**DeepFake Video detection Using Deep Learning**

**ABSTRACT**

The proliferation of deepfake technology, which uses artificial intelligence to create highly realistic synthetic videos and images, poses significant threats to privacy, security, and trust in digital media. Traditional methods for detecting these manipulations often fall short due to the sophisticated nature of deepfake algorithms. This paper proposes a novel approach for deepfake face detection using Deep Learning (DL) well-suited for sequential data analysis. Our method leverages the temporal dependencies and patterns inherent in video sequences to identify subtle inconsistencies and artifacts introduced by deepfake generation processes. By analyzing frames in a sequence rather than in isolation, the DL can capture dynamic facial features and movements that are difficult to replicate accurately in deepfakes. The proposed model is trained on a comprehensive dataset of real and deepfake videos, incorporating various scenarios and levels of manipulation. Experimental results demonstrate that our DL-based approach achieves superior accuracy and robustness compared to state-of-the-art deepfake detection techniques, particularly in challenging cases with high-quality deepfakes. Furthermore, the model exhibits strong generalization capabilities across different datasets and deepfake generation methods. This research highlights the potential of DL for enhancing the detection of deepfake content, contributing to the development of more secure and trustworthy digital media platforms.

**CHAPTER 1**

**INTRODUCTION**

**1.1 AIM OF THE PROJECT**

The primary aim of this project is to develop a robust and effective deepfake face detection system utilizing DL . By leveraging the temporal dependencies in video sequences, the project seeks to identify and distinguish deepfake videos from genuine ones. The specific objectives include: Temporal Analysis of Video Sequences: To utilize LSTM networks for analyzing the sequential frames in videos, capturing dynamic facial features and movements that are challenging to replicate accurately in deepfakes.

**1.2 SCOPE OF THE PROJECT**

The scope of this project encompasses several critical aspects of developing, evaluating, and deploying an effective deepfake face detection system using DL . The key elements of the project's scope include: Data Collection and Preprocessing: Compilation of a comprehensive dataset consisting of real and deepfake videos from various sources.

**1.3 OBJECT OF THE PROJECT**

The primary objective of this project is to design and develop an advanced deepfake face detection system using DL. The specific objectives are as follows: Develop an LSTM-Based Detection Model: Create an DL architecture that can effectively analyze the temporal patterns in video sequences to identify deepfake manipulations.

**1.4 Introduction**

Temporal Analysis of Video Sequences: To utilize DL for analyzing the sequential frames in videos, capturing dynamic facial features and movements that are challenging to replicate accurately in deepfakes. Enhancement of Detection Accuracy: To improve the accuracy and robustness of deepfake detection methods by leveraging the capabilities of DL in identifying subtle inconsistencies and artifacts introduced by deepfake generation processes.

Comprehensive Dataset Utilization: To train and validate the LSTM-based model on a diverse and extensive dataset of real and deepfake videos, ensuring the model's effectiveness across various scenarios and manipulation levels. Generalization Across Datasets: To ensure the model's strong generalization capabilities by testing its performance on different datasets and deepfake generation methods, demonstrating its applicability in real-world scenarios.

Contributing to Digital Media Security: To enhance the security and trustworthiness of digital media platforms by providing a reliable tool for detecting and mitigating the spread of deepfake content.

Data Collection and Preprocessing: Compilation of a comprehensive dataset consisting of real and deepfake videos from various sources. Preprocessing of video frames to ensure uniformity in size, format, and quality for effective training and evaluation of the LSTM model. Model Development: Design and implementation of an LSTM network architecture tailored for temporal analysis of video sequences. Integration of additional neural network layers (e.g., convolutional layers) if necessary to enhance feature extraction and improve detection performance.

Training and Optimization: Training the LSTM model using the preprocessed dataset, with a focus on optimizing hyperparameters to achieve the best possible detection accuracy. Utilization of techniques such as data augmentation and regularization to prevent overfitting and improve generalization. Evaluation and Validation: Rigorous testing of the trained LSTM model on a separate validation set to assess its accuracy, robustness, and generalization capabilities.

Comparison of the LSTM-based approach with existing state-of-the-art deepfake detection methods to highlight its advantages and limitations. Performance Metrics: Definition and use of relevant performance metrics (e.g., accuracy, precision, recall, F1-score) to quantify the effectiveness of the detection system. Analysis of the model's performance across different types of deepfake techniques and varying levels of video quality. Deployment Considerations: Exploration of potential deployment scenarios for the deepfake detection system, including integration with social media platforms, video streaming services, and forensic tools.

Consideration of computational efficiency and real-time processing capabilities to ensure practical applicability. Future Enhancements: Identification of areas for further improvement, such as incorporating additional types of neural networks (e.g., Transformer models) or exploring hybrid approaches combining LSTM with other machine learning techniques. Proposing extensions of the project to address emerging deepfake technologies and evolving manipulation techniques.

Ethical and Legal Implications: Examination of the ethical considerations related to deepfake detection, including privacy concerns and the potential impact on freedom of expression. Understanding the legal framework surrounding the use of deepfake detection technologies and ensuring compliance with relevant regulations. By addressing these elements, the project aims to develop a comprehensive and effective solution for deepfake face detection using LSTM networks, contributing to the broader efforts to safeguard digital media integrity and security.

Develop an LSTM-Based Detection Model: Create an LSTM network architecture that can effectively analyze the temporal patterns in video sequences to identify deepfake manipulations. Enhance Detection Accuracy: Achieve high accuracy in distinguishing deepfake videos from authentic ones by leveraging the LSTM network's ability to capture subtle inconsistencies and artifacts.

Construct a Comprehensive Dataset: Gather and preprocess a large dataset of both real and deepfake videos to train and evaluate the LSTM model, ensuring it performs well across various scenarios and manipulation techniques. Optimize Model Performance: Fine-tune the LSTM model's hyperparameters and employ techniques such as data augmentation and regularization to improve detection robustness and prevent overfitting.

Evaluate and Validate the Model: Conduct thorough testing on a separate validation set to assess the model's accuracy, precision, recall, and F1-score, ensuring reliable performance across different types of deepfakes and video qualities. Compare with Existing Methods: Benchmark the LSTM-based approach against current state-of-the-art deepfake detection methods to highlight its strengths and identify areas for improvement.

Ensure Real-World Applicability: Consider computational efficiency and real-time processing capabilities for practical deployment scenarios, such as integration with social media platforms and video streaming services. Address Ethical and Legal Concerns: Analyze the ethical implications of deepfake detection technology, focusing on privacy and freedom of expression, and ensure compliance with relevant legal regulations.

Propose Future Enhancements: Identify potential improvements and extensions for the deepfake detection system, including exploring hybrid approaches and incorporating emerging neural network technologies. By achieving these objectives, the project aims to create a robust, accurate, and practical deepfake face detection system using LSTM networks, contributing to the protection of digital media integrity and security.

In the recent years, the deep learning (DL) computing has been become the top standard in the machine learning community. Deep learning progress has led to the development of software, like deepfakes, that poses threats to privacy, democracy, and national security. In a narrow definition, deepfake refers to manipulated digital media such as images of videos where the image or video of a person is replaced with another person’s likeness [1]. Specifically Deep learning algorithms are employed to fabricate or alter video and audio content, making it seem as if an individual is saying or doing something they never actually did.

Deepfake technology utilizes advanced AI algorithms to create manipulated videos or audio of a person, simulating their speech and actions. It's often used for humor, pornographic, or political purposes, presenting significant ethical concerns related to privacy, consent, and misinformation. Addressing these issues requires a comprehensive approach, including regulatory measures, public awareness, and responsible AI usage to mitigate potential harm [2]. Deepfakes specifically aim at social media platforms, taking advantage of the ease with which conspiracies, rumors, and misinformation can spread due to users' tendency to follow the crowd.

The use of advanced deep neural networks and the abundance of data has made the manipulated images and videos nearly indistinguishable, fooling both humans and even advanced computer algorithms. This study promotes the research in the field of deep learning and deepfake detection. The main focus is on Long-Short Term Memory (LSTM), Convolution Neural Network (CNN), Recurrent Neural Network (RNN) algorithm which is specified to use in deepfake detection.

**1.5 Deep Learning**

**1.5.1 Deep Learning:**

Deep learning is a subset of machine learning that focuses on training deep neural networks with multiple layers to learn and represent complex patterns in data. Deep neural networks are composed of interconnected layers of artificial neurons that simulate the structure and functioning of the human brain.

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets.

Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data.

**1.5.2 Key aspects of deep learning include:**

* **Neural Networks:** Deep learning relies on neural networks with multiple hidden layers, allowing the network to learn hierarchical representations of the data. Each layer in the network extracts higher-level features from the representations learned in the previous layer. Deep neural networks can automatically learn and extract relevant features from raw data, eliminating the need for manual feature engineering.
* **Training Process:** Deep learning models are trained through a process called back propagation, where the network adjusts its internal parameters (weights and biases) to minimize the difference between the predicted output and the target output. This process involves propagating errors backward through the network and updating the parameters using gradient descent optimization algorithms.
* **Large-Scale Data:** Deep learning models typically require a large amount of labeled data for training. The availability of big data and advances in computing power have enabled the success of deep learning models. The large-scale data allows deep neural networks to learn complex representations and generalize well to new, unseen data.
* **Applications:** Deep learning has shown remarkable performance in various fields, including computer vision, natural language processing, speech recognition, and recommendation systems. It has achieved state-of-the-art results in tasks such as image classification, object detection, machine translation, and speech synthesis.
* **Deep learning excels** in handling complex and high-dimensional data, capturing intricate patterns, and achieving state-of-the-art performance in many AI tasks. However, it typically requires more computational resources and data compared to traditional machine learning approaches.

In summary, machine learning focuses on training algorithms to learn patterns and make predictions or decisions, while deep learning is a specific approach within machine learning that utilizes deep neural networks to learn complex representations. Deep learning has gained significant attention and has been particularly successful in solving tasks that involve complex data such as images, audio, and text.

Lane detection using deep learning is a popular approach that leverages the power of deep neural networks to detect and track lane markings on the road. Deep learning models excel in learning complex patterns and can effectively capture the distinctive characteristics of lane markings, making them well-suited for this task. Here is a high-level overview of the lane detection process using deep learning:

* **Dataset Preparation:** The first step is to collect or create a dataset of labeled images or videos, where the lane markings are manually annotated. The annotations typically involve marking the pixels or regions corresponding to the lane markings in the images or videos.
* **Data Pre-processing:** The collected dataset is pre-processed to prepare it for training. This may involve resizing the images, normalizing pixel values, and splitting the dataset into training and validation sets.
* **Model Architecture:** A deep learning model architecture needs to be selected or designed for lane detection. Convolutional Neural Networks (CNNs) are commonly used due to their ability to capture spatial dependencies in images. The model architecture may consist of multiple convolutional layers followed by pooling, fully connected layers, and output layers.
* **Training:** The deep learning model is trained using the labeled dataset. The training process involves feeding the input images into the model, comparing the predicted output (lane markings) with the ground truth annotations, and updating the model's weights through back propagation and gradient descent optimization algorithms. The objective is to minimize the difference between the predicted output and the ground truth annotations.
* **Post-processing:** Once the model is trained, the lane detection results may undergo post-processing steps to refine the detected lane markings. This may include techniques such as filtering outliers, curve fitting, and extrapolation to extend the detected lanes.
* **Evaluation and Testing:** The trained model is evaluated on a separate test dataset to assess its performance. Evaluation metrics such as accuracy, precision, recall, and F1 score can be used to measure the model's lane detection performance.
* **Deployment:** The trained lane detection model can be deployed in real-time applications, such as autonomous vehicles or advanced driver-assistance systems (ADAS), to detect and track lane markings in real-world scenarios.

