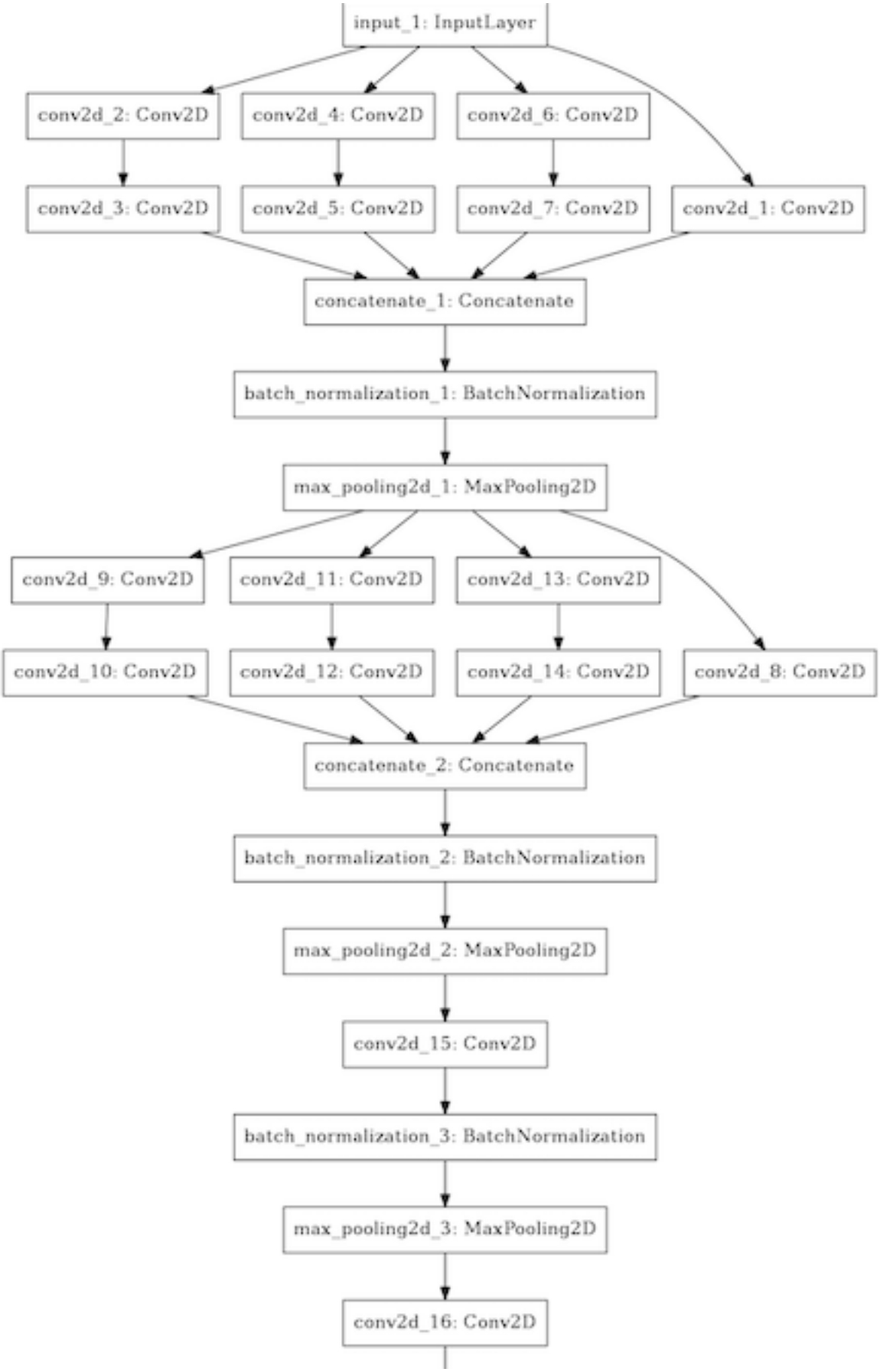


Figure 4.1: MesoInception-4 Layers pretrained weights

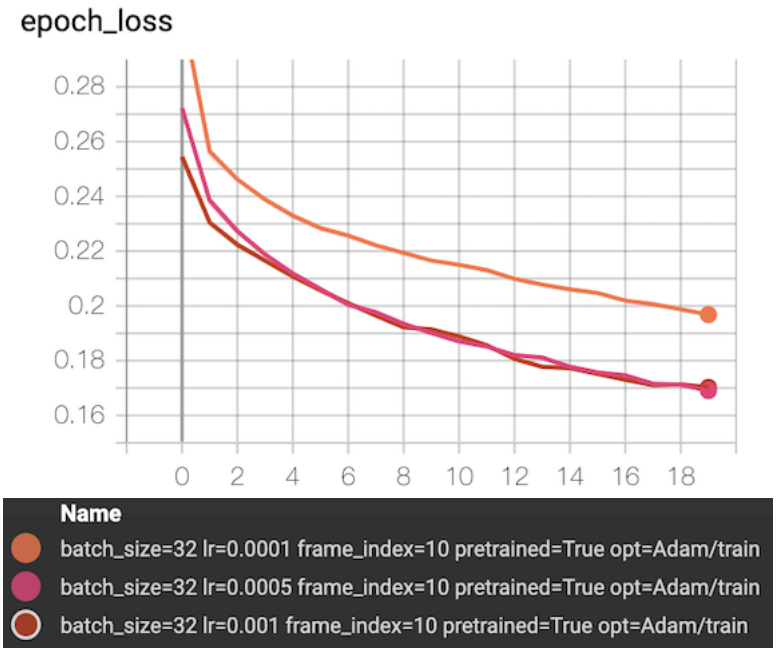


Figure 4.2: Training Loss

possible but at different epoch number, this shows that both learning rates are considered favorable but 0.0005 is able to reach that minimum faster than 0.0001 and that is due to the increase in learning rate value.

Learning rate of 0.0001 showed the poorest decrease in validation loss being lower than the previous two learning rates tested, indicating that at intervals it is taking too long to reach the minima indeed. To further understand the behaviour the learning rate is having on the kernel weights, analyzing their histogram bin counts is going to be performed next.

Comparing the weights distribution for the first conv2d inception kernel, we can see that with a learning rate of 0.001 the model is learning the most up till around epoch 12, then showing small variations in bin counts as learning becomes more difficult when reaching the minima. On the other hand, for learning rate 0.0005 the histogram shows that variations that were happening quickly similarly for learning rate of 0.001 at epoch 12, which took more epochs for learning rate 0.0001 to reach.

This confirms the hypothesis that the learning rate of 0.001 is better for this experiment as it is able to reach the minima faster than the lower learning rate of 0.0005, at the same time without overshooting the minima loss. Both show very similar behavior indicating

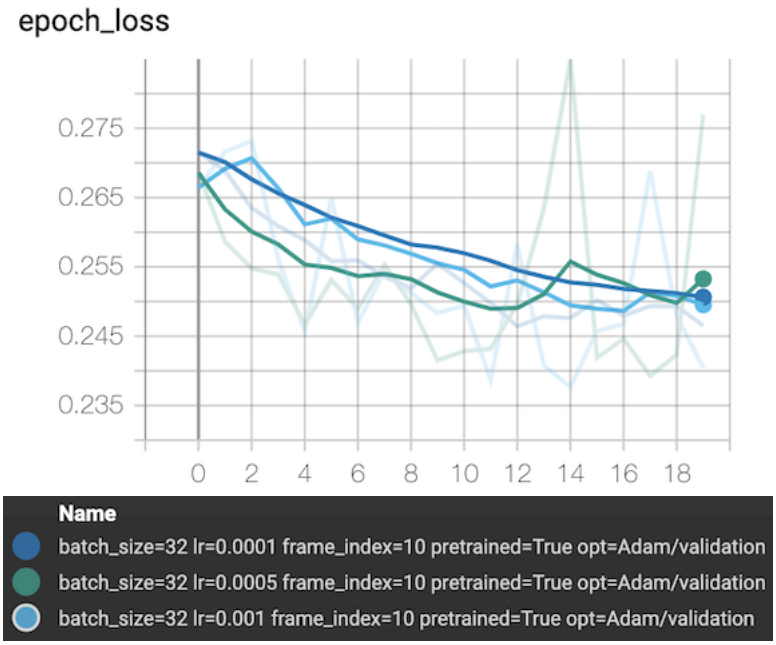


Figure 4.3: Validation Loss

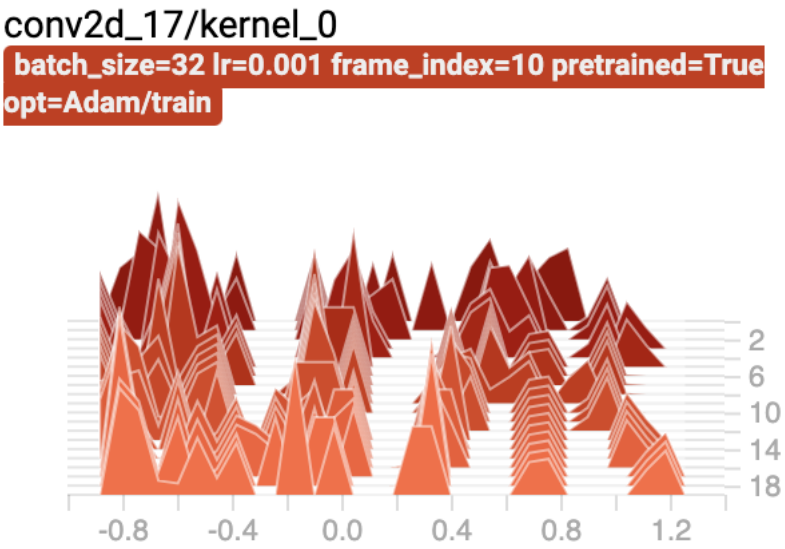


Figure 4.4: Convolutional Layer Weights (Learning Rate = 0.001)

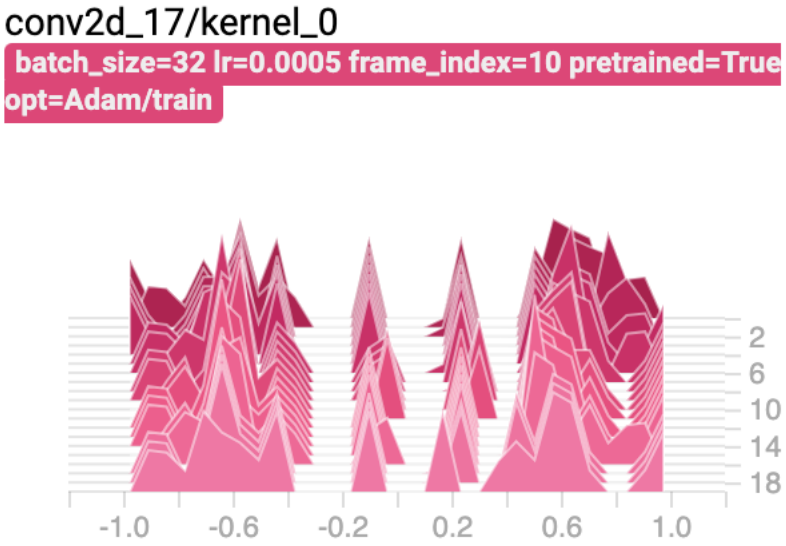


Figure 4.5: Convolutional Layer Weights (Learning Rate = 0.0005)

that this region of learning rate is the best for this algorithm and input data. Otherwise for learning rate of 0.0001 is very slow and taking much longer epoch duration to make changes/ updates to the weights kernel. Thus learning very minimal new information on every epoch.

4.1.2 MesoInception-4 - Face2face

Face2face is a generation technique which is based on tracking facial expression of the target person using a dense photo-metric consistency measure, reenactment is then achieved by fast and efficient deformation transfer between source and target (Thies, Zollhofer, Stamminger, Theobalt & Nießner 2016).

