National College of Ireland

DeepFake Detection using Deep Neural Networks

MSc Research Project
Data Analytics

Ambuj Agnihotri
Student ID: x19220073

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ambuj Agnihotri
Student ID:	x19220073
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	16/08/2021
Project Title:	DeepFake Detection using Deep Neural Networks
Word Count:	7085
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ambuj Agnihotri
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

DeepFake Detection using Deep Neural Networks

Ambuj Agnihotri
x19220073

Abstract

Deepfakes are fake images or videos created with artificial algorithms, image processing, and face swap. Deepfakes are computer-generated fake images or videos in which images are merged to create new images or videos representing events, comments, or activities that never occurred. The end product can be quite stunning. A "Generative Adversarial Network," or GAN, is an artificial intelligence technology that can be used to create fake images. GAN, a multifunction technique used to create Deep Fakes, is established to map faces using "landmark" points. Such features include the edges of a person's eyelids and mouth, nostrils, and the curve of the jawline. This research project's main objective is to employ neural networks to distinguish between fraudulent and authentic images. For deepfake image detection, a publically available Flickr Faces High Quality (FFHQ) dataset is utilized. Deepfake image detection employs a variety of pre-trained Convolutional Neural Network (CNN) architectures (EfficientNetB4, InceptionV3, and InceptionResNetV2) for feature extraction and Long Short-Term Memory (LSTM) for classification. The Classification Report including Accucary, F-1 score, and other features are used to analyze the results. To execute code with essential python libraries such as Keras, Matplotlib, sklearn, and others, Google Colab and Jupiter Notebook are utilized. EfficientNetB4-LSTM, Inceptionv3-LSTM, and InceptionResNetv2-LSTM models achieved test accuracy of 98%, 96%, and 97%, respectively.

Keywords: Deepfake, Generative Adversarial Network (GAN), Deep Learning,Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM)

1 Introduction

1.1 Motivation and Project Background

The emerging technologies have evolved so much that they have enormous power to make the nightmare come true. Privacy and security have become questionable in our nations. Facts state that 14968 fake videos were posted to social media or generally online platforms with almost 96 percent of the swaps were of the celebrity's and used for pornographic acts. The number of fake videos created is multiplying rapidly as well. It's not only the images but video recordings are transformed into something objectionable and either blackmailed or posted online. It's not only the privacy of an individual that is at stake, but sometimes it can be the entire nation suffering or the lives of innocent people that are taken away for such criminous acts. When fake videos of national leaders are created and spread across, it's the security and reputation of the country that is jeopardized but just revenge or an act of entertainment for the ones that create them. Hence classifying fake and real images is an important topic of discussion and work has

been done to make the prediction as accurate as possible in all ways including performance and prediction quality with deep learning techniques. They are so powerful and reliable that they can transform the videos thereby helping filmmakers and others reducing the manual work. But at the same time, they can be used to create deepfake videos and images. Hence as a positive sign of work, efforts are made to research how accurately the deep learning methods are used to detect fake/real images.

The word 'Deepfake' is a combination of words 'deep learning' and 'fake' which means manipulated images or any other digital representations that have fabricated piece of it which is unreal. Deepfake is a form of artificial intelligence. Anybody who has access to computers is eligible to produce deepfakes. A deepfake is a counterfeit created by deeply studying the images or the videos of the target person and then imitating the same behavior by transforming parts of it or entirely. Barrett once said that the preliminary fake videos/images are made more believable by the Generative Adversarial Networks (GAN) process. The faults in the forgery are addressed through this process several times. As a fact, AI tools are used to paste the body of a person with verbal statements of somebody else create these. Even though simple tools can be used to create fakes, deepfakes are forged in a way that it is hard to detect with bare eyes and a normal innocent man believes it as true.

The dataset used in this research, fake images were created using a Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN)(Karras et al. (2019)) using real images from Flickr-Faces-HQ dataset at 1024×1024 pixels. It offers intuitive, scale-specific monitoring of the synthesis and resulting to an automatically learnt, unsupervised distinction of properties (e.g., posture and identities when trained on facial images) and random variance in the resulting images (e.g., freckles, hair). To understand the concept of Deepfake, an example of real and fake images (generated through styleGAN) in our dataset can be seen below. The dataset contains some fake images which can be identified as fake by human eyes and some are not. Deep Learning models with proper training can help in identifying such fake and real images.

Figure 1: Real And Fake Image from the dataset

1.2 Research Question and Objectives

RQ: "How accurately can the hybrid deep learning model detect deepfake images generated by StyleGAN?"

Sub RQ: "How much training time does different CNN architecture take in a CNN-LSTM hybrid model?"

The remaining paper is arranged in the following format. The next section delves into prior research on detecting deepfakes using hybrid models. The technique as well as procedure, consisting of a thorough analysis of the dataset collection, pre-processing phases, and data mining algorithms used, is presented in sections 3 and 4 of the paper. The description of the project's implementation, evaluation methods, and outcomes is presented in sections 5 and 6 of the paper. The last section 7 concludes the paper by providing an overall summary as well as recommendations for further research.

2 Related Work

This section's topic is linked to the previous image classification literature review related to deepfake detection. This section's topic is linked to the previous image classification literature review related to deepfake detection. It is divided into three sections: 1) Deepfake Detection using different CNN Architectures 2) CNN-LSTM Hybrid Architecture in Image Classification 2) Deepfake Detection using Hybrid CNN-LSTM Architecture 4) Comparison and summary.

2.1 Deepfake Detection using different CNN Architectures

DeepFakes were introduced as part of innovations in machine learning techniques for image processing and manipulation. DeepFakes employ deep learning methods to develop fake images that can be difficult to tell apart from actual images. Khalil and Maged (2021) researched the creation and detection of deepfakes. One autoencoder understands the attributes of the input image, while the other understands the characteristics of an output image, and then the two encoders share their characteristics. For creating the deepfake image, the output image was redesigned using the input image's decoder, resulting in an output image with characteristics from the input image. The use of image enhancement techniques to increase the quality of deepfakes generated was investigated. CNN architecture named MesoNet was used to detect deepfakes. It has 4 Convolutional and pooling layers subsequently a dense network with a hidden layer. MesoNet was able to categorize deepfake images with an accuracy rate of over 80%.

Li et al. (2021) had taken the frame and clip levels using the temporal and spatial rotation angles to identify the intrinsic uniformity of face landmarks for more robust deepfake detection to tackle against frame scaling and compression. The experimental platform was a Windows 10 machine having a 3.0GHz Intel(R) Core(TM) i7-9700 processor and 32GB of RAM and the Deepfake-TIMIT dataset and FaceForensics++ datasets were used to test the model.. To obtain the spatial features, researchers initially picked a tiny proportion of steady landmarks and utilized those to create facial vectors. Second, they built the temporal rotation feature using the rotation angles of facial vectors from neighboring frames. Finally, they computed the statistics of both features independently to create the feature vector for Support Vector Machine (SVM) classification.

ul ain et al. (2021) researched real and fake facial image recognition using Error Level Analysis (ELA) with Deep learning models. The initial stage for detecting fabricated faces was image normalization for actual and fake picture assessment. Normalized pictures were then preprocessed with ELA and fed into various deep learning models that had already been trained. To measure the efficiency of models, they finetuned them for classification of two classes: fake and real. Confusion Matrix was used to evaluate models. Different CNN Architectures used were VGG-16, ResNet, ResNet50, and Inception-v3. VGG model had the best training accuracy of 91.97% with a lower number of epochs, which is significantly superior to all other approaches.

Bonettini et al. (2021) targeted current facial alteration techniques for solving the challenge of face altering identification in video sequences by ensembling different trained CNN models. Starting with a basic CNN model (EfficientNetB4), various models were created using two alternative concepts: a. Siamese training and b. Attention Layering. An attention method that provided an idea of the model while also boosting the network's training capabilities. A triplet siamese training approach that pulled feature information to improve classification results. The proposed model was implemented on 2 different datasets i.e FF++ and DFDC. AUC and Log Loss were obtained for different combinations of models for evaluation and it was observed that EfficientNetB4 with Attention Mechanism and Siamese training helped to improve the accuracy of the model. The embedding of temporal information might potentially be added to this study to improve the model's performance and accuracy.

Zhang et al. (2021) proposed a method to detect fake videos/images even when they are compressed or are of poor quality especially in social media using self-supervised decoupling networks (SSDN). The networks are trained using authenticity and compression features and a self-supervised method is used for feature decoupling. The results obtained show that the proposed method beats the state-of-the-art methods for deepfake detection for compression. For a Low-Quality setting (LQ) SSDN achieves an accuracy of 91.80% which is better than F3-net (state-of-the-art methods) by 1.4%. This method is outstanding and is considered for the review as the compression and authenticity factor is considered which other methods don't with comparatively better accuracy. With the method proposed in this paper, higher accuracy is expected with mediocre quality levels.

The DeepFake Detection challenge (DFDC) created by Facebook motivated Pokroy and Egorov (2021) to compare the performance of Efficient networks in detecting fake videos. The highest performing model was utilized and the DFDC data for training them. The method begins by framing the videos, applying a random combination of augmentation methods, use efficient networks to get a feature matrix, classify the frame using a binary classifier, and finally averaging the predictions to detect fake videos. This paper concludes that the correlation between the model performance and their size is minute and the best performances were showed by B4 and B5 models. This paper uses high-resolution frames and it is proved that with a lower number of parameters, accuracy decreases which is considered for the research conducted.

Tjon et al. (2021) described a new architecture called 'Eff-YNet' to determine the differences between the altered and the real videos. The combination of an EfficientNet encoder and a U-net is used to classify and segment deepfake videos. The spatiotemporal inconsistencies are detected using ResNet 3D networks. The results from the ensemble techniques of Eff-Y networks and Resnet 3D are quoted to be better than the baseline methods. The DFDC data is used for this and consists of real-fake pairs to create segmentation masks which might not always be available. Also, this paper does not provide

any evaluation methods other than Area Under Curve (AUC) to compare the results and evaluate them. These limitations are considered for this research and attempts have been made to overcome them.

AMTENnet, an effective fraudulent face detector, was demonstrated by combining an adaptive manipulation traces extraction network (AMTEN) with CNN to reveal face picture tampering in complicated scenarios by Guo et al. (2021). AMTEN, a face picture forensics preprocessing tool, was created to learn modification traces. AMTEN used an adaptive convolution layer to anticipate manipulating footprints in the picture, which are then utilized in future layers by changing parameters during the back-propagation pass to enhance manipulation traces. They replicate realistic face picture forensics as realistically as possible by performing some post-processing techniques such as lossy compression, scaling and, blurring to input images. AMTENnet surpasses state-of-the-art methods in spotting fraudulent facial images created by several methods, with an accuracy rate of 98.52%.

GAN-generated face recognition based on an enhanced Xception was investigated by Chen et al. (2021). The following are advancements to Xception Model: (1) To prevent overfitting, four residual blocks were excluded; (2) Inception piece with dilated convolution was used to substitute the familiar convolution layer in the Xception's pre-processing model to procure multi-scale features; (3) Feature pyramid network was used to achieve multi-level attributes for final judgment. The pluralistic image finalization technique used in the FFHQ dataset was used to create a locally GAN-based generated face (LGGF) dataset for checking model performance. All of the experiments were run in Keras on a single 11 GB GeForce GTX 1080 Ti, i7-6900 K CPU, and 64 GB RAM machine. In terms of training, they employed the Adam optimization method, with an initial learning phase rate of 0.001 and a total of 128 epochs. Three indicators were used to assess the effectiveness of the model: Precision, Accuracy, and Recall. The proposed model outperforms existing models built for entire generated faces, particularly for faces with tiny generated parts, according to experimental data.

Most of the deepfake detection models process the videos frame by frame and very few consider the temporal inconsistencies. Trinh et al. (2021) proposes a Dynamic Prototype Network (DP Net) that utilizes dynamic representations to determine temporal artifacts. On top of this the temporal logic specifications is added to check the model's compliance towards the behaviors.The prediction is based on comparing the input depictions to the prototypes to see how similar they are. The network then tries to find the real/fake by looking at the temporal behavior activated by dynamic prototypes. The summation of the similarity scores between prototypes are picked for detection. This model provides a trustworthy method for people themselves to detect deepfakes.This paper proves better than the state-of-art methods by 1-4% of AUC which is the base evaluation method used.This paper doesn't provide a strong evaluation method to compete the state-of-art methods which has been considered in this research.

Two methods for detecting deepfakes for classification tasks to automatically recognize deepfake videos by Pan et al. (2020) were Xception and MobileNet. Obtaining frames from the video, identifying faces from those individual frames, and storing face regions as photos were the three phases of the pre-processing module. Both MobileNet and Xception use depth-wise and pointwise convolutional layers and are centered on CNN Architecture. The only change is that MobileNets includes fewer features to improve the model's efficiency. Over the matching dataset being used to train each model, it performed well in classification and also examining the effects of various loss functions

and optimizers on the outcomes.

2.2 CNN-LSTM Hybrid Architecture in Image Classification

Ozkaya et al. (2021) analyzed Ground Penetrating Radar B scan (GPR B Scan) images of varying quality for soil type's performance using a hybrid residual CNN and Bi-LSTM model. This model's fundamental architecture was made up of blocks. Convolution, Rectifier Linear Unit (ReLU), and batch normalization were part of block structures. Each block comprises 20 filters, each of which is 3*3 pixels in size. To boost efficiency, residual links among the block and pooling layers were established. The challenge of gradient vanishing was attempted to be avoided via these residual connections. Three Bi-LSTM units were linked together in a parallel. There were 150 hidden units in each of these units. Finally, for the classifying task, Softmax regression was used. All of these tests were run on Matlab Software with an Intel Core i7–7700 HQ CPU running at 2.8 GHz and 16 GB RAM. Accuracy, precision, recall, and F1 score were used as performance evaluation criteria. GPR data were divided randomly split into the train (75%) and test (25%). The efficiency of the classification model was 97.31%, which was superior to any state-of-the-art method.

An approach to COVID-19 diagnosis using Lung Ultrasound (LUS) with an integrated autoencoder-based hybrid CNN-LSTM model was used by Dastider et al. (2021). The use of CNN and RNN to evaluate both spatial and temporal aspects of the LUS images was proposed. To create a robust, noise-free classifier model, CNN has used auto - encoder system and separate convolutional sections combined with a customized DenseNet-201 network. To confirm the efficacy of the proposed system, a five-fold cross-validation approach was used. To increase the classification accuracy of the traditional DenseNet network, it was suggested to add LSTM layers on top of CNN. For both CNN and LSTM, the Adam optimizer was employed for every stage, with a learning rate of 1e-3, batch size 64, and 120 epochs. To stop the overfitting of the model, numerous dropout stages were utilized after the convolutional layers. Evaluation metrics like accuracy, precision, etc. proved that the CNN-LSTM hybrid model was better than traditional DenseNet.

A study on the classification of gastrointestinal tract diseases using Residual LSTM layered CNN was conducted by Oztürk and Ozkaya (2021) to give early detection for clinical treatment, which can assist in early detection of polyps. The proposed framework used three famous CNN architectures: AlexNet, GoogLeNet, and ResNet50. These CNN architectures were utilized to strengthen the extracting features section using a transfer learning strategy. During training, the batch size was set to 16, the learning rate was set to 0 001, and the dropout parameter was set to 0.35. Every LSTM block in the structure had two LSTM components in it, and these two-layer LSTM units act in tandem. The first LSTM unit's input is adaptively changed based on the CNN architecture and positioning in the LSTM blocks. Every LSTM block had one dropout layer with a value of 0.35 to avoid overfitting. The classification problems can benefit from feature importance at all stages. CNN pooling layers comprising low, mid, and high-level image characteristics can be send to LSTM blocks which are better for classification tasks based on past learnings.

Gill and Khehra (2021) researched the classification of fruits images based on an integrated and hybrid approach of CNN, RNN, and LSTM networks. For the creation of discriminative features and sequential identifiers, CNN and RNN were used. The LSTM was explained by using a memory cell to store learning at each classification interval. The key change was that the last layer of the classic CNN architecture was replaced

by 2 layers (Fine and Coarse), and monitoring inputs were provided to the coarse and fine classes independently. RNN displays an ephemeral form with effective norms of varied lengths but incapable to dissipate, expanding gradient concerns and gradients lack to be inseminated between layers of RNN. LSTM was used to address RNN concerns by including a memory cell (C) at each level. Accuracy, F-measure, sensitivity, and specificity were among the evaluation metrics employed. The technique outperformed classic CNN, RNN, and LSTM models in terms of performance.

2.3 Deepfake Detection using Hybrid CNN-LSTM Architecture

Stanciu and Ionescu (2021) looked into whether the video's temporal features may be used to boost the effectiveness of already present deepfake detection algorithms. Instead of using the completely aligned face as input to the model and only picking selected facial areas, they tested whether some facial sections offer greater knowledge about the legitimacy of the video. (1) Face identification, (2) landmark mapping, (3) face separation, and aligning at an image size of 299 * 299, and (4) face parts separation including eyes, nose, and mouth utilizing landmark points were all part of the preprocessing. Xception network, which helped in extracting a feature vector for each image in the sequence, the LSTM block, which was a two-layer, 256-layer LSTM that produced a temporal descriptor for the sequence and was used for classification. For the CelebDF dataset, the model offered a 13.46 percent gain in AUC, while for the FF++ dataset, it yielded a virtually flawless 99.95 percent AUC.

Sanghvi et al. (2021) created a method that allows consumers to discern between machine-generated multimedia components and actual media. In preprocessing, Faces from each of the frames were collected using the pre-trained MTCNN for face recognition, and all of the extracted faces were put together to generate a face-only clip. All frames in which the face was not recognized were discarded, and a new frame for face recognition was supplied. CNN and LSTMs had worked quite well for image or video classification because CNN gathers spatial features whereas LSTMs acquire temporal features. The authors combined LSTMs with the several CNN architectures and metric learning techniques in this paper. The Triplet + LSTM model, which was based on triplet loss, was found to be the best over 60 frames and thus was considered as the best model for constructing a deepfake detection system.

Suratkar et al. (2020) used CNN architectures based on Transfer-Learning to improve the generalizability of Deepfake Detection. A technique that utilizes a CNN to collect characteristics from each frame of a clip to train a binary classifier that can effectively distinguish between real and fake videos. The approach is tested on a large number of deepfake films pulled from diverse datasets. CNN Models such as Xception, Inception v3, MobileNet, ResNet50, etc. were used as base models. Extra layers and hyperparameters such as dropout, decay rate, etc. were added on top of that for training and tested on the rescaled dataset. Except for the ResNets model every other model performed well and Inception v3 being outstanding among all of them.

Hashmi et al. (2020) used the Conv-LSTM Hybrid Framework for deepfake detection using microscopic-typo evaluation of video frames in an exploratory study. CNN feature extractors used a transfer learning strategy that starts with a pre-trained ResNet model and then passes it on to the LSTM network, which was chosen over GRU due to memory constraints. On the massive data, the training on Nvidia 1080x TI GPU lasted over seven days (Kaggle DFDC training set). To avoid memory allocation issues, the batch

size was set to one. In terms of computational complexity, the model was a heavyweight. According to evaluation results, the best traits for classification were eyes, eyebrows, head movement, and mouth movement.

Güera and Delp (2018) utilizes a CNN architecture for obtaining frame-level attributes. These attributes were then used to train RNN that learns to classify whether or not a video has been tampered with. In image processing tasks, CNNs have had a lot of success, while LSTMs are commonly utilized for long sequence processing difficulties. Random 70/15/15 split was done in preprocessing phases to generate three distinct sets, which were used for training, validation, and test, respectively. Every frame was resized to 299 299 pixels. The main problem they solved was the creation of a model that can recursively analyze a sequence in a meaningful way. They employed a 2048-wide LSTM unit with a 0.5 risk of dropout to solve this challenge, which was capable of doing exactly what they require. Training, Test, and Validation accuracies were similar with different frames per second for the Conv-LSTM model.

2.4 Summary

According to the research papers reviewed, the Deepfake idea poses a serious safety risk, and new tactics to recognize and counter it are urgently needed. In terms of identifying deepfakes, there are fewer techniques that can reliably assess whether an image or video is real or fake. To distinguish deepfake, approaches like a CNN-LSTM hybrid model appear to be efficient. We will focus on developing multiple CNN models with LSTM classifiers to recognize fake or real photos using a dataset containing real and fraudulent images (using StyleGAN) from the FFHQ dataset.

3 Methodology

This project is in the domain of data science and machine learning, Knowledge Discovery in Databases (KDD) being one of the most often utilized approaches in this domain will be considered. The reason for choosing KDD over the Cross-Industry Standard Data Mining method(CRISP-DM) is that CRISP-DM usually requires the deployment of the project because of business applications, whereas this is not an obligatory phase to finish in KDD, which aligns with our research. Below image explains the KDD architecture.

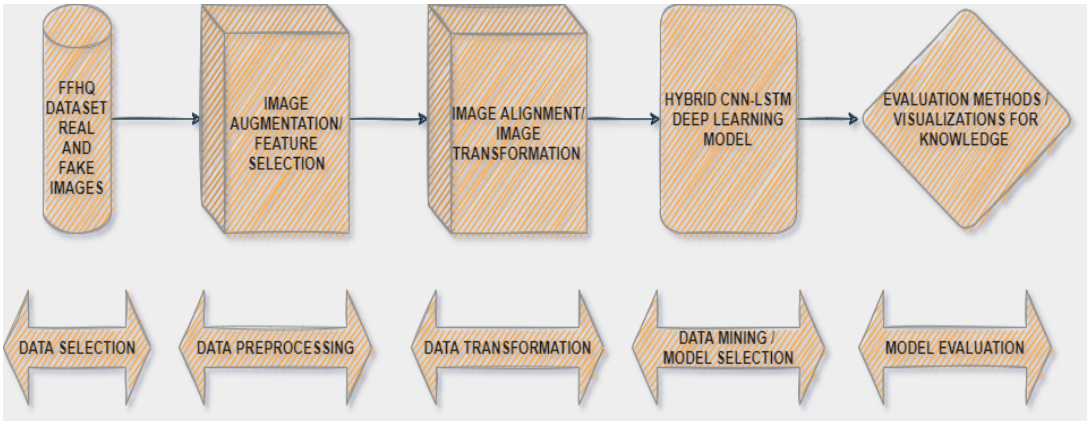


Figure 2: KDD Structure for Deepfake Detection

3.1 Data Selection

Many studies used datasets of images/videos that were either too small or of low quality. When choosing a dataset, we ensured that it is of reasonable quality and has an adequate amount of images. Deepfake detection is a rising issue for all High-tech firms, open-source datasets for this task are readily available on Kaggle, GoogleAI, FacebookAI, and other sites, alleviating any ethical concerns about data. For this project, we employed two separate publicly available datasets for real and fake images from Github. Flickr-Faces-HQ (FFHQ)[1] is a high-resolution human face image dataset extracted from the Flickr website. The dataset comprises 70,000 high-quality PNG images with a dimension of 1024 * 1024 pixels, with a wide range of ethnicity, age, and image background. It also covers eyeglasses, sunglasses, caps, and other accessories well. Karras et al. (2019) used FFHQ dataset for their Style-Based Generator Architecture to create fake images and is publicly available on GitHub[2].

3.2 Data Preprocessing

Data collected from websites are subject to a range of errors, noise, raw format, and other variables. Insufficient data preprocessing can lead to undesirable outputs due to false content or unnecessary attribute values in data sets. All images were collected from Flickr and were automatically aligned and resized using dlib, adopting all of the website's prejudices. We only selected photos with permissive licenses. To create a new dataset of 2000 images for deep learning models, 1000 real and 1000 fake images with a dimension of 1024 * 1024 were picked from the FFHQ dataset and the StyleGAN dataset, respectively.

3.3 Data Transformation

To make predicting issue patterns easier and remove the temporal framework, data transformation is required. This is the most crucial and innovative stage of the study. It's a crucial step because it removes the distracting background from the images, reducing their size. 'train valid test split' was used to split the dataset into train, validation, and test sets at a ratio of 80%, 10%, and 10%, respectively. For better training validation and testing, the train, validation, and test sets contain an equal number of real and fake images.

3.4 Model Selection

After these steps, the data becomes ready for the deep learning models for training and validation. This study presents a CNN architecture (EfficientNetB4, Inception-v3, and Inception-ResNet-v2) for extracting features from each image and forwarding the sequence to LSTM (RNN) for classification of fake and real images.

EfficientNetB4: Unlike traditional methods, which scale network parameters like breadth, depth, and resolution randomly, Tan and Le (2019) method scales each parameter evenly with a defined set of scaling factors. Authors constructed a family of networks named EfficientNets, which outperform state-of-the-art performance with up to 10x greater efficiency (smaller and faster). On a range of scales, EfficientNet gives a variety of networks (B0 to B7) that provide a reasonable mixture of efficiency and accuracy.

[1] `https://github.com/NVlabs/ffhq-dataset`

[2] `https://github.com/NVlabs/stylegan`

In general, EfficientNet models outperform previous CNNs in terms of performance and accuracy, while minimizing parameter quantity and FLOPS by an order of magnitude.[3] Performance of each model in EfficientNet was compared by Pokroy and Egorov (2021) in detecting deepfake videos. Results suggested that EfficientNet B4 and B5 were the best models for deepfake detection tasks. Due to the higher computational complexity of B6 and B7, there was a reduction of accuracy in these models. According to the dataset and computation capacity available, EfficientNetB4 was the perfect choice of CNN model with LSTM for this research.

InceptionV3: Inception-v3 is a CNN structure that improves on previous versions of Inception by streamlining the framework and deploying additional inception modules than Inception-v2. It uses Factorized 7 x 7 convolutions, Label Smoothing, and an extra auxiliary classifier to transport label information lower down the network, among many other improvements (also uses batch normalization for layers in the sidehead). Suratkar et al. (2020) used different architecture such as Inception v3, MobileNet, ResNet50, etc. in task of detecting deepfakes using transfer learning. The Inception v3 model, according to their findings, outperformed all other models in terms of accuracy and predictions while requiring less training time and computational complexity. The Adam optimizer, with a learning rate of 0.0001, was found to be the greatest fit for such a system in terms of achieving the optimal solution in a short amount of time with great precision.

Inception-ResNet-V2: The ResNet results motivated the idea of a hybrid Inception-Resnet module. There are two versions of Inception-ResNet: Inception-ResNet-v1 and Inception-ResNet-v2. Inception-ResNetv2 improved performance significantly faster and had higher accuracy than Inception-v4. The Inception-ResNet-v2 model integrates the Inception structure with a residual network (ResNet) interface to form the Inception-ResNet-v2 framework. The reason for using the residual link is that it avoids deterioration during the deeper framework and gives precise feature details like color, texture, and location. Inception-ResNet-v2 has three main frameworks: convolutional, activation, and pooling. The Convolutional Layer aids in the extraction of features from images. The activation layer is located between or at the ends of the system and assists in evaluating whether or not such a neuron will fire. The activation function for this research will be ReLu, which means that only positive values will be fired.

LSTM: An image sequence with the possibility of being a fake or not will be provided by CNN architecture. The main challenge is creating a framework that can handle a sequence iteratively and realistically. The LSTM is frequently used to examine images sequentially. Obtaining images delivering these to a network consumes a lot of memory and, due to the large datasets, it requires a lot of computing power as well. LSTM maintains patterns or features as well as face characteristics and does not preserve images on a physical server, as discussed by Hashmi et al. (2020). Before the next sequence, the LSTM memory cell captures, retains, and adapts the current sequence pattern. The current is eliminated from the cell when the next sequence arrives. These patterns are assigned values, which are subsequently saved in the LSTM memory modules. This loop repeats throughout the image sequence.

3.5 Model Evaluation

Deepfake Detection is a classifying task that attempts to determine if an image is real or fake. When making classification predictions, four types of outcomes can occur. 1)

[3] `https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html`

When the model anticipates that an image will be classified as fake and it is fake (True Positive(TP)). 2) When the model anticipates that an image as real and it is real(True Negative(TN)). 3) A prediction is given by the model that an image is fake and it is real (False Positive(FP)). 4) Model predicted an image as real but it is fake (False Negative(FN)). TP and TN should be emphasized in our system. Model's accuracy is defined as the percentage of right predictions it makes. The total number of accurate predictions (TP+TN) divided by the total number of predictions (TP+TN+FP+FN) is the idea of model accuracy. The number of positives (TP) accurately categorized out of all positives (TP+FN) is the Recall (True positive rate or Sensitivity). The recall is not a great indicator by itself. There's also Precision, which is a metric. Precision is expressed as the ratio of accurately predicted positives (TP) out of all predicted positives (TP+FP).

4 Design Specification

The figure 3 depicts the research's design specification. Following the gathering of data, the dataset is partitioned into train, validation, and test sets. For better model learning, image augmentation is used on the training set. All three sets have their images aligned and resized. To detect deepfakes, pre-trained CNN models with weights as the imagenet are picked with an LSTM classifier. To show research findings and modeling implementations, Python libraries such as Matplotlib, sklearn.metrics and others are being used. To recognize deepfakes containing various CNN frameworks with RNN (LSTM), a Hybrid Deep Learning model is applied.

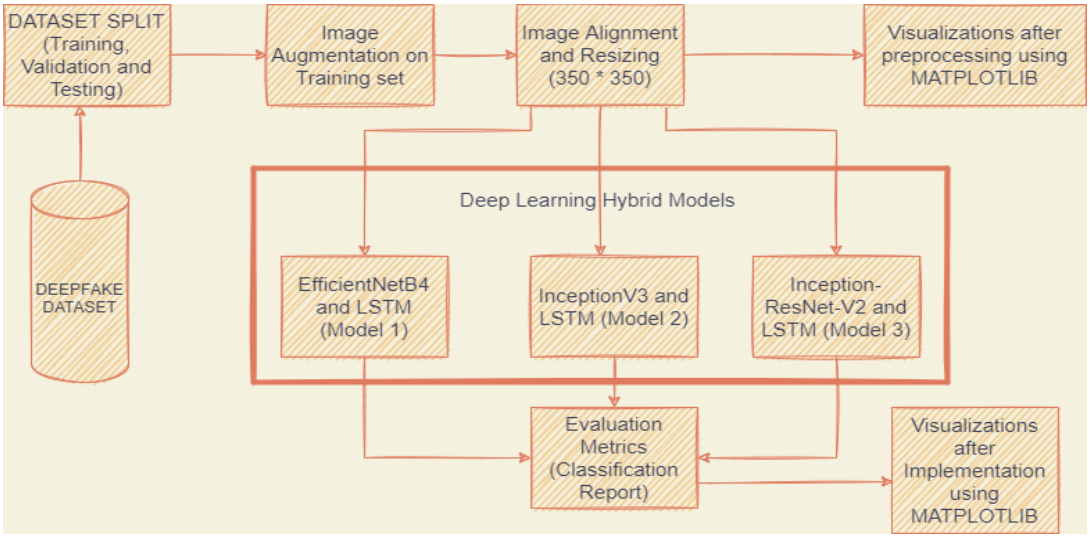


Figure 3: Design Specification

5 Implementation

5.1 Preliminary Data Exploration

The examination of the image dataset is the very first step in the project implementation. Identifying the file types in the dataset. Data analytics is supposed to be all about

analyzing data. dataset has image files of PNG format in folder named fake and real images. Dataset divided into train, validation, and test folders in proportions of 80%, 10%, and 10%, respectively, each having a labeled folder (real and fake) i.e., out of 2000 photographs, the train folder contains 1600 (800 real and 800 fake), whereas the validation and test folders each have 200 images (100 real and 100 fake images). The Google Colaboratory platform was utilized to run the model, which was aided by the GPU Tesla T4. The final dataset was uploaded to Google Drive and used with the drive mount feature during execution. To begin, all necessary libraries were imported, including Tensorflow, Keras, and Sklearn. The figure 4 shows how to use Python libraries to explore a dataset.

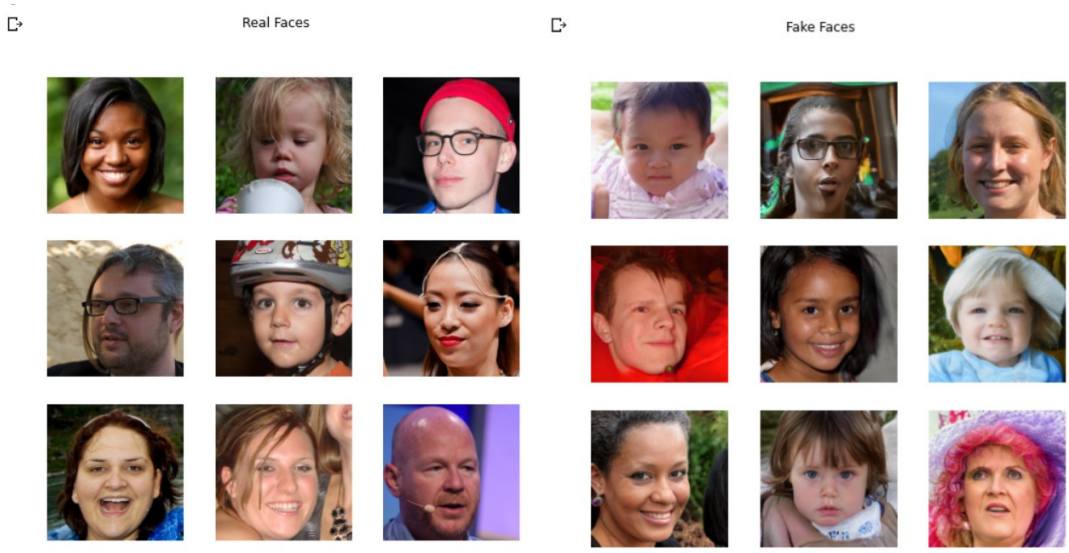


Figure 4: Data Exploration for real and fake images

5.2 Data Preprocessing

The amount of available data generally contributes to the success of deep neural network models. Data augmentation is a technique used to generate new training data from old data. This is performed by transforming images from the training samples into fresh and unique training images utilizing domain-specific techniques. Image data augmentation, which comprises changing pictures in the training set into modified duplicates that match to the same categorization as the original image, is a well-known approach of data augmentation. Transforms involve procedures such as shifts, zooms, and many other picture alteration approaches. Image data augmentation is often utilized on the training set only, not the validation or test sets. Data preprocessing, such as image resizing and pixel scaling, is distinct in that it must be done equally across all sets communicating with the model. Batch size was set to 16 and image size was chosen as 350 * 350 as we were facing GPU issues on Colaboratory in processing larger size than that. A batch size of 16 was chosen and an image size of 350 * 350 because we were having GPU troubles on Colaboratory when processing images larger than that.

Figure 5: Image Augmentation Parameters and Images after Resizing and Augmentation

5.3 Model Implementation and Evaluation

After Data Preprocessing steps, scaled and resized image data is ready for model training and vaildation steps. The LSTM is a form of RNN that can preserve long-term reliabilities. When employed in layered scenarios, LSTMs have proved to be able to supplement CNN's extracting features capabilities. LSTMs can memorize trends preferentially for a long time, while CNN's can extract the important elements from them. When utilized for image classification, the LSTM-CNN layered structure outperforms the standard CNN classifier. The suggested concept is based on Artificial Neural Networks such as Recurrent and Convolutional Neural Networks, making them resilient and applicable to a wide range of classification tasks. Below is description of such three hybrid models chosen for this research.

5.3.1 EfficientNetB4 (CNN) with LSTM (Model 1)

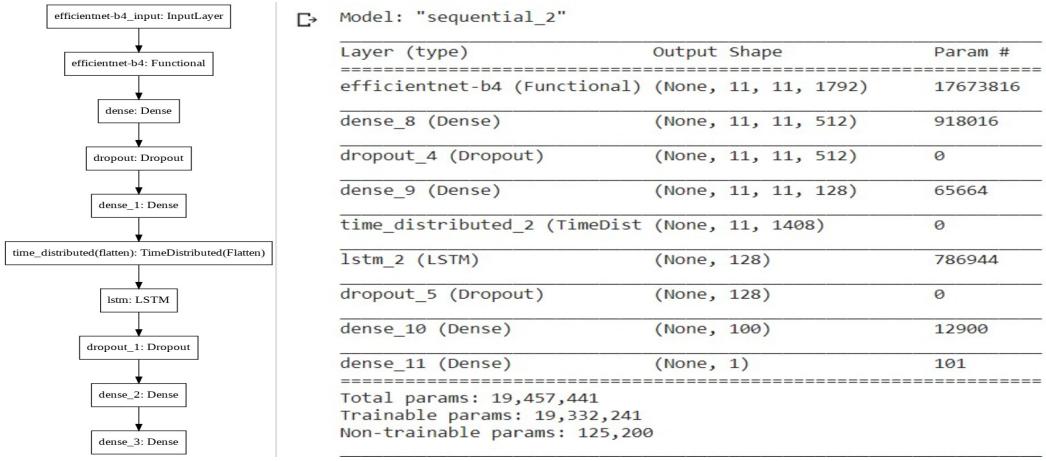


Figure 6: Model 1 Architecture

The data is delivered to the CNN model (EfficientNetB4) in the first step, which has a total of 17673816 parameters. The second layer is a dense layer of units 512 with the activation function as 'relu', and the third layer is a dropout layer of 0.5 to reduce the model overfitting scenario. We employed a dense layer of units 128 in the following layer to try to identify more hidden feature trends. The dimensions are then flattened over the next layer to prepare an input vector to the LSTM model with params 786944.

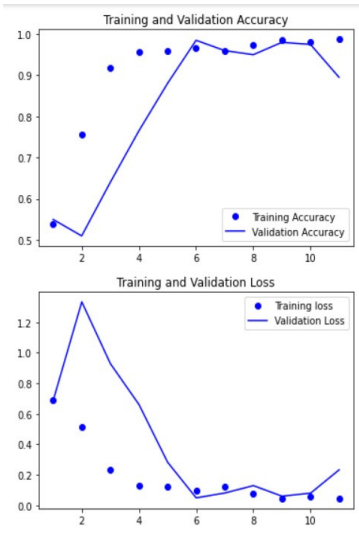
The LSTM outcome is being used in the next layers to integrate the learning from all the above layers and produce classification output. As the project's goal is to categorize real or fake images, the model compilation is done using the loss function of binary cross-entropy and Adam's optimizer. Accuracy is one of the metrics used to measure and refine model training. To ensure consistency of outcomes and training timeframe, all data sets are run using 20 epochs and a batch size of 16.

As indicated in the figure 7 below, the highest training and validation accuracy obtained in EfficientNetB4 and the LSTM hybrid model is 98.75% and 98.50%, respectively, from different epochs. The training and validation losses are respectively 0.0459 and 0.0486. Early stopping can be noticed at the ending of epochs 11, indicating that validation loss has not improved in the last 5 epochs. The optimal model for testing is saved having epochs value of 0.0486.

```
Epoch 11/20
100/100 [==============================] - 154s 2s/step - loss: 0.0459 - accuracy: 0.9875 - val_loss: 0.2328 - val_accuracy: 0.8950

Epoch 00011: val_loss did not improve from 0.04863
Epoch 00011: early stopping
Epoch 6/20
100/100 [==============================] - 153s 2s/step - loss: 0.0979 - accuracy: 0.9675 - val_loss: 0.0486 - val_accuracy: 0.9850
```

Figure 7: Training and Validation Accuracy (Model 1)

In the figure 8 below, training and validation accuracy, as well as both loss, are displayed for each epoch in a plot using the matplotlib library. Except for epochs 2, the model worked well because the training and validation values in both plots are closer to each other. The model 1 classification report, which shows that testing accuracy is 98% and F1 scores for fake and actual photos are 99% and 98%, respectively, is also presented below.




```
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(test_generator.classes, preds)
print('Confusion Matrix')
print(cm)
```

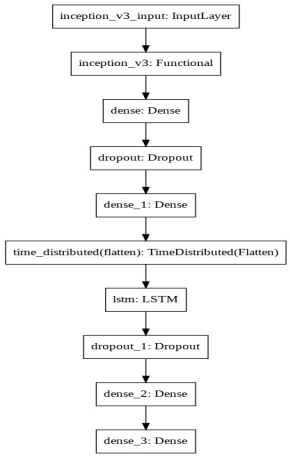
```
200/200 [==============================] - 178s 877ms/step
Confusion Matrix
[[99  1]
 [ 2 98]]
```

```
#### PRINT CLASSIFICATION REPORT
names = ['fake', 'real']
print('Classification Report')
print(classification_report(test_generator.classes, preds, target_names=names))
```

```
Classification Report
              precision    recall  f1-score   support

        fake       0.98      0.99      0.99       100
        real       0.99      0.98      0.98       100

    accuracy                           0.98       200
   macro avg       0.99      0.98      0.98       200
weighted avg       0.99      0.98      0.98       200
```

Figure 8: Plots and Classification Report (Model 1)

5.3.2 Inception V3 (CNN) with LSTM (Model 2)




```
Creating Directory: .\tmp_checkpoint
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 9, 9, 2048)        21802784
_________________________________________________________________
dense_29 (Dense)             (None, 9, 9, 512)         1049088
_________________________________________________________________
dropout_15 (Dropout)         (None, 9, 9, 512)         0
_________________________________________________________________
dense_30 (Dense)             (None, 9, 9, 128)         65664
_________________________________________________________________
time_distributed_7 (TimeDist (None, 9, 1152)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 128)               655872
_________________________________________________________________
dropout_16 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_31 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_32 (Dense)             (None, 1)                 101
=================================================================
Total params: 23,586,409
Trainable params: 23,551,977
Non-trainable params: 34,432
```

Figure 9: Model 2 Architecture

After Image augmentation, pre-processed data is passed as input to the Inception model, neural network employed pre-trained networks with initial weights derived from training on the Imagenet data, and then customized layers added on top of it. After then, the complete network can be trained without any layers being frozen. Except for the pre-trained CNN architecture with parameters 21802784, which is slightly greater than ExceptionNetB4, the model architecture above reveals that all of the layers are maintained identical to model 1. LSTM layer has 655872 parameters for model 2. With a learning rate of 0.0001, all other model compilation parameters are maintained the same. The model is run on a total of 20 epochs. The best model from the 20 epochs is saved for the final testing task on the test set with the help of the Model checkpoint feature. The patience parameter is set to 5, which means that if the model does not improve validation loss for 5 consecutive epochs, model training will be terminated with the help of the early stopping feature.

Each Epoch saves the best model for testing and also simultaneously analyzing outcomes with a 200-image validation set. The training and validation accuracy obtained in Inception v3 and the LSTM hybrid model is 98.37% and 94.50%, respectively, from different epochs, as shown in the figure below. The training and validation losses are 0.0484 and 0.1636, respectively.

```
Epoch 00011: val_loss did not improve from 0.16359
Epoch 12/20
100/100 [==============================] - 128s 1s/step - loss: 0.0484 - accuracy: 0.9837 - val_loss: 0.2343 - val_accuracy: 0.9000

Epoch 00012: val_loss did not improve from 0.16359
Epoch 7/20
100/100 [==============================] - 128s 1s/step - loss: 0.1253 - accuracy: 0.9581 - val_loss: 0.1636 - val_accuracy: 0.9450

Epoch 00007: val_loss improved from 0.21581 to 0.16359, saving model to .\tmp_checkpoint/best_model.h5
Epoch 8/20
100/100 [==============================] - 129s 1s/step - loss: 0.0908 - accuracy: 0.9694 - val_loss: 0.3110 - val_accuracy: 0.9050

Epoch 00008: val_loss did not improve from 0.16359
```

Figure 10: Training and Validation Accuracy (Model 2)

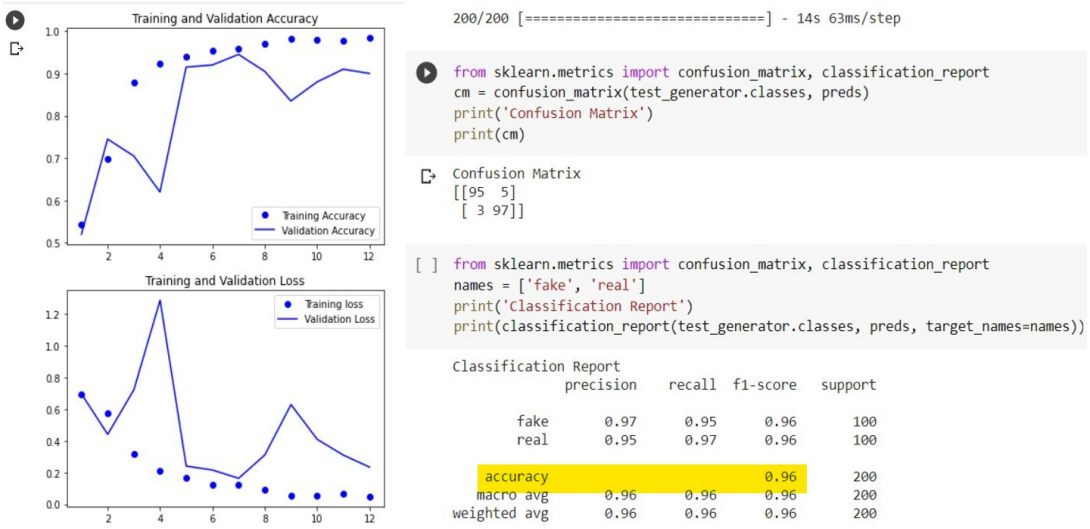


Figure 11: Plots and Classification Report (Model 2)

Training and validation accuracy, as well as training and validation loss, are plotted for each epoch in the figure below. Except for epochs 3, the model performed well, as both the training and validation points are near in both plots. The best model (from training) is used for testing the test set of 200 images after model training is completed. The model 2 classification report is also shown above in figure 11, which demonstrates that testing accuracy and F-1 are both 96%.

5.3.3 Inception-ResNet-V2 (CNN) with LSTM (Model 3)

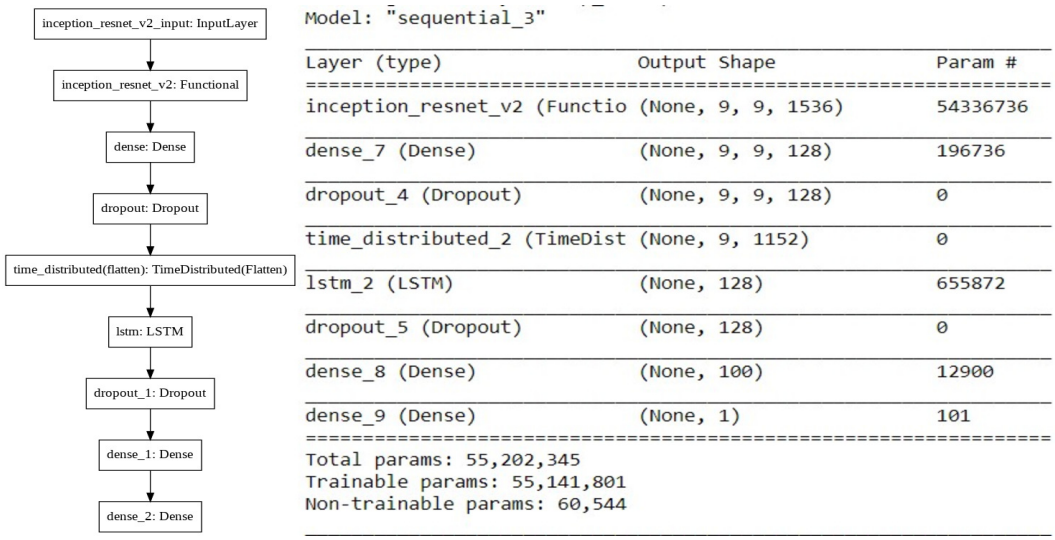


Figure 12: Model 3 Architecture

After preprocessing, pre-processed data is supplied as input to the InceptionResNet v2 model. CNN architecture used pre-trained networks with initial weights taken from Imagenet data training, and then dense and dropout layers were added after it. Pre-trained CNN architecture with 54336736 parameters, which is significantly more than

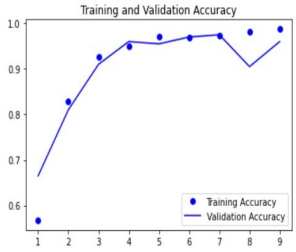
ExceptionNetB4 and inception v3. Because InceptionResNetv2 already includes a dense framework, we deleted a dense layer. Similar to Inception v3, the LSTM layer includes 655872 parameters. The rest of the model compilation parameters remain unchanged. A total of 20 epochs are used to run the model. Because of the significant computing demands, the patience parameter has been set to 3. The early stopping feature is used to halt model training.

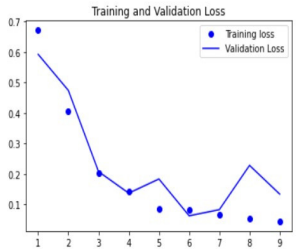
```
Epoch 00008: val_loss did not improve from 0.06265
Epoch 9/20
100/100 [==============================] - 1671s 17s/step - loss: 0.0443 - accuracy: 0.9875 - val_loss: 0.1338 - val_accuracy:
0.9600

Epoch 00009: val_loss did not improve from 0.06265
Epoch 00009: early stopping
Epoch 00005: val_loss did not improve from 0.13696
Epoch 6/20
100/100 [==============================] - 1771s 18s/step - loss: 0.0827 - accuracy: 0.9688 - val_loss: 0.0627 - val_accuracy:
0.9700
```

Figure 13: Training and Validation Accuracy (Model 3)

As shown in the figure 13 above, the training and validation accuracy of InceptionResNet-v2 and the LSTM hybrid model is 98.75 percent and 97.00 percent, respectively, from separate epochs (9 and 6). The training and validation losses are respectively 0.0443 and 0.0627. After epochs 9, the Early Stopping feature is also indicated below because there is no improvement in validation loss.

In the figure 14 below, training and validation accuracy, as well as training and validation loss, are presented on a plot using the matplotlib library for each epoch. Except for epochs 8, the model performed exceptionally well, as the training and validation plot points in both plots are nearby. The model 2 classification report, which shows that testing accuracy and F-1 are both 97%, is also presented below.






```
200/200 [==============================] - 62s 295ms/step
Confusion Matrix
[[98  2]
 [ 4 96]]

#### PRINT CLASSIFICATION REPORT
names = ['fake', 'real']
print('Classification Report')
print(classification_report(test_generator.classes, preds, target_names=names))

Classification Report
              precision    recall  f1-score   support

        fake       0.96      0.98      0.97       100
        real       0.98      0.96      0.97       100

    accuracy                           0.97       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.97      0.97      0.97       200
```

Figure 14: Plots and Classification Report (Model 3)

5.4 Model Comparision and Discussion

After model evaluation, Results are compared below in a table. All hybrid models performed exceptionally well in the task of detecting deepfakes, as training accuracy for all three models is above 98%. A slight difference is observed in test accuracy, as the EfficientNetB4-LSTM model comes out to be the best model among all with the highest test accuracy. As compared to other literature papers, the accuracy of EfficientNetB4-LSTM models was better than Bonettini et al. (2021), which implemented EfficientNetB4 with Attention and Siamese training on DFDC and FF++ datasets. Overall, Pretrained CNN architecture with LSTM hybrid models performed well in deepfake detection and classification tasks. Training time for each model also plays a vital role in the performance of deep learning models. In terms of Training time, we have chosen a batch size of 16 and epochs as 20 for all models. Model 1 and 2 trained faster (Approx. 3 mins per epochs) due to smaller architectures (lesser parameters involved) as compared to Model 3 (Approx. 28 mins per epochs).

Table 1: Results Comparision

Model	Train Accuracy (%)	Test Accuracy (%)
EfficientNetB4-LSTM (Model 1)	98.75	98.00
InceptionV3-LSTM (Model 2)	98.37	96.00
InceptionResNetv2-LSTM (Model 3)	98.75	97.00

6 Conclusion and Future Work

The major purpose of the research project is to address the research question, "How accurately can the hybrid deep learning model detect deepfake images generated by Style-GAN?" To detect deepfake images from a publically available FFHQ dataset, a hybrid deep learning model is utilized, with pre-trained CNN architectures (EfficientNetB4, Inception-v3, and InceptionResNet-v2) for extracting features and LSTM classifier for sequence classification. To discover the best potential model in detecting deepfakes, Evaluation metrics like as Classification Report, Accuracy, and F-1 Score is used to compare different CNN architectures with LSTM. In this study, EfficientNetB4 with LSTM is the best model, with 98 percent test accuracy and less training time than other models. Although InceptionResNetv2 with LSTM is slightly more accurate than Inceptionv3, Inceptionv3 requires less training time. To summarize, using pre-trained CNN architectures with LSTM is a wonderful way to detect deepfake images or videos and can assist in the development of a successful system for this purpose.

Higher training time for models with larger architecture, such as InceptionResNetv2, is a limitation of this study. Due to the lack of better system configurations, a smaller dataset of 2000 images is used for research, although CNN architectures can work on bigger datasets as well. Due to Out Of Memory exceptions, while executing code, the image size must be kept at 350 * 350 pixels. Future work can be done with a larger portion of this dataset, taking better resolution images (1024 * 1024) and better system configurations to construct a strong and reliable system that can detect deepfakes in videos or images.

7 Acknowledgement

I'd like to thank my supervisor, Dr. Christian Horn, for his guidance during the project's execution. I'd want to thank him for his regular assistance and supervision, which helped the project's execution go smoothly.

References

Bonettini, N., Cannas, E. D., Mandelli, S., Bondi, L., Bestagini, P. and Tubaro, S. (2021). Video face manipulation detection through ensemble of cnns, *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5012–5019.

Chen, B., Ju, X., Xiao, B., Ding, W., Zheng, Y. and de Albuquerque, V. H. C. (2021). Locally gan-generated face detection based on an improved xception, *Information Sciences* **572**: 16–28.

Dastider, A. G., Sadik, F. and Fattah, S. A. (2021). An integrated autoencoder-based hybrid cnn-lstm model for covid-19 severity prediction from lung ultrasound, *Computers in Biology and Medicine* **132**: 104296.

Gill, H. S. and Khehra, B. S. (2021). An integrated approach using cnn-rnn-lstm for classification of fruit images, *Materials Today: Proceedings* .

Guo, Z., Yang, G., Chen, J. and Sun, X. (2021). Fake face detection via adaptive manipulation traces extraction network, *Computer Vision and Image Understanding* **204**(5): 103170.

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

Hashmi, M. F., Ashish, B. K. K., Keskar, A. G., Bokde, N. D., Yoon, J. H. and Geem, Z. W. (2020). An exploratory analysis on visual counterfeits using conv-lstm hybrid architecture, *IEEE Access* **8**: 101293–101308.

Karras, T., Laine, S. and Aila, T. (2019). A style-based generator architecture for generative adversarial networks.

Khalil, H. A. and Maged, S. A. (2021). Deepfakes creation and detection using deep learning, *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 1–4.

Li, M., Liu, B., Hu, Y., Zhang, L. and Wang, S. (2021). Deepfake detection using robust spatial and temporal features from facial landmarks, *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, pp. 1–6.

Ozkaya, U., Oztürk, S., Melgani, F. and Seyfi, L. (2021). Residual cnn+bi-lstm model to analyze gpr b scan images, *Automation in Construction* **123**: 103525.

Oztürk, S. and Ozkaya, U. (2021). Residual lstm layered cnn for classification of gastrointestinal tract diseases, *Journal of Biomedical Informatics* **113**: 103638.

Pan, D., Sun, L., Wang, R., Zhang, X. and Sinnott, R. O. (2020). Deepfake detection through deep learning, *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pp. 134–143.

Pokroy, A. A. and Egorov, A. D. (2021). Efficientnets for deepfake detection: Comparison of pretrained models, *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pp. 598–600.

Sanghvi, B., Shelar, H., Pandey, M. and Sisodia, J. (2021). Detection of machine generated multimedia elements using deep learning, *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1238–1243.

Stanciu, D. C. and Ionescu, B. (2021). Deepfake video detection with facial features and long-short term memory deep networks, *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, pp. 1–4.

Suratkar, S., Johnson, E., Variyambat, K., Panchal, M. and Kazi, F. (2020). Employing transfer-learning based cnn architectures to enhance the generalizability of deepfake detection, *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–9.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks, *CoRR* **abs/1905.11946**.
URL: *http://arxiv.org/abs/1905.11946*

Tjon, E., Moh, M. and Moh, T.-S. (2021). Eff-ynet: A dual task network for deepfake detection and segmentation, *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pp. 1–8.

Trinh, L., Tsang, M., Rambhatla, S. and Liu, Y. (2021). Interpretable and trustworthy deepfake detection via dynamic prototypes, *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1972–1982.

ul ain, Q., Nida, N., Irtaza, A. and Ilyas, N. (2021). Forged face detection using ela and deep learning techniques, *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, pp. 271–275.

Zhang, J., Ni, J. and Xie, H. (2021). Deepfake videos detection using self-supervised decoupling network, *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6.