It's worth noting that there are different variations and approaches for lane detection using deep learning, including single-image-based methods and video-based methods. Additionally, techniques like semantic segmentation and instance segmentation can also be employed to precisely detect and differentiate lane markings from other objects on the road.

Deep learning-based lane detection has shown promising results and has been successfully applied in various real-world applications. However, it's important to fine-tune and validate the model on diverse datasets and consider factors such as different weather conditions, road types, and lighting variations to ensure robust and reliable lane detection performance.

**1.5.3 Application of Deep learning**

Deep learning is a subset of machine learning that uses artificial neural networks (ANNs) to model and solve complex problems. It is based on the idea of building artificial neural networks with multiple layers, called deep neural networks, that can learn hierarchical representations of the data.

Deep learning algorithms use a layered architecture, where the input data is passed through an input layer and then propagated through multiple hidden layers, before reaching the output layer. Each layer applies a set of mathematical operations, called weights and biases, to the input data, and the output of one layer serves as the input to the next.

The process of training a deep learning model involves adjusting the weights and biases of the model to minimize the error between the predicted output and the true output. This is typically done using a variant of gradient descent, an optimization algorithm that adjusts the weights and biases in the direction of the steepest decrease in the error.

Deep learning has a wide range of applications, including image and speech recognition, natural language processing, and computer vision. One of the main advantages of deep learning is that it can automatically learn features from the data, which means that it doesn’t require the features to be hand-engineered. This is particularly useful for tasks where the features are difficult to define, such as image recognition.

**1.5.4 Advantages of Deep Learning:**

Deep learning has several advantages over traditional machine learning methods, some of the main ones include:

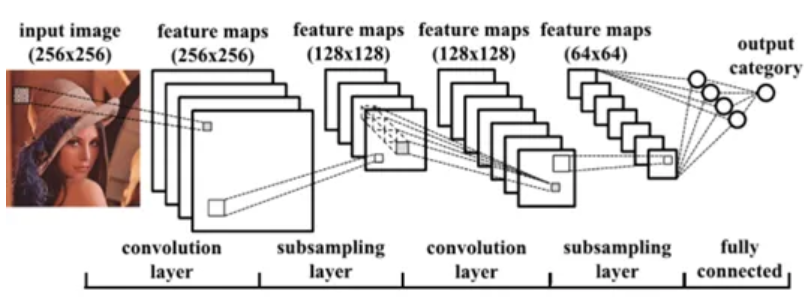
1. **Automatic feature learning:**Deep learning algorithms can automatically learn features from the data, which means that they don’t require the features to be hand-engineered. This is particularly useful for tasks where the features are difficult to define, such as image recognition.
2. **Handling large and complex data:** Deep learning algorithms can handle large and complex datasets that would be difficult for traditional machine learning algorithms to process. This makes it a useful tool for extracting insights from big data.
3. **Improved performance:**Deep learning algorithms have been shown to achieve state-of-the-art performance on a wide range of problems, including image and speech recognition, natural language processing, and computer vision.
4. **Handling non-linear relationships:**Deep learning can uncover non-linear relationships in data that would be difficult to detect through traditional methods.
5. **Handling structured and unstructured data:** Deep learning algorithms can handle both structured and unstructured data such as images, text, and audio.
6. **Predictive modeling:**Deep learning can be used to make predictions about future events or trends, which can help organizations plan for the future and make strategic decisions.
7. **Handling missing data:** Deep learning algorithms can handle missing data and still make predictions, which is useful in real-world applications where data is often incomplete.
8. **Handling sequential data:** Deep learning algorithms such as Recurrent Neural Networks (RNNs) and Long Short-term Memory (LSTM) networks are particularly suited to handle sequential data such as time series, speech, and text. These algorithms have the ability to maintain context and memory over time, which allows them to make predictions or decisions based on past inputs.
9. **Scalability:** Deep learning models can be easily scaled to handle an increasing amount of data and can be deployed on cloud platforms and edge devices.
10. **Generalization:**Deep learning models can generalize well to new situations or contexts, as they are able to learn abstract and hierarchical representations of the data.

There are three primary categories of CNN layers in total:

1) Convolutional layer

2) Pooling layer

3) Fully-connected (FC) layer



As can be seen in the above figure, a feature map is the result of the input image passing through the convolution process in the convolution layer. When the feature map reaches the last layer, a fully linked layer where the input is processed to return a probability between 0 and 1, it has already undergone subsampling in the Pooling layer (subsampling layer), which essentially reduces the size by half.

The CNN becomes more complicated with each layer, recognizing a larger area of the image. Previous layers emphasize basic elements like borders and colors. Larger components or shapes of the item are first recognized by the image data as it moves through the layers of the CNN, and eventually it recognizes the intended object.

**1) Convolutional Layer**

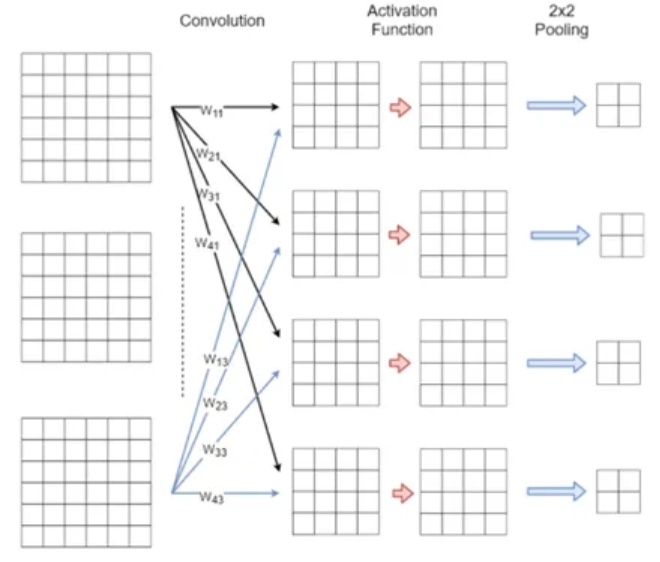
The act of projecting a filter known as a kernel across an image and carrying out mathematical operations known as convolution to create a feature map is referred to as the convolution process. It’s simpler to display this as an figure:



**2) Pooling Layer**

The pooling layer receives the feature maps produced by the convolutional layer afterward. The process of downsampling feature maps, which involves lowering their width and height, is mostly dependent on pooling layers. Reducing the number of dimensions is critical in order to maintain translation invariance, manage computational complexity, and highlight significant local characteristics in the feature maps.

A single convolution and pooling sequence is depicted in the graphic below.



While the pooling operation sweeps a filter across the entire input, it differs from the convolutional layer in that the filter is weightless. Rather, the values in the receptive field are subjected to an aggregation function by the kernel, which then fills the output array. Additionally, in a pooling layer, the kernel typically does not overlap.

Two primary categories of pooling exist:

**Max pooling:** The filter chooses the pixel with the highest value to send to the output array as it passes through the input. In addition, this method is typically applied more frequently than ordinary pooling.

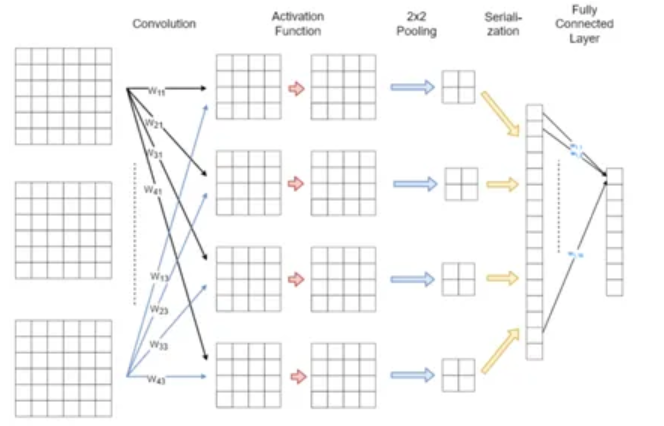
**Average pooling:** The filter determines the average value in the receptive field as it passes through the input and sends it to the output array.

**3) Fully Connected Layer**

Features continue to be the end product of the convolution/pooling layer combo. In order to categorize or reach a decision, we must consider every piece of information or feature we have so far gathered and consider every combination that might exist. The Fully Connected layer, which is essentially our conventional neural network that we learned before CNN and in which every node is connected to every other node, is responsible for this. The output of the final pooling layer is serialized before being fed into a fully linked layer, as the image below illustrates.

There is just one completely connected layer used in this example. This sample architecture can be used for a categorization into 8 classes because the fully linked layer’s output has 8 neurons. In this instance, a softmax-activation function — which isn’t shown in the graphic below — processes the output.

Be aware that only One-dimensional data is accepted by the fully connected layer. Our 3D data can be converted to 1D using Python’s flatten function. Our 3D volume is effectively arranged into a 1D vector as a result.



**CHAPTER 2**

**LITERATURE SURVEY**

**[1] Nguyen, T.T., Nguyen, Q.V.H., Nguyen, D.T., Nguyen, D.T., Huynh-The, T., Nahavandi, S., Nguyen, T.T., Pham, Q.V. and Nguyen, C.M., 2022. Deep learning for deepfakes creation and detection: A survey. Computer Vision and Image Understanding, 223, p.103525.**

Deep learning has been successfully applied to solve various complex problems ranging from big data analytics to computer vision and human-level control. Deep learning advances however have also been employed to create software that can cause threats to privacy, democracy and national security. One of those deep learning-powered applications recently emerged is deepfake. Deepfake algorithms can create fake images and videos that humans cannot distinguish them from authentic ones.

The proposal of technologies that can automatically detect and assess the integrity of digital visual media is therefore indispensable. This paper presents a survey of algorithms used to create deepfakes and, more importantly, methods proposed to detect deepfakes in the literature to date. We present extensive discussions on challenges, research trends and directions related to deepfake technologies.

By reviewing the background of deepfakes and state-of-the-art deepfake detection methods, this study provides a comprehensive overview of deepfake techniques and facilitates the development of new and more robust methods to deal with the increasingly challenging deepfakes. In a narrow definition, deepfakes (stemming from “deep learning” and “fake”) are created by techniques that can superimpose face images of a target person onto a video of a source person to make a video of the target person doing or saying things the source person does.

This constitutes a category of deepfakes, namely faceswap. In a broader definition, deepfakes are artificial intelligence-synthesized content that can also fall into two other categories, i.e., lip-sync and puppet-master. Lip-sync deepfakes refer to videos that are modified to make the mouth movements consistent with an audio recording. Puppet-master deepfakes include videos of a target person (puppet) who is animated following the facial expressions, eye and head movements of another person (master) sitting in front of a camera [1].

While some deepfakes can be created by traditional visual effects or computer-graphics approaches, the recent common underlying mechanism for deepfake creation is deep learning models such as autoencoders and generative adversarial networks (GANs), which have been applied widely in the computer vision domain [2– 8].

**[2] Westerlund, M., 2019. The emergence of deepfake technology: A review. Technology innovation management review, 9(11).**

Novel digital technologies make it increasingly difficult to distinguish between real and fake media. One of the most recent developments contributing to the problem is the emergence of deepfakes which are hyper-realistic videos that apply artificial intelligence (AI) to depict someone say and do things that never happened. Coupled with the reach and speed of social media, convincing deepfakes can quickly reach millions of people and have negative impacts on our society.

While scholarly research on the topic is sparse, this study analyzes 84 publicly available online news articles to examine what deepfakes are and who produces them, what the benefits and threats of deepfake technology are, what examples of deepfakes there are, and how to combat deepfakes. The results suggest that while deepfakes are a significant threat to our society, political system and business, they can be combatted via legislation and regulation, corporate policies and voluntary action, education and training, as well as the development of technology for deepfake detection, content authentication, and deepfake prevention.

The study provides a comprehensive review of deepfakes and provides cybersecurity and AI entrepreneurs with business opportunities in fighting against media forgeries and fake news. In recent years, fake news has become an issue that is a threat to public discourse, human society, and democracy (Borges et al., 2018; Qayyum et al., 2019). Fake news refers to fictitious news style content that is fabricated to deceive the public (Aldwairi & Alwahedi, 2018; Jang & Kim, 2018). False information spreads quickly through social media, where it can impact millions of users (Figueira & Oliveira, 2017).

Presently, one out of five Internet users get their news via YouTube, second only to Facebook (Anderson, 2018). This rise in popularity of video highlights the need for tools to confirm media and news content authenticity, as novel technologies allow convincing manipulation of video (Anderson, 2018). Given the ease in obtaining and spreading misinformation through social media platforms, it is increasingly hard to know what to trust, which results in harmful consequences for informed decision making, among other things (Borges et al., 2018; Britt et al., 2019).

Indeed, today we live in what some have called a “post-truth” era, which is characterized by digital disinformation and information warfare led by malevolent actors running false information campaigns to manipulate public opinion (Anderson, 2018; Qayyum et al., 2019; Zannettou et al., 2019).

[3] Thippanna, G., Priya, M.D. and Srinivas, T.A.S., An Effective Analysis of Image Processing with Deep Learning Algorithms. International Journal of Computer Applications, 975, p.8887.

Image processing finds applications in various fields, including medicine, remote sensing, surveillance, entertainment, and scientific research by using various algorithms and techniques. It involves transforming, enhancing, and extracting information from images to improve their quality, interpret their content, or make them suitable for specific applications. Deep learning algorithms are designed to automatically learn hierarchical representations of data through multiple layers of interconnected artificial neurons, known as artificial neural networks.

These networks are organized into input, hidden, and output layers, with each layer consisting of numerous interconnected nodes or units called neurons. Each neuron applies a mathematical operation to the inputs it receives and passes the result to the next layer. When deep learning algorithms are applied to image processing, they can perform a wide range of tasks such as image classification, object detection, image segmentation, image generation, and image enhancement.

Image processing is a field of computer science that deals with analysing, manipulating, and enhancing digital images. Deep learning, on the other hand, is a subfield of machine learning that focuses on training artificial neural networks to automatically learn and extract meaningful representations from data. When combined, deep learning and image processing have revolutionized the way we approach various tasks, such as object recognition, image segmentation, and image generation. Deep learning algorithms excel in image processing tasks due to their ability to automatically learn hierarchical representations from raw image data.

Traditional image processing techniques often require handcrafted features and heuristics, which can be time-consuming and may not generalize well to new datasets. Deep learning, on the other hand, learns these features directly from the data, making it more flexible and adaptable. Convolution Neural Networks (CNNs)[A] are the cornerstone of deep learning in image processing.

CNNs are specifically designed to process grid-like data, such as images. They consist of multiple layers of interconnected neurons that learn to recognize patterns and features at different levels of abstraction. Convolutional layers in CNNs apply filters to input images to extract local features, while pooling layers down sample the feature maps to reduce spatial dimensions

**[4] Indolia, S., Goswami, A.K., Mishra, S.P. and Asopa, P., 2018. Conceptual understanding of convolutional neural network-a deep learning approach. Procedia computer science, 132, pp.679-688.**

Deep learning has become an area of interest to the researchers in the past few years. Convolutional Neural Network (CNN) is a deep learning approach that is widely used for solving complex problems. It overcomes the limitations of traditional machine learning approaches. The motivation of this study is to provide the knowledge and understanding about various aspects of CNN. This study provides the conceptual understanding of CNN along with its three most common architectures, and learning algorithms. This study will help researchers to have a broad comprehension of CNN and motivate them to venture in this field. This study will be a resource and quick reference for those who are interested in this field.

With the rapidly growing demand for learnable machines for solving many complex problems, deep learning has evolved itself as an area of interest to the researchers in the past few years. As researchers tend to mimic human behavior, a major question arises that how do the humans acquire knowledge? The answer to this question is an essential ability of humans i.e. learning, which needs to be incorporated in machines, hence the term machine learning was coined.

Machine learning promises to reduce the efforts by making the machines to learn themselves through past experiences [2] using three approaches of learning namely, learning under supervision, without supervision and semi-supervised learning [4]. The conventional machine learning techniques need feature extraction as the prerequisite, and this requires a domain expert [16].

Furthermore, selection of appropriate features for a given problem is a challenging task. Deep learning techniques overcome the problem of feature selection by not requiring pre-selected features but extracting the significant features from raw input automatically for a problem in hand [24]. Deep learning model consists of a collection of processing layers that can learn various features of data through multiple levels of abstraction [15]. Multiple levels allow the network to learn distinct features.

Deep learning has emerged as an approach for achieving promising results in various applications like image recognition [31], speech recognition [9], topic classification, sentiment analysis [27], language translation, natural language understanding, signal processing [40], face recognition [13], prediction of bioactivity of small molecules [38] etc. There are different deep learning architectures such as deep belief networks, recurrent neural networks, convolution neural networks etc.

**[5] Ralf C. Staudemeyer, “Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks”, arXiv:1909.09586v1 [cs.NE] 12 Sep 2019**

Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) are one of the most powerful dynamic classifiers publicly known. The network itself and the related learning algorithms are reasonably well documented to get an idea how it works. This paper will shed more light into understanding how LSTM-RNNs evolved and why they work impressively well, focusing on the early, ground-breaking publications. We significantly improved documentation and fixed a number of errors and inconsistencies that accumulated in previous publications. To support understanding we as well revised and unified the notation used.

This article is an tutorial-like introduction initially developed as supplementary material for lectures focused on Artificial Intelligence. The interested reader can deepen his/her knowledge by understanding Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) considering its evolution since the early nineties. Todays publications on LSTM-RNN use a slightly different notation and a much more summarized representation of the derivations. Nevertheless the authors found the presented approach very helpful and we are confident this publication will find its audience.

Machine learning is concerned with the development of algorithms that automatically improve by practice. Ideally, the more the learning algorithm is run, the better the algorithm becomes. It is the task of the learning algorithm to create a classifier function from the training data presented. The performance of this built classifier is then measured by applying it to previously unseen data. Artificial Neural Networks (ANN) are inspired by biological learning systems and loosely model their basic functions.

Biological learning systems are complex webs of interconnected neurons. Neurons are simple units accepting a vector of real-valued inputs and producing a single real-valued output. The most common standard neural network type are feed-forward neural networks. Here sets of neurons are organised in layers: one input layer, one output layer, and at least one intermediate hidden layer. Feed-forward neural networks are limited to static classification tasks. Therefore, they are limited to provide a static mapping between input and output. To model time prediction tasks we need a so-called dynamic classifier.

**[6] Güera, D. and Delp, E.J., 2018, November. Deepfake video detection using recurrent neural networks. In 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS) (pp. 1-6). IEEE.**

In recent months a machine learning based free software tool has made it easy to create believable face swaps in videos that leaves few traces of manipulation, in what are known as "deepfake" videos. Scenarios where these realistic fake videos are used to create political distress, blackmail someone or fake terrorism events are easily envisioned. This paper proposes a temporal-aware pipeline to automatically detect deepfake videos. Our system uses 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 if a video has been subject to manipulation or not. We evaluate our method against a large set of deepfake videos collected from multiple video websites. We show how our system can achieve competitive results in this task while using a simple architecture. The first known attempt at trying to swap someone’s face, circa 1865, can be found in one of the iconic portraits of U.S. President Abraham Lincoln. The lithography, as seen in Figure 1, mixes Lincoln’s head with the body of Southern politician John Calhoun. After Lincoln’s assassination, demand for lithographies of him was so great that engravings of his head on other bodies appeared almost overnight [27].

Recent advances [21, 42] have radically changed the playing field of image and video manipulation. The democratization of modern tools such as Tensorflow [6] or Keras [12] coupled with the open accessibility of the recent technical literature and cheap access to compute infrastructure have propelled this paradigm shift. Convolutional autoencoders [38, 37] and generative adversarial network (GAN) [17, 7] models have made tampering images and videos, which used to be reserved to highly-trained professionals, a broadly accessible operation within reach of almost any individual with a computer. Smartphone and desktop applications like FaceApp [1] and FakeApp [2] are built upon this progress.

FaceApp automatically generates highly realistic transformations of faces in photographs. It allows one to change face hair style, gender, age and other attributes using a smartphone. FakeApp is a desktop application that allows one to create what are now known as “deepfakes” videos. Deepfake videos are manipulated videoclips which were first created by a Reddit user, deepfake, who used TensorFlow, image search engines, social media websites and public video footage to insert someone else’s face onto preexisting videos frame by frame.

**[7] Mallet, J., Dave, R., Seliya, N. and Vanamala, M., 2022, November. Using deep learning to detecting deepfakes. In 2022 9th International Conference on Soft Computing & Machine Intelligence (ISCMI) (pp. 1-5). IEEE.**

In the recent years, social media has grown to become a major source of information for many online users. This has given rise to the spread of misinformation through deepfakes. Deepfakes are videos or images that replace one person’s face with another computer-generated face, often a more recognizable person in society. With the recent advances in technology, a person with little technological experience can generate these videos. This enables them to mimic a power figure in society, such as a president or celebrity, creating the potential danger of spreading misinformation and other nefarious uses of deepfakes.

To combat this online threat, researchers have developed models that are designed to detect deepfakes. This study looks at various deepfake detection models that use deep learning algorithms to combat this looming threat. This survey focuses on providing a comprehensive overview of the current state of deepfake detection models and the unique approaches many researchers take to solving this problem. The benefits, limitations, and suggestions for future work will be thoroughly discussed throughout this paper.

Over the past few years, technology has advanced in a way that enables users with little technological background to effortlessly create a deepfake. A deepfake is a video or image that replaces one person's face with a synthetic, computergenerated version of the authentic person. Deepfakes, commonly created by a generative adversarial network (GAN), hold the capability of fooling an audience into believing a recognizable figure in society is speaking, when they are not.

As society continues to heighten its dependency on social media for providing important information, the threat of misinformation stemming from deepfakes continues to grow. People like presidents, celebrities, and other powerful figures in society can all have their likeness mimicked in a deepfake video that is often indistinguishable from the original to the human eye, creating the potential of misuse. This imminent threat of misinformation has produced the need for software to be developed that could uncover these deepfakes that the human eye is unable to detect. As artificial intelligence continues to advance in related fields, as seen in [1-9], it’s clear this is the tool necessary to combat deepfake misuse.

**[8] Abir, W.H., Khanam, F.R., Alam, K.N., Hadjouni, M., Elmannai, H., Bourouis, S., Dey, R. and Khan, M.M., 2023. Detecting Deepfake Images Using Deep Learning Techniques and Explainable AI Methods. Intelligent Automation & Soft Computing., pp.2151-2169.**

Nowadays, deepfake is wreaking havoc on society. Deepfake content is created with the help of artificial intelligence and machine learning to replace one person’s likeness with another person in pictures or recorded videos. Although visual media manipulations are not new, the introduction of deepfakes has marked a breakthrough in creating fake media and information. These manipulated pictures and videos will undoubtedly have an enormous societal impact. Deepfake uses the latest technology like Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) to construct automated methods for creating fake content that is becoming increasingly difficult to detect with the human eye.

Therefore, automated solutions employed by DL can be an efficient approach for detecting deepfake. Though the “black-box” nature of the DL system allows for robust predictions, they cannot be completely trustworthy. Explainability is the first step toward achieving transparency, but the existing incapacity of DL to explain its own decisions to human users limits the efficacy of these systems. Though Explainable Artificial Intelligence (XAI) can solve this problem by interpreting the predictions of these systems.

This work proposes to provide a comprehensive study of deepfake detection using the DL method and analyze the result of the most effective algorithm with Local Interpretable Model-Agnostic Explanations (LIME) to assure its validity and reliability. This study identifies real and deepfake images using different Convolutional Neural Network (CNN) models to get the best accuracy. It also explains which part of the image caused the model to make a specific classification using the LIME algorithm.

To apply the CNN model, the dataset is taken from Kaggle, which includes 70 k real images from the Flickr dataset collected by Nvidia and 70 k fake faces generated by StyleGAN of 256 px in size. For experimental results, Jupyter notebook, TensorFlow, NumPy, and Pandas were used as software, InceptionResnetV2, DenseNet201, InceptionV3, and ResNet152V2 were used as CNN models. All these models’ performances were good enough, such as InceptionV3 gained 99.68% accuracy, ResNet152V2 got an accuracy of 99.19%, and DenseNet201 performed with 99.81% accuracy. However, InceptionResNetV2 achieved the highest accuracy of 99.87%, which was verified later with the LIME algorithm for XAI, where the proposed method performed the best. The obtained results and dependability demonstrate its preference for detecting deepfake images effectively.

**[9] Gong, D., Kumar, Y.J., Goh, O.S., Ye, Z. and Chi, W., 2021. DeepfakeNet, an efficient deepfake detection method. International Journal of Advanced Computer Science and Applications, 12(6).**

Different CNNs models do not perform well in deepfake detection in cross datasets. This paper proposes a deepfake detection model called DeepfakeNet, which consists of 20 network layers. It refers to the stacking idea of ResNet and the split-transform-merge idea of Inception to design the network block structure, That is, the block structure of ResNeXt. The study uses some data of Face Forensics++, Kaggle and TIMIT datasets, and data enhancement technology is used to expand the datasets for training and testing models.

The experimental results show that, compared with the current mainstream models including VGG19, ResNet101, ResNeXt50, XceptionNet and GoogleNet, in the same dataset and preset parameters, the proposed detection model not only has higher accuracy and lower error rate in cross dataset detection, but also has a significant improvement in performance. Face is one of the most representative features in human beings' biometrics, with high recognition. At the same time, with the rapid development of face synthesis technology, the security threat brought by face tampering is becoming more and more serious.

Especially in the era of mobile phones highly popular and social networks increasingly mature, the deepfake video using deep network model to replace face spreads rapidly in social media and the Internet, such as deepfacelab, and faceswap [1]. At the end of 2018, the Dutch Deeptrace laboratory released a Deepfake development report[2] showing: Deepfake's global search volume has stabilized to as much as 1 million by December 2018, with at least 14678 fake videos, including 96% of pornography, and risk content such as violence, political sensitivity, advertising, contraband, etc., disguised in the video. The appearance of AI face changing is undoubtedly a great impact on its objective authenticity.

**[10] Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Nießner, M., 2019. Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1-11).**

The rapid progress in synthetic image generation and manipulation has now come to a point where it raises significant concerns for the implications towards society. At best, this leads to a loss of trust in digital content, but could potentially cause further harm by spreading false information or fake news. This paper examines the realism of state-of-the-art image manipulations, and how difficult it is to detect them, either automatically or by humans. To standardize the evaluation of detection methods, we propose an automated benchmark for facial manipulation detection. In particular, the benchmark is based on Deep-Fakes, Face2Face, FaceSwap and NeuralTextures as prominent representatives for facial manipulations at random compression level and size.

The benchmark is publicly available and contains a hidden test set as well as a database of over 1.8 million manipulated images. This dataset is over an order of magnitude larger than comparable, publicly available, forgery datasets. Based on this data, we performed a thorough analysis of data-driven forgery detectors. We show that the use of additional domain-specific knowledge improves forgery detection to unprecedented accuracy, even in the presence of strong compression, and clearly outperforms human observers.

**CHAPTER 3**

**PROPOSED METHOD**

**Upload Dataset**

**Preprocessing Dataset**

**Apply DL Training**

**Performance**

**Test Videos**

**Fig 3.1 Block diagram of proposed method**

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

**Fig 4. SDLC stands for Software Development Process.**

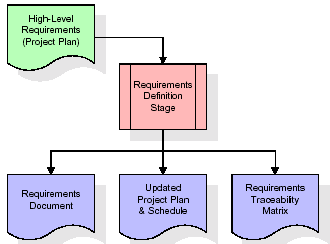
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



**Fig 5. Collecting method uses the objectives specified**

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

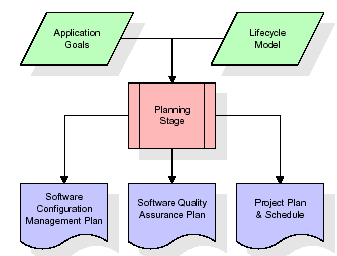
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



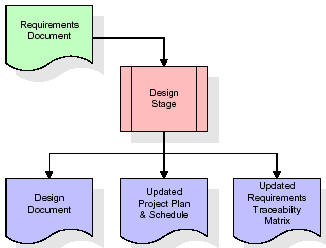
**Fig.6 Planning creates a bird's-eye perspective of the desired software platform**

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included.

The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

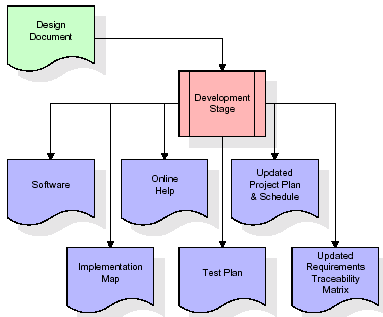
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



**Fig.7 listed in the accepted specification document serve**

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

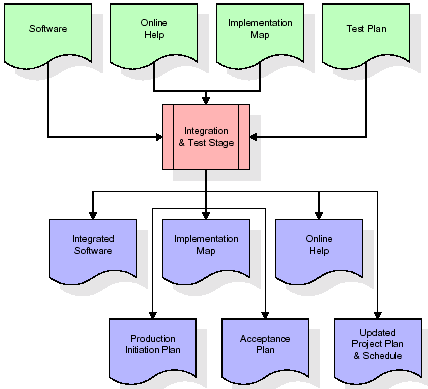


**Fig.8 design features mentioned in the approved planning**

The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



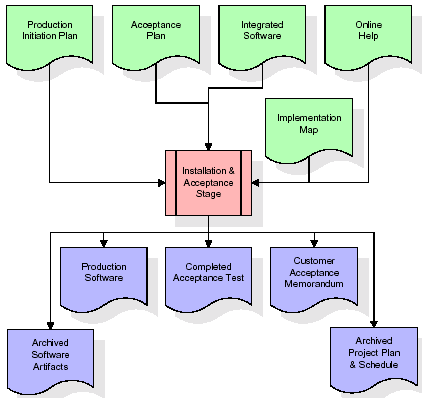
**Fig 9. software artefacts, online support, and test data**

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

# **CHAPTER 4**

**SOFTWARE AND HARDWARE REQUIRMENT**

**4.1 HARDWARE REQUIREMENTS:**

* System : Pentium Dual Core.
* Hard Disk : 120 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* Ram : 4 GB

**4.2 SOFTWARE REQUIREMENTS:**

* Operating system : Windows 10
* Coding Language : python
* Tool : Python
* Database : dataset
* Server : Flask

**4.3 Hardware Interfaces**

Intel Core i5 2.00GHz Processor or each and every other processor and 200 GB min RAM 20GB Hard plate, and mouse is required.

**Software Interfaces**

The Python IDLE is an open-supply web utility that allows you to make and charge records that be essential for stay code, circumstances, portrayals, and story-printed content. Uses encompass realities cleansing and exchange, numerical re-establishment, quantifiable illustrating, realities conviction, framework examining, and divides

**CHAPTER 5**

**RESULT**

In this project DL algorithm is used to detect Deepfake faces detection from video input.

To train above model we have used Deepfake faces dataset from KAGGLE repository which contains more than 95000 images and this dataset can be downloaded from below URL

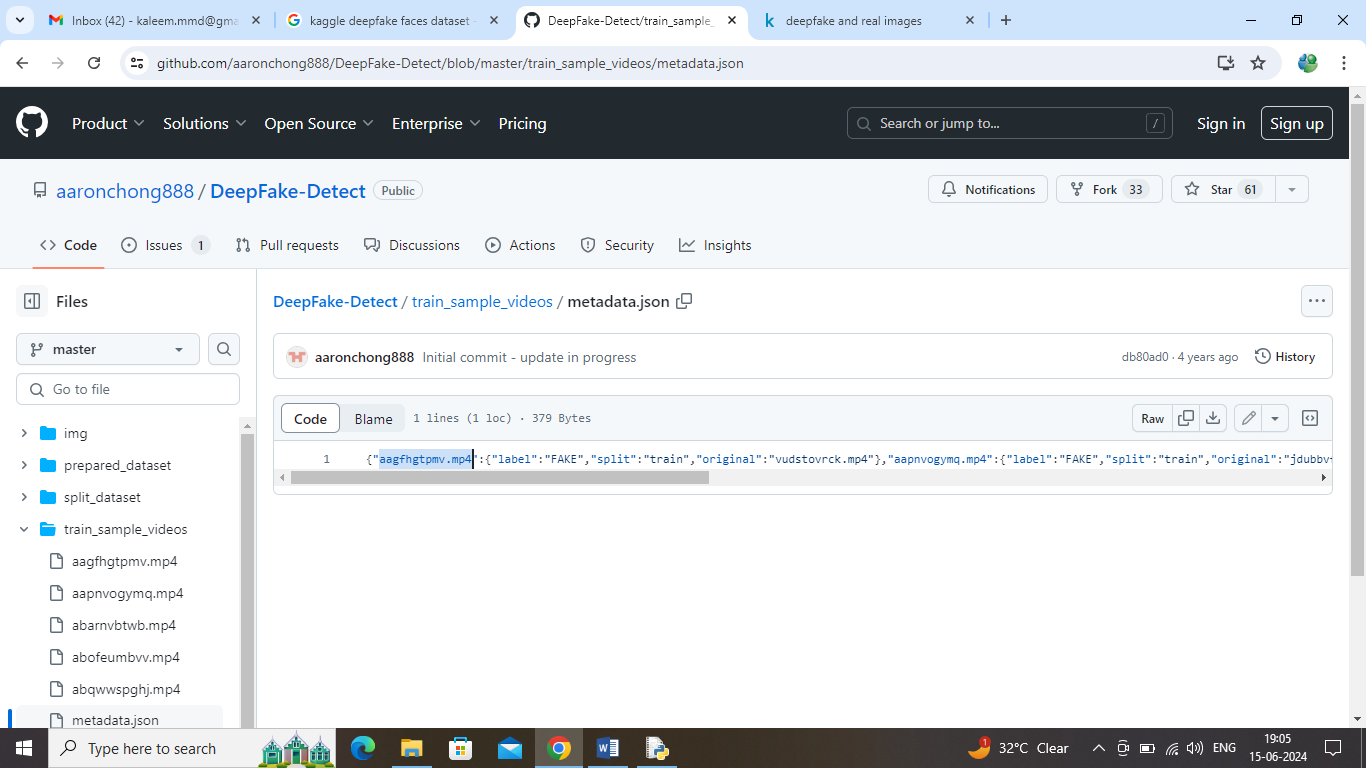
<https://www.kaggle.com/datasets/dagnelies/deepfake-faces>

Above dataset contains two different class labels such as Fake and Real

To detect Deepfake faces we have downloaded videos from below URL

<https://github.com/aaronchong888/DeepFake-Detect/blob/master/train_sample_videos/metadata.json>

From above URL file we can see fake and real videos which we downloaded and tested with our model

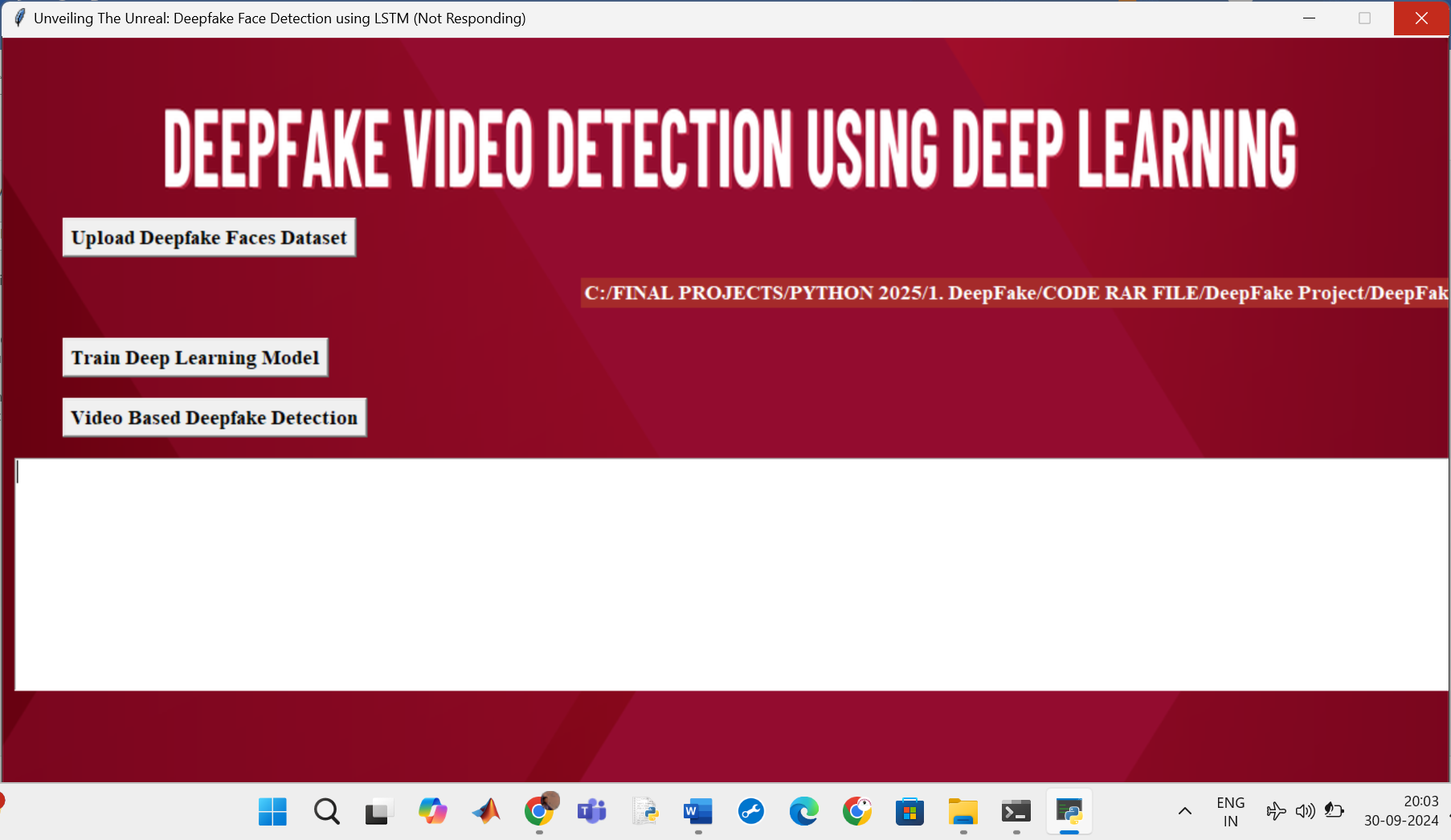


In above URL screen you can see ‘aagfhgtpmv.mp4’ video is fake and ‘abarnvbtwb.mp4’ is the real video and this model is successfully predicting this videos.

To implement this project we have designed following modules

1. Upload Deepfake Faces Dataset: using this module will upload dataset images to application and then application will read all images and then resize to equal sizes and then creating X and Y training array
2. Train DL Model: this module will shuffle, normalize and then split all images into 80:20 percent train and test ratio. 80% images will be input to DL algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy
3. Video Based Deepfake Detection: using this module will upload test video and then DL model will analyse faces from each frame slowly and then predict video as Real or Deepfake. Once after prediction will get video playing output with result as fake or real.

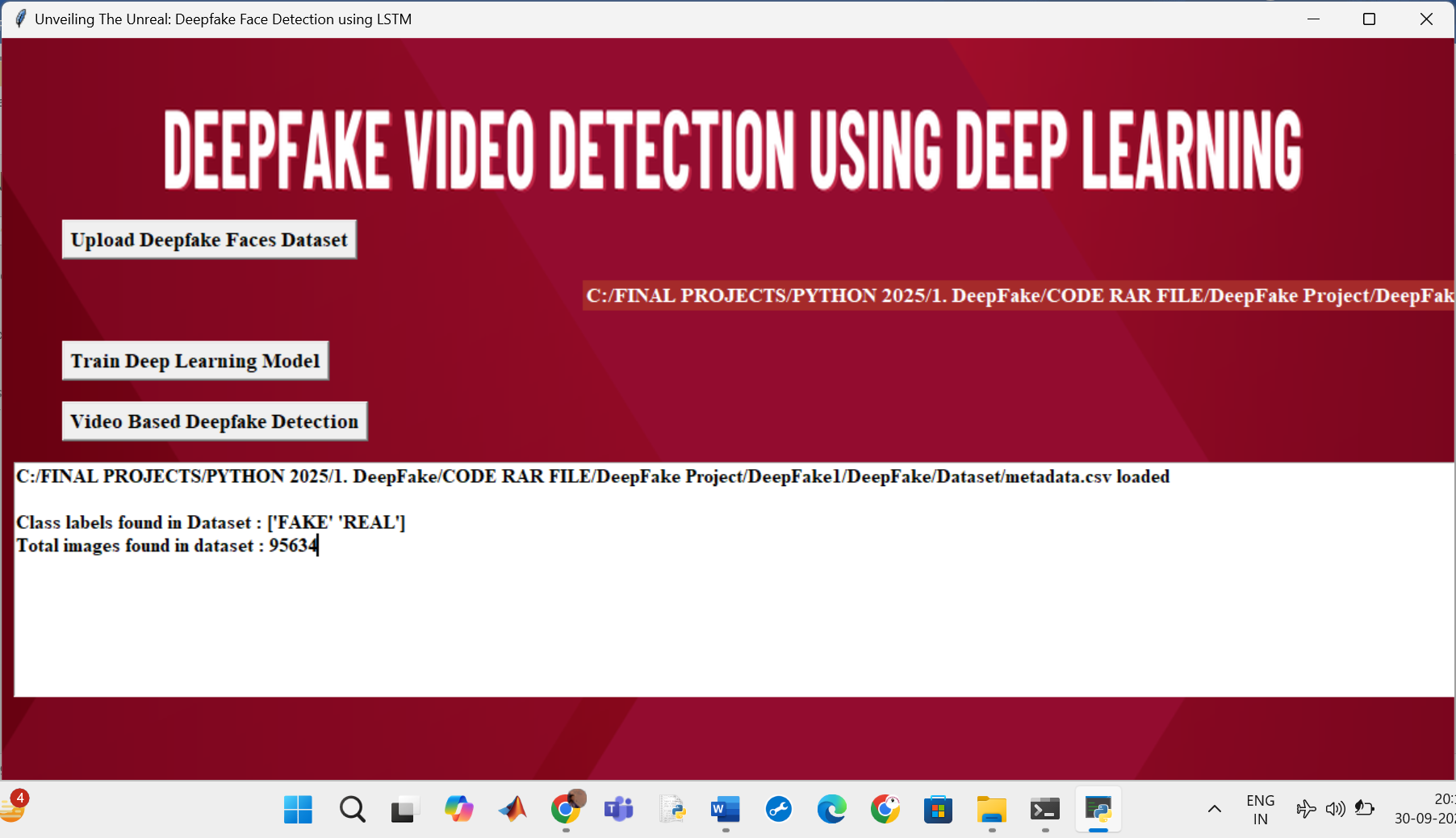
To run project double click on run.bat file to get below screen



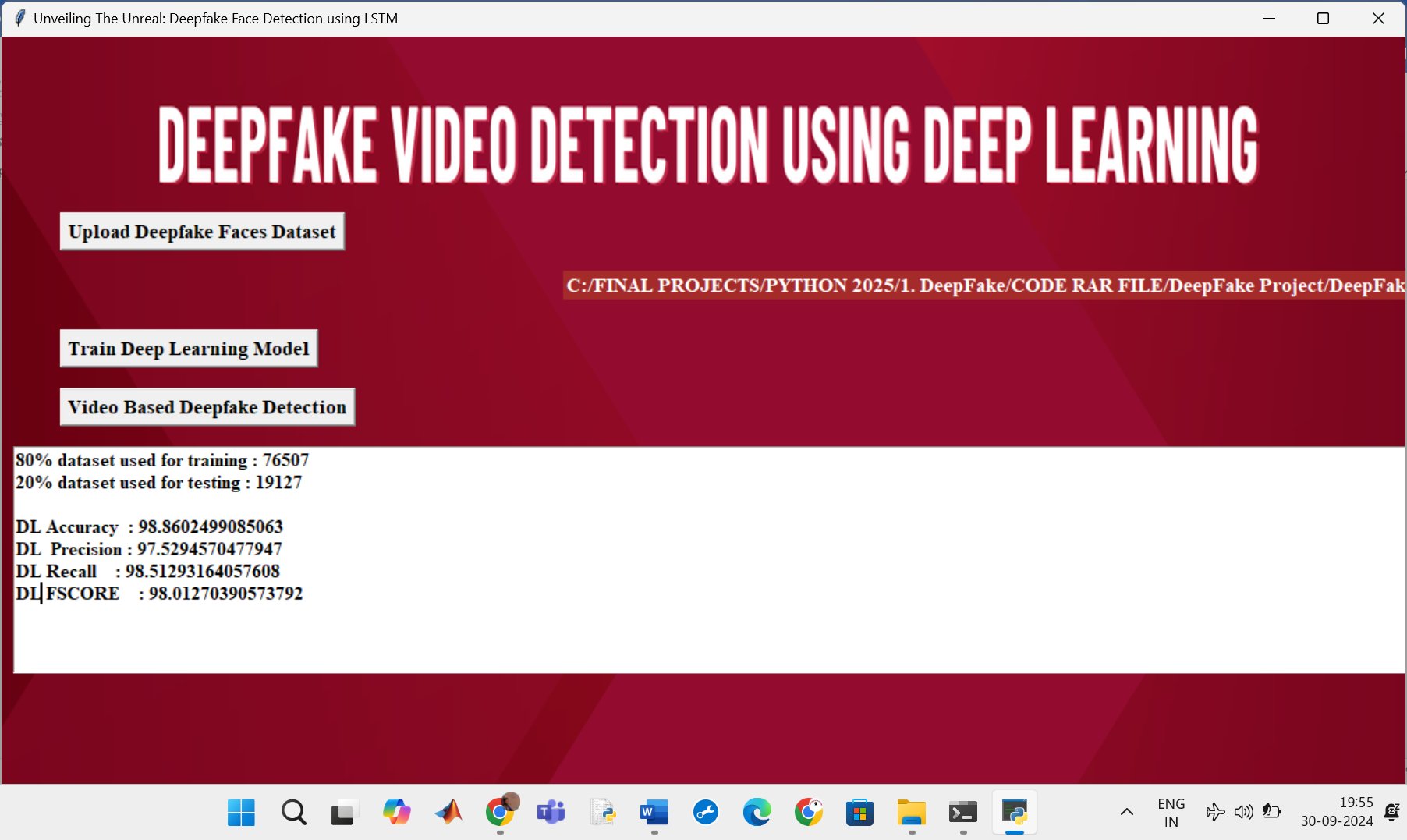
In above screen click on ‘Upload Deepfake Faces Dataset’ button to load dataset and get below page



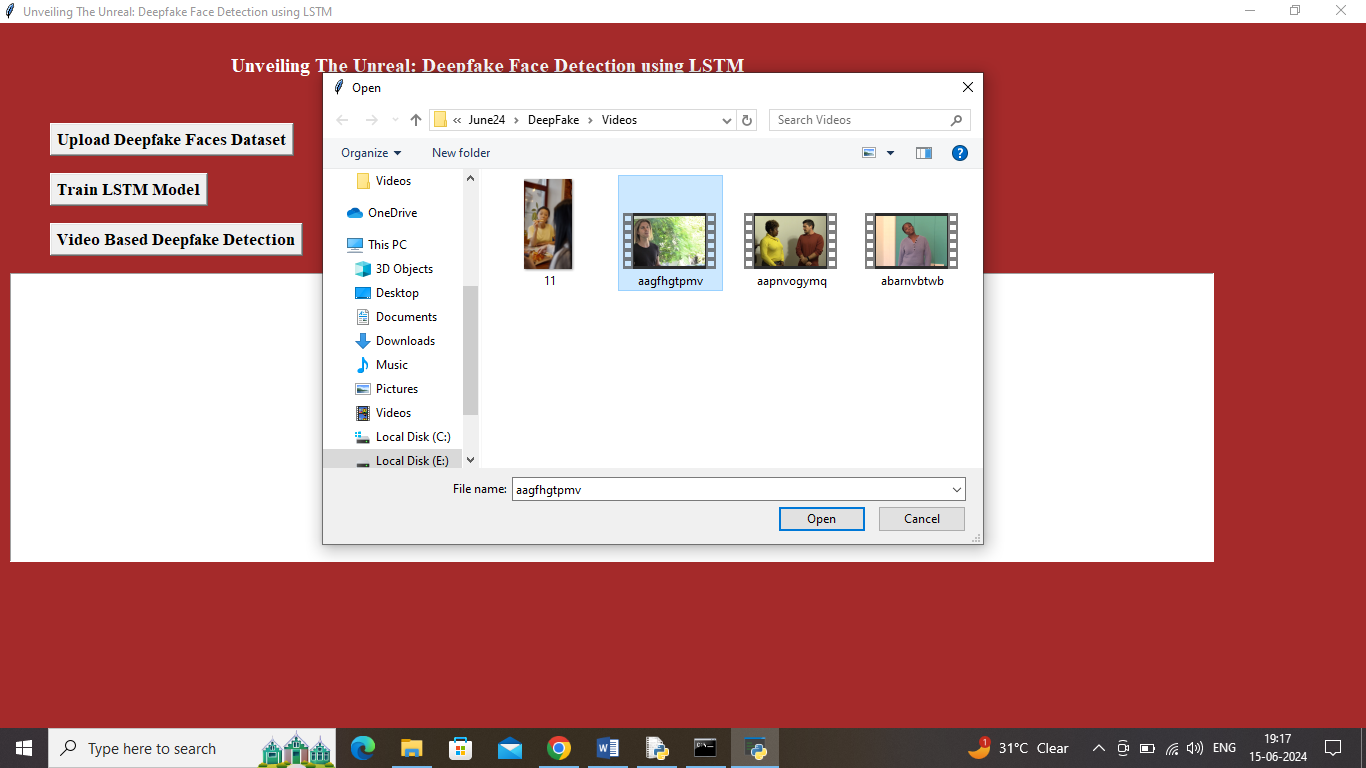
In above screen selecting and uploading dataset annotation file and then click on ‘Open’ button to get below output



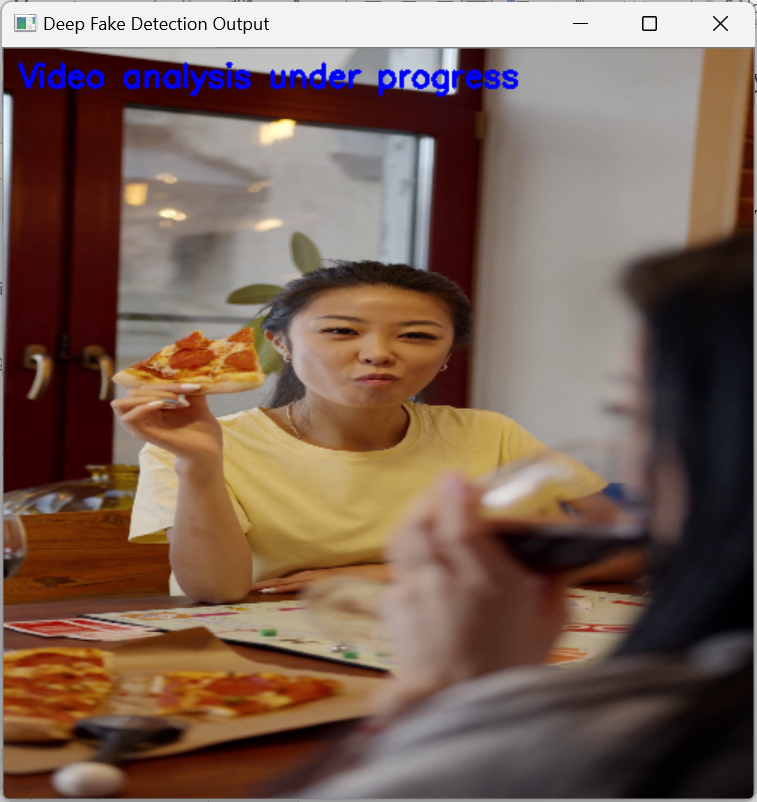
In above screen can see dataset loaded and can see different class labels found in dataset and then can see number of images found in dataset and now click on ‘Train DL Model’ button to train algorithm and get below page



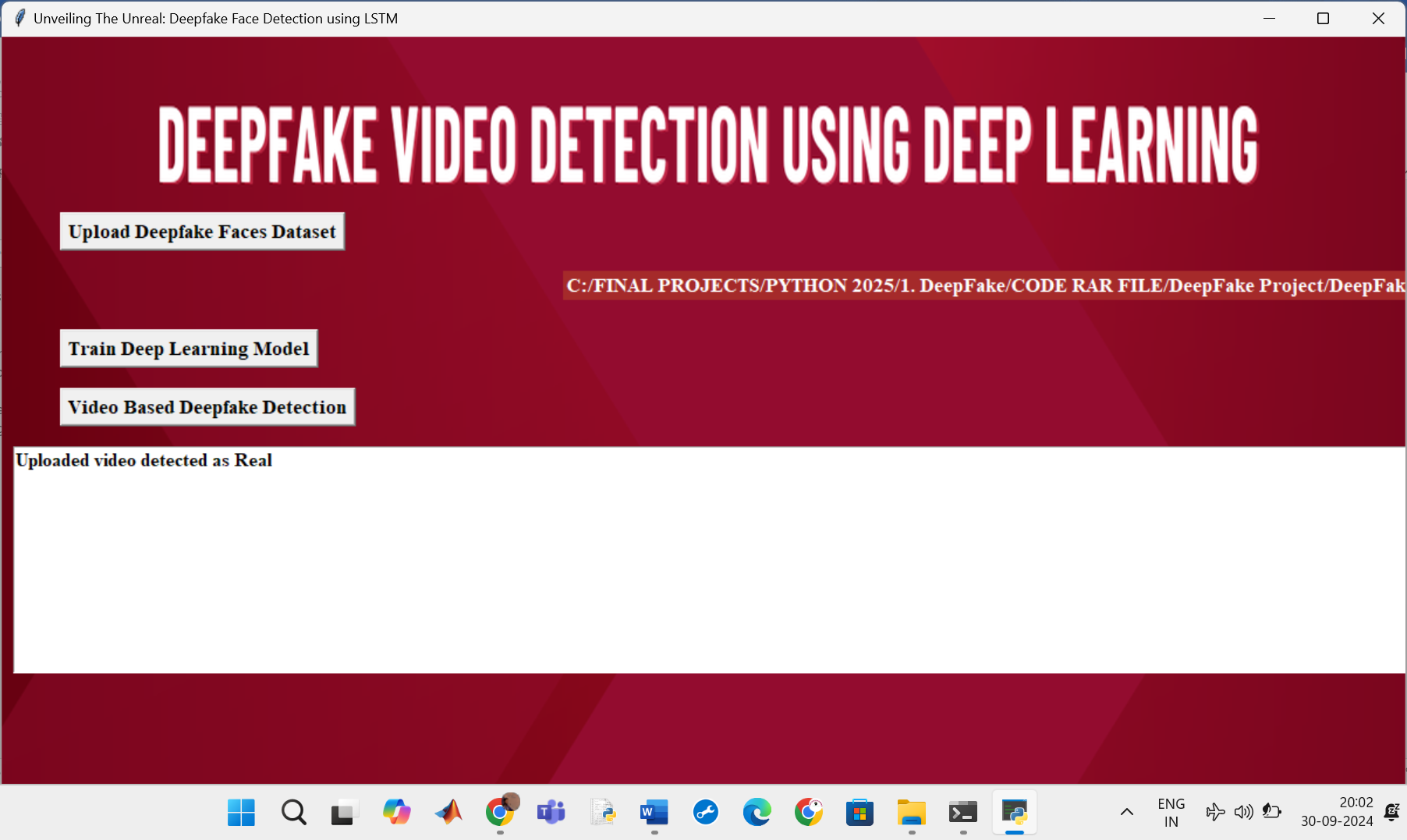
In above screen can see train and test dataset size and then can see DL got 99% accuracy and can see other metrics like precision, recall and FSCORE. Now click on ‘Video Based Deepfake Detection’ button to upload test video and get below page

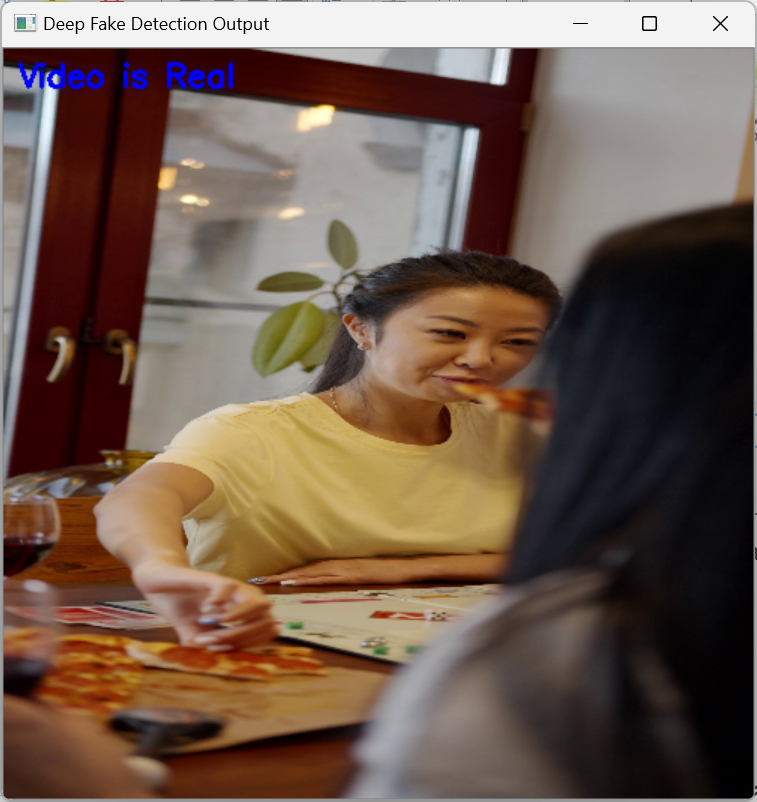


In above screen selecting and uploading 11.mp4 video and then click on ‘Open’ button to start analysing video and get below page

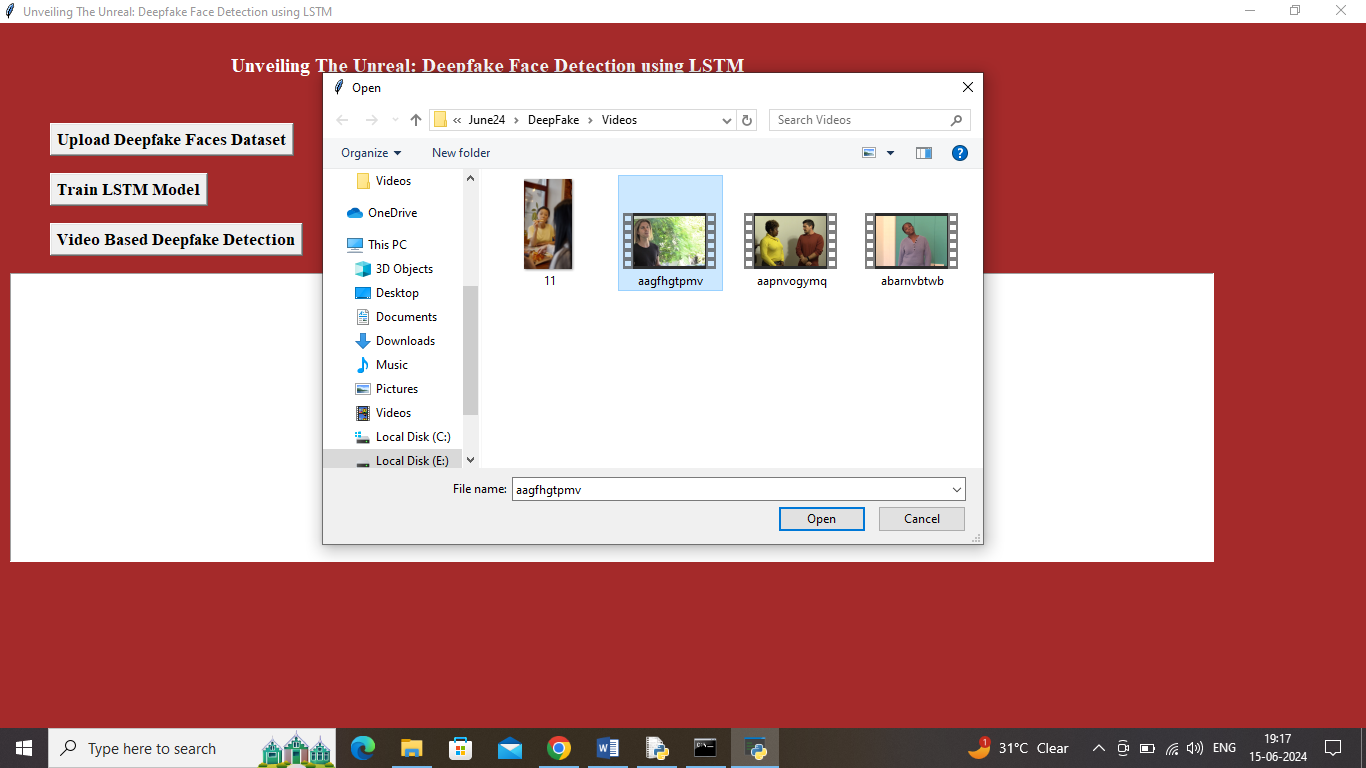


In above screen in blue colour text can see video analysis started and after thorough analysis by DL will get below output

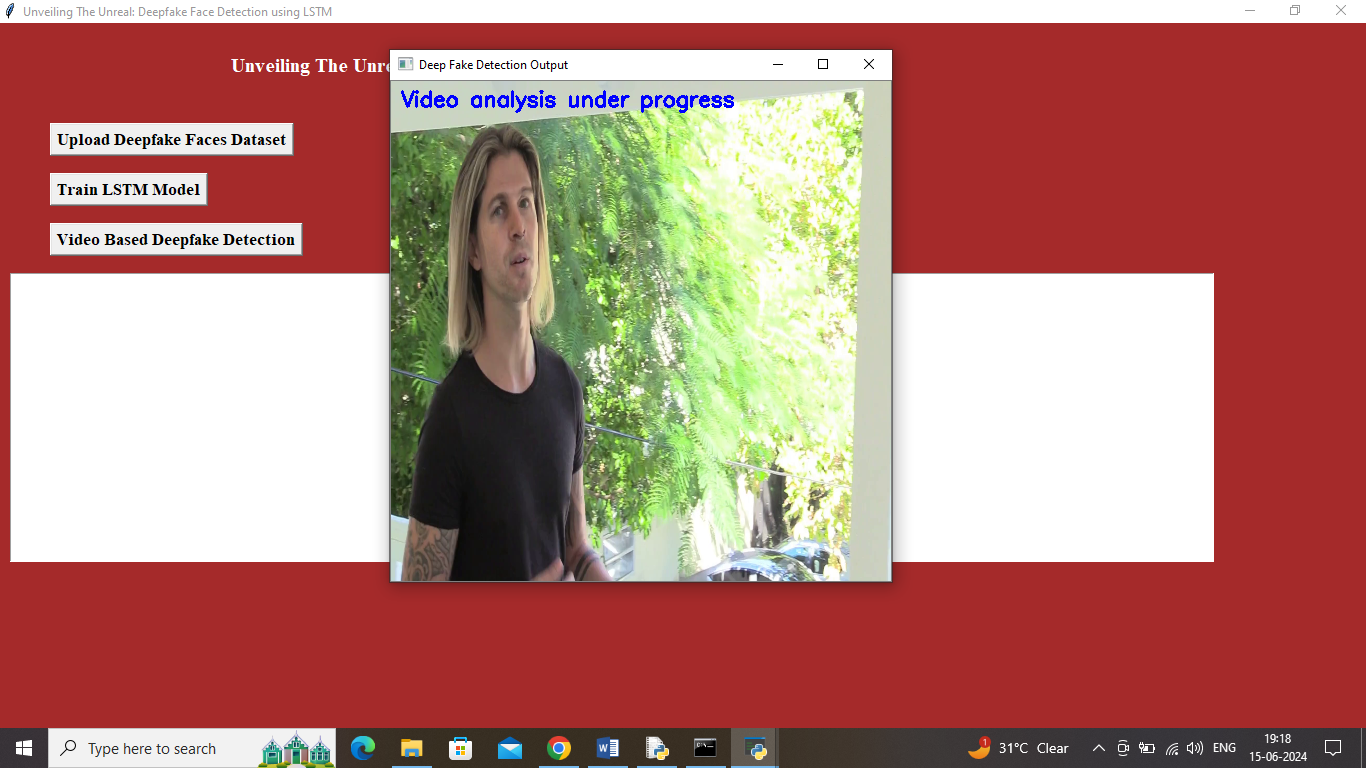




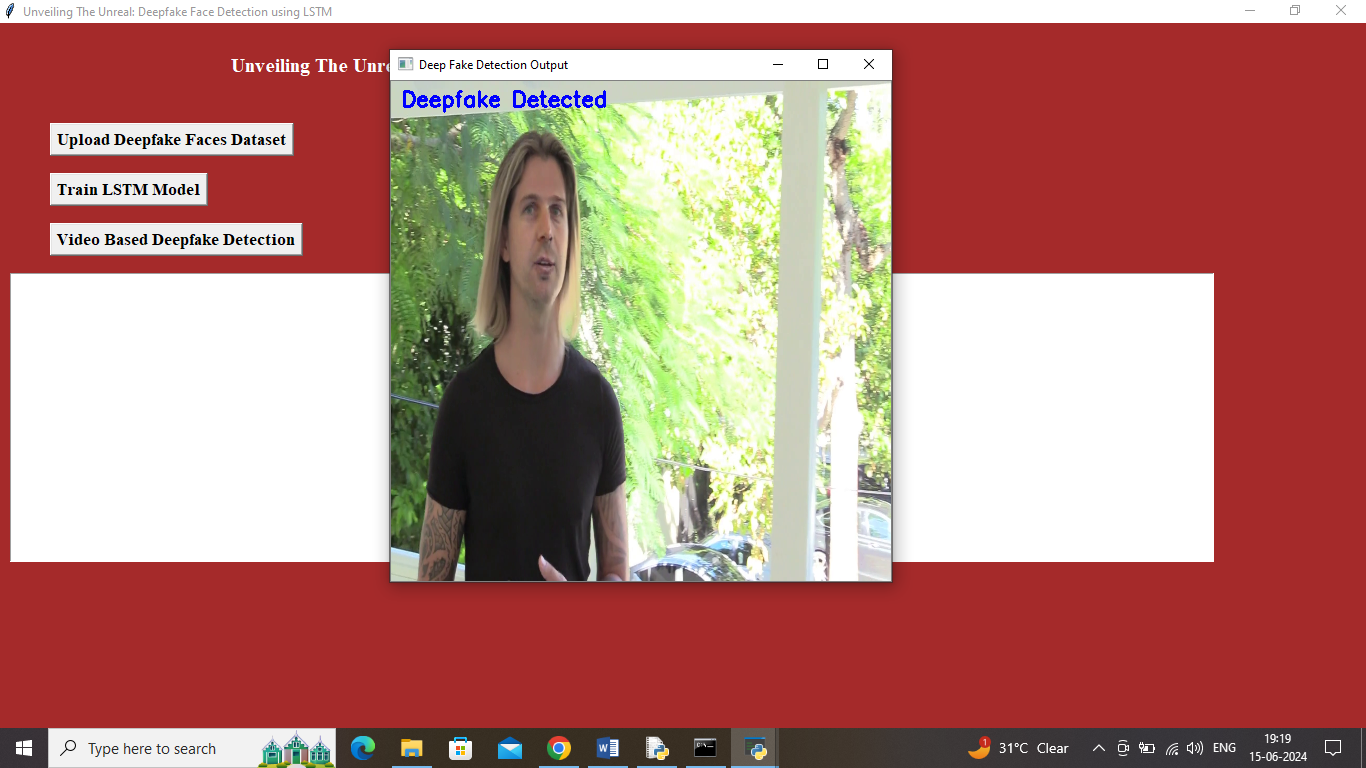
In above screen uploaded video predicted as ‘Real’ and now test with other fake video



In above screen uploading ‘aagfhgtpmv.mp4’ video and then click on ‘Open’ button to load video and get below output



In above screen video analysis under progress and once after analysis will get below output



In above screen uploaded video is detected as Deepfake and similarly you can upload and test other video

**CHAPTER 6**

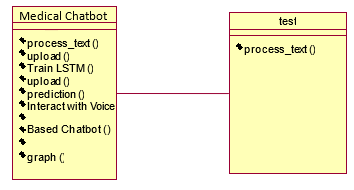
**SYSTEM DESIGN**

**UML Diagram:**

**Class Diagram:**

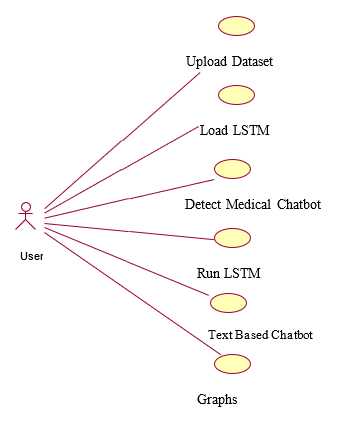
As the foundation of device modelling, the diagram serves as a starting point. In addition to modelling, it is also used to translate the images into computer code in a more thorough manner. It is also possible to simulate data using graphs and charts. In a gantt chart, the essential concepts and transactions in the software are represented by the classes, which are also the schools that need to be written. Class are depicted in the illustration as three columns:

* The identifier of both the object is shown at the top, while the properties of something like the class are located in the centre.
* Class procedures are listed in the bottom half of the page.

****

**Fig 6.1 Class diagram**

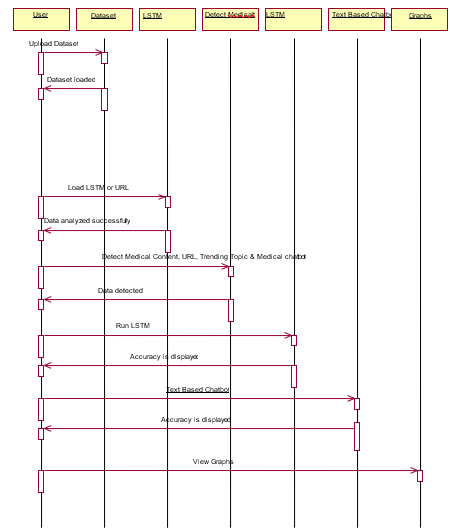
**Use case Diagram:** With in simplistic definition, a use case chart illustrates a customers happy and satisfied with a computer and the use case parameters. An example of a use wireframe is a depiction of a platform's participants and their interactions with it. When used in combination with a written use scenario, this figure is commonly found in group pictures though too.

****

**Fig.6.2 Use case Diagram**

**Sequence Diagram:**

In a schematic, processes are depicted in relation to one other in whatever chronology. It's a Process Flow Chart style. A pattern diagram is a visual representation of how objects interact with each other over time. It represents the situations class diagrams, as well as the data sent and received by those objects in order to effectively carry out the scenario's functions. There in Functional Viewpoint under operation, flow charts are often related with use case realisations. Flow charts were often termed er diagram, happening situations, & timings drawings.

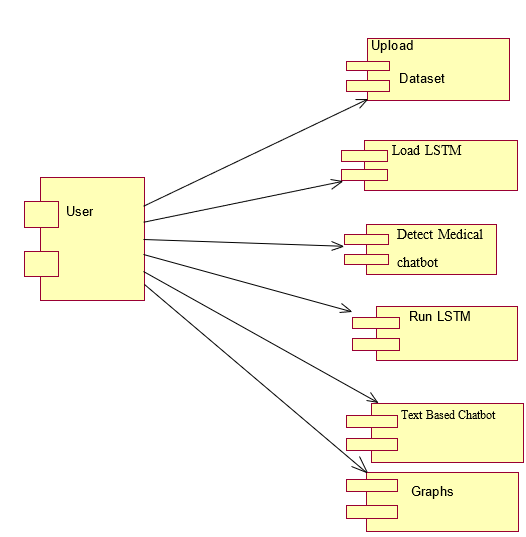


**Fig. 6.3 Sequence diagram**

**Component Diagram:**

It is capable of simulating software and bigger modules using the Interface Definition Idiomatic unit diagrams. A wide range of systems may be shown using these models.

An construction connect is used to link one function's necessary connectivity to this other element's available port. This is a good example of the customer experience in action.

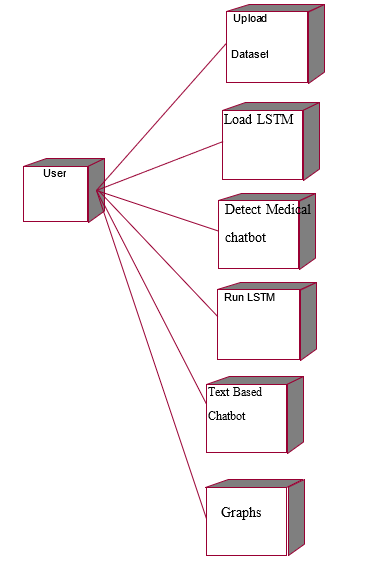


**Fig 6.4** **Component Diagram**

**Deployment Diagram:**

A deployments chart in the Uniform Modelling Tool (UML) depicts how objects are physically deployed on a network node. Dispatch diagrams indicate what embedded systems ("nodes") reside, whether apps ("artefacts") operate to every site, and how well the various pieces that infrastructure ("artefacts") are joined to represent a web page, for example (e.g. JDBC, REST, RMI).

It is possible to see the nodes as square brackets, and the artefacts assigned from each box. Inter show as nesting boxes in the tree. Nodes can represent many devices, other than a whole cluster of dbms, when they are shown in the distribution diagram.



**Fig 6.5 Deployment Diagram**

**CHAPTER 7**

**CONCLUSION**

The conclusion of deep fake face detection from videos using deep learning (DL) emphasizes the importance of leveraging advanced neural networks to detect subtle facial manipulations. Deep learning models, such as CNNs and RNNs, have shown promising results in accurately identifying fake faces by analyzing temporal inconsistencies, texture anomalies, and facial landmarks across frames. These approaches help mitigate the spread of misinformation, improve security in digital platforms, and enable automated real-time detection, proving essential in combating the evolving threats posed by deep fakes.

**REFERENCE**

[1] Nguyen, T.T., Nguyen, Q.V.H., Nguyen, D.T., Nguyen, D.T., Huynh-The, T., Nahavandi, S., Nguyen, T.T., Pham, Q.V. and Nguyen, C.M., 2022. Deep learning for deepfakes creation and detection: A survey. Computer Vision and Image Understanding, 223, p.103525.

[2] Westerlund, M., 2019. The emergence of deepfake technology: A review. Technology innovation management review, 9(11).

[3] Thippanna, G., Priya, M.D. and Srinivas, T.A.S., An Effective Analysis of Image Processing with Deep Learning Algorithms. International Journal of Computer Applications, 975, p.8887.

[4] Indolia, S., Goswami, A.K., Mishra, S.P. and Asopa, P., 2018. