DEEPFAKE IMAGE DETECTION USING DEEP LEARNING

A PROJECT-II REPORT SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

by

Batch–A17

G. Anandita Dakshayani (19JG1A0541) CH. Eswari (19JG1A0525)

A.V.S. S. Amrutha(19JG1A0504)

Under the esteemed guidance of
Dr. N. Sharmili

Associate Professor

CSE Department

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]
[Accredited by National Board of Accreditation (NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-22 to2022-25]
[Accredited by National Assessment and Accreditation Council [NAAC] with A Grade – Valid from 2022-27]
Kommadi, Madhurawada, Visakhapatnam–530048

2019–2023

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project Work-II report titled **“DEEPFAKE IMAGE DETECTION USING DEEP LEARNING”** is a bonafide work of following IV B. Tech II Semester students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2022- 2023, in fulfilment of the requirement for the award of the degree of Bachelor of Technology of this university.

G. Anandita Dakshayani (19JG1A0541)   CH. Eswari (19JG1A0525)

A.V.S. S. Amrutha(19JG1A0504)

Signature of the Guide   Signature of the HOD

Dr. N. Sharmili MTech., Ph.D.,
Associate Professor
(Internal Guide)

Dr. P. V. S. Lakshmi Jagadamba MTech., Ph.D.,
Professor
(Head of the department)

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **Dr. N. Sharmili,** Associate Professor for encouragement all the way during analysis of the project. Her annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We would like to take this opportunity to extend our gratitude to Project Coordinator, **Mrs. K. Rohini,** Assistant Professor of Computer Science and Engineering for making us to follow a defined path and for advising us to get better improvements in the project.

We express our deep sense of gratitude and thanks to **Dr. P. V. S. Lakshmi Jagadamba**, Professor and Head of the Department of Computer Science and Engineering for her guidance and for expressing her valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to **Dr.R. K. Goswami,** Principal and **Dr. G. Sudheer,** Vice Principal for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We are also thankful to both teaching and non-teaching faculty of the Department of Computer Science and Engineering for giving valuable suggestions from our project.

G. Anandita Dakshayani (19JG1A0541) **CH. Eswari** (19JG1A0525)

A.V.S. S. Amrutha(19JG1A0504)

TABLE OF CONTENTS

ABSTRACT

DeepFakes are fake computer-generated images created using the algorithms such as Generative Adversarial Networks (GAN) and Autoencoders. These algorithms can be used to create scandalous photos of innocent people and celebrities alike. The faked images appear to be so realistic that it is virtually impossible to detect them with the naked eye. DeepFakes are a fairly new issue that needs to be investigated in depth. Two methods are proposed to distinguish between the real and fake images one – by using SVM and DenseNets. In the SVM method, to uncover the distinguishing characteristics between false and real photos, the approach preprocesses the images using image processing. In order to identify between a fake and a real image, the SVM classifier is then trained on these detecting features. The three implemented classifiers are 1) Linear Kernel 2) Gaussian Radial Basis Function Kernel (RBF) 3) The Polynomial Kernel. The other approach, DenseNets involves preprocessing the input images by using Data Normalisation. The DenseNet algorithm is applied to the preprocessed images for classification into fake and real images.

LIST OF FIGURES

LIST OF TABLES

LIST OF SCREENS

1. INTRODUCTION

For a long time, people have been fascinated by the alteration of photo, audio, and video material. While manipulating photographs using software like Adobe Photoshop is relatively simple, audio and video modification may be challenging at times. To introduce special effects into the cinema, videos had to be manipulated frame by frame in the beginning. Almost everyone nowadays has the ability to do some form of video editing on their computer. With the advancement of Artificial Intelligence (AI) and Machine Learning algorithms, a new method for editing photo, audio, and video content has just been launched. It first became known to the public when Face Swap Machine Learning algorithms were used to project the faces of celebrities into obscene films. However, one noteworthy example of these techniques is a false video of President Obama warning against DeepFakes [Jordan Peele 2018].

DeepFake is a catch-all phrase for pictures, videos, and audio files that havebeen altered using deep learning techniques. There are various techniques for creating totally computer-generated photos that are remarkably identical to actual photographs. Professionals can generate such photographs and films, but there are tools available that allow even inexperienced users to create reliable DeepFakes one such application is Zao, a Chinese DeepFake software, has taken the internet by surprise. It allows one to generate DeepFake videos in seconds and the results appear extremely authentic.

Facebook has started a Deepfake Detection Challenge, a collaborative effort with top scholars, business specialists, and policymakers to create improved tools for detecting deepfakes. Facebook is also using machine learning tools to detect manipulated media on its site.

Similarly, Google has been focusing on creating deepfake detection technology. In 2019, Google published a collection of deepfake videos to assist academics in developing improved detection techniques. Furthermore, Google has been using machine learning techniques to detect altered media on YouTube. Amazon and Apple are also spending in the development of deepfake identification technology. Amazon's Rekognition service contains a feature for identifying altered pictures and videos, and

Apple is allegedly working on a machine learning tool for detecting deepfakes.

Netflix is also engaged in the development of deepfake identification technology. Netflix started a challenge in 2020 to promote the creation of improved deepfake detection algorithms.

Overall, FAANG firms are addressing the problem of deepfakes by spending in research and development of deepfake detection technology. However, detecting deepfakes remains a complex and difficult issue, with much more work to be done in this field.

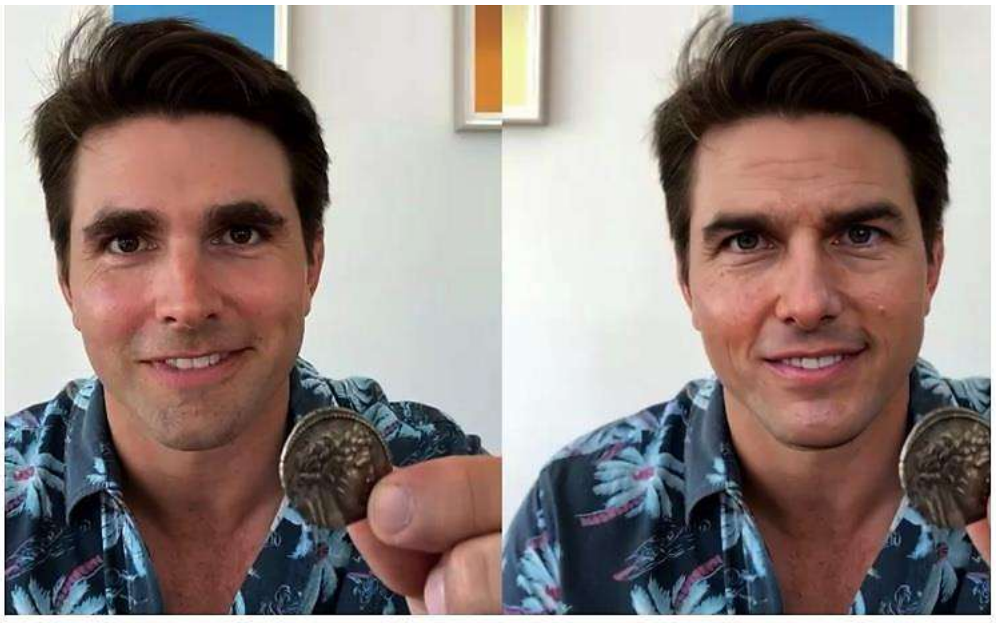
1.1 PROBLEM DEFINITION

DeepFakes currently provide a serious risk due to their extensive ability to increasingly resemble individuals in terms of image, video, and voice. Thus, it is important to start looking for solutions to this issue as soon as possible. In this project, wewant to put into practise two systems: a system that creates progressively plausible pictures and a system that accurately recognises these images.

1.2 MOTIVATION

There are several harmful situations in which DeepFakes may be utilized to harm our society and people, hence study into identifying these manipulations appears to be critical. This subject is extremely intriguing from a technological standpoint. Because of its freshness, it is a very active subject of study, both in terms of creation and detection. The technique of producing DeepFakes is also quite interesting in order to completely comprehend their flaws. There are several generative ways to learn, including Autoencoders and Generative Adversarial Networks. However, our primary goal is to detect DeepFakes, hence the emphasis is on Support Vector Machines for classification, as well as approaches to improve their classification performance.

There are various dangerous circumstances in which DeepFakes might be used to harm our society and the people in it such as identity theft, hence research into identifying these manipulations is critical. Another instance of DeepFakes that came to light was the incident of a person who spent months to DeepFake himself to look like Tom Cruise as shown in Figure 1.



Want to see a magic trick? Tom Cruise impersonator Miles Fisher (left) and the deepfake Tom Cruise created by Chris Ume (right). | Image: Chris Ume

Figure 1: *Real and DeepFake image comparison*

1.3 ORGANIZATION OF DOCUMENTATION

The report is structured as follows:

Chapter 1 contains the Introduction of the project, and it contains an overview of the project.

Chapter 2, provides a Literature Study, and crucial variables for making stronger contributions are indicated.

Chapter 3, documents the Methodology employed in the project.

Chapter 4, the need of Requirement Analysis for hardware and software requirements isunderlined.

Chapter 5, includes the Designing of the project

Chapter 6, shows Implementation containing the step-by-step process and screenshots of the output.

Chapter 6, covers future scope.

Chapter 7, constitutes the Conclusion of the project.

Chapter 8, provides the citations and a basis article.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEMS

Katarya and Lal, “**A Study on Combating Emerging Threat of DeepFake Weaponization”**, proposed a DeepFake detection model based on SSTNets that use spatial, temporal, and steganalysis for detection and GANs and autoencoders were used for the DeepFake image creation [1]. Nevertheless, more tools and their upgraded versions are required to assist in the detection process. The SSTNet model provides flexibility and can be used to classify both fake videos and images. However, the model’s accuracy is at 90%.

Aduwalaet al., “**DeepFake Detection using GAN Discriminators”**, explored a solution for a DeepFake detection system based on GAN discriminators to detect DeepFake videos. Using MesoNet as a baseline, a GAN was trained, and a discriminator was extracted as a dedicated module to detect DeepFakes [2]. The results indicate that the discriminator accuracy on known datasets is high while accuracy on never-before seen datasets is low. Nonetheless, a GAN discriminator is not robust enough to be used as a general-purpose DeepFake detector.

Agarwal et al., “**DeepFake Detection Using SVM”**, introduced a method known as frequency domain analysis after which a classifier will be used to differentiate the real and fake image [3]. DFTs and FFTs were used to pre-process the collected datasets. The proposed pipeline does not include nor requires tremendous amounts of information. This method can distinguish the high resolution DeepFake images with 99.76% accuracy. However, the acceptable images can only be square shaped.

Güera and Delp, “**DeepFake Video Detection Using Recurrent Neural Networks**”, proposed a temporal-aware pipeline to automatically detect DeepFake videos [4]. Thesystem used a convolutional neural network (CNN) to extract frame-level features. These features are then used to train a recurrent neural network (RNN) that learns to classify whether a video has been subject to manipulation or not. The experimental results using a large collection of manipulated videos had shown that using a simple convolutional LSTM structure can accurately predict if a video has been subject to manipulation or not with as few as 2 seconds of video data. However, the robustness if this model is debatable.

Al-Dhabi and Zhang, “**DeepFake Video Detection by Combining Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)**”, presented a system based on a combination of CNN and RNN, as research shows that using CNN and RNN combined achieve better results [5]. The experimental results using a large collection of manipulated videos had obtained competitive results while utilizing a simplisticarchitecture. However, there is room for improvement of the proposed model.

Li et al., “**DeepFake-o-meter: An Open Platform for DeepFake Detection**”, proposed a DeepFake-o-meter which is an online platform where for developers of DeepFake detection algorithm it provides an API architecture to wrap individual algorithm and run onthird party remote server-Frontend, Backend and Data Synchronizing [6]. The front- end isthe website portal to interact with users. The back end is the core component of this platform, which calls corresponding detection methods to the submitted videos and thedata synchronizing is the protocol for exchanging the interested data between front-endand back-end.

Rana et al., “**DeepFake Detection Using Machine Learning Algorithms**”, explored a process that involves creating a unique set of features by combining HOG, Haralic, Hu Moments, and Color Histogram features [7] that lessen the data's complexity and making patterns recognizable by splitting a task into two stages: object detection and objectrecognition. The object detection phase scans an entire image and identifies all possible objects. The object recognition step identifies relevant objects. a CNN that extracts the most critical features and an LSTM for sequential analysis.in four distinct CNNs were trained using real and fake images.

Rafique, et al., “**DeepFake Detection Using Error Level Analysis and Deep Learning**”, in their paper proposed the model using convolution neural network (CNN) models including Alex Net and Shuffle Net [8]. These are used to recognize genuine and counterfeit face images in this article. Two strategies for detecting deep fakes utilizing ELA and DL techniques in this paper. In ELA digital data, i.e., images are compressed at a positive degree of image quality, and then we take the difference of this compressed data with original data. In Dl used a YOLO face detector to extract face area from video frames and then pass these faces are passed to the InceptionResNetV2 CNN model to extract visual features.

Khichi and Yadav, “**A Threat of DeepFakes as a Weapon on Digital Platform and their Detection Methods**”, in their paper observed that researchers use the face-swapping technique with Generative Adversarial Networks (GANs)to build DeepFakes [9]. While a detector or discriminative network is another. These two networks collaborate to create realistic-looking fake videos photographs. That discriminative network decides whether generated representation is accurate and believable, while the generative system uses two CNNs the Encoder and the Decoder to create images.

Yang et al., Exposing “**Deep Fakes Using Inconsistent Head Poses**”, in their approach used an intrinsic limitation in the deep neural network face synthesis models [10]. SVM based classifier is used to differentiate between real and fake images based on eye and mouth. Generative neural networks are used to generate fake images.

Li et al., “**DeepFake Detection Using Robust Spatial and Temporal Features from Facial Landmarks**”, suggested a novel method for detecting DeepFakes using biometric characteristics [11]. A collection of chosen face landmarks was used to generate thebiometric traits. They first proposed a measure for selecting strong face landmarks, and then used the landmarks to generate facial feature vectors. The spatial and temporalrotation degrees are included to make the creation of the SVM feature vector easier. In essence, they employed spatial and temporal rotation angles to define the underlyingconsistency of face landmarks at both the frame and video levels. The experimentalfindings show that their detector has the highest resilience when compared to six other approaches at the time. It also has high AUC (Area Under the Receiver Operating Characteristic Curve) and detection accuracy.

Pan, et al., “**DeepFake Detection through Deep Learning**”, in their paper a total of eight DeepFake video classification models were trained, evaluated, and compared based onfour fake video generation methods and two neural networks [12]. Each model exhibited satisfactory classification performance over the corresponding dataset used to train it. Specifically, for the Xception models, the overall fake detection accuracy was above 90% and the model performed slightly better at detecting real videos compared to fake videos with a 2-3% increase in accuracy. The NeuralTextures model was an outlier in this test with the same detection rate of 91% for both true positive rates and true negative rates.

Joseph and Nyirenda, "**DeepFake Detection using a Two-Stream Capsule Network**", in their paper addressed the challenge of DeepFake Detection in their study by operating a Two-Stream Capsule Network in parallel that takes in both RGB pictures or frames and Error Level Analysis images [13]. The suggested technique achieves detection accuracy of 73.39% for the DeepFake Detection Challenge (DFDC) and 57.45% for the Celeb-DF datasets. However, these results are from a preliminary application of the proposed technique. The findings, however, were acquired from models that had not been optimised due to a lack of computing resources to train the model.

DeepFake detection was defined as a one-class anomaly detection issue by Khalid and Woo, "**OC-FakeDect: Classifying DeepFakes Using One-class Variational Autoencoder"** [14]. They propose OC-FakeDect, which employs a one-class Variational Autoencoder (VAE) to train solely on actual face photos and detects non-real images as anomalies. Their first results demonstrate that our one class-based technique can be promising for identifying DeepFakes, obtaining 97.5% accuracy on the well-known FaceForensics++benchmark dataset's NeuralTextures data without utilising any fake photos in the training procedure.

Ajoy et al., "**DeepFake Detection using a frame-based approach involving CNN**", suggested a CNN-based algorithm in their article that learns essentially unique patterns that differ between a DeepFake and a real video [15]. Pixel distortion, discrepancies with facial superimposition, skin colour variances, blurring, and other visual aberrations are among these distinguishing traits. The proposed model has trained a CNN (Convolutional Neural Network) to efficiently differentiate DeepFake movies based on various attributes using a frame-based technique. The model's initial findings, acquired after being trained on around 700 films, are encouraging, with a validation accuracy of 85.8 percent, and a validation loss of 0.3403.A.

Guo et al., "**Robust Attentive Deep Neural Network for Detecting GAN- Generated Faces"**, in their article proposed architecture for GAN-generated face detection [16]. They used DLib to detect faces and localize eyes, and use Mask R-CNN to segment out the iris regions. A Residual Attention Network (RAN) then performs binary classification on the extracted iris pair to determine if the face is real or fake. The trainingis carried out using a joint loss combining the Binary Cross-Entropy (BCE) loss and the ROC-AUC loss with WMW relaxation for better accuracy

Ma et al., "**Multi-Perspective Dynamic Features for Cross-Database Face Presentation Attack Detection**", proposed a novel scheme to extract the facial motion patterns in a video by analysing the pixel value changes throughout the video [17]. The scheme globally maps the temporal motion information into a single image by integratingeach pixel in the time domain. Considering the complementarity between the motion pattern and the noise pattern, they further combine the motion pattern with the noise pattern, thereby further improving the performance of cross-database face representation attack detection.

Zhang et al., "**No One Can Escape: A General Approach to Detect Tampered and Generated Image**", in their article brought forward a single model which is used detect tampered images and GANs generated images [18]. They found a general method which introduces a strong generalization ability into the detection of images generated byvarious GANs. They put forward a structure with depth wise separable convolutions as the main component which with a smaller number of parameters compared with the one consists of traditional convolution. Compared with other work, this method has good performance in detecting tampered images.

Guarnera et al., "**Fighting DeepFake by Exposing the Convolutional Traces on Images**", in their work offered an algorithm based on Expectation-Maximization to extract the Convolutional Traces (CT) - a unique fingerprint useable to identify the fakes among real, if an image is a DeepFake but also the GAN architecture that generated it [19]. The CT extracted the fingerprint demonstrated to have high discriminative power.

Kim and Cho, “**Exposing Fake Faces Through Deep Neural Networks Combining Content and Trace Feature Extractors**”, in their paper paper proposed a hybrid face forensics framework based on a Convolutional Neural Network combining the two forensics approaches [20]. To validate the proposed framework, they used a public Face2Face dataset and a custom DeepFake dataset collected on their own. Face authenticity classifier combines content feature extractor (CFE) and trace feature extractor (TFE). A convolution is depicted by a square containing its detail in the two feature extractors. Their proposed system is shown below,

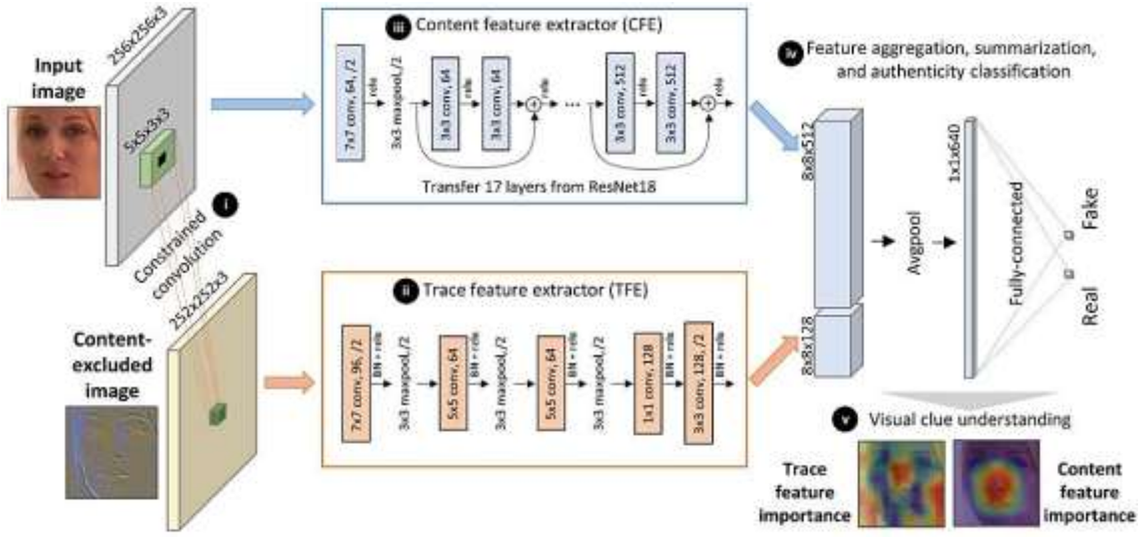


Figure 2: *Existing System [20]*

J. Kang et al., “**Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces”** proposed a technique for detecting various types of deepfake images using three common traces generated by deepfakes: residual noise, warping artifacts, and blur effects [21] by employing steganalysis to detect pixel-wise residual-noise traces. However, it detects only three types of deepfake techniques: face swap, puppet-master, and attribute change.

V. -N. Tran et al., “**Generalization of Forgery Detection with Meta Deepfake Detection Model**”, explored a deepfake detection method based on meta learning named Meta Deepfake Detection (MDD) [22]. Its goal was to develop a generalized model capable of solving new unseen domain directly without the need for model updates. Nevertheless, experimentation with new benchmarks required. DFDC, Celeb-DF-v2, and FaceForensics++ datasets were used to train and test.

T. Jung et al., "**DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern"** introduced a method called DeepVision is implemented to verify an anomaly based on the period, repeated number, and elapsed eye blink time when eye blinks were continuously repeated within a very short period of time [23]. FaceForensics and Kaggle's Eye blinking prediction datasets were employed in this model. However, integrity verification may not be applicable to people with mental illnesses.

A. Groshev et al., "**GHOST—A New Face Swap Approach for Image and Video Domains**", proposed a new one-shot pipeline for image-to-image and image-to-video face swap solutions [24]. VGGFace2, FaceForensics++, FaceSwap datasets were used in their work. Fine tuning required for more real look images.

M. Jiwtode et al., "**Deepfake Video Detection using Neural Networks"**, in their paper proposed a model to distinguish between a real and a Deepfake video using RNN and CNN [25]. DFDC, FaceForensics++, Custom Image datasets were used in this system. InceptionResnet V2 CNN was used in this work. However, the model cannot identify audio DeepFake within the video.

A. Trabelsi et al., **"Improving Deepfake Detection by Mixing Top Solutions of the DFDC",** experimented with various assembly strategies [26]. UADFV, DeepfakeTI MIT, FaceForensi cs++, Google DFD, Celeb-DFD, DeeperForensics-1.0, DFDC datasets were used in this model. However, the methods are not well generalized.

Y. -X. Luo and J. -L. Chen, "**Dual Attention Network Approaches to Face Forgery Video Detection**", designed a Dual Attention Forgery Detection Network (DAFDN) [27]. DFDC and FaceForensics++ datasets were made use of in this paper. However, DAFDN focuses on the modified face positions alone.

I. -J. Yu et al., "**Manipulation Classification for JPEG Images Using Multi-Domain Features",** proposed a manipulation classification network is order to distinguish between a manipulated JPEG image from a real JPEG image using MCNet to classify manipulation [28]. DFDC, ALASKA dataset, FiveK datasets were used. Its disadvantage lies in the fact that it is ineffective for real world scenarios.

D. Du et al., **"Multi Branch Deepfake Detection Based On Double Attention Mechanism",** proposes a multi branch artifact detection algorithm based on double attention mechanism, which can detect subtle artifact to detect deepfakes [29]. FaceForensics++ dataset was used to create multi-branch network based on double attention. Nevertheless, there is still scope for improvement.

In conclusion, the research papers mentioned above detail the various techniques used to detect DeepFakes. Most of the above works use Neural Networks as means to distinguish between real or fake images and videos.

Table 1: *A Comparative Study of Related Works*

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
1	A Study on Combating Emerging Threat of Deep fake Weaponization	2020	2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)	R. Katarya , A. Lal	MNIST datset	Generative Adversial Networks	Cannot identify hidden features in images
2	DeepFake Detection using GAN Discriminators	2021	2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)	S. A. Aduwala, M. Arigala, S. Desai, H. J. Quan, and M. Eirinaki	CelebA, StyleGAN, DFDC, 70K Real Faces Datasets.	GAN Discriminators	Not robust enough to work as a general-purpose DeepFake detector.
3	DeepFake Detection Using SVM	2021	2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)	H. Agarwal, A. Singh, and R. D	CelebAHQ(1000), Flicker(1000)	Support Vector Machine	May not detect newfake features generated by new GAN

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
4	DeepFake Video Detection Using Recurrent Neural Networks	2018	2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)	D. Güera and E. J. Delp	HOHA dataset	CNN for feature extraction and LSTM for sequence processing	Performance is yet to be investigated on replay attacks and spoofing speech
5	DeepFake Video Detection by Combining Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)	2021	2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)	Y. Al-Dhabi and S. Zhang	UADFV, FaceForensic++, DFDC, Celeb-DF	Combination of CNN and RNN	Varying results with different datasets
6	DeepFake-o-meter: An Open Platform for DeepFake Detection	2021	2021 IEEE Security and Privacy Workshops (SPW)	Y. Li et al	FaceForensics ++ dataset	DeepFake-o-meter composedby front-end andback-end	Not very large increment in performance compared to other similar approaches

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
7	DeepFake Detection Using Machine Learning Algorithms	2021	2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI)	M. S. Rana, B. Murali and A. H. Sung	FaceForencics++, DFDC, Celeb-DF, VDFD datasets	SVM, RF, ERT, DT, MLP, SGB, KNN algorithms are used	When training the SVM classifier for greater precision,multi-feature point detectors will not be used as the input dataset
8	DeepFake Detection Using Error Level Analysis and Deep Learning	2021	2021 4th International Conference on Computing & Information Sciences (ICCIS)	R. Rafique, M. Nawaz, H. Kibriya	CASIA dataset	Convulutional neural networkis used	Limited to uncompreddes data
9	A Threat of DeepFakes asa Weapon on Digital Platform and their Detection Methods	2021	2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)	M. Khichi and R. Kumar Yadav	DFDC, StyleGAN datasets	Combination of CNN and GAN algorithms	This relay on high-level semantics derived from images provably

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
10	Exposing Deep Fakes Using Inconsistent Head Poses	2019	ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)	X. Yang, Y. Li, and S. Lyu	UADFV, CelebA datasets	Support Vector Machine classifier is used	Cannot identify the hidden features in the images
11	DeepFake Detection Using Robust Spatial and Temporal Features from Facial Landmarks	2021	2021 IEEE International Workshop on Biometrics and Forensics (IWBF)	M. Li, B. Liu, Y. Hu, L. Zhang, and S. Wang	UADFV dataset	LSTM and SVM algorithmsare used	Doesn’t take into account the inconsistencies in blinking rate
12	DeepFake Detection through Deep Learning	2020	2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologie s(BDCAT)	D. Pan, L. Sun, R. Wang, X. Zhang and R. O. Sinnott	FaceForensi cs++	Xception and MobileNet	Is based on isolated pictures obtained from videos

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
13	DeepFake Detection using a Two-Stream Capsule Network	2021	2021 IST-Africa Conference (IST-Africa)	Z. Joseph and C. Nyirenda	DFDC(120 Kvideos), Celeb-DF datasets	Single stream capsule network	Large amount of data is required for preprocessing
14	OC-FakeDect: Classifying DeepFakes Using One-class Variational Autoencoder	2020	2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)	H. Khalid and S. S. Woo	FaceForensics++	Variational Autoencoder(VAE)	The model considered only real images and notfake images
15	DeepFake Detection using a frame-based approach involving CNN	2021	2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)	A. Ajoy, C. U. Mahindrakar, D. Gowrish, and V. A	DeepFake Detection Challenge	CNN, SGD, MTCNN algorithms are used	Does not use any fake images in the training process instead it is trained using real images
16	Robust Attentive Deep Neural Network for Detecting GAN-Generated Faces	2022	IEEE Access, vol. 10	H. Guo, S. Hu, X. Wang, M. - C. Chang and S. Lyu	Flickr- faces-HQ,FFHQ-GAN datasets	RAN and GAN algorithms are used	This model is very large, and if it isn’t constructed carefully, it will result in overfitting

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
17	Multi-Perspective Dynamic Features for Cross-Database Face Presentation Attack Detection	2020	IEEE Access, vol. 8	Y. Ma, Y. Xu and F. Liu	Replay-Attack, CASIA-FASD, MSU-MFSD datasets	CNN is for temporal mapping and SVM is used for classification purpose	Only pristine face images were used in the training
18	No One Can Escape: A General Approach to Detect Tampered and Generated Image	2019	IEEE Access, vol. 7	K. Zhang, Y. Liang, J. Zhang, Z. Wang and X. Li	CACIA, LSUN, PGGAN, SNGAN, StyleGAN datasets	Default setting stride is SAME padding in convolution operation and Relu activate function be used	It is not possible to obtaina mask for precise facial positioning and cutting
19	Fighting DeepFake by Exposing the Convolutional Traces on Images	2020	IEEE Access, vol. 8	L.Guarnera, O. Giudice and S. Battiato	DRESDEN, UCID, VISION datasets	Convulutional Neural Network	Only considers lack of blinking for detection and not the frequency of blinking

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
20	Exposing Fake Faces Through Deep Neural Networks Combining Content and Trace Feature Extractors	2021	IEEE Access, vol. 9	E. Kim and S. Cho	Face2Face and createda tailor- made DeepFake dataset	Convulutional Neural Network	Accuracy decreased when the video is compressed
21	Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces	2022	*IEEE Access*, vol. 10	J. Kang et al.,	Custom dataset	Steganalysis to detect pixel-wise residual-noise traces.	Detects only three types of deepfake techniques: face swap, puppet-master, and attribute change
22	Generalization of Forgery Detection with Meta Deepfake Detection Model	2022	IEEE Access	V. -N. Tran et al.,	DFDC, Celeb-DF-v2, FaceForensics ++	Meta Deepfake Detection.	Experimentation with new benchmarks required

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
23	DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern	2020	IEEE Access, vol. 8	T. Jung et al.,	FaceForensics, Kaggle's Eye blinking prediction,	Tracking Significant changes in the eye blinking patterns in deepfakes.	The integrity verification may not be applicable to people with mental illnesses
24	GHOST—A New Face Swap Approach for Image and Video Domains	2022	*IEEE Access*, vol. 10	A. Groshev et al.,	VGGFace2, FaceForensics ++, FaceSwap	GHOST (Generative High-fidelity One Shot Transfer).	Fine tuning required for more real look images
25	Deepfake Video Detection using Neural Networks	2022	2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)	M. Jiwtode et al.,	DFDC, FaceForensics ++, Custom Image dataset	InceptionResnetV2 CNN	Cannot identify Audio DeepFake

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
26	Improving Deepfake Detection by Mixing Top Solutions of the DFDC	2022	2022 30th European Signal Processing Conference (EUSIPCO)	A. Trabelsi et al.,	UADFV, DeepfakeTIMIT, FaceForensics++, Google DFD, Celeb-DFD, DeeperForensics-1.0, DFDC	Assembly strategies	Methods are not well generalized.
27	Dual Attention Network Approaches to Face Forgery Video Detection	2022	*IEEE Access*, vol. 10	Y. -X. Luo, J. -L. Chen	DFDC and FaceForensics++	Dual Attention Forgery Detection Network (DAFDN)	DAFDN focuses on the modified face positions
28	Manipulation Classification for JPEG Images Using Multi-Domain Features	2020	IEEE Access, vol. 8	I. -J. Yu et al.,	DFDC, ALASKA dataset, FiveK,	MCNet to classify manipulation algorithms applied to JPEG compressed image	Ineffective for real world scenarios.
29	Multi branch deepfake detection based on double attention mechanism	2021	2021 International Conference on Electronic Information Engineering and Computer Science	D. Du et al.,	FaceForensics++	Multi-branch network based on double attention	Scope for improvement

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
30	An Exploratory Analysis on Visual Counterfeits Using Conv-LSTM Hybrid Architecture	2020	IEEE Access, vol. 8,	M. F. Hashmi et al.,	DFDC dataset	Conv-LSTM which uses facial landmarks and convolutional features	Challenge in prediction without real video for having precise results.
31	Convolutional Neural Network Based on Diverse Gabor Filters for Deepfake Recognition	2022	IEEE Access, vol. 10	A. H. Khalifa et al.,	CD2, DFDC, FF++ and WDF	Convolutional Neural Network is used with a unified Gabor function	Small number of parameters
32	DeepFake Detection for Human Face Images and Videos: A Survey	2022	IEEE Access, vol. 10	A. Malik et al.,	UADFV, DF-TIMIT-(LQ), DF-TIMIT-(HQ)	DNN models are used to analyze spatial characteristics, boost detection efficacy and improve the generalization capacity to detect deepfakes. These methods are entirely data-driven.	Lack of datasets, unknown types of attacks on media, temporal aggregation and unlabeled data.

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
33	A Heterogeneous Feature Ensemble Learning based Deepfake Detection Method	2022	IEEE International Conference on Communications	J. Zhang et al.,	CelebA	extracting heterogeneous features integrating them into a ensemble feature vector by a flatten process, and finally sending the feature vector to a back-propagation	Not robust enough to work as general deepfake detector.
34	Comparative Analysis on Different DeepFake Detection Methods and Semi Supervised GAN Architecture for Deepfake Detection	2022	2022 Sixth International Conference on I-SMAC	J. John and B. V. Sherif	Flicker-Faces-HQ(FFHQ) dataset.	Semi Supervised Generative Adversial Network	Unable to identify the hidden features.

S.NO	Title	Year Of Publication	Journal Name	Author(s)	Dataset(s)	Methodology	Limitation(s)
35	Deepfake Generation, Detection and Datasets: a Rapid-review	2021	2021 First International Conference on Advances in Computing and Future Communication Technologies	A. Verma et al.,	FaceForensics ++, DFDC datasets, UADFV, Deepfake TIMIT, Celeb-DF.	CNN, RNN, LSTM, GAN are used.	May not detect the new features that are adding upon the data.
36	Deepfake Detection using Inception-ResnetV2	2022	2022 15th International Conference on Information Security and Cryptography	A. KoÇak and M. Alkan	DFDC dataset	ResnetV2 is used.	Doesn’t consider the inconsistencies in blinking rate.

2.1 LIMITATIONS OF EXISTING SYSTEM

The existing system cannot identify hidden features in the images such as moles and scars present on the face and the length of eyelashes, eyebrows etc. This system does not take into consideration temporal inconsistencies across frames such as any sudden changes occur in the image it unable to classify the image. It only considers the track of blinking for detection and not the frequency of blinking which results reducing the accuracy of the system. It does not encode the position and orientation of an image which requires lots of data for training. There isn't a significant improvement in performance when contrasted with other similar techniques.

2.2 PROPOSED SYSTEM

One of the Proposed methods uses simple supervised classifier after analyzing the frequency domain. The steps included for the preprocessing the data are Frequency Domain Analysis, Discrete Fourier Transform, Azimuthal Average, Classifier Algorithm, Support Vector Machine. This proposed system does not include nor requires tremendous data which is an advantageous property for those situations that experience the ill effects of information shortage.

The other method uses data normalization as a preprocessing step in order to extract the essential features from the image. In this method, DenseNets are used as a classification algorithm. This method requires huge amounts of data to be trained due to the classifier being a Deep Learning Classifier.

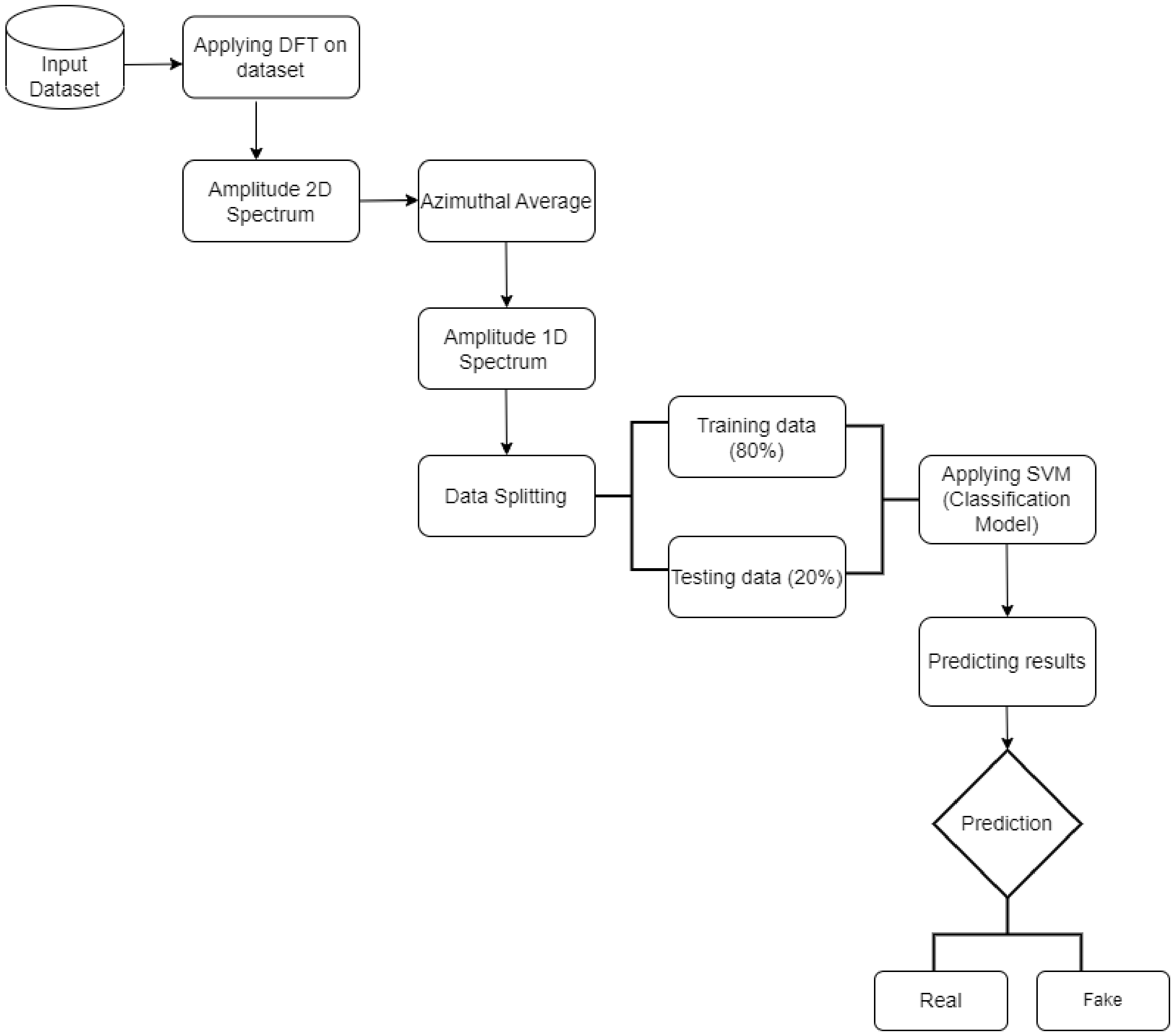


Figure 3: *Proposed System of Support Vector Classifier*

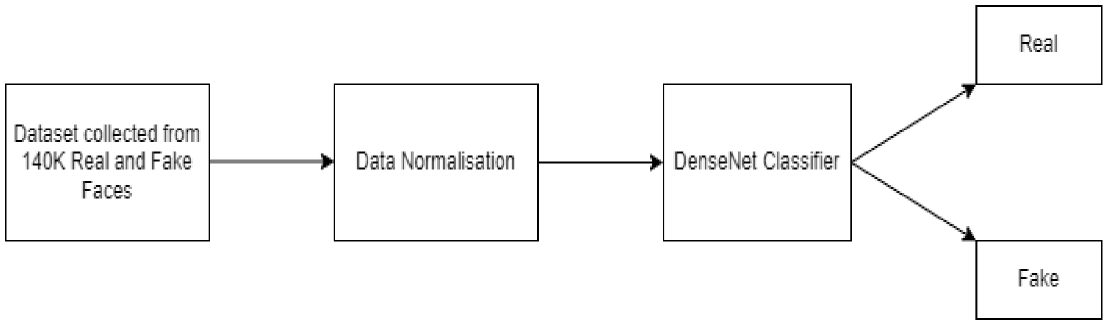


Figure 4: *Proposed System of DenseNet Classifier*

To examine whether the photographs in the proposed system are false or real, wetook into account the dataset known as CelebA. Simple supervised and unsupervised classifier frequency analysis of images was employed in this system. Support vector machines (SVM) are employed in this system to categorise fake and authentic pictures. The SVM method is built using kernel to change an input data space into the necessary structure. Given that data may be separated linearly, we employed a linear kernel in the suggested system. This technique has high accuracy.

This project is divided into three modules:

- **Module-1:** Data Collection

  In this module, the dataset was collected from the CelebA website for the Support Vector Classifier. The dataset consists of both real and fake data. The real image data consists of one thousand real images gathered from the CelebA website. The fake data contains one thousand computer generated images.

  The DenseNet Classifier uses 140K Real and Fake Faces a Kaggle dataset, containing 70 thousand real faces and 70 thousand fake faces.

- **Module-2:** Data Pre-processing

  Various preprocessing techniques were used in this module to prepare the data for classification. This module employs the use of image processing to prepare the images in the dataset before they are sent to train the Support Vector Machines model.

  The dataset is rescaled in the case of the DenseNet classifier.

- **Module-3:** Classifier Building

  In this module, we will use Support Vector Machines to train the model to distinguish between actual and false images. Linear, Gaussian RBF and Polynomial Kernels were used to train the model.

  The DenseNet121 pretrained model of the tenserflow library of python was used to train the model. None of the pretrained weights were used to build the model.

3. METHODOLOGY

3.1 SYSTEM OVERVIEW

The aim of the system is to differentiate between real and fake images generated by deep learning techniques. The algorithm of Support Vector Machines, a machine learning algorithm is being employed to achieve this. The dataset that is being used is the CelebA dataset collected from the CelebA website comprising of over 56 thousand images that are being used for training the model.

3.2ARCHITECTURE DIAGRAM

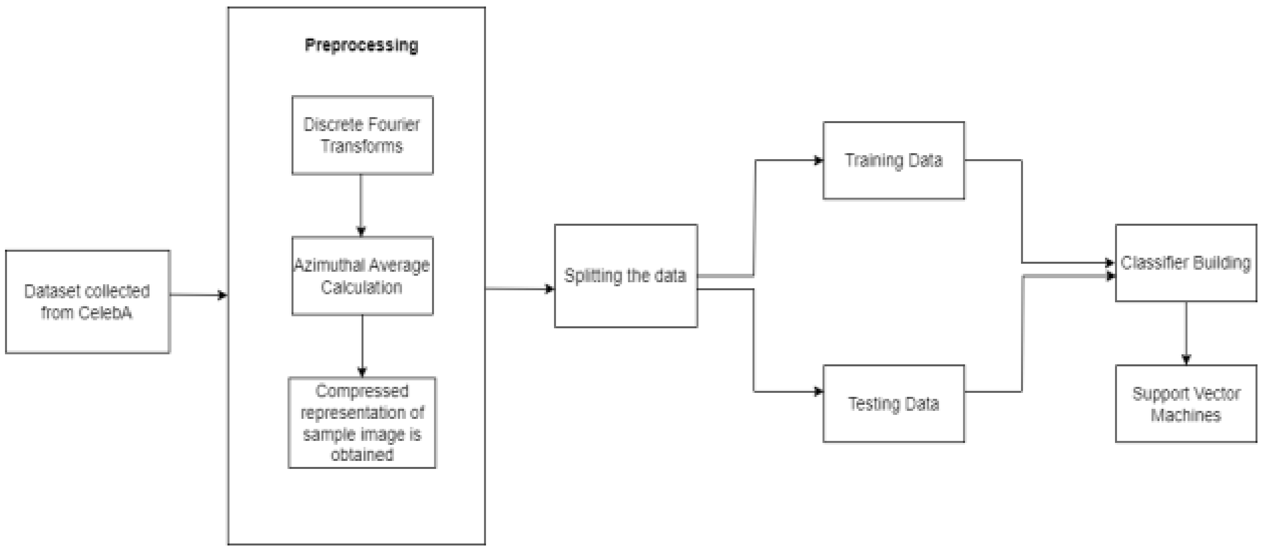

Dataset collected from CelebA
Preprocessing
Discrete Fourier Transforms
Azimuthal Average Calculation
Compressed representation of sample image is obtained
Splitting the data
Training Data
Testing Data
Classifier Building
Support Vector Machines

Figure 5: *Architecture Diagram*

The suggested architectural diagram is a graphical representation of the physical implementation of several software system components. This is a formal description and representation of the system, which is built up to support examination ofthe structures and behaviors of its components.

3.3IDENTIFICATION OF MODULES

3.3.1 Data Collection

There are over a thousand celebrity images of resolution 1024*1024 with 722 variations in facial attribute in CelebA Attributes Dataset as real images and another thousand DeepFaked images. 80% of the data is utilized for training, and the unused 20% is used for testing.

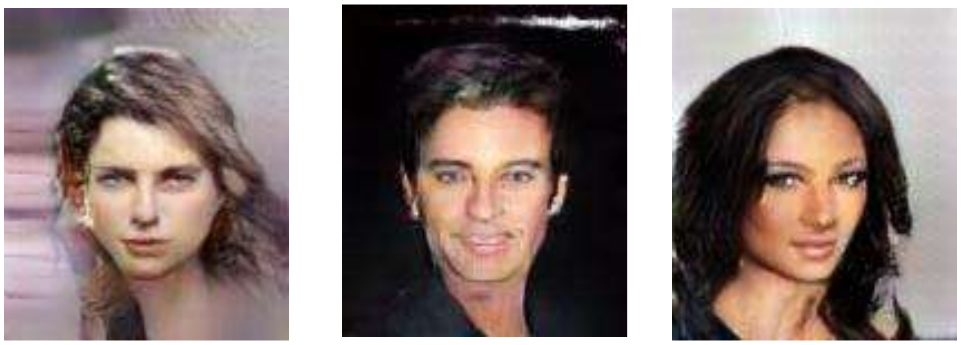


Figure 6*: Fake Data*

Figure 7*: Real Images*

3.3.2 Data Pre-processing

The actions taken to prepare pictures before they are used by model training and inference are referred to as image preprocessing. This covers resizing, orienting, and colour adjustments, among other things.

Data preprocessing for our project has been carried out in the following major steps which are steps involved in Frequency Domain Analysis: Discrete Fourier Transforms and Azimuthal Average.

3.3.2.1 Frequency Domain Analysis: The image is converted from the spatial domain to the frequency domain. A digital picture is transformed from the spatial domain to the frequency domain in the frequency domain. This is illustrated in Figure 4. Image filtering in the frequency domain is used to improve images fora specific purpose.

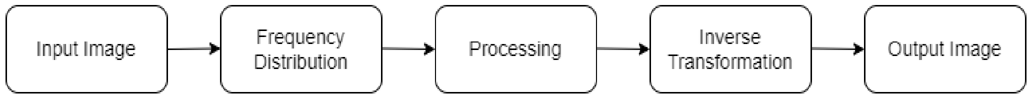


Figure 8: *Frequency Domain Analysis*

a) **Discrete Fourier Transform:** Images can be processed using the Fourier transformation. In order to break down a picture into sine and cosine components, it is employed. The output image is represented in the Fourier or frequency domain, while the input image is a spatial domain. Numerous applications, including picture reduction and filtering, use the Fourier transformation. A signal’s frequency spectrum can be calculated using Discrete Fourier Transforms.

The input and output values are discrete samples, whichmake it convenient for computer manipulation. The formula used is,

$$X_{k,l} = \sum_{n=0}^{N-1}\sum_{0}^{M-1} x_{n,m}.e^{-\frac{i2\pi}{N}kn}.e^{-\frac{i2\pi}{M}lm} \tag{1}$$

b) **Azimuthal Average:** When working with photos, the output is presented as 2D data. Fast Fourier Transforms are represented effectively by applying a magnitude spectrum azimuthal average. Azimuthal average is utilized to produce a 1D representation of the magnitude spectrum of the Discrete Fourier Transform (DFT).

3.3.3 Classification

The Classification method, a Supervised Learning approach, is used to categorize fresh observations in light of training data. In classification, a program makes use of the dataset or observations that are provided to learn how to interpret freshobservations into various classes or groups. A classifier algorithm is an important algorithm in the machine learning which is used to arrange or classify information into at least one of a set of a classes. In this

Support Vector Machine: Support vector machines are a type of supervised learning algorithms for classification, regression, and detecting outliers. SVMs vary from other classification methods in that they select the decision boundary that optimizes the distance from the nearest data points for all classes. The maximum margin classifier or maximum margin hyper plane is the decisionboundary generated by SVMs. A simple linear SVM classifier connects twoclasses by drawing a straight line between them. That is, all of the data points on one side of the line will be assigned to a category, while the data points located onthe other side of the line will be classified to a different class.

SVM Kernels: A kernel function is a way for taking input data and transforming it into the needed form of processing data. The term "kernel" refers to a collection of mathematical functions used in Support Vector Machine to provide a window to alter data.

In general, the Kernel Function modifies the training set of data so that a non-linear decision surface can be transformed to a linear equation in a larger number of dimension spaces. It essentially returns the inner product of two points in a standard feature dimension. The Kernel Functions used in the model are Linear, Gaussian RBF and Polynomial Functions.

(i) **Linear Kernel Function:** When the data is linearly separable, that is, it can be separated using a single line, a Linear Kernel is applied. It is one of the most commonly used kernels. It is typically employed when a data set has a large number of features. Text Classification is one of the applications of Linear Kernel. It is calculated as follows:

$$K(x_i,x_j) = (x_i*x_j) \qquad (2)$$

(ii) **Gaussian Radial Basis Function Kernel:** It is typically used with non-linear data. When there is no prior information about the data, it is applied to perform transformation. It improves the transformation by using the radial basis approach. It is computed as follows:

$$K(x, y) = e^{-\|x-y\|2/\sigma 2} \qquad (3)$$

(iii) **Polynomial Function:** It is a more general description of the linear kernel. It shows the similarity of vectors in the training set of data in a feature space over polynomials of the kernel's original variables. It is determined as follows:

$$K(x, y) = (x*y+1)^d \qquad (4)$$

DenseNets: DenseNets, short for Dense Convolutional Networks, are a type of neural network architecture that were introduced in 2016 by Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. The key innovation in DenseNets is the use of "dense blocks," which are made up of multiple convolutional layers that are connected in a dense fashion, with each layer receiving inputs from all preceding layers. By doing so, DenseNets can leverage the features learned by earlier layers more effectively and reduce the number of parameters required to achieve high accuracy on a given task.

DenseNets are deep neural networks that improve upon traditional convolutional neural networks (CNNs) by establishing dense connections between layers. In a DenseNet, each layer receives inputs from all previous layers and passes its outputs to all subsequent layers. This allows information to be efficiently propagated through the network and reduces the vanishing gradient problem, which can hinder training of very deep networks.

DenseNets are composed of dense blocks, which contain a series of convolutional layers followed by a concatenation operation that aggregates the outputs from all previous layers. The dense blocks are connected by transition layers, which reduce the dimensionality of the feature maps before passing them on to the next dense block.

DenseNets have achieved state-of-the-art performance on a variety of computer vision tasks, such as image classification, object detection, and semantic segmentation. They are particularly useful when working with limited data and can be trained more efficiently than traditional CNNs, making them an attractive choice for real-world applications.

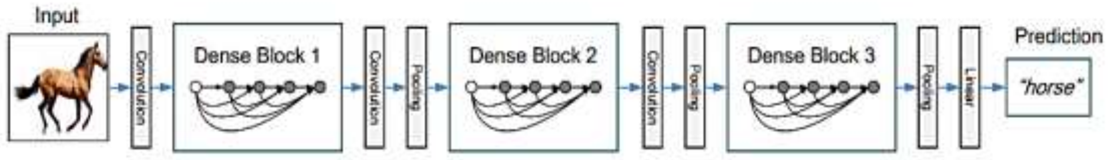


Figure 9: *Working of DenseNets [37]*

Dense connectivity is a type of neural network architecture where every layer is directly connected to all subsequent layers in the network. This means that the output of each layer is passed as input to every subsequent layer, resulting in a dense network of connections.

In a dense network, the ℓ-th layer receives the feature-maps of all preceding layers, denoted as $x_0, x_1, \ldots, x_{\ell-1}$, as input. The output of the ℓ-th layer is then calculated using a function Hℓ that takes the concatenation of the feature-maps of all preceding layers as input:

$$x_\ell = H_\ell([x_0, x_1, \ldots, x_{\ell-1}]), \qquad (5)$$

where [x0, x1, …, xℓ−1] refers to the concatenation of the feature-maps produced in layers 0 to ℓ−1. The multiple inputs of $H_\ell(\cdot)$ are concatenated into a single tensor for ease of implementation.

Because of its dense connectivity, this network architecture is referred to as a Dense Convolutional Network (DenseNet). Dense connectivity can improve the information flow between layers and facilitate feature reuse, which can lead to better performance on various tasks such as image classification, object detection, and segmentation [31].

DenseNet in our project has been implemented using the DenseNet121 pretrained model of tenserflow library of python. However, the pretrained weights haven't been considered leading to us training the model from scratch.

4. REQUIREMENTS

Each computer system necessitates the presence of specific hardwarecomponents or software resources. System requirements might include both hardware and software. Prerequisites are another term for these system needs. The demand for system requirements grows in tandem with the demand for increased CPU power and resources in successive versions.

4.1 SOFTWARE REQUIREMENTS

Software requirements are concerned with identifying the software capabilities and dependencies that should be installed onto a computer in order for a program to run as efficiently as desirable. Given that they are not usually included in the program installation package, these requirements must be installed separately before the package can be installed. Languages, libraries, packages, and operating systems, as well as other tools for project development, are all part of the software requirements.

Table 2: *Software Requirements*

SOFTWARE USED	VERSION
Jupyter Notebook	6.4.5
Google Colab	1.0.0
Anaconda Framework	4.10.3
Operating System(s)	Windows 10, Windows 11

Table 3: *Packages Used*

S. No	Package	Description
1	numpy	numpy package is fast and accurate, used for scientific computing in python.
2	cv2	cv2 is a Python computer vision library for image and video processing, such as object detection, tracking, and facial recognition.
3	glob	glob is a Python module that allows users to search for files with specific patterns in a directory using pathname pattern matching.
4	pickle	Pickle is a Python module that allows users to save and load complex data structures through object serialisation and deserialization.
5	scipy	scipy is a Python scientific computing library used for numerical integration, optimization, signal processing, linear algebra, and other purposes.
6	matplotlib	It is a cross platform, data visualization and graphical plotting library for python and its numerical extension
7	sklearn	Scikit-learn is an open source learn is an open-source Python package functionality supporting supervised and unsupervised learning.
8	seaborn	Seaborn is a matplotlib-based Python data visualization package. It has a high-level interface for creating visually appealing and instructive statistics visuals.
9	tensorflow	It is a python-based package which provides implementing different machine learning and AI tools such as neural layers.
10	keras	Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It consists of all the neural network layers and other packages used for deep learning computations.
11	json	JSON is a lightweight data interchange format used in web development for data serialisation and communication between client-server applications.
12	django	Django is a high-level Python web framework that promotes rapid development and simple, practical design
13	pil	PIL (Python Imaging Library) is a Python module that handles image processing tasks like opening, manipulating, and saving images.

a. Python

A high-level, all-purpose programming language is Python. Code readability is prioritised in its design philosophy, which makes heavy use of indentation. Python uses garbage collection and has dynamic typing. It combines modules, exceptions, dynamic typing, high level dynamic data types, and classes. It supports a variety of programming paradigms, including structured, object-oriented, and functional programming.

b. Anaconda Framework

Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment. The package management system, conda,manages the versions of packages in Anaconda. Conda examines the current environment before completing an installation to prevent causing disruption to other frameworks and packages.

c. Jupyter Notebook

The first web application for producing and sharing computational documents was called a Jupyter Notebook. It provides a straightforward, efficient, document-focused experience.

d. Google Colab

It is a free cloud-based platform that lets users write and run Python code through a web browser. It comes pre-installed with necessary libraries for machine learning tasks and provides access to powerful GPUs and TPUs to speed up training times. It's a convenient and efficient way to conduct machine learning experiments.

4.2 HARDWARE REQUIREMENTS

The most common set of parameters provided by every operating system or software package is the physical computer resources, sometimes known as hardware. The next subsections go through the various components of the hardware requirements.

Table 4: *Hardware Requirements*

HARDWARE REQUIREMENTS	**CONFIGURATION**
Processor	Intel® Core™ i5
System Type	64-bit operating system, x64-based processor
RAM	8.00 GB

5. DESIGN

5.1 INTRODUCTION

Unified Modelling Language (UML) is used to represent the software in a graphical format, that is it provides a standard way to visualize how a system is designed. Good UML design is followed for a better and precise software output. The UML provides different constructs for specifying, visualizing, constructing and documenting the artifacts of software systems. UML diagrams are used to better understand the system, to maintain the document information about the system and emphasizes on the roles, actors, actions, actions and process. The UML diagrams considered are,

• Use-case Diagram

• Class Diagram

• Sequence Diagram

Relationships

The relationships among different entities in the following diagrams are,

• **Association -** This relationship tells how two entities interact with each other.

• **Dependency -** An entity is dependent on another if the change of that is reflecting the other.

• **Aggregation -** It is a special kind of association which exhibits part-of relationship.

• **Generalization -** This relationship describes the parent- child relationship among different entities.

• **Realization -** It is a kind of relationship in which one thing specifies the behaviour or a responsibility to be carried out, and the other thing carries out that behaviour.

The relationships are represented as follows,

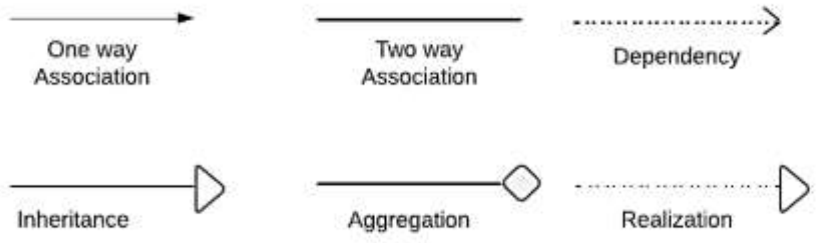


Figure 10: *Relationships in UML*

5.2 UML DIAGRAMS

Unified Modeling Language (UML) diagrams are a standardized way of visualizing and modeling software systems. They include a variety of diagrams such as class diagrams, sequence diagrams, and use case diagrams that allow developers and stakeholders to better understand the architecture and behavior of a system. UML diagrams are an essential tool for software development teams to communicate and collaborate effectively.

5.2.1 USE-CASE DIAGRAMS

Use case diagrams capture the behaviour of a system and help to emphasize the requirements of the system. It describes the high-level functionalities of the system. It consists of the following artifacts,

- Actor
- Use case
- Secondary Actor and Use case Boundary

Actor

A role of a user that interacts with the system you're modeling is represented by an actor. The primary actor in this project is Admin. The admin has full access to the system such that all the operations are executed under the surveillance of the admin.

Use-Case

A use case is a function that a system does to help the user achieve their goal. They are 10 use-cases which has their own functionality that can be useful in accomplishing the goals. The use cases Request for Uploading image are use cases which extend and includes other use cases.

Secondary Actor

Actors who are not directly interacting with the system are secondary actors and are placed on the right side of the use case boundary. The secondary actor in the system is User. The User interacts with the system with limited access only. The user can upload image once after his registration and model will analyze whether the image is real or fake.

Here, the use case diagram consists of different use cases, actors and secondary actors. The user can upload image as input while the admin is responsible for extracting, pre-processing, and feeding data into the model and obtaining the results. The admin also takes care when there are abnormal situations raised by different users accessing the system. The system should predict that the image is given by user as input is realistic or fake image. So, this can be accomplished by visualize results use case. The prediction, training model, pre-processing and data extraction is achieved using the Data Extraction, Data Pre-processing, Feeding models and other use cases in the system.

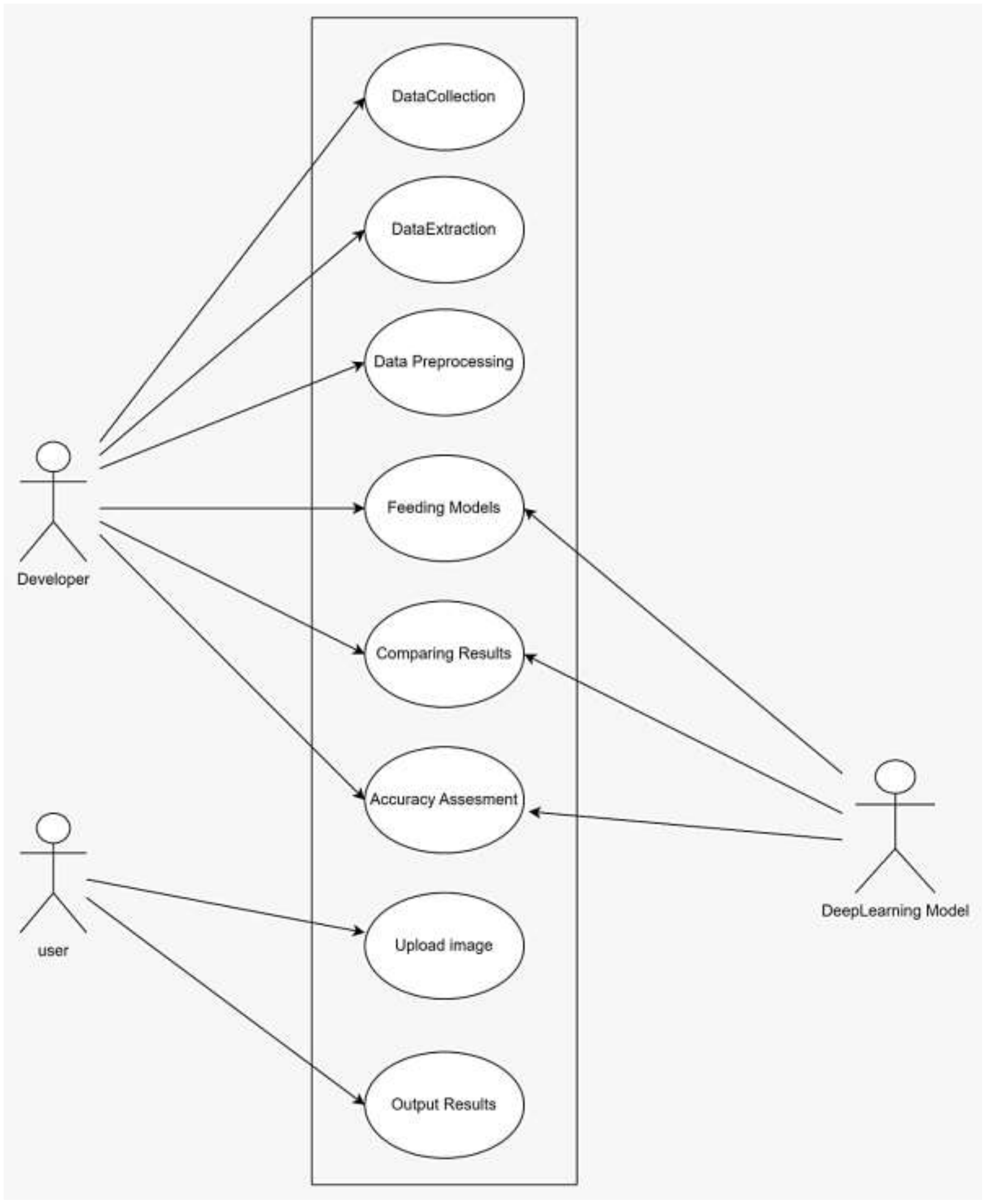


Figure 11: *Use Case Diagram*

5.2.2 CLASS DIAGRAM

Class diagram is a static overview of the system used to highlight the important aspects as well as executables in the system. These are the only diagrams which can be mapped with the object-oriented systems. These diagrams show the responsibilities of a class and relationships among different classes in the system.

Class

The class represents similar objects and consists of name, responsibilities and attributes. The classes considered in the project are User, admin, Dataset, Data collection, Data Pre-processing, Training Model, External Resources and Webpage. These are shown diagrammatically in the figure below

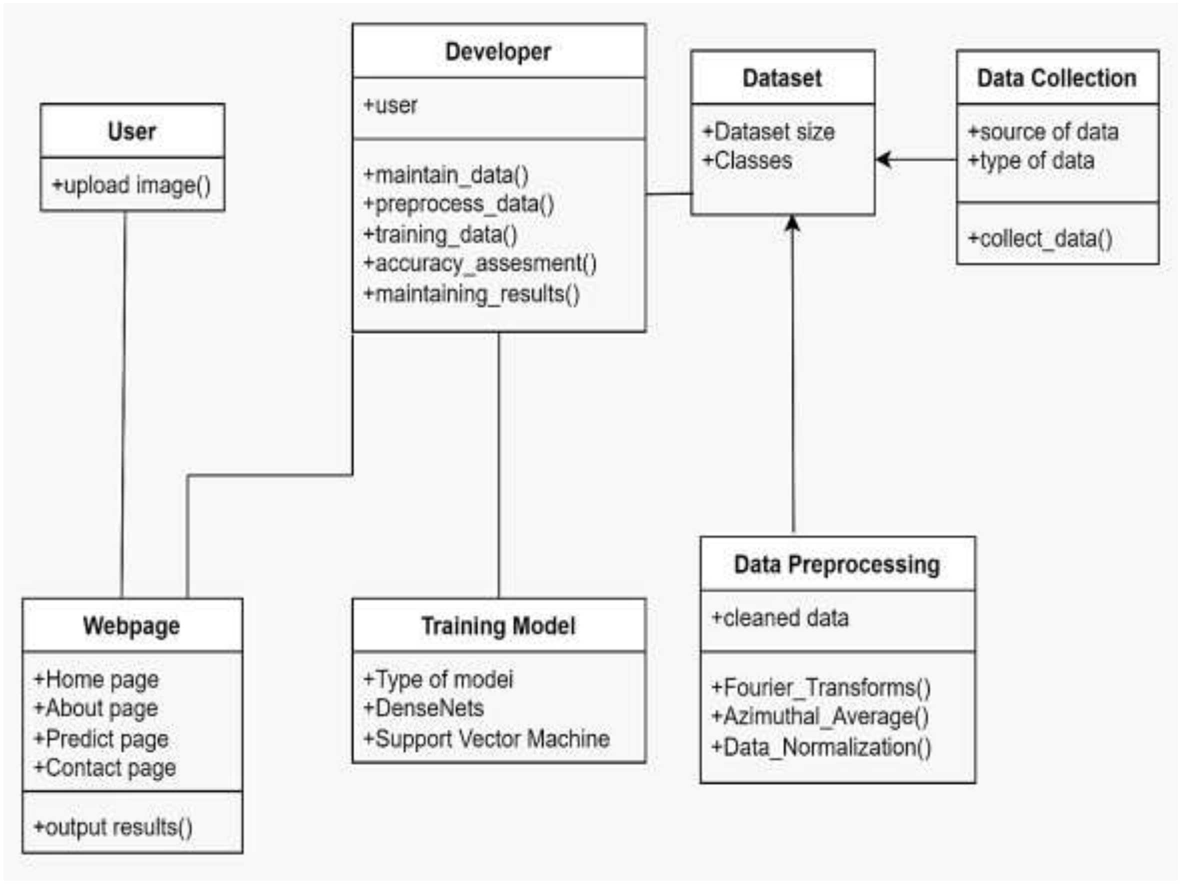


Figure 12: *Class Diagram*

5.2.3 SEQUENCE DIAGRAM

The sequence diagram, also known as an event diagram, depicts the flow of messages through the system. It aids in the visualization of a variety of dynamic scenarios. It depicts communication between any two lifelines as a time-ordered series of events, as if these lifelines were present at the same moment. The message flow is represented by a vertical dotted line that extends across the bottom of the page in UML, whereas the lifeline is represented by a vertical bar. It encompasses both iterations and branching.

A sequence diagram in Fig. simply displays the order in which things interact, or the order in which these interactions occur. A sequence diagram can also be referred to as an event diagram or an event scenario. Sequence diagrams show how and in what order the components of a system work together.

Notations of Sequence Diagram are,

Lifeline - A lifeline represents an individual participant in the sequence diagram. It is at the very top of the diagram. The lifeline for different participants starts at different times.

Actor - An actor is a character who interacts with the subject and plays a part. It isn't covered by the system's capabilities. It depicts a role in which human users interact with external devices or subjects. Here the user and admin are the actors.

Message - Message denotes the interactions between the messages. They are in the sequential order in the timeline. Here there are 10 messages which are both call and return messages with respective to the participants in the system.

Call message

By defining a specific communication between the interaction's lifelines, it shows that the target lifeline has invoked an operation. Here the call messages are Data extraction, Data cleaning, feed model and results.

Return Message

It specifies a specific communication between the interaction lifelines, which reflect the flow of data from the receiver of the associated caller message. The return messages are extracted data, cleaned data, embedding matrix, result analysis and display results.

The sequence of messages is shown in the figure. The system here represents the GUI that the user interacts with and also predict whether the image is real or fake.

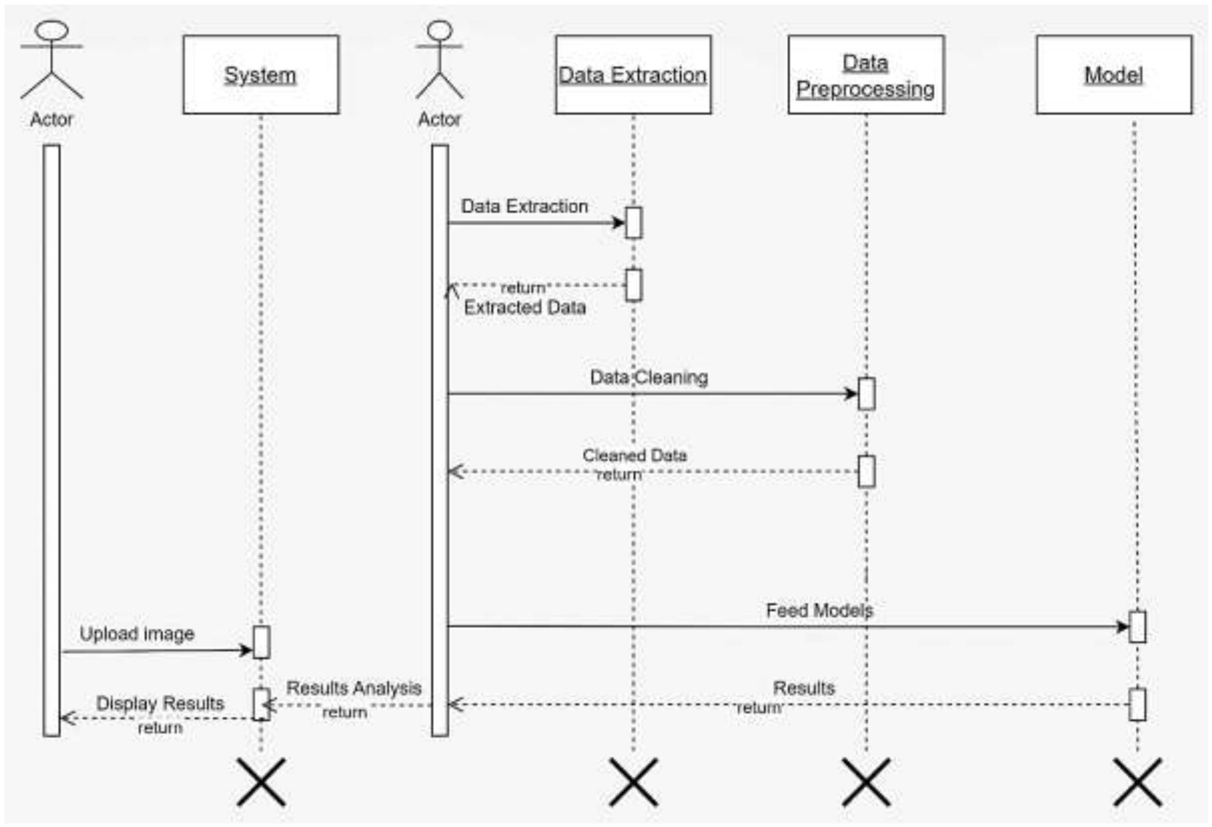


***Figure 13**: Sequence Diagram*

6. IMPLEMENTATION AND RESULTS

INTRODUCTION

The implementation stage of an ML project involves turning the developed algorithm into a working model that can be integrated into a larger system. This stage includes creating and testing the model, fine-tuning its parameters, and deploying it to a production environment. The implementation stage is critical to the success of an ML project, as it determines whether the model can function effectively in real-world scenarios.

6.1 SOFTWARE DESIGN

6.2.1 Front-End Design

HTML

HTML (Hypertext Markup Language) is the standard language used to create web pages. It defines the structure and content of web pages using tags, which allow developers to define headings, paragraphs, links, images, and other elements. HTML is a fundamental building block for web development and is essential for creating interactive, visually appealing websites.

CSS

CSS (Cascading Style Sheets) is a language used to style and format web pages created with HTML. It allows developers to define the visual appearance of HTML elements, including colors, fonts, layout, and other design elements. CSS is essential for creating visually appealing and engaging websites that provide an exceptional user experience.

Django

Django is a high-level Python web framework that helps developers build web applications quickly and efficiently. It includes a range of built-in tools and modules, such as an ORM (Object-Relational Mapping) system, a templating engine, and an administration interface. Django follows the Model-View-Controller (MVC) architectural pattern and is highly scalable and secure.

6.2.2 Back-End Design

a. Data Collection

Data collection is the process of gathering information and data from various sources. It is a critical step in the data analysis process and involves identifying relevant data sources, selecting appropriate data collection methods, and ensuring the accuracy and completeness of the collected data. The collected data can be used for a variety of purposes, including research, analysis, and decision-making. Effective data collection requires careful planning and attention to detail to ensure that the resulting data is accurate, reliable, and relevant to the problem or question being addressed.

b. Data Preprocessing

Data preprocessing of image data involves several techniques, including resizing, cropping, normalization, and augmentation. These techniques are used to clean and transform raw image data into a format that is suitable for use in machine learning models. Preprocessing techniques help to reduce noise, improve image quality, and enhance the features of the image data, resulting in more accurate and reliable machine learning models.

c. Classification

Classification is a type of machine learning technique that involves categorizing data into distinct classes or groups based on specific characteristics or attributes. It is a supervised learning technique that relies on labeled training data to develop a model that can classify new, unlabeled data accurately. Classification is used in a variety of applications, including image recognition, spam filtering, and sentiment analysis.

6.2IMPLEMENTATION

To implement our project, we employed Django to handle the front-end, while Jupyter Notebook and Google Colab were utilized for the back-end programming.

6.2.1 Front-End Code

The front end of our project uses 4 html pages. The **home** page is the landing page of our project followed by the **about** page which gives a brief description of the implemented models. The **predict** page takes an image as an input in jpeg format and outputs real if the input image is real and fake if the real image is fake. The **contact** page provides the contact information of the developers and a form to contact the developers.

home.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">

<head>
  <title>DeepFake Detection | Home</title>

</head>
<body>
  <nav>
    <ul>
      <li><a href="{% url 'home' %}">Home</a></li>
      <li><a href="{% url 'about' %}">About</a></li>
      <li><a href="{% url 'index' %}">Predict</a></li>
      <li><a href="{% url 'contact' %}">Contact</a></li>
    </ul>
  </nav>

    <section id="about" class="about">
      <h1>DeepFake Detection</h1>
      <div class="container">

        <div class="row position-relative">
          <div class="col-lg-7">
            <h2>About Us</h2>
            <div class="our-story">
              <h4>Est 2022</h4>
              <h3>Our Story</h3>
              <p>DeepFakes are fake computer-generated images created
                using the algorithms such as Generative Adversarial
            Networks (GAN) and Autoencoders.
            DeepFakes are already a severe issue because of their
              great capacity to increasingly resemble people
            There are various dangerous circumstances in which
              DeepFakes might be used to harm our society and the people
              in it, hence research into identifying these manipulations is
              critical.

                processing to pre-process the images to find the discerning
                features between the fake and real images. The SVM
                classifier is then trained on these discerning characteristics
                to distinguish between a fake and a real image.</p>

        </div>
      </div>

    </div>

  </div>
</section>

<section id="team" class="team">
  <div class="container" data-aos="fade-up">

    <div class="section-header">

      <h2>Our Team</h2>

    </div>

    <div class="row gy-5">

      <div class="col-lg-4 col-md-6 member" data-aos="fade-up" data-aos-delay="100">
        <div class="member-img">
          <img src="mem-img" class="img-fluid" alt="">

        </div>
        <div class="member-info text-center">
          <h4>Dr. N. Sharmili</h4>
          <span>Guide</span>


        </div>
      </div><!-- End Team Member -->
    <div class="row gy-5">

      <div class="col-lg-4 col-md-6 member" data-aos="fade-up" data-aos-delay="100">
        <div class="member-img">
```

```
          <img src="{% static 'adtl.jpg' %}" class="img-fluid" alt="">

        </div>
        <div class="member-info text-center">
          <h4>Anandita Dakshayani</h4>
          <span>Team Leader</span>

        </div>
      </div><!-- End Team Member -->

      <div class="col-lg-4 col-md-6 member" data-aos="fade-up" data-aos-delay="200">
        <div class="member-img">
          <img src="assets/img/team/team-2.jpg" class="img-fluid" alt="">

        </div>
        <div class="member-info text-center">
          <h4>Eswari Chirukuri</h4>

        </div>
      </div><!-- End Team Member -->

      <div class="col-lg-4 col-md-6 member" data-aos="fade-up" data-aos-delay="300">
        <div class="member-img">
          <img src="assets/img/team/team-3.jpg" class="img-fluid" alt="">

        </div>
        <div class="member-info text-center">
          <h4>Amrutha Ariselli</h4>

        </div>
      </div><!-- End Team Member -->


    </div>

  </div>
</section><!-- End Our Team Section -->


</body>

</html>
```

about.html

```
<!DOCTYPE html>
{% load static%}
<html>
<head>
        <title>DeepFake Detection | About</title>
</head>
  <body>
    <nav>
      <ul>
        <li><a href="{% url 'home' %}">Home</a></li>
        <li><a href="{% url 'about' %}">About</a></li>
        <li><a href="{% url 'index' %}">Predict</a></li>
        <li><a href="{% url 'contact' %}">Contact</a></li>
      </ul>
     </nav>
<h1>DeepFake Detection Model Performance and Model Comparison</h1>
<h2>Model Performance</h2>
<div class="flip-card-container">
  <div class="flip-card">
   <div class="flip-card-inner">
    <div class="flip-card-front">
      <h1 class="title">Support Vector Machine - Linear Kernel</h1>
      <p>Model Performance</p>
    </div>
    <div class="flip-card-back">
      <div class="box-1">
        <img class="image" src="{%static 'linear.jpg'%}">
      </div>
      <div class="box-2">
        <img class="image-2" src="{%static 'linearcr.png'%}">
      </div>
    </div>
   </div>
  </div>

  <div class="flip-card">
   <div class="flip-card-inner">
    <div class="flip-card-front">
      <h1 class="title">Support Vector Machine - Gaussian RBF Kernel</h1>
      <p>Model Performance</p>
    </div>
    <div class="flip-card-back">
      <div class="box-1">
        <img class="image" src="{%static 'rbf.jpg'%}">
```

```
          </div>
          <div class="box-2">
            <img class="image-2" src="{%static 'rbfcr.png'%}">
          </div>
        </div>
      </div>
    </div>
  </div>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br>
  <h2>Model Performance</h2>
  <div class="flip-card-container">

    <div class="flip-card">
      <div class="flip-card-inner">
        <div class="flip-card-front">
          <h1 class="title">Support Vector Machine - Polynomial Kernel</h1>
          <p>Model Performance</p>
        </div>
        <div class="flip-card-back">
          <div class="box-1">
            <img class="image" src="{%static 'poly.jpg'%}">

          </div>
          <div class="box-2">
            <img class="image-2" src="{%static 'polynomialcr.png'%}">

          </div>
        </div>
      </div>
    </div>
    <div class="flip-card">
      <div class="flip-card-inner">
        <div class="flip-card-front">
          <h1 class="title">DenseNet</h1>
          <p>Model Performance</p>
        </div>
        <div class="flip-card-back">
          <div class="box-1">
            <img class="image" src="{%static 'dense.jpg'%}">

          </div>
          <div class="box-2">
            <img class="image-2" src="{%static 'densecr.png'%}">

          </div>
        </div>
      </div>
    </div>
  </div>
  <br><br>

  <div class="table-container">
    <h2> Model Comparision</h2>
    <table>
      <thead>
        <tr>
          <th>Model</th>
          <th>Accuracy Score</th>
          <th>Precision Score</th>
          <th>Recall Score</th>
          <th>RMSE Score</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td> SVM - Linear Kernel</td>
          <td> 0.950</td>
          <td> 0.984</td>
          <td> 0.950</td>
          <td>0.223</td>
        </tr>
        <tr>
          <td> SVM - Gaussian RBF Kernel</td>
          <td>0.992</td>
          <td id="high">1.00</td>
          <td>0.992</td>
          <td>0.086</td>
        </tr>
        <tr>
          <td> SVM - Polynomial Kernel</td>
          <td id="high">1.00</td>
          <td id="high">1.00</td>
          <td id="high">1.00</td>
          <td id="high">0.00</td>
        </tr>
        <tr>
          <td> DenseNet </td>
          <td>0.98</td>
          <td>0.97</td>
          <td>0.98</td>
          <td>0.135</td>
        </tr>
      </tbody>
    </table>
    </div>
  </body>
  </html>
```

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <link href= "predict/template/favicon.jpg" rel="icon">
    <title>DeepFake Detection</title>
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="{% url 'home' %}">Home</a></li>
        <li><a href="{% url 'about' %}">About</a></li>
        <li><a href="{% url 'index' %}">Predict</a></li>
        <li><a href="{% url 'contact' %}">Contact</a></li>
      </ul>
    </nav>
    <div class="container">
    <h1>DeepFake Detection Project</h1>
     {% if captured_image_url %}
    <img src="{{ captured_image_url }}" alt="Captured Image">
      {% endif %}

    <form method="POST" class="form" enctype="multipart/form-data">
      {% csrf_token %}
      <span class="form-title">Upload your file</span>
      <p class="form-paragraph">
          File should be an image in .jpg format
        </p>
       <label for="file-input" class="drop-container">
      <span class="drop-title">Drop files here</span>
      or
      <input type="file" name="image" accept=".jpg" required="" id="file-input">
       <button type="button" class="btn" id="capture-btn" capture>Capture from Camera</button>
        <button type="button" class="btn" id="save-btn">Save</button>
        <input type="hidden" name="image_data" id="image-data">
    </label>
    <br><br>

      <input style= "display:block; margin: auto" class="btn"type="submit" value="Predict"/>
    </form>
    <br><br>
  </div>
```

```
    <div class="imgcontainer">
    {% if image_data_url %}
    <img class="img" src="{{ image_data_url }}">
{% endif %}

<br>
<br>

    {% if prediction %}
      <p>{{ prediction }}</p>
    {% endif %}
  </div>

  <script>
    function captureImage() {
  navigator.mediaDevices.getUserMedia({video: true}).then((stream) => {
      let video = document.createElement('video');
      video.srcObject = stream;
      video.onloadedmetadata = () => {
          let canvas = document.createElement('canvas');
          canvas.width = video.videoWidth;
          canvas.height = video.videoHeight;
          canvas.getContext('2d').drawImage(video, 0, 0, canvas.width, canvas.height);
          let imageDataUrl = canvas.toDataURL('image/jpeg');  // save as JPEG
          document.querySelector('#image-data').value = imageDataUrl;
          document.querySelector('#file-input').value = '';
          video.srcObject.getTracks()[0].stop();
      };
      video.play();
  }).catch((err) => {
      console.log(err);
  });
}

</script>
<div class="canvas-container">
  <canvas id="canvas"></canvas>
</div>
<script>
// get references to the DOM elements
const captureBtn = document.getElementById('capture-btn');
const saveBtn = document.getElementById('save-btn');
const canvas = document.getElementById('canvas');

// set the canvas dimensions
canvas.width = 640;
canvas.height = 480;
```

```
let imageData = null;

// add click event listener to the capture button
captureBtn.addEventListener('click', async () => {
  // get access to the camera
  const stream = await navigator.mediaDevices.getUserMedia({ video: true });

  // create a video element to display the camera stream
  const video = document.createElement('video');
  video.srcObject = stream;
  video.autoplay = true;

  // wait for the video to start playing
  await new Promise(resolve => video.addEventListener('playing', resolve));

  // draw the current video frame to the canvas
  const context = canvas.getContext('2d');
  context.drawImage(video, 0, 0, canvas.width, canvas.height);

  // stop the camera stream
  stream.getTracks().forEach(track => track.stop());

  // remove the video element
  video.remove();

  // save the canvas image data
  imageData = canvas.toDataURL();
});

// add click event listener to the save button
saveBtn.addEventListener('click', () => {
  if (imageData) {
    // create a temporary link element to download the image
    const link = document.createElement('a');
    link.download = 'image.jpeg';
    link.href = imageData;
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
  } else {
    alert('Please capture an image first!');
  }
});
</script>

  </body>
</html>
```

contact.html

```
<!DOCTYPE html>
<html>
<head>
        <title>DeepFake Detection | Contact Us</title>
</head>

<body>

  <nav>
    <ul>
      <li><a href="{% url 'home' %}">Home</a></li>
      <li><a href="{% url 'about' %}">About</a></li>
      <li><a href="{% url 'index' %}">Predict</a></li>
      <li><a href="{% url 'contact' %}">Contact</a></li>
    </ul>
  </nav>
  <h1 class="title">Contact Us!</h1>
    <div class="card">
          <div class="box-1">
            <h5>Our Address</h5>
            <p>Visakhapatnam, Andhra Pradesh, India</p>
          </div>
          <div class="box-2">
            <h5>Email Us</h5>
            <p>ananditad21@gmail.com</p>
          </div>
          <div class="box-3">
            <h3>Call Us</h3>
            <p>+91 630 955 9548</p>
          </div>
    </div>

    <div class="card2">


        <div class="part2">
            <span class="tit">Leave Us a Comment!</span>
            <form class="form">
              <div class="group">
              <input placeholder="" type="text" required="">
              <label for="name">Name</label>
              </div>
          <div class="group">
              <input placeholder="" type="email" id="email" name="email" required="">
              <label for="email">Email</label>
      </div>
    <div class="group">
      <textarea placeholder="" id="comment" name="comment" rows="5" required=""></textarea>
      <label for="comment">Comment</label>
    </div>
      <button type="submit">Submit</button>
     </form>
    </div>
</div>
  <script>
     const form = document.querySelector('.form');

     form.addEventListener('submit', (e) => {
      e.preventDefault();

      const nameInput = form.querySelector('#name');
      const emailInput = form.querySelector('#email');
      const commentInput = form.querySelector('#comment');

      const nameValue = nameInput.value.trim();
      const emailValue = emailInput.value.trim();
      const commentValue = commentInput.value.trim();

      if (nameValue === '' || emailValue === '' || commentValue === '') {
        alert('Please fill in all fields');
        return;
      }

      const data = {
        name: nameValue,
        email: emailValue,
        subject: subjectValue,
        message: messageValue
      };

      fetch('http://127.0.0.1:8000/contact/', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
      })
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.json();
      })
```

```
            .then(data => {
              console.log(data);
              alert('Form submitted successfully');
            })
            .catch(error => {
              console.error('There was a problem submitting the form:', error);
              alert('There was a problem submitting the form');
            });

            form.reset();
          });;
      </script>

</body>
</html>
```

views.py

```
import io
import json
from tensorflow import keras
import numpy as np
from django.http import JsonResponse
from django.shortcuts import render
from PIL import Image

import base64
#from keras.preprocessing.image import img_to_array
from keras.models import load_model
def home(request):
    return render(request, 'home.html')

def about(request):
    return render(request, 'about.html')

def contact(request):
    return render(request, 'contact.html')

model = load_model('C:/Users/daksh/Desktop/predict3.1/predict/densenet.h5',compile=False)
def index(request):
    prediction = None
    image_data_url = None
    if request.method == 'POST':
        image_file = request.FILES['image']
        image_bytes = image_file.read()
        image = Image.open(io.BytesIO(image_bytes))
        image = image.convert('RGB').resize((224, 224))

        image = np.array(image)
        image = image/255.0
        image_data_url = f"data:image/jpeg;base64,{base64.b64encode(image_bytes).decode()}"

        prediction = model.predict(image[np.newaxis])[0][0]
        predicted_class = 'real' if prediction > 0.5 else 'fake'
        prediction = f'The given image is predicted to be  {predicted_class}'

    context = {'prediction': prediction, 'image_data_url': image_data_url}
    return render(request, 'index.html', context)
```

6.2.2 Back-End Code

The models are used to distinguish between a real and a fake image by preprocessing the images after which the models are trained on 80% the data following which the model was tested on the remaining data.

SVM

```
import cv2# for image processing
import numpy as np#For converting the data into numpy arrays
import glob#used to read all the files given the folder / directory
import pickle#to store the data extracted from each of the files
from scipy.interpolate import griddata# algorithms for interpolation ,Interpolate unstructured
D-D data.
data= {}
epsilon = 1e-8
N = 80
y = []
error = []

number_iter = 1000#number of images

psd1D_total = np.zeros([number_iter, N])
```

```
label_total = np.zeros([number_iter])
psd1D_org_mean = np.zeros(N)
psd1D_org_std = np.zeros(N)


cont = 0
        #Calculate the azimuthally averaged radial profile.
        def azimuthalAverage(image, center=None):
          # Calculate the indices from the image
          y, x = np.indices(image.shape)
          # Calculate the center indices from the image
          if not center:
              center = np.array([(x.max()-x.min())/2.0, (y.max()-y.min())/2.0])

          r = np.hypot(x - center[0], y - center[1])

         #for sorted radii
          ind = np.argsort(r.flat)
          r_sorted = r.flat[ind]
          i_sorted = image.flat[ind]

          # For the integral part of the radii
          r_int = r_sorted.astype(int)

           # Find all pixels that fall within each radial bin.
              #Binning groups continuous data into categories defined by specific ranges
          deltar = r_int[1:] - r_int[:-1]
          rind = np.where(deltar)[0]
          nr = rind[1:] - rind[:-1]

          # Cumulative sum to get each radius bin
          csim = np.cumsum(i_sorted, dtype=float)
          tbin = csim[rind[1:]] - csim[rind[:-1]]

          radial_prof = tbin / nr

          return radial_prof
        #fake data
        rootdir = 'C:/Users/daksh/Desktop/PROJECT-1 CODING/dataset_celebA1/'

        for filename in glob.glob(rootdir+"*.jpg"):
```

```
    img = cv2.imread(filename,0)

    f = np.fft.fft2(img)#Compute the 2-dimensional discrete Fourier Transform.
    fshift = np.fft.fftshift(f)#Shift the zero-frequency component to the center of the spectrum
    fshift += epsilon

    magnitude_spectrum = 20*np.log(np.abs(fshift))
    # Calculate the azimuthally averaged 1D power spectrum
    psd1D = azimuthalAverage(magnitude_spectrum)


    points = np.linspace(0,N,num=psd1D.size) #co ordinates of a point
    xi = np.linspace(0,N,num=N) #co ordinates for interpolation

    interpolated = griddata(points,psd1D,xi,method='cubic')#Interpolate unstructured D-D data.
    interpolated /= interpolated[0]

    psd1D_total[cont,:] = interpolated
    label_total[cont] = 1
    cont+=1

    if cont == number_iter:
        break

for x in range(N):
    psd1D_org_mean[x] = np.mean(psd1D_total[:,x])
    psd1D_org_std[x]= np.std(psd1D_total[:,x])
psd1D_total2 = np.zeros([number_iter, N])#Return a new array of given shape and type, filled
with zeros.
label_total2 = np.zeros([number_iter])
psd1D_org_mean2 = np.zeros(N)
psd1D_org_std2 = np.zeros(N)

cont = 0
#real data
rootdir2 = 'C:/Users/daksh/Desktop/PROJECT-1 CODING/img_align_celeba1/'


for filename in glob.glob(rootdir2+"*.jpg"):
    img = cv2.imread(filename,0)
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    fshift += epsilon

    magnitude_spectrum = 20*np.log(np.abs(fshift))


    psd1D = azimuthalAverage(magnitude_spectrum)

    points = np.linspace(0,N,num=psd1D.size)
    xi = np.linspace(0,N,num=N)

    interpolated = griddata(points,psd1D,xi,method='cubic')

    interpolated /= interpolated[0]

    psd1D_total2[cont,:] = interpolated
    label_total2[cont] = 0
    cont+=1

    if cont == number_iter:
        break

for x in range(N):
    psd1D_org_mean2[x] = np.mean(psd1D_total2[:,x])
    psd1D_org_std2[x]= np.std(psd1D_total2[:,x])


y.append(psd1D_org_mean)
y.append(psd1D_org_mean2)
error.append(psd1D_org_std)
error.append(psd1D_org_std2)

psd1D_total_final = np.concatenate((psd1D_total,psd1D_total2), axis=0)
label_total_final = np.concatenate((label_total,label_total2), axis=0)

data["data"] = psd1D_total_final
data["label"] = label_total_final

output = open('celeba_low_1000.pkl', 'wb')
pickle.dump(data, output)
```

```
output.close()

print("DATA Saved")
##########################

pickle_off = open("celeba_low_1000.pkl", 'rb')
unpickle = pickle.load(pickle_off)
print(unpickle)

import numpy as np
import matplotlib.pyplot as plt
import pickle


pkl_file = open('celeba_low_1000.pkl', 'rb')
data = pickle.load(pkl_file)
pkl_file.close()
X = data["data"]
y = data["label"]
plt.plot(y)
y

num = int(X.shape[0]/2)
num_feat = X.shape[1]

psd1D_org_0 = np.zeros((num,num_feat))
psd1D_org_1 = np.zeros((num,num_feat))
psd1D_org_0_mean = np.zeros(num_feat)
psd1D_org_0_std = np.zeros(num_feat)
psd1D_org_1_mean = np.zeros(num_feat)
psd1D_org_1_std = np.zeros(num_feat)

cont_0=0
cont_1=0

# We separate real and fake using the label
for x in range(X.shape[0]):
    if y[x]==0:
        psd1D_org_0[cont_0,:] = X[x,:]
        cont_0+=1
    elif y[x]==1:
        psd1D_org_1[cont_1,:] = X[x,:]
        cont_1+=1

# We compute statistcis
for x in range(num_feat):
    psd1D_org_0_mean[x] = np.mean(psd1D_org_0[:,x])
    psd1D_org_0_std[x]= np.std(psd1D_org_0[:,x])
    psd1D_org_1_mean[x] = np.mean(psd1D_org_1[:,x])
    psd1D_org_1_std[x]= np.std(psd1D_org_1[:,x])

# Plot
x = np.arange(0, num_feat, 1)
fig, ax = plt.subplots(figsize=(15, 9))
ax.plot(x, psd1D_org_0_mean, alpha=0.5, color='m', label='Fake', linewidth =2.0)
ax.fill_between(x, psd1D_org_0_mean - psd1D_org_0_std, psd1D_org_0_mean +
psd1D_org_0_std, color='red', alpha=0.2)
ax.plot(x, psd1D_org_1_mean, alpha=0.5, color='c', label='Real', linewidth =2.0)
ax.fill_between(x, psd1D_org_1_mean - psd1D_org_1_std, psd1D_org_1_mean +
psd1D_org_1_std, color='blue', alpha=0.2)

plt.tick_params(axis='x', labelsize=20)
plt.tick_params(axis='y', labelsize=20)
ax.legend(loc='best', prop={'size': 20})
plt.xlabel("Spatial Frequency", fontsize=20)
plt.ylabel("Power Spectrum", fontsize=20)
plt.savefig('1000_celeba.png', bbox_inches='tight')

import numpy as np
import matplotlib.pyplot as plt
import pickle

num = 10
SVM = 0
SVM_r = 0
SVM_p = 0


for z in range(num):
    # read python dict back from the file
    pkl_file = open('celeba_low_1000.pkl', 'rb')
    data = pickle.load(pkl_file)
    pkl_file.close()
    X = data["data"]
    y = data["label"]
    try:

        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
        #Linear
        from sklearn.svm import SVC
        svclassifier = SVC(kernel='linear')
        svclassifier.fit(X_train, y_train)
        #Gaussian RBF
        from sklearn.svm import SVC
        svclassifier_r = SVC(C=6.37, kernel='rbf', gamma=0.86)
        svclassifier_r.fit(X_train, y_train)
        #Polynomial
        from sklearn.svm import SVC
        svclassifier_p = SVC(kernel='poly')
        svclassifier_p.fit(X_train, y_train)
        SVM+=svclassifier.score(X_test, y_test)
        SVM_r+=svclassifier_r.score(X_test, y_test)
        SVM_p+=svclassifier_p.score(X_test, y_test)



print("Average SVM: "+str(SVM/num))
print("Average SVM_r: "+str(SVM_r/num))
print("Average SVM_p: "+str(SVM_p/num))


########FOR LINEAR##############
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
pred_l=svclassifier.predict(X_test)
print(classification_report(y_test,pred_l))
cm_l=confusion_matrix(y_test,pred_l)
import seaborn as sns
sns.heatmap(cm_l,cmap=" BuGn ",annot=True)

########FOR GAUSSIAN RBF##############
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
pred_r=svclassifier_r.predict(X_test)
print(classification_report(y_test,pred_r))
cm_r=confusion_matrix(y_test,pred_r)
import seaborn as sns
sns.heatmap(cm_r,cmap=" GnBu ",annot=True)

########FOR POLYNOMIAL##############
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
pred_p=svclassifier_p.predict(X_test)
print(classification_report(y_test,pred_p))
cm_p=confusion_matrix(y_test,pred_p)
import seaborn as sns
sns.heatmap(cm_p,cmap="YlGn_r",annot=True)
#SAVING MODEL AS A PICKLE FILE
with open("svclassifier_p.pkl","wb") as json_file:
    #json_file.write(svclassifier_p.to_pd())
    pickle.dump(svclassifier_p,json_file)
print("Saved model")
# LOADING SAVED MODEL
loaded_model = pickle.load(open('svclassifier_p.pkl','rb'))
```

DENSENETS

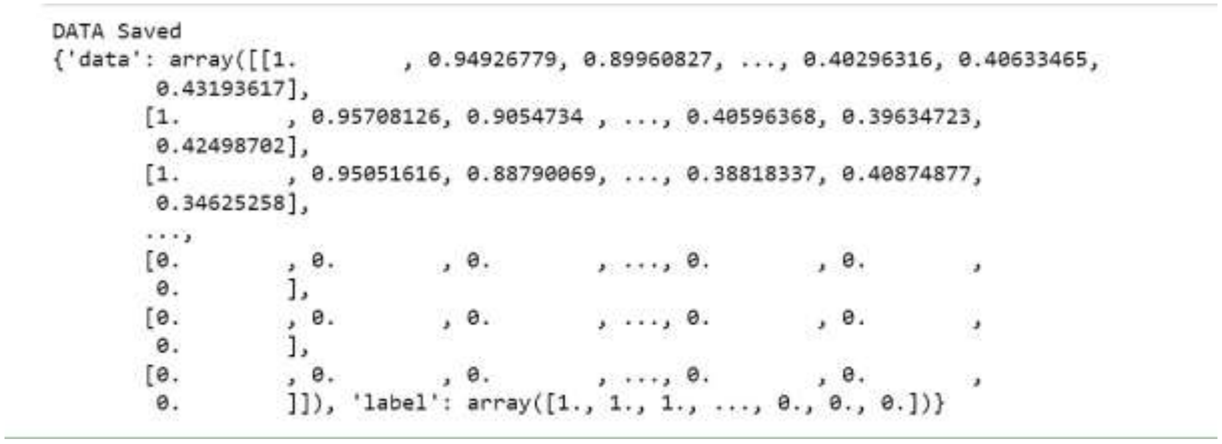
```
#install Kaggle
!pip install -q kaggle
from google.colab import files
files.upload()
# create a kaggle folder
! mkdir ~/.kaggle
# copy kaggle.json to folder created
! cp kaggle.json ~/.kaggle/
#permissions for json to act
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets download -d xhlulu/140k-real-and-fake-faces #downloading the
dataset on to the envt
! unzip 140k-real-and-fake-faces.zip #unzipping
```

```
import cv2
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.applications import DenseNet121 #pretrained model
from tensorflow.keras.callbacks import Callback, ModelCheckpoint
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential # in neural networks we have 2 types
of model:Sequential- where we specify the how the layers are connected. in parallel
the same is unspecified
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from sklearn import metrics
import tensorflow as tf

def build_model(pretrained):
   model = Sequential([
      pretrained,
      layers.GlobalAveragePooling2D(),
      layers.Dense(1, activation='sigmoid')
   ])

   model.compile(
      loss='binary_crossentropy'

      optimizer=Adam(),
      metrics=['accuracy']
   )

   return model


base_path = '/content/real_vs_fake/real-vs-fake/'
image_gen = ImageDataGenerator(rescale=1./255.)#preprocessing by rescaling.

train_flow = image_gen.flow_from_directory(
   base_path + 'train/',
   target_size=(224, 224),
   batch_size=64,
   class_mode='binary'
)
```

```
image_gen1 = ImageDataGenerator(rescale=1./255.)

valid_flow = image_gen1.flow_from_directory(
    base_path + 'valid/',
    target_size=(224, 224),
    batch_size=64,
    class_mode='binary'
)

densenet = DenseNet121(
    weights=None,
    include_top=False,
    input_shape=(224,224,3)
)
model = build_model(densenet)
model.summary()

train_steps = 100000//64 #=1562
valid_steps = 20000//64

history = model.fit_generator(
    train_flow,
    epochs = 11,# iterations
    steps_per_epoch =train_steps,
    validation_data =valid_flow,
    validation_steps = valid_steps
)
model.save('completed_trained_model.h5')
test_flow = image_gen1.flow_from_directory(
    base_path + 'test/',
    target_size=(224, 224),
    batch_size=1,
    shuffle = False,
    class_mode='binary'
)
y_pred = model.predict(test_flow)
y_test = test_flow.classes
print(metrics.classification_report(y_test, y_pred > 0.5))
```

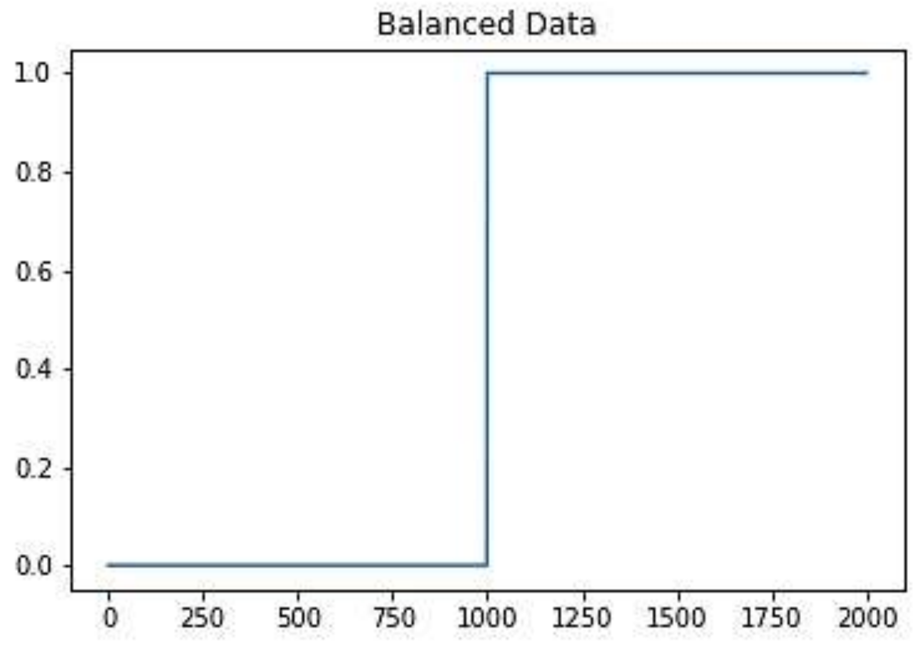
6.3 RESULTS

The below Screens represent the unpickled data after preprocessing by Discrete Fourier Transforms and Azimuthal Averaging



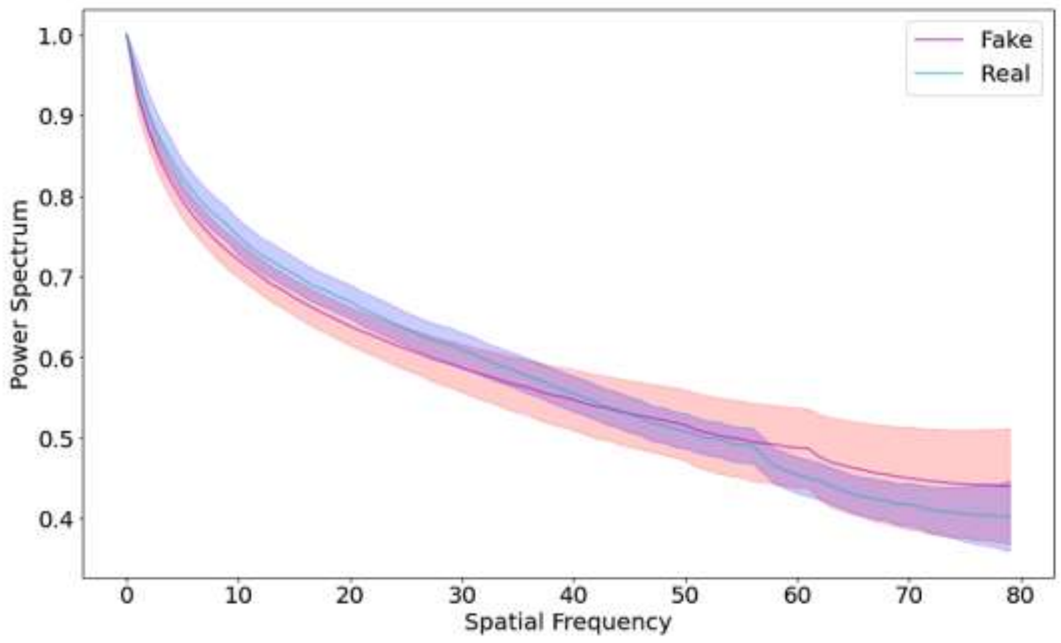
```
DATA Saved
{'data': array([[1.        , 0.94926779, 0.89960827, ..., 0.40296316, 0.40633465,
         0.43193617],
       [1.        , 0.95708126, 0.9054734 , ..., 0.40596368, 0.39634723,
        0.42498702],
       [1.        , 0.95051616, 0.88790069, ..., 0.38818337, 0.40874877,
        0.34625258],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ]]), 'label': array([1., 1., 1., ..., 0., 0., 0.])}
```

***Screen 1**: Unpickled Data*



***Screen 2:** Balanced Data*

The below screen represents the 1D Power Spectrum generated by the Real and Fake images.

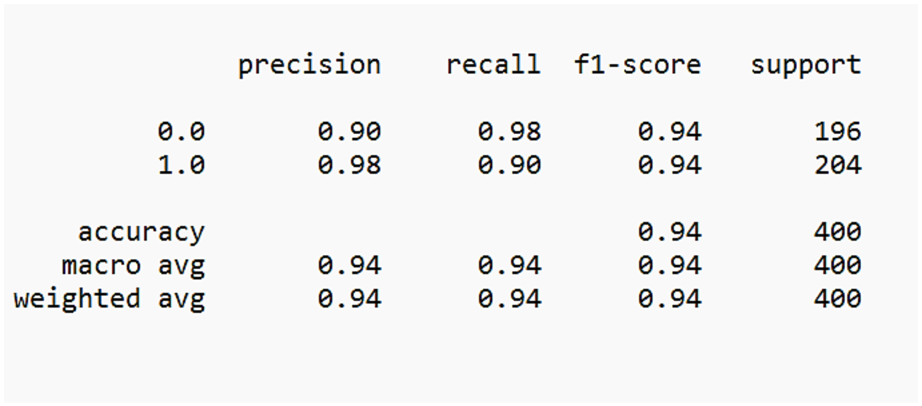


Screen 3: *1D Power Spectrum*

The linear kernel was utilized to analyze the model's behavior and sensitivity and was used as a baseline model with an accuracy of 94.45% on the given dataset. To achieve a higher rate of accuracy the Gaussian Radial Basis Function Kernel was employed. With a gamma value of 0.85, it presents an accuracy of 99.32%. From the above models, it is evident that, the data was not entirely linear and hence requires dimensional transformation. In order to transform the dataset into a higher dimensional space, the Polynomial Kernel was used at a degree of 3 which has resulted in an accuracy of 99.975%.

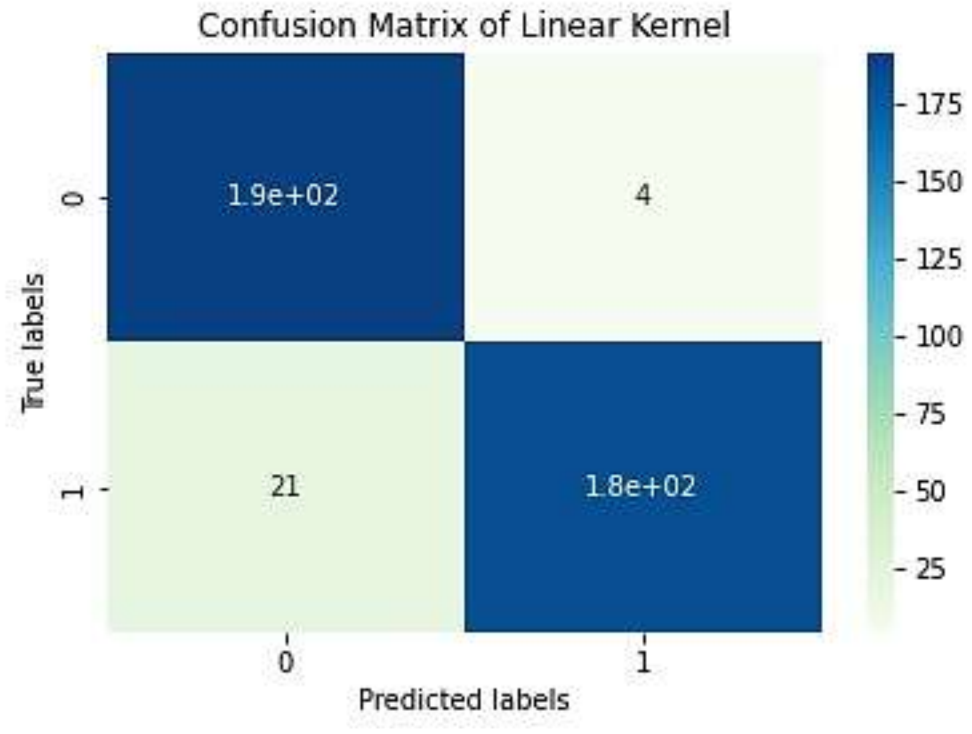
The DenseNet classifier has resulted in a high accuracy of 98% and a final loss of 0.0268. Furthermore, during the validation process, the model achieved a loss of 0.1164 and an accuracy of 0.9617 on the validation set.

These results indicate that the model is performing very well on the training set, with a high accuracy and a low loss. The validation set results are also impressive, with a high accuracy and a relatively low loss, suggesting that the model is generalizing well to new data.

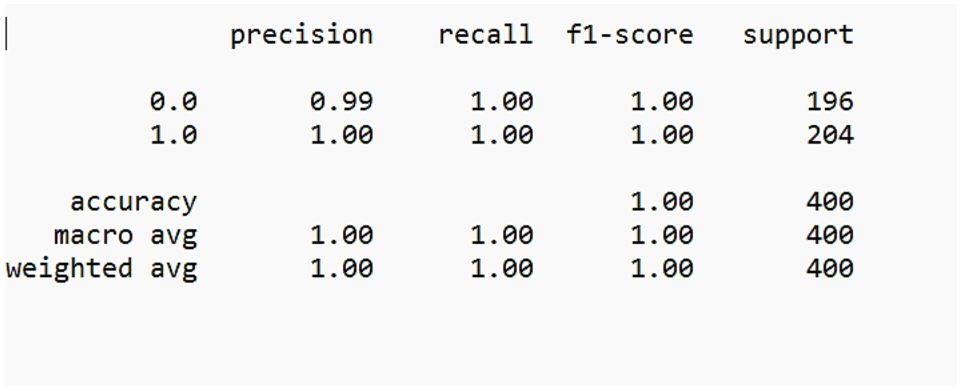


```
              precision    recall  f1-score   support

         0.0       0.90      0.98      0.94       196
         1.0       0.98      0.90      0.94       204

    accuracy                           0.94       400
   macro avg       0.94      0.94      0.94       400
weighted avg       0.94      0.94      0.94       400
```

Screen 4: *Classification Report of Linear Kernel*

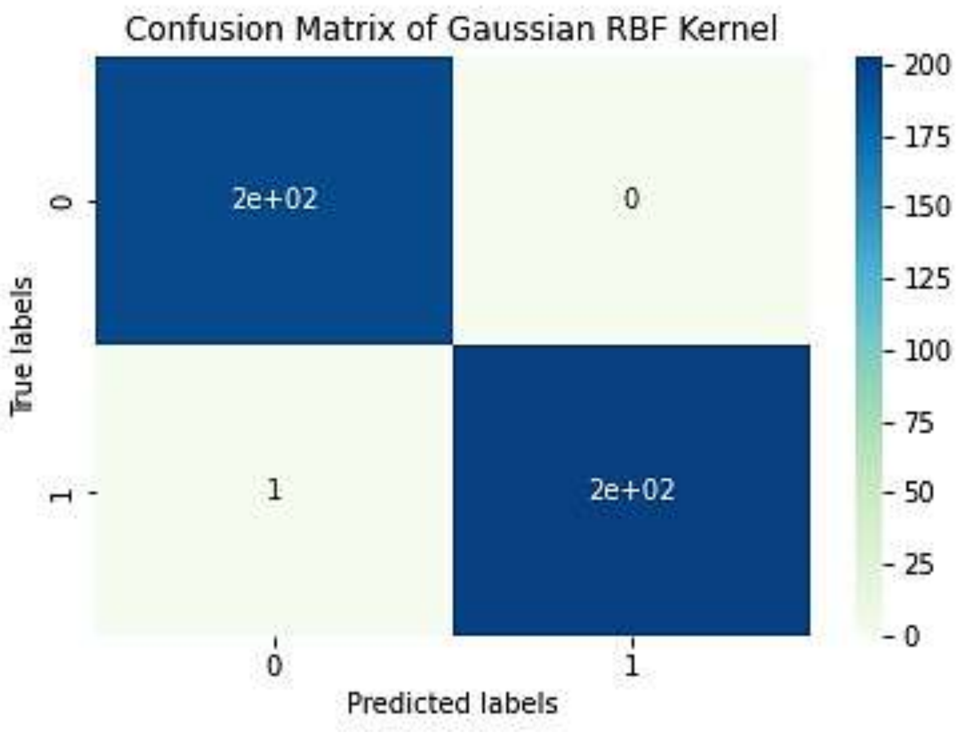


Screen 5: *Confusion Matrix of Linear Kernel*

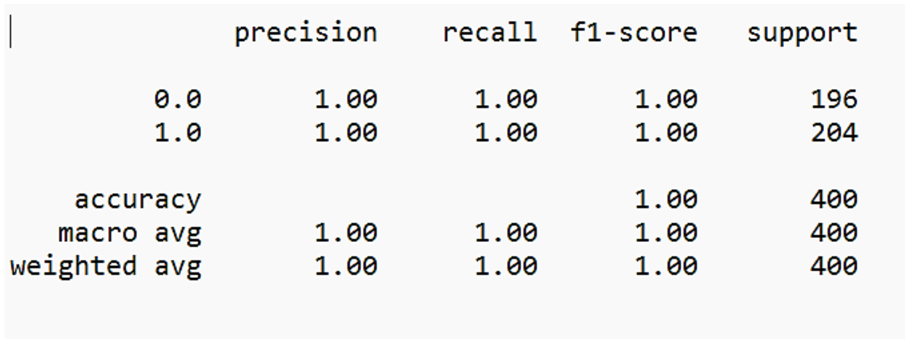


```
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00       196
         1.0       1.00      1.00      1.00       204

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

Screen 6: *Classification Report of Gaussian RBF Kernel*

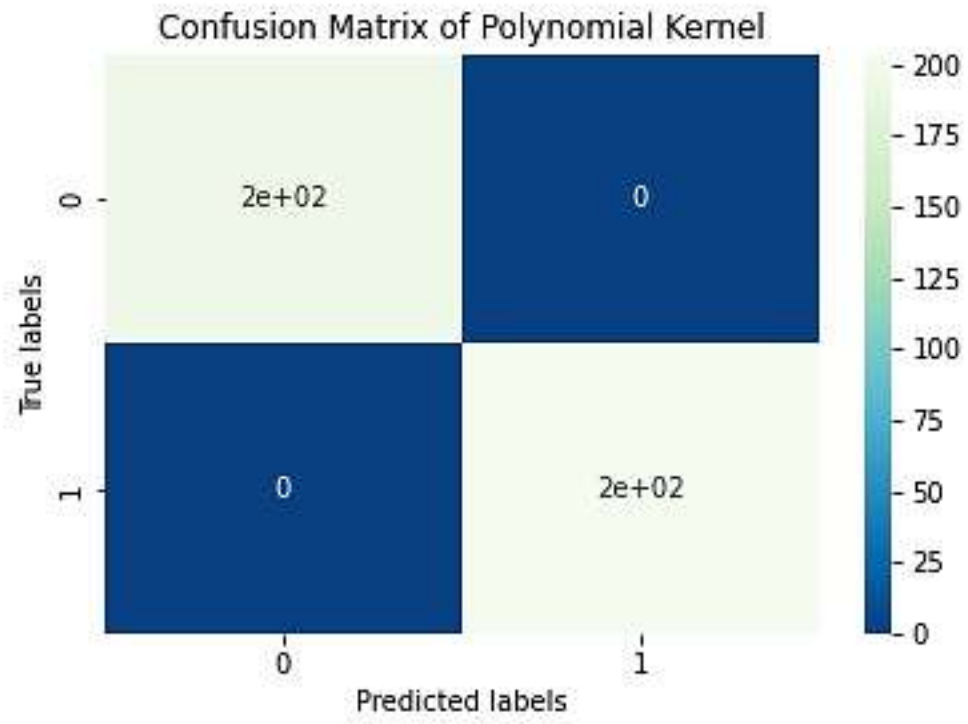


Screen 7: *Confusion Matrix of Gaussian RBF Kernel*

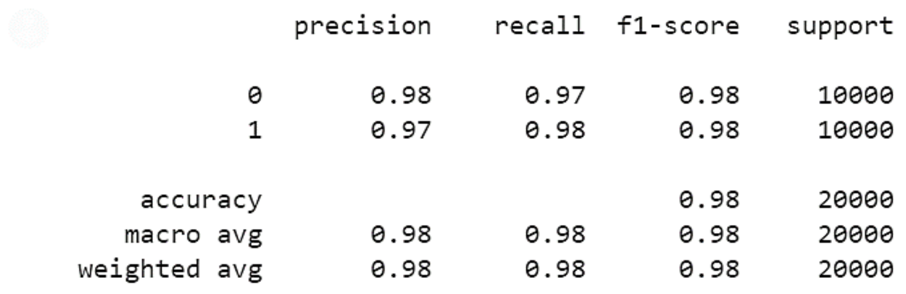


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       196
         1.0       1.00      1.00      1.00       204

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

***Screen 8**: Classification Report of Polynomial Kernel*



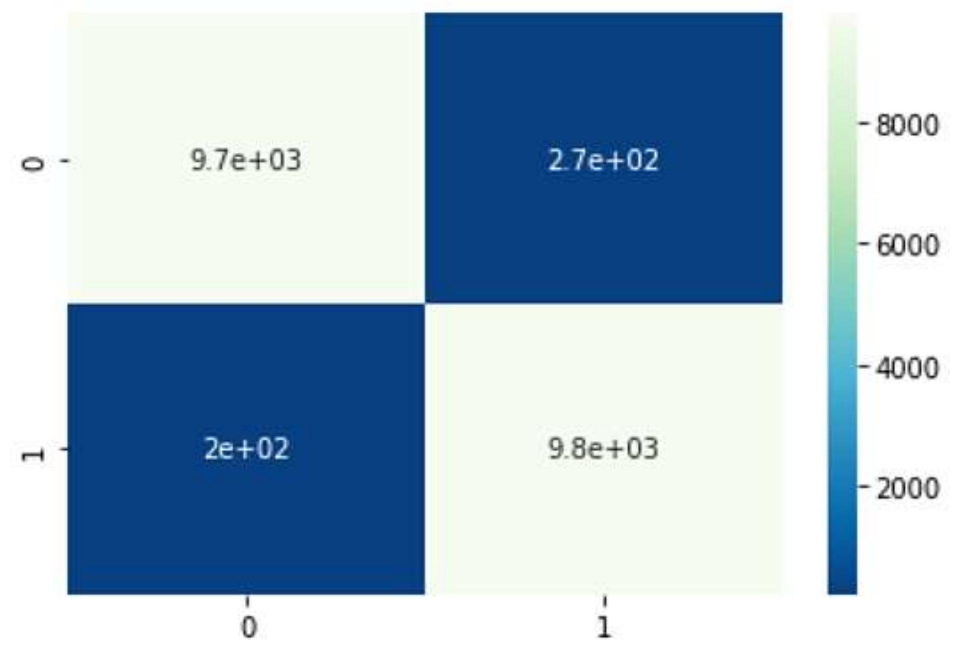
***Screen 9:** Confusion Matrix of Polynomial Kernel*



```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98     10000
           1       0.97      0.98      0.98     10000

    accuracy                           0.98     20000
   macro avg       0.98      0.98      0.98     20000
weighted avg       0.98      0.98      0.98     20000
```

***Screen 10**: Classification Report of DenseNet Classifier.*

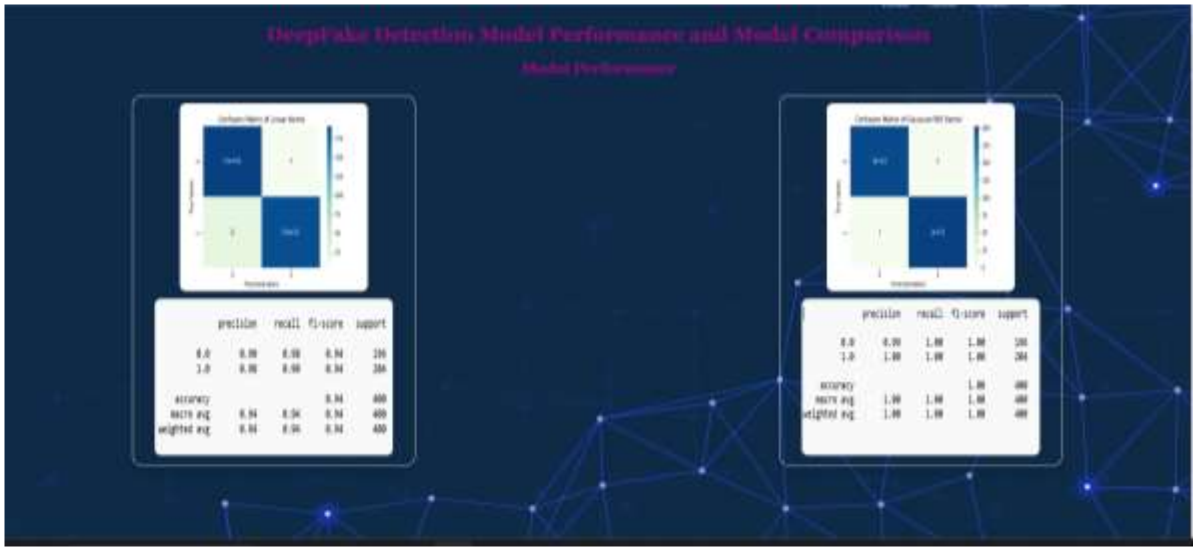
```
<Axes: >
```



***Screen 11:** Confusion Matrix of DenseNet Classifier.*

***Screen 12**: Home Page of Front-End*

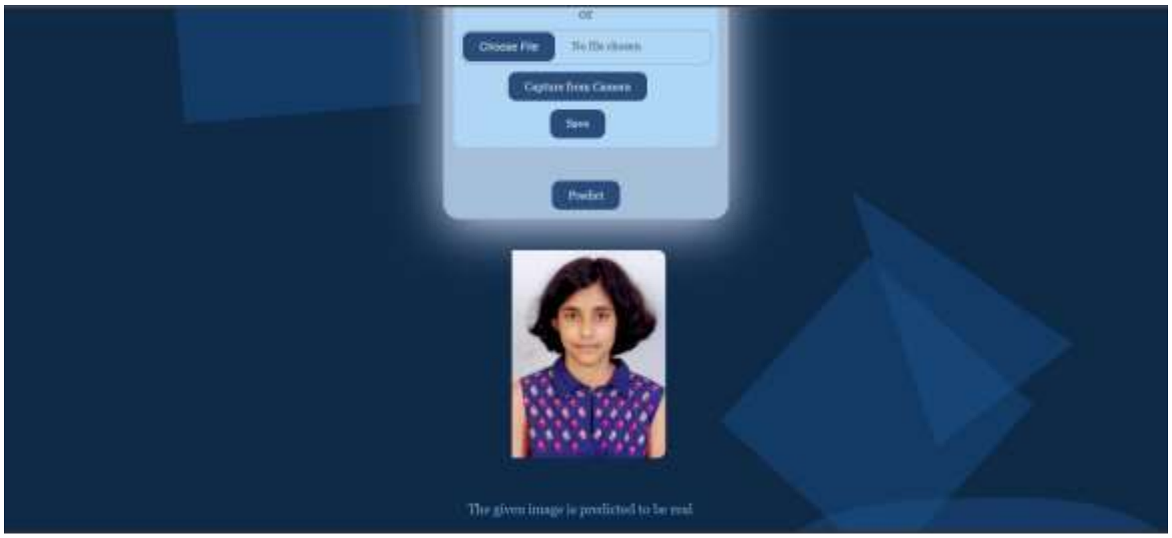
.



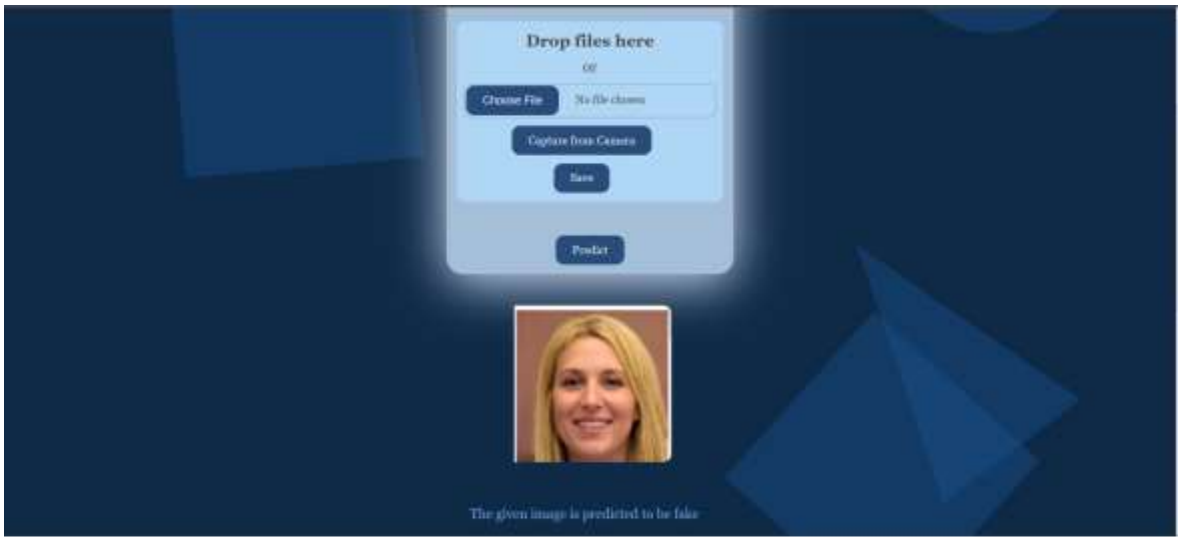
```
              precision    recall  f1-score   support

         0.0       0.98      0.90      0.94       196
         1.0       0.91      0.98      0.94       204

    accuracy                           0.94       400
   macro avg       0.94      0.94      0.94       400
weighted avg       0.94      0.94      0.94       400
```

```
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00       196
         1.0       1.00      1.00      1.00       204

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

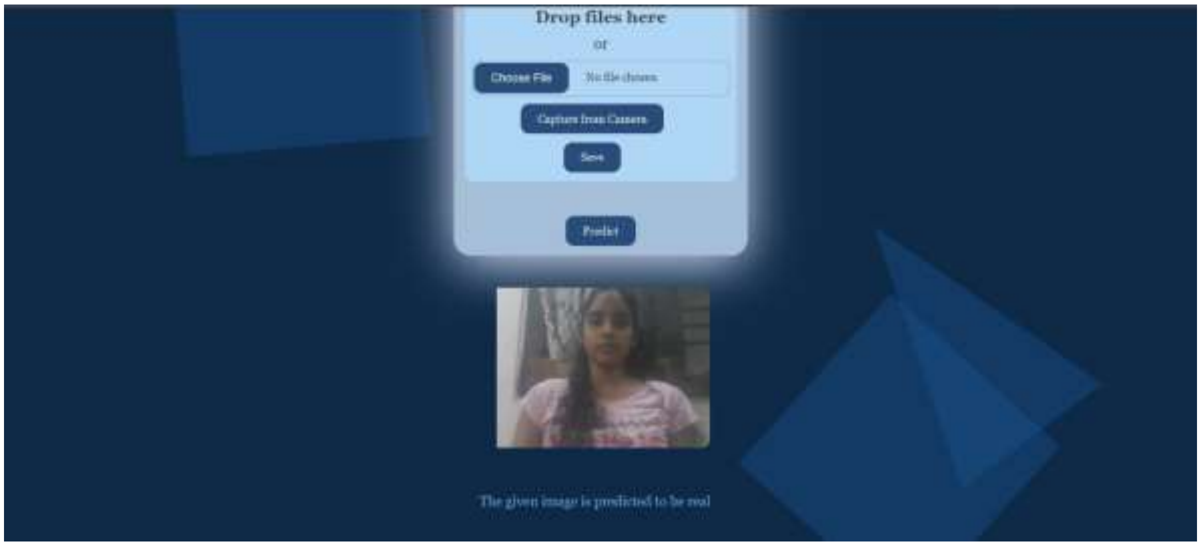
***Screen 13**: About Page of Front-End*



Screen 14: *Predict Page of Front-End-Real Image*

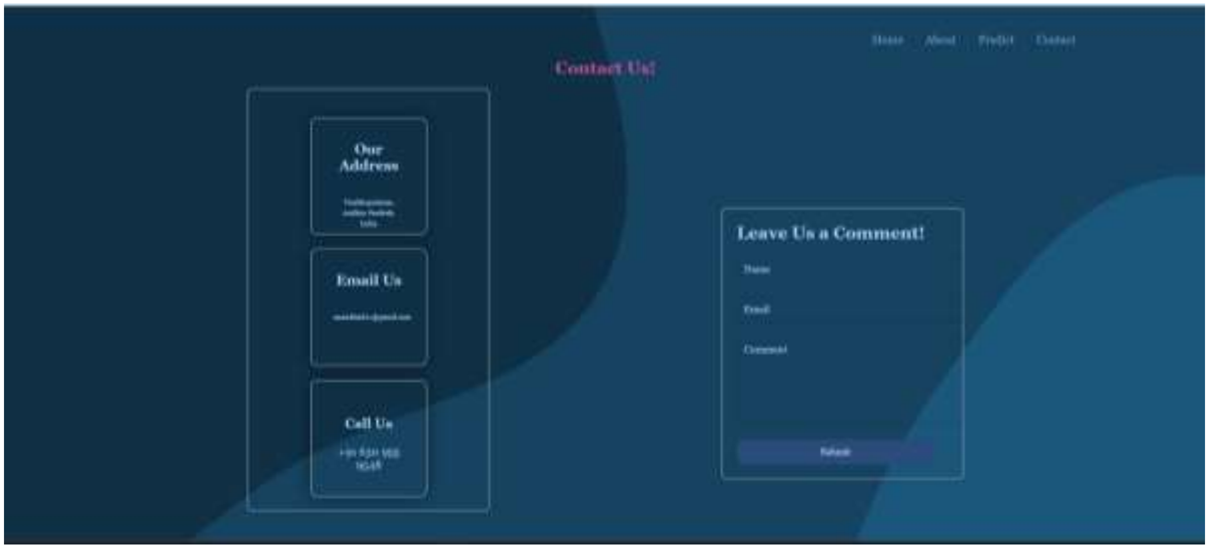


Screen 15*: Predict Page of Front-End-Fake Image*



The given image is predicted to be real

***Screen 16**: Capture Image*



***Screen 17:** Contact Us Page of Front-End*

7. TESTING AND VALIDATION

7.1INTRODUCTION

Scope

One of the main goals of testing is to find software flaws so that faults can be found and fixed. This is not a simple task. Testing proves whether the project works properly in all circumstances or not. Examining code as well as executing it in various environments and conditions, as well as examining the characteristics of code, does it accomplish what it is supposed to do and what it needs to do, is all part of software testing. In today's world, a testing organization may be separate from the development team, depending on the software development culture.

Defects and Failures

Coding faults do not cause all software issues. Need gaps, such as unrecognized requirements that result in omission of errors by the programmer designer, are a common source of costly faults. Non-functional needs including testability, scalability, maintainability, usability, performance, and security are a major source of requirements gaps.

●The following processes cause software errors.

●A programmer makes a mistake in the software source code, resulting in a defect (fault, bug). If this flaw is exploited, the system will produce incorrect results in some circumstances, resulting in a failure.

●Failures are not always the result of flaws

Defects in dead code, for example, will never cause failures. A fault might become a failure when the environment changes.

Compatibility

Compatibility with another application, a new operating system, or, increasingly, a new web browser version is a common cause of software failure. When it comes to backward compatibility, this can happen because programmers have only considered creating or testing their software for the latest version of the operating system. The unintentional effect of this fact is that their most recent work may not be fully compatible with other software/hardware combinations, or with another significant operating system. From the above cases, these variations, whatever they were, may have resulted in software failures, as seen by a large number of computer users.

Input Combinations and Pre-Conditions

Even with a simple product, verifying all possible inputs and preconditions is impossible. As a result, the number of flaws in a software product might be enormous, and defects that occur seldom are difficult to detect during testing.

Static vs. Dynamic Testing

Software testing can be done in a variety of ways. Two of them are static and dynamic. Static testing includes things like reviews, walk through, and inspections, whereas dynamic testing involves actual running of the code with a set of test cases. The former can be skipped, whereas the latter occurs when users are used for the first time, which is usually regarded as the start of the testing phase. This could even start before the user is finished in order to test a certain area.

7.2 TYPES OF TESTING

Unit Testing

i. Black Box Testing

This technique generates some test cases as input conditions that fully execute all of the program's functional requirements. This testing was used to identify faults in the following areas.

- Missing or incorrect functionalities
- Interaction issues
- Data structure or external database access errors
- Errors in performance
- Errors in initialization and termination

Only the output is examined in this testing and the logical flow of the data is not checked.

ii. White Box Testing

Test cases are built for each module's logic by generating flow graphs for that module, and logical judgments are tested on all situations. It was used to create test cases in the following scenarios,

- Ascertain that all independent paths have been followed.
- Carry out all logical decisions, both true and false.
- Complete all loops inside their operational constraints and at their boundaries.

iii. Gray Box Testing

It's a method of testing a software product or application with only a rudimentary understanding of its internal workings. The goal of this testing is to look for faults caused by incorrect code structure or improper application usage.

Context-specific issues relating to web systems are frequently detected throughout this process.

It expands the testing coverage by focusing on all layers of a complicated system.

Integration Testing

Integration testing guarantees that software and its subsystems work together efficiently. It checks the interfaces of all the modules to ensure that they work together properly.

Software verification and validation - Verification and validation are used in conjunction with software testing.

Verification - It deals with the correctness of the model and the procedures applied.

These two are the required steps done in software testing and validation at the process and product level.

Validation-The Polynomial SVM Kernal outperforms the highest performing individual model in the present in the in all the specified performance metrics. This model is accessible through a front-end that has also been effectively tested and installed on the local system, guaranteeing that all of the requirements specified in the software requirements specification have been fulfilled. Warning notifications are displayed to the user in order to steer them in the path of the software's intended use.

7.3 DESIGN OF TEST CASES AND SCENARIOS

The test cases are formatted so that the user can enter values in the proper format. The table below specifies the test cases applied for our project.

Table 5: Test Cases and Scenarios

S.No	**Test Type**	**Test Case**	**Expected Result**	**Observed Result**	**Positive/Negative**	**Pass/Fail**
1	Black Box Testing	Making prediction on user uploaded Real image.	Return prediction value Real.	Pass	Positive	Pass
2	Black Box Testing	Making prediction on user uploaded Fake image.	Return prediction value Fake.	Pass	Positive	Pass
3	Black Box Testing	Making prediction on user uploaded image of invalid format	Raise an Error	Pass	Positive	Pass
4	Black Box Testing	Making prediction on user uploaded image of invalid image (non-facial data)	Raise an Error	Pass	Positive	Pass
5	Black Box Testing	Making prediction on user uploaded computer-generated image of non-facial data	Raise an Error	Fail	Negative	Fail

7.3 CONCLUSION

This chapter focuses on the certification of systems using various testing methods such as accuracy and F1 score. The best Accuracy score was achieved by the Polynomial SVM Model, which was 99.87% followed by the DenseNet Model which has presented an accuracy of 98%.

8. FUTURE WORK

We have considered the domain of Machine Learning for our project of DeepFake Detection. In order to undertake this project, we surveyed several articles in the latest areas of Generative Adversarial Network (GAN) altered images using a plethora of algorithms. We undertook a literature survey of nearly forty IEEE journal and conference articles published between 2017 and 2022. The flowchart of the proposed system and architecture diagram were the starting points for the project. The next step was the acquisition of the dataset from the CelebA website for training the Support Vector Machine model. The acquired dataset was then cleaned and pre-processed into a workable dataset using the image processing techniques of Fourier Transforms and Azimuthal Averages. Three Kernels namely, Linear, Gaussian RBF and Polynomial were trained on the data to achieve a accuracy and a low RMSE value.

We have then chosen a Deep Learning Model namely DenseNet which is a Deep Neural Network to train the 140K Real and Fake Faces dataset which has also proven to be a robust model for detecting GAN generated images.

As the field of computer science continues to advance, increasingly complex algorithms are likely to be developed in the future. These new algorithms could be incorporated into the existing model, allowing for the creation of an even more sophisticated ensemble model. With the necessary modifications, this model could be used to recognize DeepFaked videos.

9. CONCLUSION

DeepFakes are manipulated data in the form of images, videos or audios by employing complex deep learning techniques such as Autoencoders and GANs (Generative Adversarial Networks). They can be malicious in the wrong hands because of their capacity to progressively resemble authentic photos, videos, and audios. Therefore, DeepFakes are a social menace that disrupt the peace of mind of the victims and can have terrible consequences. The ill effects of DeepFakes that might cause unnecessary trauma for individuals in society must be eliminated as they create realistic looking fake images.

Support Vector Machines and DenseNets have been utilized to identify DeepFakes. They have both proven to be highly robust with accuracies of 99.97% and 98% respectively. In the future, we would like to explore other technologies to detect DeepFaked videos.

10. REFERENCES

[1] R. Katarya and A. Lal, "A Study on Combating Emerging Threat of DeepFake Weaponization," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 485-490, doi: 10.1109/I-SMAC49090.2020.9243588.

[2] S. A. Aduwala, M. Arigala, S. Desai, H. J. Quan, and M. Eirinaki, "DeepFake Detection using GAN Discriminators," 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), 2021, pp. 69-77, doi: 10.1109/BigDataService52369.2021.00014.

[3] H. Agarwal, A. Singh, and R. D, "DeepFake Detection Using SVM," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 1245-1249, doi: 10.1109/ICESC51422.2021.9532627.

[4] D. Güera and E. J. Delp, "DeepFake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018, pp. 1-6, doi: 10.1109/AVSS.2018.8639163.

[5] Y. Al-Dhabi and S. Zhang, "DeepFake Video Detection by Combining Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)," 2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE), 2021, pp. 236-241, doi: 10.1109/CSAIEE54046.2021.9543264.

[6] Y. Li et al., "DeepFake-o-meter: An Open Platform for DeepFake Detection," 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 277-281, doi: 10.1109/SPW53761.2021.00047.

[7] M. S. Rana, B. Murali and A. H. Sung, "DeepFake Detection Using Machine Learning Algorithms," 2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI), 2021, pp. 458-463, doi: 10.1109/IIAI-AAI53430.2021.00079.

[8] R. Rafique, M. Nawaz, H. Kibriya and M. Masood, "DeepFake Detection Using

Error Level Analysis and Deep Learning," 2021 4th International Conference on Computing & Information Sciences (ICCIS), 2021, pp. 1-4, doi: 10.1109/ICCIS54243.2021.9676375.

[9] M. Khichi and R. Kumar Yadav, "A Threat of DeepFakes as a Weapon on Digital Platform and their Detection Methods," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021, pp. 01-08, doi: 10.1109/ICCCNT51525.2021.9580031.

[10] X. Yang, Y. Li, and S. Lyu, "Exposing Deep Fakes Using Inconsistent Head Poses," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 8261-8265, doi: 10.1109/ICASSP.2019.8683164.

[11] M. Li, B. Liu, Y. Hu, L. Zhang, and S. Wang, "DeepFake Detection Using Robust Spatial and Temporal Features from Facial Landmarks," 2021 IEEE International Workshop on Biometrics and Forensics (IWBF), 2021, pp. 1-6, doi: 10.1109/IWBF50991.2021.9465076.

[12] D. Pan, L. Sun, R. Wang, X. Zhang and R. O. Sinnott, "DeepFake Detection through Deep Learning," 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), 2020, pp. 134-143, doi: 10.1109/BDCAT50828.2020.00001.

[13] Z. Joseph and C. Nyirenda, "DeepFake Detection using a Two-Stream Capsule Network," 2021 IST-Africa Conference (IST-Africa), 2021, pp. 1-8.

[14] H. Khalid and S. S. Woo, "OC-FakeDect: Classifying DeepFakes Using One-class Variational Autoencoder," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 2794-2803, doi: 10.1109/CVPRW50498.2020.00336.

[15] Ajoy, C. U. Mahindrakar, D. Gowrish, and V. A, "DeepFake Detection using a frame-based approach involving CNN," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021, pp. 1329-1333, doi:

10.1109/ICIRCA51532.2021.9544734.

[16] H. Guo, S. Hu, X. Wang, M. -C. Chang and S. Lyu, "Robust Attentive Deep Neural Network for Detecting GAN-Generated Faces," in IEEE Access, vol. 10, pp. 32574-32583, 2022, doi: 10.1109/ACCESS.2022.3157297.

[17] Y. Ma, Y. Xu and F. Liu, "Multi-Perspective Dynamic Features for Cross-Database Face Presentation Attack Detection," in IEEE Access, vol. 8, pp. 26505-26516, 2020, doi: 10.1109/ACCESS.2020.2971224.

[18] K. Zhang, Y. Liang, J. Zhang, Z. Wang and X. Li, "No One Can Escape: A General Approach to Detect Tampered and Generated Image," in IEEE Access, vol. 7, pp. 129494-129503, 2019, doi: 10.1109/ACCESS.2019.2939812.

[19] L. Guarnera, O. Giudice and S. Battiato, "Fighting DeepFake by Exposing the Convolutional Traces on Images," in IEEE Access, vol. 8, pp. 165085-165098, 2020, doi: 10.1109/ACCESS.2020.3023037.

[20] E. Kim and S. Cho, "Exposing Fake Faces Through Deep Neural Networks Combining Content and Trace Feature Extractors," in IEEE Access, vol. 9, pp. 123493- 123503, 2021, doi: 10.1109/ACCESS.2021.3110859.

[21] J. Kang, S. -K. Ji, S. Lee, D. Jang and J. -U. Hou, "Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces," in IEEE Access, vol. 10, pp. 69031-69040, 2022, doi: 10.1109/ACCESS.2022.3185121.

[22] V. -N. Tran, S. -G. Kwon, S. -H. Lee, H. -S. Le and K. -R. Kwon, "Generalization of Forgery Detection with Meta Deepfake Detection Model," in IEEE Access, doi: 10.1109/ACCESS.2022.3232290.

[23] T. Jung, S. Kim and K. Kim, "DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern," in IEEE Access, vol. 8, pp. 83144-83154, 2020, doi: 10.1109/ACCESS.2020.2988660.

[24] A. Groshev, A. Maltseva, D. Chesakov, A. Kuznetsov and D. Dimitrov, "GHOST—A New Face Swap Approach for Image and Video Domains," in IEEE Access, vol. 10, pp. 83452-83462, 2022, doi: 10.1109/ACCESS.2022.3196668.

[25] M. Jiwtode, A. Asati, S. Kamble and L. Damahe, "Deepfake Video Detection using Neural Networks," 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), 2022, pp. 1-5, doi: 10.1109/ICBDS53701.2022.9935984

[26] A. Trabelsi, M. M. Pic and J. -L. Dugelay, "Improving Deepfake Detection by Mixing Top Solutions of the DFDC," 2022 30th European Signal Processing Conference (EUSIPCO), 2022, pp. 643-647, doi: 10.23919/EUSIPCO55093.2022.9909905.

[27] Y. -X. Luo and J. -L. Chen, "Dual Attention Network Approaches to Face Forgery Video Detection," in IEEE Access, vol. 10, pp. 110754-110760, 2022, doi: 10.1109/ACCESS.2022.3215963.

[28] I. -J. Yu, S. -H. Nam, W. Ahn, M. -J. Kwon and H. -K. Lee, "Manipulation Classification for JPEG Images Using Multi-Domain Features," in IEEE Access, vol. 8, pp. 210837-210854, 2020, doi: 10.1109/ACCESS.2020.3037735.

[29] D. Du, H. Cai, G. Chen and H. Shi, "Multi branch deepfake detection based on double attention mechanism," 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), 2021, pp. 746-749, doi: 10.1109/EIECS53707.2021.9587946.

[30] M. F. Hashmi, B. K. K. Ashish, A. G. Keskar, N. D. Bokde, J. H. Yoon and Z. W. Geem, "An Exploratory Analysis on Visual Counterfeits Using Conv-LSTM Hybrid Architecture," in IEEE Access, vol. 8, pp. 101293-101308, 2020, doi: 10.1109/ACCESS.2020.2998330.

[31] A. H. Khalifa, N. A. Zaher, A. S. Abdallah and M. W. Fakhr, "Convolutional Neural Network Based on Diverse Gabor Filters for Deepfake Recognition," in IEEE Access, vol. 10, pp. 22678-22686, 2022, doi: 10.1109/ACCESS.2022.3152029.

[32] A. Malik, M. Kuribayashi, S. M. Abdullahi and A. N. Khan, "DeepFake Detection for Human Face Images and Videos: A Survey," in IEEE Access, vol. 10, pp. 18757-18775, 2022, doi: 10.1109/ACCESS.2022.3151186.

[33] J. Zhang, K. Cheng, G. Sovernigo and X. Lin, "A Heterogeneous Feature Ensemble Learning based Deepfake Detection Method," ICC 2022 - IEEE International Conference on Communications, 2022, pp. 2084-2089, doi: 10.1109/ICC45855.2022.9838630.

[34] J. John and B. V. Sherif, "Comparative Analysis on Different DeepFake Detection Methods and Semi Supervised GAN Architecture for DeepFake Detection," 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2022, pp. 516-521, doi: 10.1109/I-SMAC55078.2022.9987265.

[35] A. KoÇak and M. Alkan, "Deepfake Generation, Detection and Datasets: a Rapid-review," 2022 15th International Conference on Information Security and Cryptography (ISCTURKEY), 2022, pp. 86-91, doi: 10.1109/ISCTURKEY56345.2022.9931802.

[36] A. Verma, D. Gupta and M. K. Srivastava, "Deepfake Detection using Inception-ResnetV2," 2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT), 2021, pp. 39-41, doi: 10.1109/ICACFCT53978.2021.9837351.

[37] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

Support Vector Machine Aided DeepFake Detection

Dr. Sharmili Nukapeyi
Department of Computer Scienece and Engineering
GVP College of Engineering for Women
Visakhapatnam, India
logintosharmi@gmail.com

Anandita Dakshayani Garimella
Department of Computer Scienece and Engineering
GVP College of Engineering for Women
Visakhapatnam, India
dakshugarimella@gmail.com

Eswari Chirukuri
Department of Computer Scienece and Engineering
GVP College of Engineering for Women
Visakhapatnam, India
eswarichirukuri@gmail.com

Amrutha Ariselli
Department of Computer Scienece and Engineering
GVP College of Engineering for Women
Visakhapatnam, India
swarupaamrutha@gmail.com

Abstract— DeepFakes is a serious concern in recent times that affects society and steals people's identities by disseminating false images. A detailed investigation is essential, since DeepFakes are a relatively recent problem. The notion of DeepFake Detection using Support Vector Machines (SVM) is discussed in this work. Using methods like Generative Adversarial Networks (GAN) and Autoencoders, fake computer-generated pictures known as DeepFakes are created. Scandalous images of both celebrities and average individuals may be produced using these algorithms. It is practically very difficult to identify the falsified images with the naked eye since they look so authentic. To uncover the distinguishing characteristics between false and real photos, the proposed approach pre-processes the images using image processing. In order to identify between a fake and a real image, the SVM classifier is then trained on these detecting features. Only celebrity facial image data were used to train the model. The three implemented classifiers are as follows: 1) Linear Kernel 2) Gaussian Radial Basis Function Kernel (RBF) 3) The Polynomial Kernel. The Polynomial Kernel classifier generated the most precise findings, with a 99.975% accuracy.

Keywords— DeepFake, GAN generated images, SVM, classification.

I. INTRODUCTION

People have long been intrigued by the altering of photo, audio, and video information. While manipulating images is now very simple, audio and video alteration is still be daunting. It originally came to public attention when Face Swap Machine Learning algorithms were used to project the faces of celebrities into obscene films. One notable example of these approaches is a spoof film of President Obama warning about DeepFakes [21] when in reality it was a completely different person, this is depicted in Figure 1.

There are several methods for making completely computer-generated images that seem eerily similar to real photographs. Professionals may make such photos and videos, but there are tools available that allow even untrained people to create solid DeepFakes. One such app is Zao, [22] a Chinese picture and video editing software that has taken the internet by surprise. It enables the creation of DeepFake videos in seconds, with the results appearing incredibly real.

DeepFakes are already a severe issue because of their great capacity to increasingly resemble people in terms of image, video, and audio. As a result, it is critical to start looking for feasible solutions to this problem.

There are various dangerous circumstances in which DeepFakes might be used to harm our society and the people in it such as identity theft [21] [23], hence research into identifying these manipulations is critical. Another instance of DeepFakes that came to light was the incident of a person who spent months to DeepFake himself to look like Tom Cruise. From a technology viewpoint, this topic is incredibly exciting. It is a particularly active field of study because of its novelty, both in terms of generation and detection of images, videos and audio.

The method of creating DeepFakes is also highly fascinating in order to fully understand their limitations. Autoencoders and Generative Adversarial Networks are two generative learning methods that are extensively used in generating DeepFakes.

Figure 1: Real and DeepFake image comparison [28]

The proposed method uses simple supervised classifier after analyzing the frequency domain to distinguish between a fake and a real image. The steps used for the preprocessing the data are the frequency domain analysis steps of discrete Fourier transform and azimuthal average. It uses SVM Linear, Gaussian RBF and Polynomial Kernels for classification which exhibit a higher rate of accuracy unlike most models that were researched which primarily use Neural Networks and image processing as a pre-processing step.

The works related to the DeepFake Detection, methodology employed in the proposed model and the experimentation and the results of the same are discussed in the subsequent sections.

II. LITERATURE SURVEY

Katarya and Lal, A Study on Combating Emerging Threat of Deepfake Weaponization, proposed a deepfake detection model based on SSTNets that use spatial, temporal, and steganalysis for detection and GANs and autoencoders were used for the deepfake image creation [1]. Nevertheless, more tools and their upgraded versions are required to assist in the detection process. The SSTNet model provides flexibility and can be used to classify both fake videos and images. However, the model's accuracy is at 90%.

Aduwala et al., Deepfake Detection using GAN Discriminators, explored a solution for a deepfake detection system based on GAN discriminators to detect Deepfake videos. Using MesoNet as a baseline, a GAN was trained, and a discriminator was extracted as a dedicated module to detect Deepfakes [2]. The results indicate that the discriminator accuracy on known datasets is high while accuracy on never-before seen datasets is low. Nonetheless, a GAN discriminator is not robust enough to be used as a general-purpose Deepfake detector.

Agarwal et al., Deepfake Detection Using SVM, introduced requency domain analysis after which a classifier will be used to differentiate the real and fake image [3]. DFTs were used to pre-process the collected datasets. The proposed pipeline does not include nor requires huge amounts of information. This method can distinguish the high resolution deepfake images with 99.76% accuracy. However, the acceptable images can only be square shaped.

Güera and Delp, Deepfake Video Detection Using Recurrent Neural Networks, proposed a temporal-aware pipeline to automatically detect deepfake videos [4]. [5] [20]. The system used a convolutional neural network (CNN) to extract frame-level features. These features are then used to train a recurrent neural network (RNN) that learns to classify whether a video has been subject to manipulation or not. The experimental results using a large collection of manipulated videos had shown that using a simple convolutional LSTM structure can accurately predict if a video has been subject to manipulation or not with as few as 2 seconds of video data. However, the robustness if this model is debatable.

Li et al., DeepFake-o-meter: An Open Platform for DeepFake Detection, proposed a Deepfake-o-meter which is an online platform where for developers of deepfake detection algorithm it provides an API architecture to wrap individual algorithm and run on third party remote server- Frontend, Backend and Data Synchronizing [6].

Rana et al., Deepfake Detection Using Machine Learning Algorithms, explored a process that involves creating a unique set of features by combining HOG, Haralic, Hu Moments, and Color Histogram features [7] that makes patterns recognizable by splitting a task into two stages: object detection and object recognition. The object detection phase scans an entire image and identifies all possible objects. The object recognition step identifies relevant objects. a CNN that extracts the most critical features and an LSTM for sequential analysis.

Rafique, et al., DeepFake Detection Using Error Level Analysis and Deep Learning, in their paper proposed the model using convolution neural network (CNN) models including Alex Net and Shuffle Net [8]. These are used to recognize genuine and counterfeit face images. Two strategies for detecting deepfakes utilizing ELA and DL techniques in this paper.

Khichi and Yadav, A Threat of Deepfakes as a Weapon on Digital Platform and their Detection Methods, in their paper observed that researchers use the face-swapping technique with Generative Adversarial Networks (GANs)to build Deepfakes [9]. While a detector or discriminative network is another. These two networks collaborate to create realistic-looking fake videos. That discriminative network decides whether generated representation is accurate and believable, while the generative system uses two CNNs the Encoder and the Decoder to create images.

Yang et al., Exposing Deep Fakes Using Inconsistent Head Poses, in their approach used an intrinsic limitation in the deep neural network face synthesis models [10]. SVM based classifier is used to differentiate between real and fake images based on eye and mouth.

Li et al., Deepfake Detection Using Robust Spatial and Temporal Features from Facial Landmarks, suggested a novel method for detecting deepfakes using biometric characteristics [11]. A collection of chosen face landmarks was used to generate the biometric traits. The spatial and temporal rotation degrees are included to make the creation of the SVM feature vector easier. It has high AUC (Area Under the Receiver Operating Characteristic Curve) and detection accuracy.

Pan, et al., Deepfake Detection through Deep Learning, in their paper a total of eight deepfake video classification models were trained, evaluated, and compared based on four fake video generation methods and two neural networks [12]. Each model exhibited satisfactory classification performance over the corresponding dataset used to train it. Specifically, for the Xception models, the overall fake detection accuracy was above 90% The NeuralTextures model was an outlier in this test.

[13] Joseph and Nyirenda, Deepfake Detection using a Two-Stream Capsule Network, in their paper addressed the challenge of Deepfake Detection in their study by operating a Two-Stream Capsule Network in parallel that takes in both RGB pictures or frames and Error Level Analysis images [13]. The suggested technique achieves detection accuracy of 73.39% for the Deepfake Detection Challenge (DFDC) and 57.45% for the Celeb-DF datasets. However, these results are from a preliminary application of the proposed technique.

Deepfake detection was defined as a one-class anomaly detection issue by Khalid and Woo, OC-FakeDect: Classifying Deepfakes Using One-class Variational Autoencoder [14]. They propose OC-FakeDect, which employs a one-class Variational Autoencoder (VAE) to train solely on actual face photos and detects non-real images as anomalies. Their first results demonstrate that our one class-based technique can be promising for identifying Deepfakes, obtaining 97.5% accuracy on the well-known FaceForensics++ benchmark dataset's NeuralTextures data without utilizing any fake photos in the training procedure.

Ajoy et al., DeepFake Detection using a frame-based approach involving CNN, [15]. Pixel distortion, discrepancies with facial superimposition, skin color variances, blurring, and other visual aberrations are among these distinguishing traits. The proposed model has trained a CNN (Convolutional Neural Network) to efficiently differentiate DeepFakes based on various attributes using a frame-based technique. The model's initial findings, acquired after being trained on around 700 films, are encouraging, with a validation accuracy of 85.8%,

Guo et al., Robust Attentive Deep Neural Network for Detecting GAN-Generated Faces, in their article proposed architecture for GAN-generated face detection [16]. They used DLib to detect faces and localize eyes, and use Mask R-CNN to segment out the iris regions. A Residual Attention Network (RAN) then performs binary classification on the extracted iris pair to determine if the face is real or fake. The training is carried out using a joint loss combining the Binary Cross-Entropy (BCE) loss and the ROC-AUC loss with WMW relaxation for better accuracy

Ma et al., Multi-Perspective Dynamic Features for Cross-Database Face Presentation Attack Detection, proposed a novel scheme to extract the facial motion patterns in a video by analyzing the pixel value changes throughout the video [17]. The scheme globally maps the temporal motion information into a single image by integrating each pixel in the time domain.

Zhang et al., No One Can Escape: A General Approach to Detect Tampered and Generated Image, in their article brought forward a single model which is used detect tampered images and GANs generated images [18]. They found a general method which introduces a strong generalization ability into the detection of images generated by various GANs. They put forward a structure with depth wise separable convolutions as the main component which with a smaller number of parameters compared with the one consists of traditional convolution. Compared with other work, this method has good performance in detecting tampered images.

Guarnera et al., Fighting Deepfake by Exposing the Convolutional Traces on Images, in their work offered an algorithm based on Expectation-Maximization to extract the Convolutional Traces (CT) - a unique fingerprint useable to identify the fakes among real, if an image is a Deepfake but also the GAN architecture that generated it [19]. The CT extracted the fingerprint demonstrated to have high discriminative power.

The research papers mentioned above detail the various techniques used to detect DeepFakes. Most of the works use Neural Networks. This paper aims to design and develop a system that makes use of SVM Linear, Gaussian RBF and Polynomial Kernels for the classification of DeepFakes.

III. Methodology

The objective of the model is to distinguish between real and fake photos created by deep learning algorithms such as Autoencoders and GANs. To do this, the Support Vector Machines algorithm, a machine learning method, is used. The dataset employed is a part of the CelebA dataset, which was acquired from the CelebA [24] website and contains over a thousand real and another thousand fake images collected from the web for training the model.

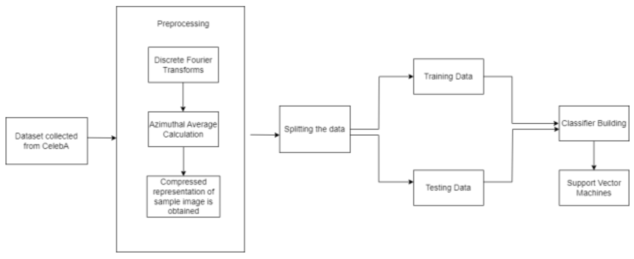

Preprocessing
Discrete Fourier Transforms
Azimuthal Average Calculation
Compressed representation of sample image is obtained
Dataset collected from CelebA
Splitting the data
Training Data
Testing Data
Classifier Building
Support Vector Machines

Figure 2: Architecture Diagram

The above diagram describes the overall architecture of the model. The model is divided into two modules, 1) Data Pre-processing after which the data is split into Training and Testing Data, 2) Classification. These modules are described in detail in the subsequent sections.

A. Data Pre-processing

The actions taken to prepare pictures before they are used by model training and inference are referred to as image preprocessing. It covers resizing, orienting, and color adjustments, among other things.

Data preprocessing for our project has been carried out in the following major steps: Discrete Fourier Transforms and Azimuthal Average.

1) Discrete Fourier Transforms

The Fourier transforms can be used for image processing. It is used to deconstruct an image into sine and cosine components.

The input image which is in the spatial domain is transformed into the frequency domain. The Fourier transformation is used in a variety of applications, including image reduction and filtering. The frequency spectrum of a signal may be computed using Discrete Fourier Transforms. The values of input and output are discrete values.

2) Azimuthal Average

Azimuthal average is utilized to produce a 1D representation of the magnitude spectrum of the Discrete Fourier Transform (DFT).

B. Classification

The model presented in this paper uses Support Vector Machines for classification.

Support vector machines are a type of supervised learning algorithms for classification, regression, and detecting outliers. SVMs vary from other classification methods in that they select the decision boundary that optimizes the distance from the nearest data points for all classes. The maximum margin classifier or maximum margin hyper plane is the decision boundary generated by SVMs.

A simple linear SVM classifier connects two classes by drawing a straight line between them. That is, all of the data points on one side of the line will be assigned to a category, while the data points located on the other side of the line will be classified to a different class.

1) SVM Kernels

A kernel function is a way for taking input data and transforming it into the needed form of processing data.

The term "kernel" refers to a collection of mathematical functions used in Support Vector Machine to provide a window to alter data.

In general, the Kernel Function modifies the training set of data so that a non-linear decision surface can be transformed to a linear equation in a larger number of dimension spaces. It essentially returns the inner product of two points in a standard feature dimension.[25]

The Kernel Functions used in the model are Linear, Gaussian RBF and Polynomial Functions, this is displayed in Figure 6.

a) Linear Kernel Function

When the data is linearly separable, that is, it can be separated using a single line, a Linear Kernel is applied. It is one of the most commonly used kernels. It is typically employed when a data set has a large number of features. Text Classification is one of the applications of Linear Kernel [27]. It is calculated as follows:

$$K(x_i,x_j)= \Sigma(x_i*x_j) \quad (1)$$

b) Gaussian Radial Basis Function Kernel

It is typically used with non-linear data. When there is no prior information about the data, it is applied to perform transformation. It improves the transformation by using the radial basis approach. It is computed as follows:

$$K(x, y) = e^{-\| x-y\|2/2\sigma2} \quad (2)$$

c) Polynomial Function

It is a more general description of the linear kernel. It shows the similarity of vectors in the training set of data in a feature space over polynomials of the kernel's original variables. It is determined as follows:

$$K(x, y) = (x*y+1)^d \quad (3)$$

IV. EXPERIMENTATION AND RESULTS

A. Dataset

There are over a thousand celebrity images of resolution 1024*1024 with 722
variations in facial attribute in CelebA Attributes Dataset as real images and another thousand DeepFaked images. 80% of the data is utilized for training, and the unused 20% is used for testing.

Figure 3: Real images

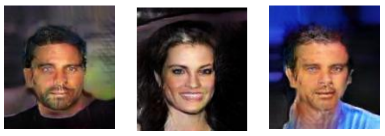


Figure 4: Fake images

B. Training

Real and fake photos are utilized to train the model. The colorful image data is first converted to its greyscale counterpart. The dataset is then subjected to the data preprocessing stages of Discrete Fourier Transforms and Azimuthal Average. The data set is then divided into training and testing data. To train the model, the classification algorithm of Support Vector Machines is used, in which Linear, Gaussian Radial Basis Function and Polynomial Kernels are used

C. Power Spectrum Analysis

Images from their corresponding collection, i.e., fake images and real images, each have distinctive features.

Real and fake pictures may easily be distinguished at high frequencies since they each have their own distinct spectrum. Figure 5 presents the power spectrum of real and fake images taking the data as a whole.

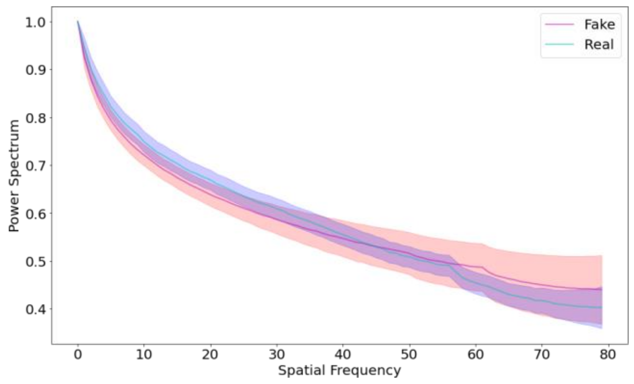
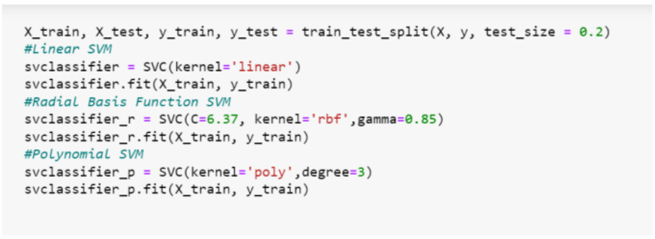


Figure 5: 1D Power Spectrum of the dataset

D. Results

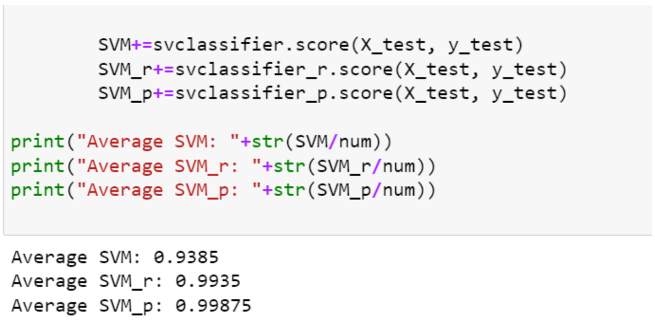
The linear kernel was utilized to analyze the model's behavior and sensitivity and was used as a baseline model with an accuracy of 94.45% on the given dataset. To achieve a higher rate of accuracy the Gaussian Radial Basis Function Kernel was employed. With a gamma value of

0.85, it presents an accuracy of 99.32%. From the above models, it is evident that, the data was not entirely linear and hence requires dimensional transformation. In order to transform the dataset into a higher dimensional space, the Polynomial Kernel was used at a degree of 3 which has resulted in an accuracy of 99.975%, the same has been depicted in Figure 7 below.



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
#Linear SVM
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
#Radial Basis Function SVM
svclassifier_r = SVC(C=6.37, kernel='rbf',gamma=0.85)
svclassifier_r.fit(X_train, y_train)
#Polynomial SVM
svclassifier_p = SVC(kernel='poly',degree=3)
svclassifier_p.fit(X_train, y_train)
```

Figure 6: Kernel Implementation



```
        SVM+=svclassifier.score(X_test, y_test)
        SVM_r+=svclassifier_r.score(X_test, y_test)
        SVM_p+=svclassifier_p.score(X_test, y_test)

print("Average SVM: "+str(SVM/num))
print("Average SVM_r: "+str(SVM_r/num))
print("Average SVM_p: "+str(SVM_p/num))
```

```
Average SVM: 0.9385
Average SVM_r: 0.9935
Average SVM_p: 0.99875
```

Figure 7: Kernel accuracies

V. CONCLUSION

DeepFakes are a social menace that disrupt the peace of mind of the victims and can have terrible consequences. The deleterious effects of deepfakes that might cause unnecessary trauma for individuals in society must be eliminated. as they create realistic looking fake images that must be identified. The proposed system achieves the premise of DeepFake Detection by using some of the best SVM Kernels and image processing. This paper has focused exclusively on celebrity facial image data and has applied Support Vector Machines. However, more work needs to be done on different datasets to improve the model behavior to all kinds of facial image data.

REFERENCES

[1] R. Katarya and A. Lal, "A Study on Combating Emerging Threat of Deepfake Weaponization," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 485-490, doi: 10.1109/I-SMAC49090.2020.9243588.

[2] S. A. Aduwala, M. Arigala, S. Desai, H. J. Quan, and M. Eirinaki, "Deepfake Detection using GAN Discriminators," 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), 2021, pp. 69-77, doi: 10.1109/BigDataService52369.2021.00014.

[3] H. Agarwal, A. Singh, and R. D, "Deepfake Detection Using SVM," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 1245-1249, doi: 10.1109/ICESC51422.2021.9532627.

[4] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018, pp. 1-6, doi: 10.1109/AVSS.2018.8639163.

[5] Y. Al-Dhabi and S. Zhang, "Deepfake Video Detection by Combining Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)," 2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE), 2021, pp. 236-241, doi: 10.1109/CSAIEE54046.2021.9543264.

[6] Y. Li et al., "DeepFake-o-meter: An Open Platform for DeepFake Detection," 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 277-281, doi: 10.1109/SPW53761.2021.00047.

[7] M. S. Rana, B. Murali and A. H. Sung, "Deepfake Detection Using Machine Learning Algorithms," 2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI), 2021, pp. 458-463, doi: 10.1109/IIAI-AAI53430.2021.00079.

[8] R. Rafique, M. Nawaz, H. Kibriya and M. Masood, "DeepFake Detection Using Error Level Analysis and Deep Learning," 2021 4th International Conference on Computing & Information Sciences (ICCIS), 2021, pp. 1-4, doi: 10.1109/ICCIS54243.2021.9676375.

[9] M. Khichi and R. Kumar Yadav, "A Threat of Deepfakes as a Weapon on Digital Platform and their Detection Methods," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021, pp. 01-08, doi: 10.1109/ICCCNT51525.2021.9580031.

[10] X. Yang, Y. Li, and S. Lyu, "Exposing Deep Fakes Using Inconsistent Head Poses," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 8261-8265, doi: 10.1109/ICASSP.2019.8683164.

[11] M. Li, B. Liu, Y. Hu, L. Zhang, and S. Wang, "Deepfake Detection Using Robust Spatial and Temporal Features from Facial Landmarks," 2021 IEEE International Workshop on Biometrics and Forensics (IWBF), 2021, pp. 1-6, doi: 10.1109/IWBF50991.2021.9465076.

[12] D. Pan, L. Sun, R. Wang, X. Zhang and R. O. Sinnott, "Deepfake Detection through Deep Learning," 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), 2020, pp. 134-143, doi: 10.1109/BDCAT50828.2020.00001.

[13] Z. Joseph and C. Nyirenda, "Deepfake Detection using a Two-Stream Capsule Network," 2021 IST-Africa Conference (IST-Africa), 2021, pp. 1-8.

[14] H. Khalid and S. S. Woo, "OC-FakeDect: Classifying Deepfakes Using One-class Variational Autoencoder," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 2794-2803, doi: 10.1109/CVPRW50498.2020.00336.

[15] A. Ajoy, C. U. Mahindrakar, D. Gowrish, and V. A, "DeepFake Detection using a frame-based approach involving CNN," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021, pp. 1329-1333, doi: 10.1109/ICIRCA51532.2021.9544734.

[16] H. Guo, S. Hu, X. Wang, M. -C. Chang and S. Lyu, "Robust Attentive Deep Neural Network for Detecting GAN-Generated Faces," in IEEE Access, vol. 10, pp. 32574-32583, 2022, doi: 10.1109/ACCESS.2022.3157297.

[17] Y. Ma, Y. Xu and F. Liu, "Multi-Perspective Dynamic Features for Cross-Database Face Presentation Attack Detection," in IEEE Access, vol. 8, pp. 26505-26516, 2020, doi: 10.1109/ACCESS.2020.2971224.

[18] K. Zhang, Y. Liang, J. Zhang, Z. Wang and X. Li, "No One Can Escape: A General Approach to Detect Tampered and Generated Image," in IEEE Access, vol. 7, pp. 129494-129503, 2019, doi: 10.1109/ACCESS.2019.2939812.

[19] L. Guarnera, O. Giudice and S. Battiato, "Fighting Deepfake by Exposing the Convolutional Traces on Images," in IEEE Access, vol. 8, pp. 165085-165098, 2020, doi: 10.1109/ACCESS.2020.3023037.

[20] E. Kim and S. Cho, "Exposing Fake Faces Through Deep Neural Networks Combining Content and Trace Feature Extractors," in IEEE Access, vol. 9, pp. 123493-123503, 2021, doi: 10.1109/ACCESS.2021.3110859.

[21] https://www.businessinsider.in/tech/a-video-that-appeared-to-show-obama-calling-trump-a-dipsh-t-is-a-warning-about-a-disturbing-new-trend-called-deepfakes/articleshow/63807263.cms

[22] https://zao.en.softonic.com/android

[23] https://kslnewsradio.com/1945169/what-is-a-deep-fake-how-how-do-i-spot-one-and-why-should-i-care/

[24] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[25] https://www.geeksforgeeks.org/major-kernel-functions-in-support-vector-machine-svm

[26] https://data-flair.training/blogs/svm-kernel-functions/

[27] https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/

[28] https://www.vox.com/2018/4/18/17252410/jordan-peele-obama-deepfake-buzzfeed