In 4.6 the training loss is decreasing gradually and consistently for both learning rates tested (0.001 and 0.0001). This shows that the model's weight initialisation for the 33 layers is helping the training loss to decrease and enabling faster learning on the dataset.

However, in 4.7 the validation loss for learning rate 0.001 is decreases in the first epochs, then starts fluctuating and increasing in value up till the last epoch. This can indicate that although the training loss is decreasing gradually; this learning rate of 0.001 might be too large for the model, thus overshooting some of the minimas when it comes to

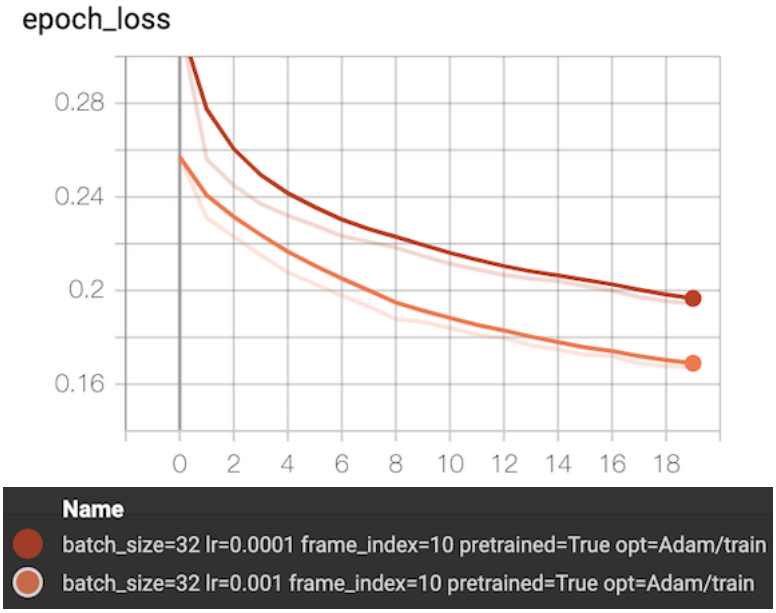


Figure 4.6: Training Loss

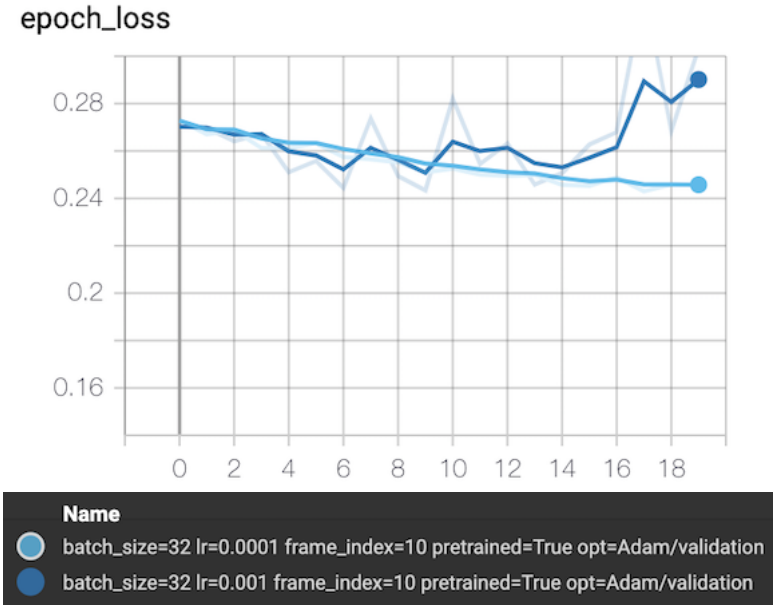


Figure 4.7: Validation Loss

validation and casuing overfitting in the epochs 6 to 20. The validation loss for the learning rate 0.0001 kept on decreasing smoothly reaching a final value of 0.2408 at the last epoch, while the lowest loss for 0.001 was at epoch 6 with 0.2508.

Therefore, in this experiment using the pretrained weights for the Face2Face model, to

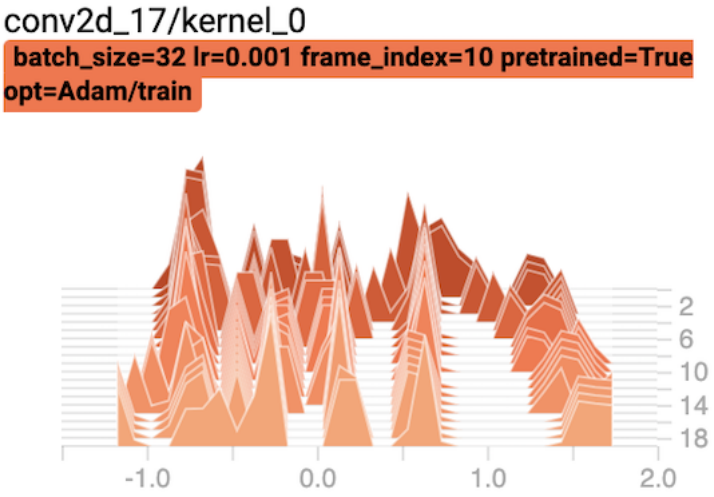


Figure 4.8: Convolutional Layer Weights (Learning Rate = 0.001)

initailise the weights of the model, a learning rate of 0.001 seemed to be the optimal value. Causing the training and validation loss both to decrease gradually with almost no fluctuations.

Looking at 4.8 we can see that the model's weight distribution is changing rapidly and bins are accumulating from 0 parts of the graph, to form peaks later on in the training session this might be an indication that the model is learning fast at 0.001 or overshooting the minimum, but when compared to 4.9, it becomes evident that the learning rate of 0.001 is indeed too large.

That is due to the fact that the histogram counts for learning rate of 0.0001 is more steady and doesn't change much; this is expected because the model's weight initialization is based on the deepfake problem, therefore it might be the case that the initial weights of the model were indeed the most optimum for that very first layer extracting basic features from images.

4.2 Training from scratch

In this section the MesoInception-4 model is going to be trained from scratch with random weights initialization for all of its layers. There are several hyperparameters that are going to be altered accordingly to check what improves the validation loss as measured in Log

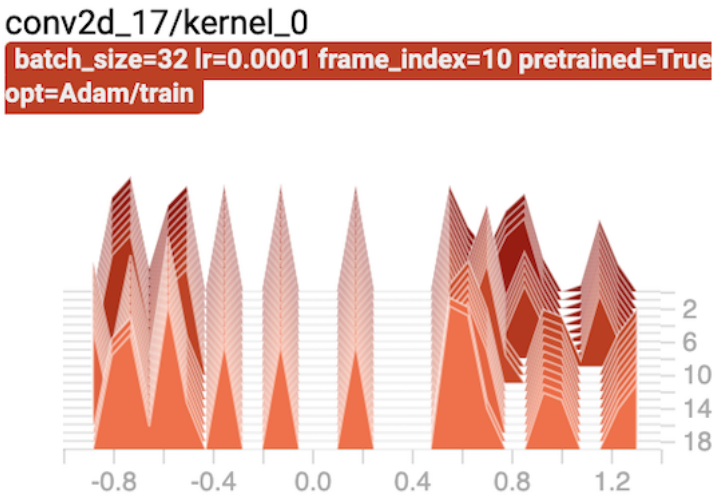


Figure 4.9: Convolutional Layer Weights (Learning Rate = 0.0001)

Loss or Binary Crossentropy.

The first hyperparameter that is going to be changed is the learning rate. Starting with 1e-4 up to 1e-3 and recording its performance while the rest of the parameters are going to be the same. Constant parameters are going to be the batch size set at 32, running the models for 20 epochs for comparison and using Adam as an optimizer.

4.2.1 Learning rate 0.0001 - Adam Optimizer

Looking at the training loss graph for the first run with learning rate set as 0.0001 we can see that the loss is decreasing gradually from around 0.26 up till almost 0.2 on the final training epoch, which shows that the model is learning the data behavior with room for improvement 4.10.

The validation loss is decreasing rapidly in the first 10 epochs which shows that this learning rate is not too small for the model, then the validation loss it is not decreasing much after epoch 10 with 0.2504. In epoch 18 the validation loss drops the lowest at 24.86. Between epochs 10 to 18 the validation loss graph doesn't decrease, rather it fluctuates going up and down until reaching the lowest on epoch 18.

In order to gain better understanding of the networks behavior and how the learning rate is affecting the learning process, examining the early convolutional layer's weights is

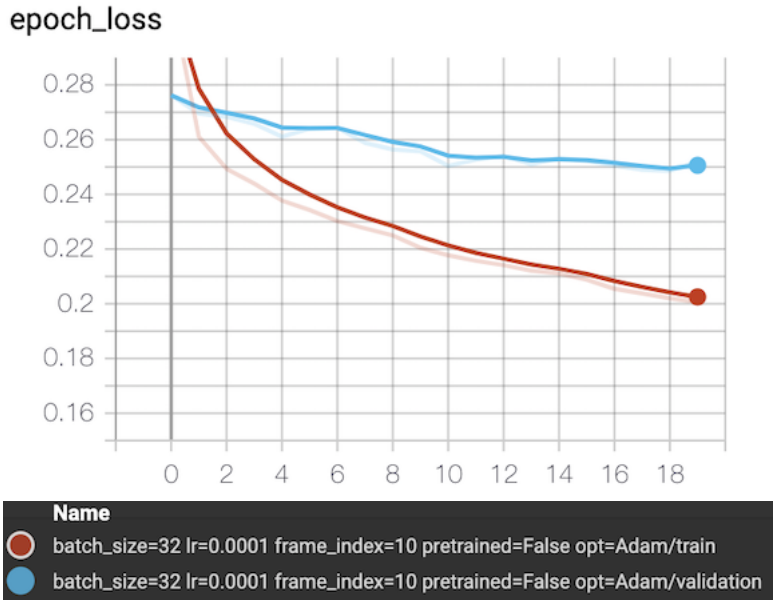


Figure 4.10: Loss (Learning Rate = 0.0001)

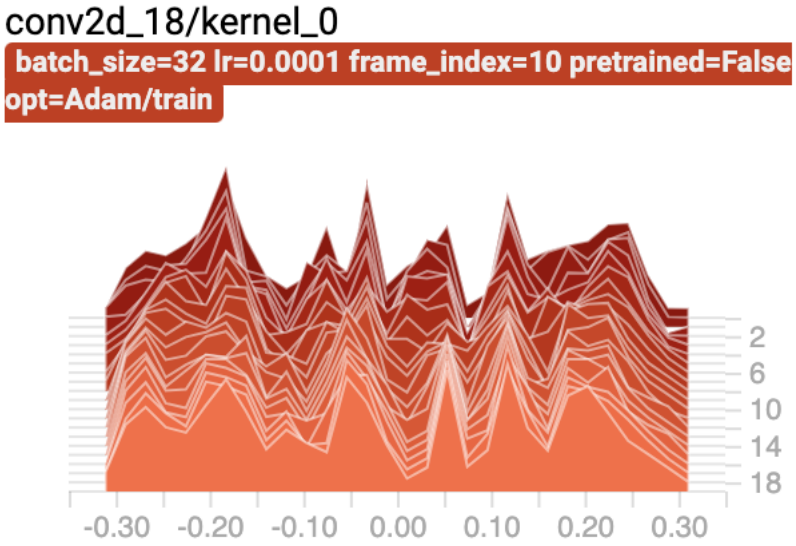


Figure 4.11: First Convoloutial Layer weights

essential. Starting with the second convolutional layer in the inception module , which is a Conv2D operation after a pointwise Conv2D operation was performed as to reduce the inputs size. Looking at 4.11 we can see that the layers' weights range from -0.30 to 0.30 with irregular updates and changes to the weights bin count in the histogram as the epochs increase; which is a good sign that the model is learning and updating weights gradually to decrease the training loss.

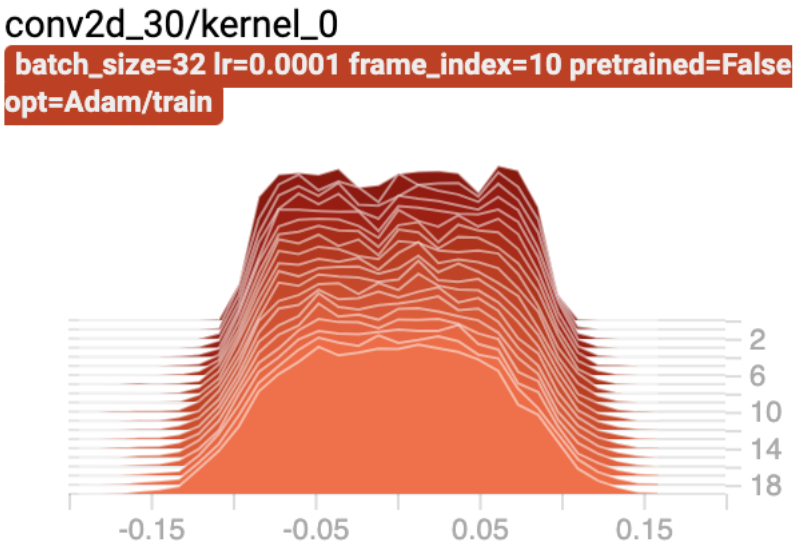


Figure 4.12: Last Convoloutial Layer weights

Weights are distributed with along the standard deviation of the graph with peaks were the model thinks that is the best value for some of the neurons in the layer. When comparing back with the validation loss graph, we can also see that also in the histogram the model is not learning much (i.e. changing the bin counts) at almost after epoch 10, thus confirming the hypothesis that the number of epochs can be reduced for this learning rate.

Looking at 4.12 we can see the last convolutional layer weights (Conv15); is centered around 0 with standard deviation of 0.15 at epoch 20. The standard deviation didn't change much from epoch 1 when it was almost 0.11. This shows that the model's last layers aren't learning much as the distribution remains almost the same for the whole training session. This also demonstrates that the last convolutional layers aren't learning much about the data but rather the initial convolutional layers were the ones extracting the features and learning more about the data. No significant changes is encountered in the histogram bins count throughout the epochs.

These graphs in conjunction with the training and validation loss graph 4.10 show that the model was able to learn about the data and loss was decreasing gradually up till epoch 10, but the model is not yet using it's full potential. Experimenting with different learning rates or training the model for longer epochs iterations might be necessary. In

the next experiment I am going to increase the learning rate from 0.0001 to 0.001 to get an idea on whether the network's learning process is slow.

4.2.2 Learning rate 0.001 - Adam Optimizer

The training loss is decreasing steadily all over the epochs from 1 to 20. While the validation loss starts of decreasing rapidly with a large gradient up till epoch 6, then the graph encounters some fluctuations but decreasing overall in loss up till epoch 16. The validation loss graph is fluctuating right after epoch 6 but decreasing overall.

Increasing the learning rate had an effect on the model's loss by making it reach a lower value in the same number of epochs. This might be an indication that a higher learning rate might be more optimal for the model to learn on the data 4.13. The learning rate of 0.001 enabled the model's validation loss to get lower with 0.2377 when compared to 0.2506 at learning rate of 0.0001.

The first convolutional layer's weights are distributed from -0.6 to 1 4.14, with peaks gradually changing to different areas of the histogram where the model thinks that is the best weight for the network. This convolutional layer weights demonstrate that the model is successful learning and extracting features from video frames. The model is regularly updating weights and distribution is changing up till the last epoch, which when analyzed with the validation loss we can see that the model is learning successfully the feature of the data input and haven't reached a plateau yet. At the last epoch the model is changing the weights aggressively which is a hint that the model had great potential to further improve and larger epoch number might be a good idea, despite the fluctuations in the validation loss graph.

The last layer is showing a uniform distribution around 0 in the early epochs with standard deviation of 0.2, while during the very last epochs the model's standard deviation increases almost to 1 lowering the sharp peaks at the 0 bin. This is another sign that the model was on the edge of further improving at the last epoch, therefore the probability that this is an optimal learning rate is going to be considered while running the model for more than 20 epochs to examine the improvement in loss and weight updates.

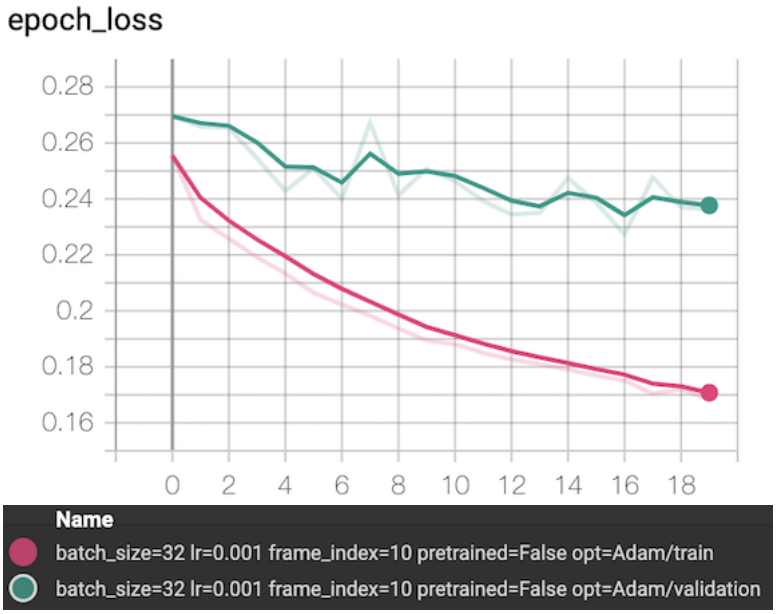


Figure 4.13: Loss (Learning Rate = 0.001)

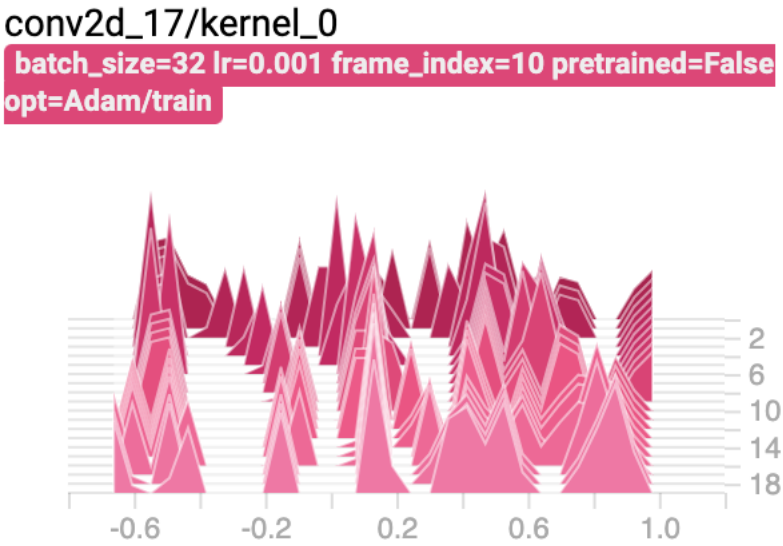


Figure 4.14: First Convoloutial Layer weights

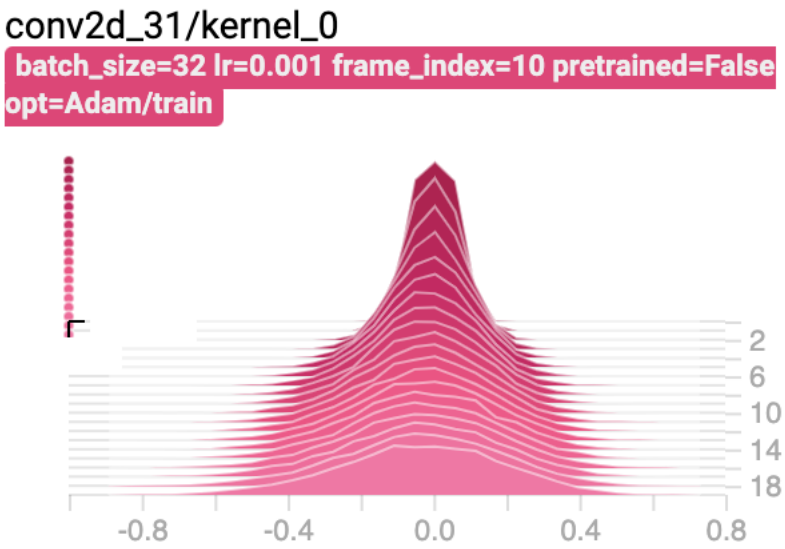


Figure 4.15: Last Convoloutial Layer weights

4.2.3 Comparison on Validation Accuracy

To further confirm that 0.001 is a good value for the learning rate. Testing the model with a learning rate in between 0.001 and 0.0001 is important as to understand the effect of balancing the learning rate between those two experiments conducted. Learning rate of 0.0005 showed a constant decrease in validation loss up till epoch 14 then the loss increases again up till the last epoch. This experiment of setting the learning rate to a value in between did not improve the validation loss.

Setting the learning rate to 0.001 proved to yield the lowest validation loss 4.16. Scoring 0.2386 when compared with learning rate of 0.0001 with loss at 0.2549 and 0.0005 with loss at 0.251. Therefore it can be confirmed that 0.001 is the optimal learning rate for this model with Adam as an optimizer.

Summary

It can be concluded from these various experiments using pretrained and untrained versions of Meso-Inception-4 that the optimal set up is the following:

1. **Deepfake Pretrained** has an optimal learning rate of **0.001** and batch size of **32** with **Adam** as an optimizer

Figure 4.16: Model Validation Loss

2. **Face2Face Pretrained** has an optimal learning rate of **0.0001** and batch size of **32** with **Adam** as an optimizer

3. **Untrained version** has an optimal learning rate of **0.001** and batch size of **32** with **Adam** as an optimizer

Chapter 5

XceptionNet

XceptionNet is one of the standard SOTA models for image classification. With the concept of Inception Modules taken one step further with deep separable convolutions it outperforms other models in this domain.

In this section I am going to explore the use of XceptionNet on the deepfake classification task due to its high performance in image classification and its ability to extract salient features from images. Experimenting with several hyper parameters and pre trained layers is going to highlight how useful this model is in this domain. Noting that it has scored the highest on the FaceForensics++ dataset, which has similar characteristics to the DFDC dataset considered in this project.

5.1 Transfer Learning

Previous model knowledge in the area of image classification is going to be harnessed through the use of the same weights initialization for the model layers, except replacing the last Xception block of the network with new a block that is capable of producing the binary output desired instead of classifying 1000 classes of images as trained on ImageNet. This will give the opportunity for the model to fine-tune the weights for the layers by continuing backpropagation, thus enabling the model network gaining knowledge of the deepfake characteristics while benefiting from it being pretrained on numerous classes.

5.1.1 Learning rate 0.0001 - Adam Optimizer

Initially setting the learning rate at a low value of 0.0001 was chosen as to enable the pretrained model update weights slowly and not to distort the initialized weights 5.1. The training loss graph shows decrease in loss, while the validation loss graph 5.2 shows fluctuations around a steady loss value over the epochs and doesn't show any significant decrease over the epochs.

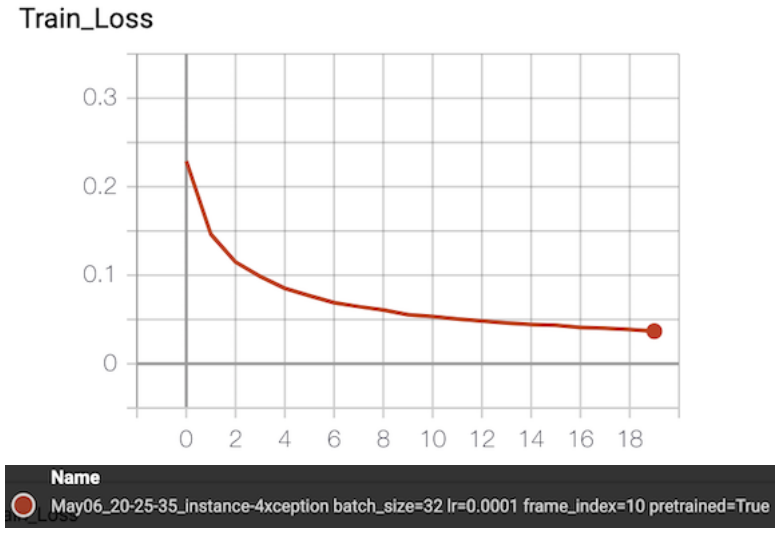


Figure 5.1: Training Loss

Potentially, this might be a sign of over-fitting that the network's training loss is decreasing but the validation loss is not decreasing although it started at a low value of 0.2065, but over the epochs the loss keeps fluctuating and not decreasing overall.

Further examining the model's weights at the final Xception block 5.3;which performs a depth wise convolution on the previous layer, it shows that the model's parameters aren't changing much and the histogram's weights are centred around 0 with a standard deviation of 0.2 which is not changing across epochs.

The same for the layer's gradient's histogram 5.4, it shows the same gradient counts over the epochs which indicated that the model is not particularly learning anything new after the first 1 or two epochs with the highest bin count at 0, with no significant updates to the model. This might be because of the small learning rate set to the model so it's unable

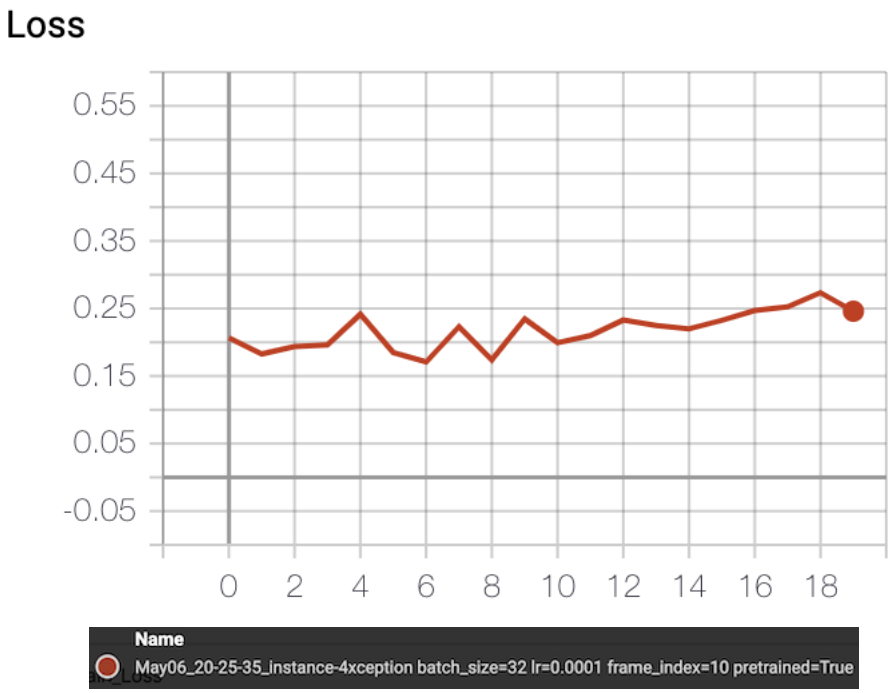


Figure 5.2: Validation Loss

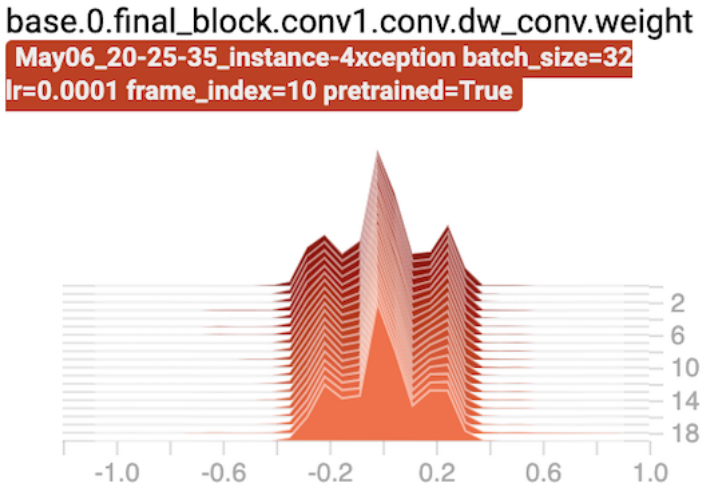


Figure 5.3: Last DepthWise Convoloutial Layer weights

to update weights effectively during each epoch, therefore teh next step is to increase the learning rates and note what effect does that have on the validation loss and the model's layers weights.

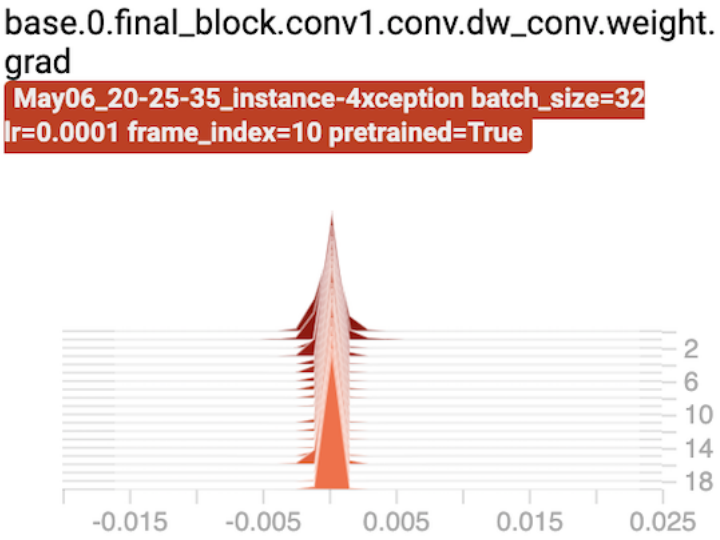


Figure 5.4: Last DepthWise Convoloutial Layer weights gradients

5.1.2 Increasing Learning rate

Increasing the learning rate to 0.01 and 0.001 as an attempt to examine whether this case of fluctuating validation accuracy is due to the learning rate being too low and thus affecting the model's layers weights that they fail to update accordingly.

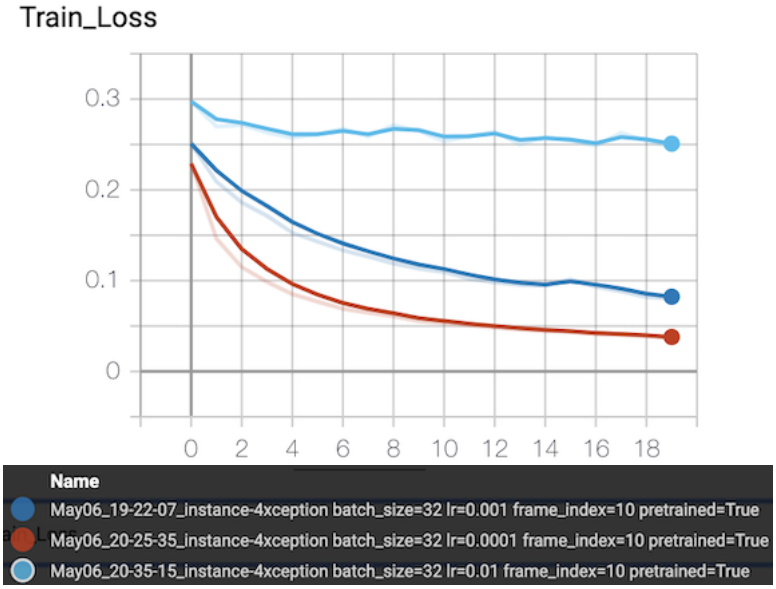


Figure 5.5: Training Loss

Looking at 5.5 the training loss decreases for both learning rates where learning Rate = 0.001 in blue and learning Rate = 0.001 in light blue, where as training loss is the lowest for learning rate of 0.0001. While the validation loss also indicates that increasing the learning rate did not have an improvements on making the loss graph behave normally as in the training loss graph.

Specifically looking at the validation loss for learning rate 0.01 we can see that the model keep on overshooting the minimum represented as fluctuations in the graph. These fluctuations might indicate that somewhere between 0.0001 and 0.001 is the optimal learning rate, but the issue is that the validation loss for 0.001 and 0.0001 is not decreasing steadily, it decreases for the first few epochs then starts on fluctuating. The validation loss typically should be decreasing in parallel with training loss but that is not evident in all experiments up till now. Unfortunately this might be an indication that the there is an overfitting problem.

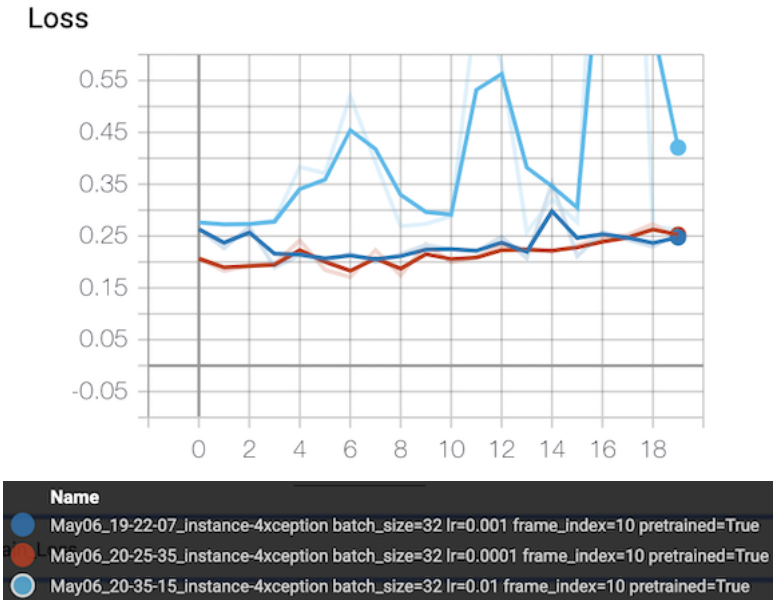


Figure 5.6: Validation Loss

When analyzing the validation loss graphs in conjunction with model's weights gradient updates at 5.4, it raises another suggestion which might be that the data is insufficient for such big model. This case of overfitting might be due to the Xception model's architecture. XceptionNet has 36 convolutional layers which yields a huge number of parameters

around 23 Million which requires a lot of input data to avoid overfitting. For example this model was trained on ImageNet with almost 1.3 Million images.

Therefore the possibility of increasing the dataset size is going to be considered, but also another probable cause for the steady validation loss might be due that the model is able to achieve the best loss it can in the very first few epochs and thus, only training for shorter number of epochs is sufficient and more training starts to cause this problem of overfitting.

5.2 Training from scratch

In this section XceptionNet is going to be trained from scratch, with random weights initialization instead. First the model is going to be trained for 20 epochs with an initial learning rate of 0.0001 and a batch size of 32, while tracking the models' loss metric on training. Optimizer Adam have proven to work best with XceptionNet on various image classification tasks, therefore it is going to be used in this first experiment.

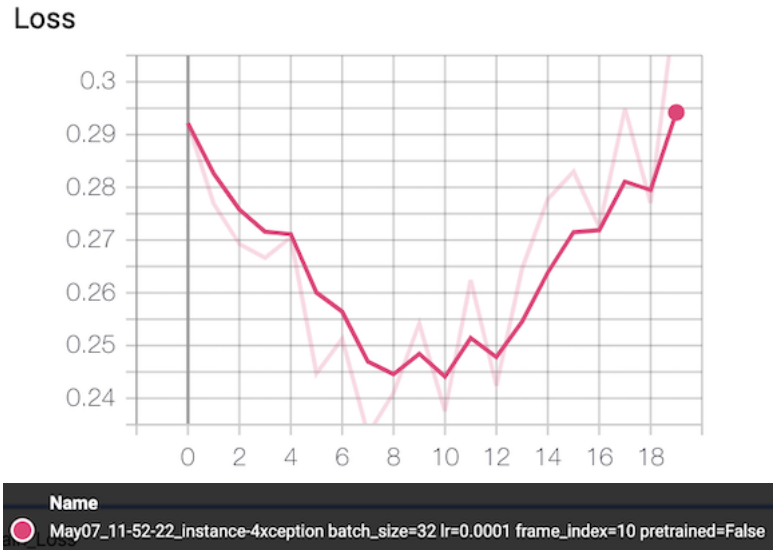


Figure 5.7: Training Loss

When inspecting the training loss graph we can see that the loss is going down smoothly while the validation loss is fluctuating. Typically, it is expected that both metrics decrease in conjunction, the validation loss 5.8 shows that after epoch 10 the model starts

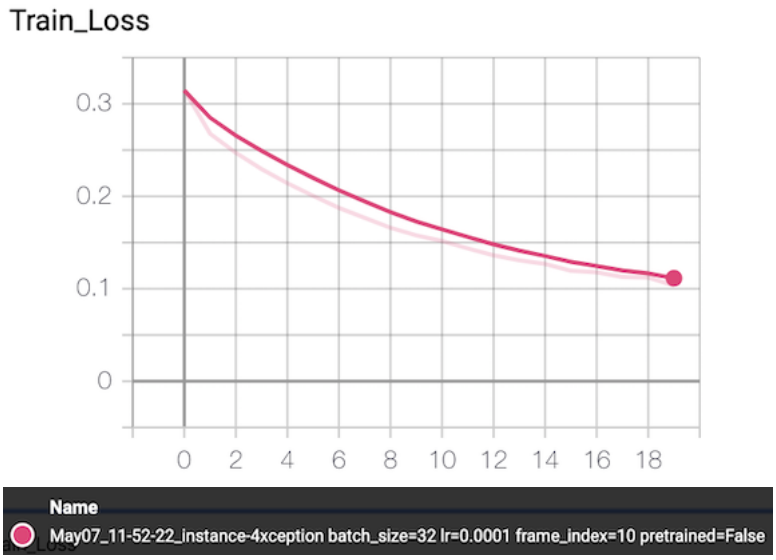


Figure 5.8: Validation Loss

overfitting and loss is increasing significantly up till it initial value as it started in epoch 0.

The validation loss graph indicates that the dataset size might be insufficient to train the model from scratch, therefore the model's training loss 5.7 is decreasing while the validation loss reaches lowest on epoch 10 then the model starts to overfit, leading to increase in loss.

Therefore, in this network it is considered best if the model is trained on a pretrained version of xception as in the previous section and we cannot afford to train the model fom scratch due to dataset size limitation.

Summary

After experimenting with various set ups for the Xception model, it can be concluded that the optimum training conditions for XceptionNet model on this dataset is when using a **pretrained version** for model weight's initialization, using a learning rate of **0.001** and a batch size of **32** with **Adam** as an optimizer.

Chapter 6

LSTM

After baseline models have been replicated on this dataset, the next step is developing a different architecture deep model which is able to exploit intra-frame inconsistencies rather inter-frame inconsistencies utlised in the baseline models in sections 4 and 5 by MesoNet and XceptionNet which are variants of convolutional neural networks.

In this section an attempt at creating an LSTM model which is able to learn the differences between fakes and originals when frames of a video are progressing. Such inconsistencies could be the eye blinking, the face movement as a whole or in face regional inconsistencies.

As discussed in literature review LSTMs have the capability of storing temporal information. This fetaure is going to be particularily useful when looking at deepfakes as to manipulate such inconsistencies in a way that have them stored in memory cells over time. Only teh first 10 frames are going to be considered as when increasing to a value more than 10 frames start dropping and this will reduce the data size enormously

6.1 Preprocessing

The process of preparing the data for the LSTM begins with extracting the number of frames required. Unlike previous CNN models where prediction was performed on only 1 frames; in LSTM the data input is going to be a series of frames. In order to enable the LSTM to learn the most the first 10 face frames of a video are going to be extracted

as a series, then calculating the CNN embedding for facial images. In this experiment InceptionResnetV1 is going to be used as a feature extractor, extracting 512 feature vectors for each image. Yielding a dataset size of (10, 512) instead of (10, 160, 160, 3). Each image has a size of (160, 160, 3).

The next step of preparing the data for the LSTM is to pass the video frames sequence to a rolling window function which will create sub sequences of the video frames to enable the LSTM to learn more on the data. As discussed in literature review; passing a (10, 512) to the rolling window function with width 5 will return (10-5+1) sub sequences. With an image frame sequence of (10, 512) the resulting tensor shape is going to be (6, 5, 512) where 6 is the number of sequences, 5 is the width of the sequence (i.e. the number of frames), and 512 is the number of feature vectors extracted.

6.2 Extracting feature vectors from frames

For extracting the CNN features of faces; a pretrained model is going to be utilized. InceptionResnetv1 has been trained on VGGFace2 which is a database of over 3.3 million faces. This model is going to be used as a featuer extractor yielding 512 feature vectors for each video frame. These feature vectors extracted are going to be then transformed into subsequences through rolling window for the TimeDistributed layer and LSTM as input.

6.3 Architecture

The model is created so it can receive an input of (supersteps , window size , 512) where supersteps is the number of subsequences created by the rolling window function, window size is the length of the video sequence and 512 is the number of features vectors extracted by InceptionresnetV1.

The TimeDistributed layers is a wrapper that allows to apply a layer to every temporal slice of an input, which is the subsequences created by the rollimg window function. Inside that wrapper is an LSTM layer which is applied to every part of this sequence. This allows LSTM to learn temporal data sequence applied to every sub-sequence.

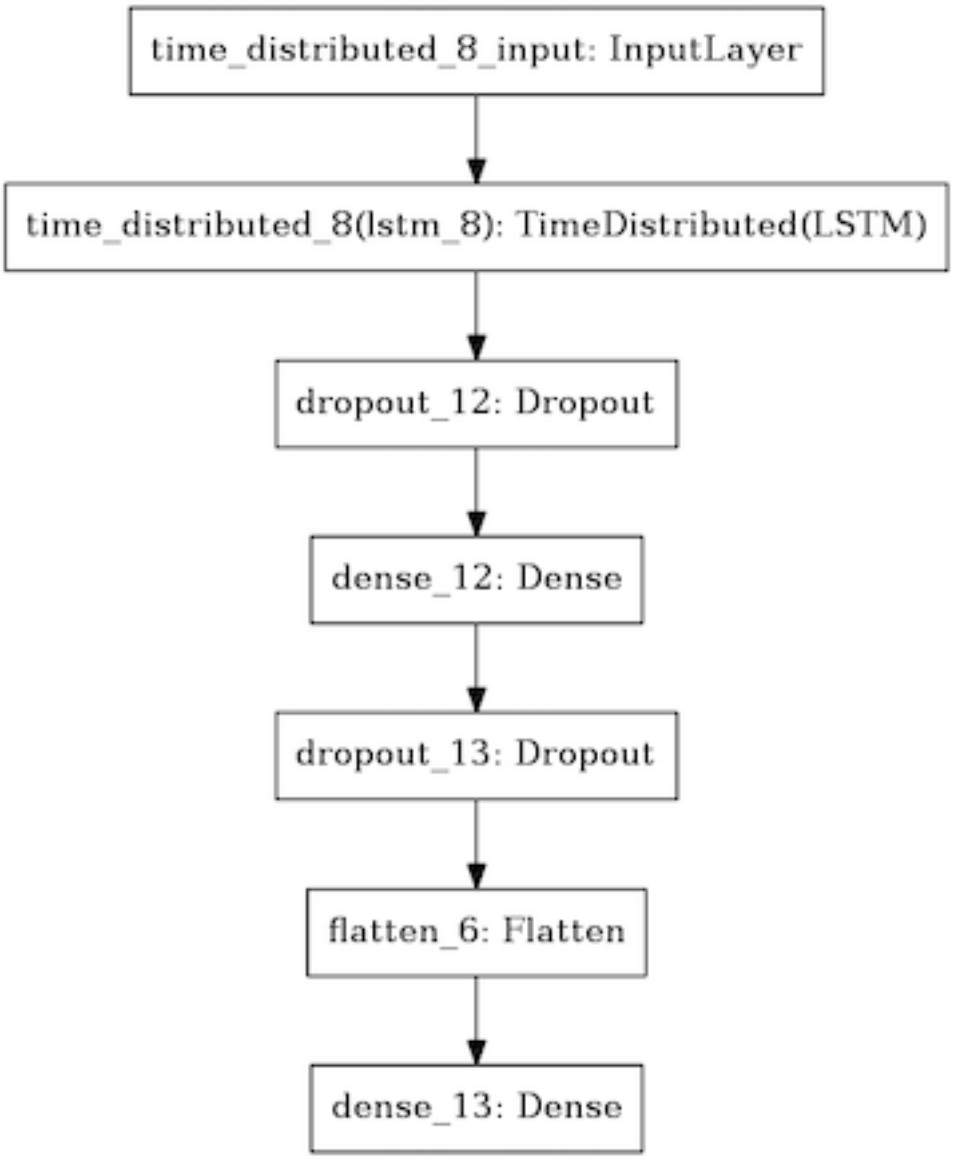


Figure 6.1: LSTM Model architecture

6.4 Comparison of Learning Rates

Looking at figure 6.2, it can be spotted that the training loss is decreasing gradually and consistently for all learning rates tested. When looked at in conjunction with figure 6.3, the validation loss is not decreasing at all. In some learning rates the validation loss is increasing throughout the epochs which is a sign of over fitting; as the training loss is decreasing but the validation loss is not.

Multiple architectures encompassing the LSTM layer where tested with various number of neurons for each layer, but all of them were demonstrating similar behaviour of decreasing

Figure 6.2: Training Loss

training loss but steady or increasing validation loss. As LSTMs have a memory cell built in that is capable of storing temporal information, indeed the model is memorizing the data rather than memorizing the sequence behind deepfakes or real videos.

Observing the weight updates from the TimeDistributed layer wrapper 6.4 and the LSTM layer 6.5, it can be noted that the model is not learning over the epochs, the distribution for both remains the same throughout the training session only making very small changes which confirms the hypothesis that the model is not learning, but rather memorized the data from the first epochs and is making no improvement at all.

Therefore, in conclusion the following LSTM set up is considered useless for this problem due to its lack of ability of generalizing well to the unseen validation set. In order to harness the benefit of LSTM another dataset up might be required or the gathering of a larger dataset that is more varied, to force the model to learn the temporal sequence behind deepfakes rather than memorizing single values for the training dataset.

Additionally, in this experiment, unlike previous ones with XceptionNet and MesoNet

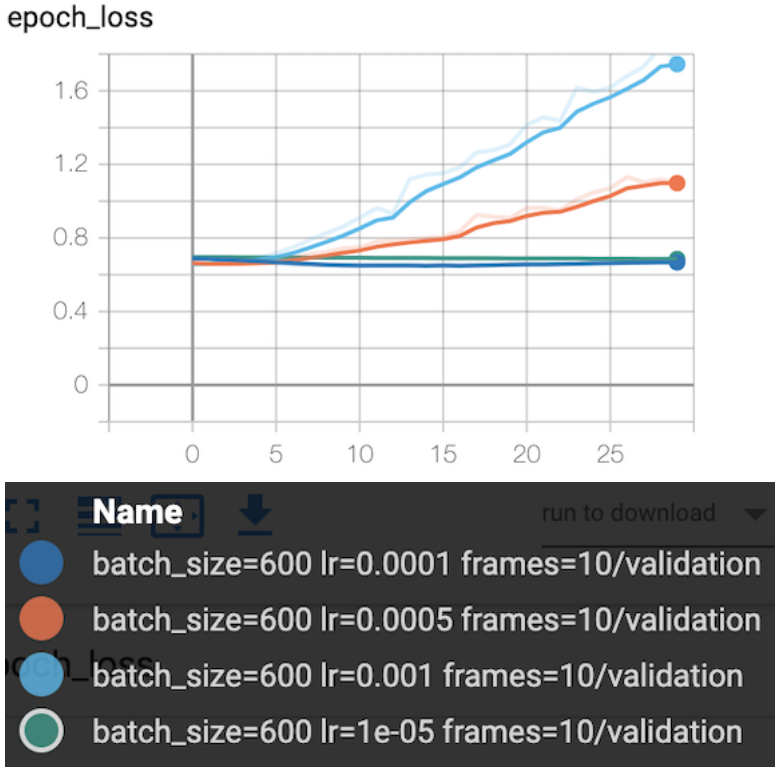


Figure 6.3: Validation loss

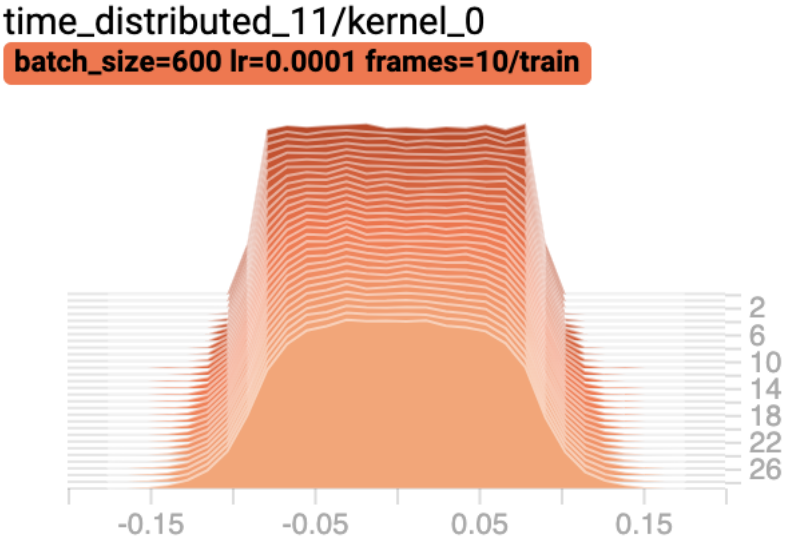


Figure 6.4: Time Distributed Kernel

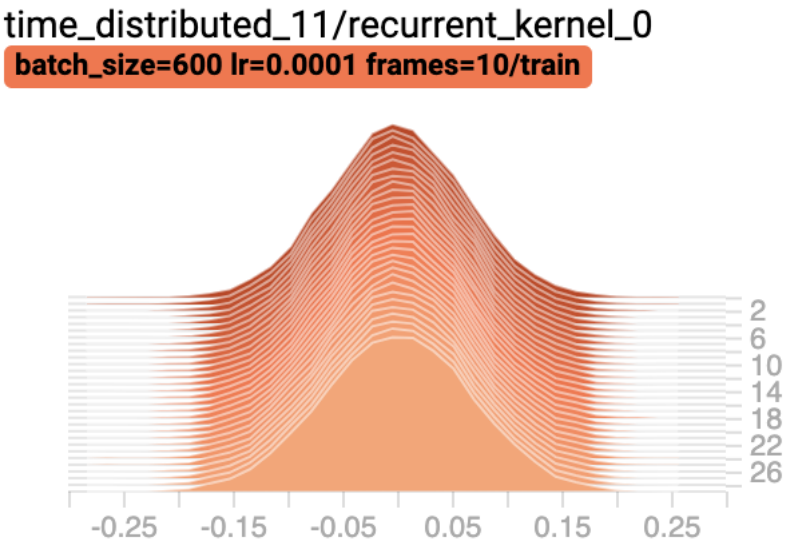


Figure 6.5: LSTM Kernel

the dataset was underbalanced due to the unavailability of an additional dataset to over-balance it with, thus this reducuction of data could have become harmful for the model leading to overfitting of this model.

Chapter 7

Results

In evaluating the created models performance several metrics were used. The first being the loss metric which is measured by the log loss equation as mentioned before, which can also be regarded as binary cross entropy. In previous sections this was conducted on validation data which is also used to validate the models performance prior to final presentation of results on the test set which is completely unseen by the model.

In this section detailed analysis of the models results is going to be carried out using various methods of evaluating the final performance of the model. Based on previous sections; the results discussed here are going to be based on the optimal hyper parameters found previously such as the learning rate and batch size for each model.

The Receiver Operating Characteristic (ROC) curve is a plot of the True Positive Rate against the False Positive Rate of the model. True positives are the correctly identified fake videos in the dataset, while the False positives are the misclassfied real videos regarded by the model as fake. Larger area under the curve (AUC), the better the model's performance is with minimal FPR and maximum TPR.

7.1 MesoNet

In experimenting with MesoNet two main variations were considered. A pretrained variant which backropagation of loss was allowed throughout the network while training and the second being an untrained network starting completely from scratch. MesoNet was

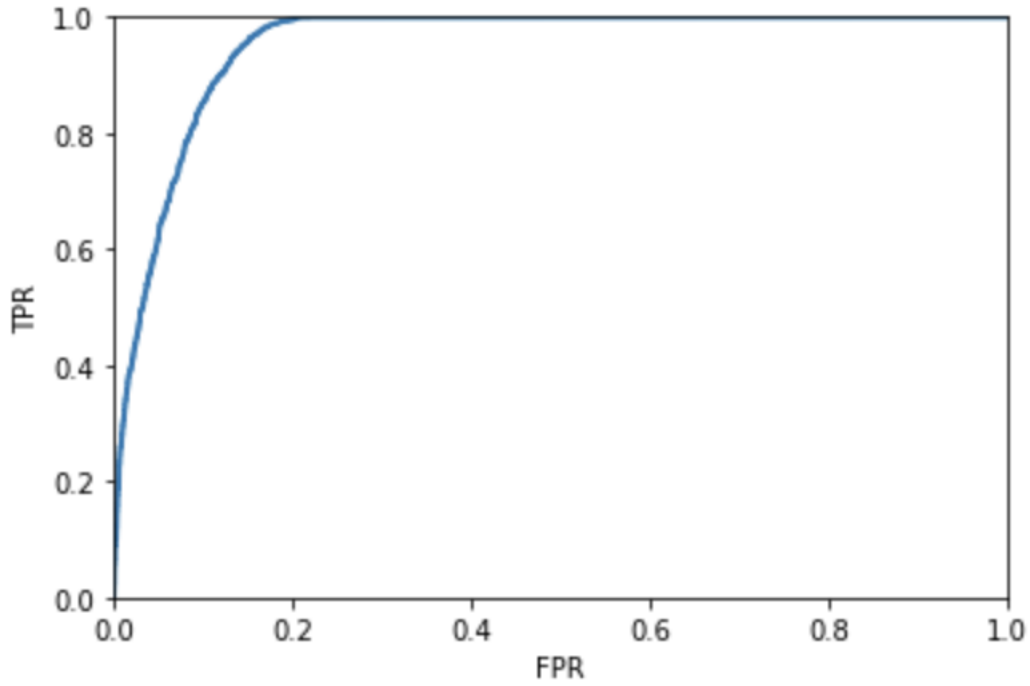


Figure 7.1: ROC Curve Face2Face

trained on two datasets for deepfakes Face2face and Deepfake dataset.

In figure 7.1 the ROC curve plot for Face2Face version of MesoInception-4 shows that the ideal point is almost at 0.2 FPR while it reaches 1 for the True positive rate. Beyond this point the model cannot improve more as it has reached the maximum TPR while the FPR keeps on increasing. This is the optimal point for the model.

The ROC curve for Deepfake testing is almost the same as Face2Face training, with an optimal point at around 0.2 FPR where the TPR reaches 1.

MesoNet	Face2face	Deepfake	Untrained
AUC	0.93	0.93	0.95

Table 7.1: AUC MesoInception-4

Comparing the two pretrained models, it is clear that the model pretrained on the Deepfake dataset is outperforming the other model trained on Face2Face dataset in overall accuracy and precision. Both models have a recall value of 1 which indicates that there are no false negatives produced and models are able to identify deep fakes accurately.

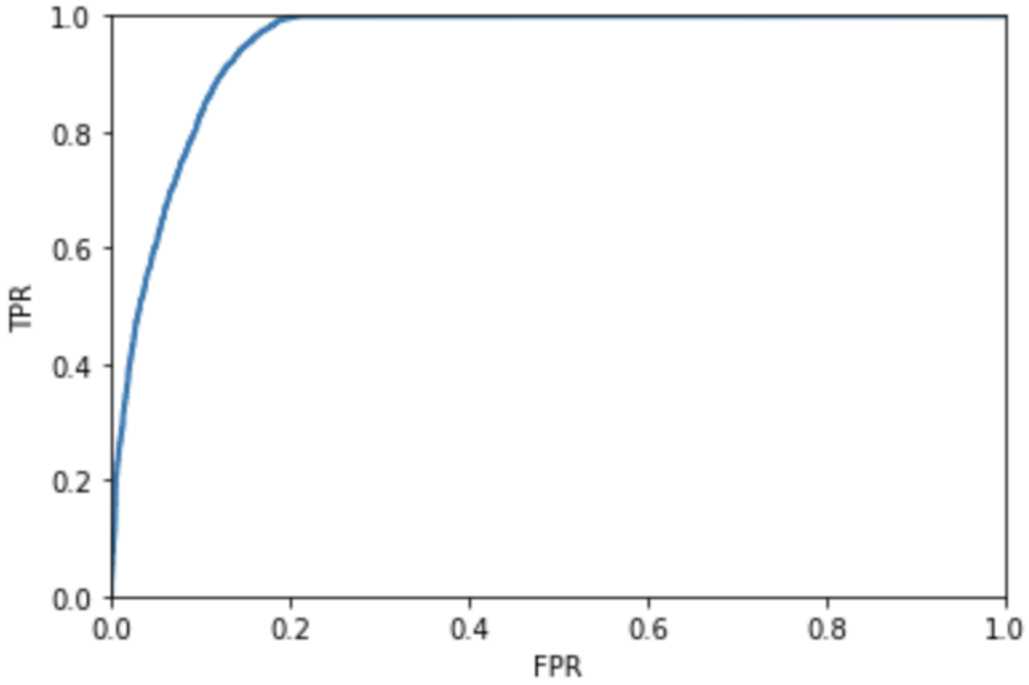


Figure 7.2: ROC Curve Deepfake

MesoNet	Face2face	Deepfake	Untrained
Accuracy	0.86	0.87	0.89
Recall	1	1	1
Precision	0.78	0.79	0.80

Table 7.2: Results MesoInception-4

These results indicate that both models are successfully utilizing the knowledge from previous training into this dataset being considered and resulting in very high scores which proves the models excellent capability at identifying deepfakes.

However, the untrained model achieves almost similar values for the auc score, but slightly outperforms the two pretrained models in accuracy and precision. With also a prefect recall score of 1, which indicates that this model architecture is suitable for training under conditions where all deepfakes need to be identified.

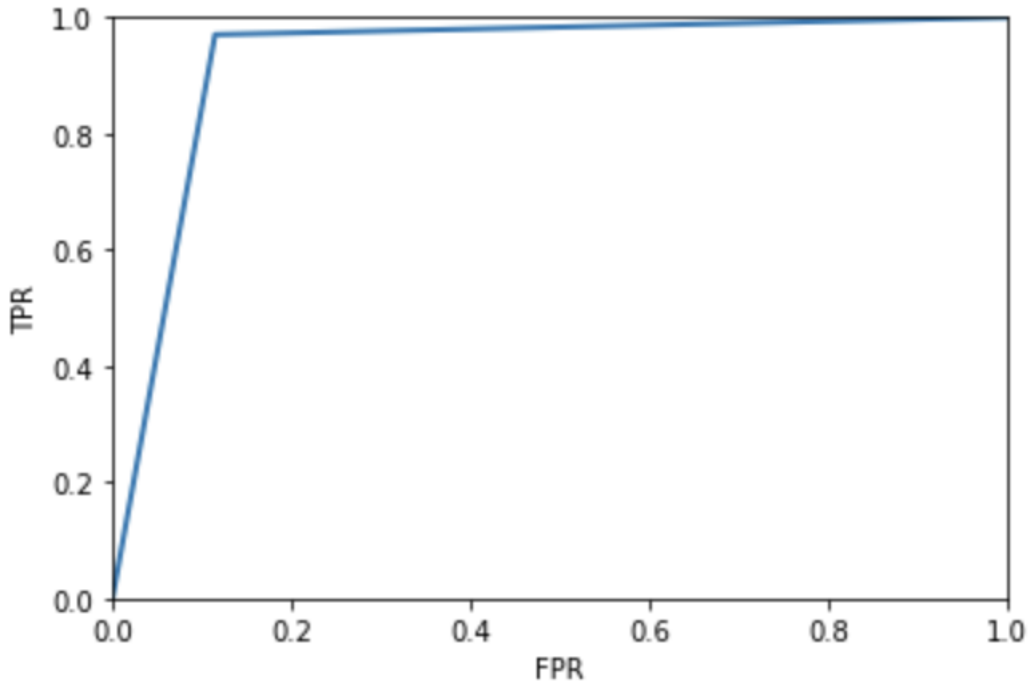


Figure 7.3: ROC Curve XceptionNet Pretrained

7.2 XceptionNet

During the training process of XceptionNet, it has become clear that finetuning a pretrained version is the more viable option as the untrained version produced extensive fluctuations in the validation loss which indicated that the model is potentially over fitting due to the limited size of the data.

Figure 7.3 shows the ROC curve for the pretrained XceptionNet with an optimal point at almost 0.10 FPR and the TPR reaches a value of 0.98. The area under the curve is 0.927 which confirms the capability of the model in identifying deep fakes correctly.

The model achieves superior results with an accuracy of 0.92 on detecting deepfakes 7.3, recall of 0.97 which indicates that there is a small percentage of results falling under the False negatives, and precision of 0.89.

Scores	XceptionNet
Accuracy	0.92
Recall	0.97
Precision	0.89

Table 7.3: Results LSTM

7.3 LSTM

As discussed in the previous section, the LSTM's validation loss indicated that the model might be memorizing the data rather than actually learning the feature progression within frames. Therefore it is expected that the model's results are not as good as the other models created before.

Looking at table 7.4, it is noted that the LSTM results are much lower than expected. The accuracy of the model is at 0.68 which is quite low compared to the other SOTA models in detecting deepfakes. The recall is at 0.83 which is quite a good measure of how much deepfakes are returned in a given dataset but when put into perspective with the accuracy and precision this indicates that the model is classifying many data instances as deepfakes without much understanding of the sequence characteristics.

Scores	LSTM
Accuracy	0.68
Recall	0.83
Precision	0.74

Table 7.4: Results LSTM

In figure 7.4 it can be concluded that the model is not performing very well on the test data. The optimal point appears to be at 0.6 FPR and 0.8 TPR. The FPR is too high and cannot be afforded to be incorporated in a deepfake detection system. To further confirm the area under the curve is around 0.59 which is too low.

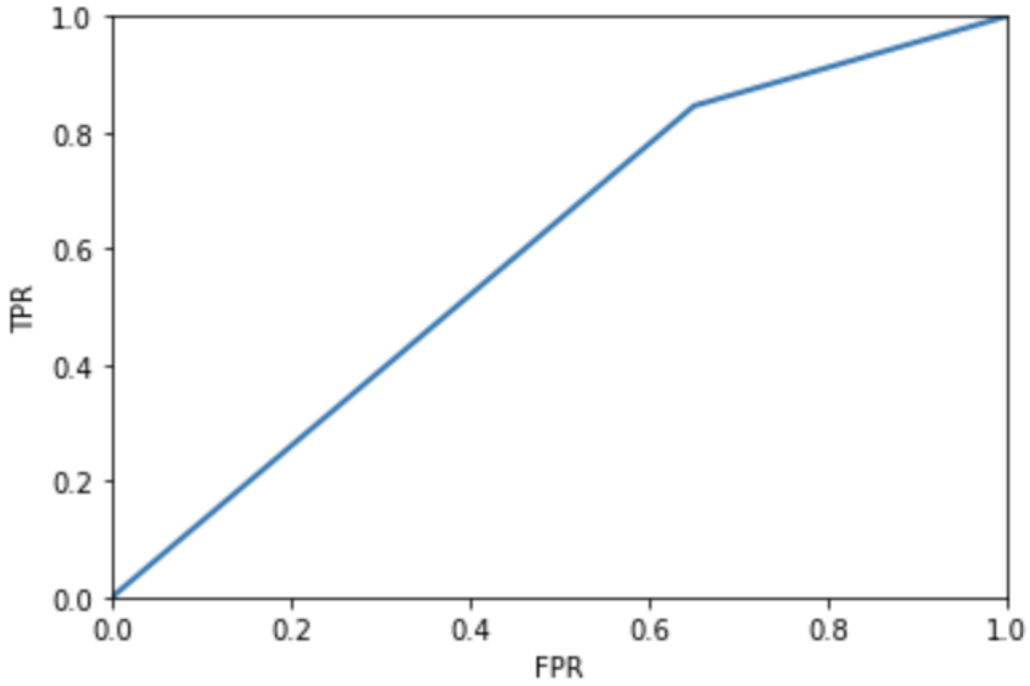


Figure 7.4: ROC Curve LSTM

7.4 Comparison

When comparing the previous implementations, we can conclude that the LSTM model is not performing very well when compared to the CNN models as standalone.

However, MesoNet and XceptionNet are achieving outstanding results in accuracy and recall. MesoNet implementations were significantly good at the recall rate which was 1, meaning that they were able to identify all deepfakes present in the test data. XceptionNet recall was slightly lower at 0.97 which is still very good.

The AUC score for both CNN models was very high at over 0.9 meaning that they can perform very good with a low FPR rate. XceptionNet slightly outperformed the two MesoNet models by scoring an accuracy of 0.92 when compared to the 0.89 for MesoNet implementations.

These scores confirm the outstanding capability of CNN models in detecting deepfakes based on a single frame rather than several frames for LSTM.

Chapter 8

Evaluation

This section will discuss the overall objectives and aims of the project and discuss the overall success of the project, including any shortcomings as well as future directions of research.

Step 1

Acquire Data which to base Deepfake detection on:

Data was acquired successfully from many third party sources. The main database which was used in this project was the DeepFake Detection Challenge database, this database was used as the primary dataset when evaluating the SOTA models and their performance in this project.

When examining the DFDC dataset it has become clear that it would be unpractical to train the models on this data set only as it was imbalanced. The number of fake videos was extremely greater than the number of real videos in this dataset. With a ratio of 1:0.28, it might have caused difficulties for the model to train on and would have caused a bias in the model towards deepfakes. Generally Machine Learning techniques find it difficult to train on imbalanced datasets therefore the possibility of including another dataset by which to balance the dataset was considered.

The second data set acquired was the Flickr-Faces-HQ Dataset FFHQ which contains real faces captured in high quality. This dataset was used to address the imbalanced original DFDC dataset problem, by adding a proprtion of real faces to make the dataset have equal samples for each class, by adding the missing number of real faces in DFDC

dataset using the FFHQ dataset making the ratio 1:1.

Step 2

Explore the latest state-of-the-art techniques used to Deepfakes in videos:

This section was conducted in the literature review which included thorough review of the top performing models in this domain as well the methodology behind each. After finishing this section it has become clear that there were 2 Deep Learning techniques with promising architecture and results that are capable of identifying deepfakes successfully.

The first was MesoNet which has performed very well on similar datasets for deepfake detection. The second model was XceptionNet which have also had very high accuracies among detecting deepfakes in a given dataset. These 2 model were replicated later on in order to measure their capability of detecting deepfakes on our dataset.

Step 3

Face Detection, Cropping and Alignment:

In this section various face detecting libraries were considered, as this is one of the crucial aspects when tackling this problem as the model can only get good as much as the dataset provided to is. Face detetcion libraries were tested for speed due to the size of the database being 490GB. More importantly, they were tested for recall, the number of faces retrieve in a given video dataset.

MTCNN library have proven to yield the best recall rates with its ability to detect faces in numerous cases were illumination and background were very dark, and failing to detect faces in extremely rare cases. With a slight modification to the original MTCNN algorithm based on exploiting face coordinates in previous frames to make the next prediction it achieves superior speed as well compared to other algorithms with almost 55 frames per second.

Step 4

Manipulate the Data to contain all key elements in modeling human signals:

Irrelevant data from the dataset was disregarded such as audio accompanying the .mp4 videos in this dataset due to the nature of the models being developed. Also where necessary data sets were combined as to balance the dataset for the CNN models that

are basing the predictions on a single frame.

Step 5

Create model which improves upon the latest SOTA Deepfake detection:

Two of the SOTA models were replicated that were previously tested among other deepfake datasets where replicated and tested on the balanced version of the dataset.

MesoNet was the first model to be replicated and have its performance tested in this project, which yielded quite outsanding results notably with a recall rate of 1, meaning that it was able to obtain all deepfake videos present in a dataset with no false negatives. A variant of MesoNet MesoInception-4 was used due to its superior performance to the other MesoNet implementations.

The Second SOTA model that was replicated in this project was XceptionNet which have also demonstrated outstanding performance in detetcing deepfakes with accuracy over 0.90 and recall at 0.97.

As an attempt to come up with a unique implementation to improve upon the latest SOTA an LSTM CNN model was implemented in this project. The idea behind it is as to enable the model to detect sequences in fake videos rather than exploit the shortcomings of the generation techniques. However, due to the limited sample size this LSTM model did not perform very well and was rather memorizing the data instead of understanding the underlying sequence behind it. This might have been due to the limited size of the dataset as in tacklig this problem with the LSTM model under balancing technique was used instead of overbalancing. This was due to the limited nature of similar dataset with videos.

Chapter 9

Conclusion

In this project it has been confirmed that Deep Learning techniques can successfully be employed to detect alterations in deepfake videos. Deep Learning CNN have demonstrated that in order to produce reliable results the data must be preprocessed accordingly. Working with the given dataset it was essential to detect, crop and align faces so the data is ready for the CNN models to work on.

Preprocessing the data also included extensive data augmentations such as flipping, contrast and resolution reduction, in order to model the real world data the model faces in real life situation, and also to make the model more robust to changes in input images.

Two of the three developed models have produced outstanding results with very high recall rate, being able to retrieve almost every single deepfake video, those were MesoNet and XceptionNet. The LSTM model combined with a CNN feature extractor have proven to need more data in order to avoid the model from over fitting and memorizing the data as predictions were based on a series of consective frames, unlike the two other CNN standalone models which based prediction on single frame.

MesoNet and XceptionNet have proven to be the best models when it comes to deepfake detection, pretrained versions of both proved to yield the best results as they were trained on another dataset but with similar characteristics. Finetuning of the models was also performed which enabled the model to generalize well on the DFDC data set being considered in this project.

Chapter 10

Statement of Ethics

Due to the nature of the project including real data comprised of videos and audio for actors, this project was carried out with respect to ethical concerns where various areas were taken into concern such as Legal, Social, Ethical aspects of the project to ensure its compliance with effective policy codes.

10.1 General Ethical Principles

10.1.1 Contribute to society and to human well-being

This project tackles an important threat to human privacy which is deepfakes. The availability of deepfake generation tools have increased dramatically over the last few years allowing anyone with access to a computer to create a deepfake video. This potentially has harmed many people, whether being victims of revenge attacks or celebrities or even politicians during a campaign.

The general public needs to be able to distinguish between fake videos and real ones as to preserve the sense of integrity and honesty. In this project a deepfake detection tool have been developed which benefits the advancement of deepfake detetcion techniques thus helping society to maintain privacy.

10.1.2 Avoid Harm

The system developed enables users to detect deepfake videos, but the accuracy of the system should be considered. As the system is prone to false positives and false negatives, which may affect the people by providing inaccurate information. Therefore such misguidance is considered as the system is still under development, which incurrs no harm to any individual, but on the contrary it helps the society in detetcing deepfakes.

10.1.3 Confidentiality

The main dataset obtained which is the DFDC dataset, which is publicly available subject to downloading it from kaggle. Therefore this is not an issue consideration in this project.

10.1.4 Respect Privacy

Unlike other deepfake datasets collected which is comprised of videos gathered from YouTube without knowledge from participants, this project utilities the DFDC dataset which have been shot by actors participating in creating the dataset with full knowledge with the motives for this dataset collection, which is to support the development of more robust deepfake detection techniques.

Privacy is enforced through the anonymity of the actors in the videos, as well as with their knowledge, they had the decision to shoot the video wherever they wanted, so not exploiting some of their home/ outdoor privacy. Additionally, after the completion of the project and release of the application the data is going to be removed and deleted permanently from the hardware devices stored on, which comprises of the remote working Google VM Machine.

10.1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts

In this project the main aim was to provide a deepfake detection tool that is unlike any produced before. Through exploiting intra-frame inconsistencies and training a model on learning the sequence progression in real vs fake videos, this way the models will

become robust against future developed generation techniques, due to the model's ability to generalise to sequence detetction rather than inter-frame inconsistencies. However, this method is still in its early stages, therefore the ensemble of three systems was taken instead, one being the LSTM sequence detetcion model.

10.1.6 Be honest and trustworthy

Honesty is crucial aspect when it comes to deepfake detection systems due to the conflict this can create between people and the spread of false information. Therefore, in the system careful consideration have been put in indicating the probability of the video being fake, and also not to over exaggerate the performance of the system and inform the actual performance of the system that it is still prone to false positives and false negatives.

10.2 BCS Code of Conduct

10.2.1 Public Interest

The development of deepfake detection techniques serves in the intest of any individual who is affected by false information based on deepfake videos. Deepfake videos spread online could be very fast and also the spread of misinformation whether regarding politics or other. Therefore this project aims to serve in the public interest by assisting in the development of deepfake detection techniques; having a due regard to the public's privacy and security.

10.2.2 Professional Competence and Integrity

All work done in this project was within my professional scope of knowledge, and where there was not enough personal knowledge to complete parts of the project these were not implemented, not claiming any level of competence that is not available. In this project there were several areas that required further research to improve personal knowledge, whenever this was necessary this have been carried out to a professional level of knowledge.

Appendix A

Appendices

SAGE

Response ID	Completion date
514292-514283-59548558	7 May 2020, 14:41 (BST)

1	**Applicant Name**	Ahmed Moahmed
1.a	**University of Surrey email address**	am02150@surrey.ac.uk
1.b	**Level of research**	Undergraduate
1.b.i	**Please enter your University of Surrey supervisor's name. (If you have more than one supervisor, enter the details of the supervisor who will check this submission).**	H Lilian Tang
1.b.ii	**Please enter your supervisor's University of Surrey email address. (if you have more than one supervisor, enter the details of the supervisor who will check this submission)**	h.tang@surrey.ac.uk
1.c	**School or Department**	Computer Science

2	**Project title**	Deepfake Detection

3	**For Undergraduate and Masters students, will your student research project be conducted according to a faculty standard study protocol? Your module lead or supervisor can advise if you are unsure.**	NO

4	**Are you making an amendment to a project with a current University of Surrey/NHS REC/other favourable ethical opinion in place?**	NO

5	**Does your research involve any animals, animal data or animal derived tissue, including cell lines?**	NO

6	**This question is deliberately left blank.**	Please click here to continue

7	**Does your project involve* human participants, their data and/or any human tissue?**	NO

8	**Does your funder, collaborator or other stakeholder require a mandatory ethics review (e.g. Institutional Review Board (IRB) review) to take place at the University of Surrey?**	NO

9	**Does your project process personal data1? Processing covers any activity performed with personal data, whether digitally or using other formats, and includes contacting, collecting, recording, organising, viewing, structuring, storing, adapting, transferring, altering, retrieving, consulting, marketing, using, disclosing, transmitting, communicating, disseminating, making available, aligning, analysing, combining, restricting, erasing, archiving, destroying.**	NO

10	**Does your project require the processing of special category2 data?**	NO

11	**If you are an undergraduate or Masters student, are you ONLY using name and contact details for recruitment purposes, and no other personal data is being collected as listed in questions 9 and 10 above?**	NO

12	**Does your project involve any type of human tissue? This includes Human Tissue Authority (HTA) relevant, or irrelevant tissue (e.g. non-cellular such as plasma or serum), any genetic material, samples that have been previously collected, samples being collected directly from the donor or obtained from another researcher, organisation or commercial source.**	NO

13	**Does your research involve exposure of participants to any hazardous materials e.g. chemicals, pathogens, biological agents or does it involve any activities or locations that may pose a risk of harm to the researcher or participant?**	NO

14	**Will you be accessing any organisations, facilities or areas that may require prior permission? This includes organisations such as schools (Headteacher authorisation), care homes (manager permission), military facilities etc. If you are unsure, please contact RIGO.**	NO

15	**Will you be working with any collaborators or third parties to deliver any aspect of the research project?**	NO

16	**Will you be travelling to non-UK countries for any of your research activities?**	NO

17	**Will any research activities be conducted outside of the UK?**	NO

18	**Does your research involve lone working?**	NO

19	**Certain types of research require ethics approval from a nationally recognised research ethics committee (REC) which operates to standards set out by the Department of Health's Governance Arrangements for Research Ethics Committees. Recognised research ethics committees (REC) include NHS RECs and the MoDREC. Does your research involve any of the following? (select all that apply)**	None of the above

20	**Have you selected any of the options between A-O from question 19?**	NO

21	**Does your project require ethics review from another institution?**	NO

28	**Declarations**	• *I confirm that I have read the University's Code on Good Research Practice and ethics policy and all relevant professional and regulatory guidelines applicable to my research and that I will conduct my research in accordance with these. • I confirm that I have provided accurate and complete information regarding my research project • I understand that a false declaration or providing misleading information will be considered potential research misconduct resulting in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies • I understand that if my answers to this form have indicated that I must submit an ethics and governance application, that I will NOT commence my research until a Favourable Ethical Opinion is issued and governance checks are cleared. If I do so, this will be considered research misconduct and result in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies. • I understand that if any of my responses

		to the governance questions have requested additional documents, that these will be provided with my ethics and governance application if my project is to proceed. • I understand that if I have selected any options from Qu 22-27 I MUST submit an ethics and governance application (EGA) for review in order to proceed with this research project UNLESS I am an undergraduate or Masters student, in which case I have completed Qu 29 below.

29	**If I am conducting research as a student:**	I confirm that I have discussed my responses to the questions on this form with my supervisor to ensure they are correct.

Bibliography

617, M. S. (2018), 'Deep learning', https://www.math.purdue.edu/~nwinovic/deep_learning_optimization.html.

Afchar, D., Nozick, V., Yamagishi, J. & Echizen, I. (2018), 'Mesonet: a compact facial video forgery detection network', *CoRR* **abs/1809.00888**.
URL: http://arxiv.org/abs/1809.00888

Amidi, A. & Amidi, S. (2017), 'Recurrent neural networks cheat-sheet', https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

Amor, E. (2020), 'Convolutional neural networks', *Medium* .

Cao, Q., Shen, L., Xie, W., Parkhi, O. M. & Zisserman, A. (2018), Vggface2: A dataset for recognising faces across pose and age, Technical report, Visual Geometry Group, Department of Engineering Science, University of Oxford.

Chollet, F. (2016), 'Xception: Deep learning with depthwise separable convolutions', *CoRR* **abs/1610.02357**.
URL: http://arxiv.org/abs/1610.02357

Data Policyl (n.d.), https://deepfakedetectionchallenge.ai/data-policy.

Dolhansky, B., Howes, R., Pflaum, B., Baram, N. & Ferrer, C. C. (2019), The deepfake detection challenge (dfdc) preview dataset, Technical report, AI Red Team and Facebook AI.

Faceswap: Deepfakes software for all (n.d.), https://github.com/deepfakes/faceswap.

Gron, A. (2017), *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edn, O'Reilly Media, Inc.

Guera, D. & Delp, E. J. (2018), Deepfake video detection using recurrent neural networks, *in* 'EEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)', Vol. 15, IEEE, pp. 1–6.

Hao, J. & Wang, B. (2014), 'Parameter sensitivity analysis on deformation of composite soil-nailed wall using artificial neural networks and orthogonal experiment', *Mathematical Problems in Engineering* **2014**, 1–8.

Nguyen, H. H., Yamagishi, J. & Echizen, I. (2019), Capsule-forensics: Using capsule networks to detect forged images and videos, *in* 'IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 2307–2311.

Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I. & Natarajan, P. (2019), Recurrent convolutional strategies for face manipulation detection in videos, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops', IEEE, pp. 80–87.

Saha, S. (2018), 'Convolutional neural networks', *Towards Data Science* .

Sample, I. (2020), 'What are deepfakes – and how can you spot them?', *The Guardian* .

Schroff, F., Kalenichenko, D. & Philbin, J. (2015), Facenet: A unified embedding for face recognition and clustering, Technical report, Google Inc.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2014), 'Going deeper with convolutions', *CoRR* **abs/1409.4842**.
URL: `http://arxiv.org/abs/1409.4842`

Thi, T., M., C., Tien, D., Nguyen, D. T. & Nahavandi, S. (2019), Deep learning for deepfakes creation and detection, Technical report, School of Information Technology, Deakin University, Victoria, Australia and School of Engineering, Deakin University,

Victoria, Australia and Institute for Intelligent Systems Research and Innovation, Deakin University, Australia.

Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C. & Nießner, M. (2016), 'Face2face: Real-time face capture and reenactment of rgb videos'.
URL: http://www.graphics.stanford.edu/~niessner/papers/2016/1facetoface/thies2016face.pdf

Tsang, S.-H. (2018), 'Review: Xception — with depthwise separable convolution, better than inception-v3 (image classification)', https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-ima

Wu, Y. & Radewagen, R. (2017), '7 techniques to handle imbalanced data', https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html.

Yang, X., Li, Y. & Lyu, S. (2019), Exposing deep fakes using inconsistent head poses, *in* 'IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 8261–8265.

Zhang, K., Zhang, Z., Li, Z. & Qiao, Y. (2016), 'Joint face detection and alignment using multi-task cascaded convolutional networks', *CoRR* **abs/1604.02878**.
URL: http://arxiv.org/abs/1604.02878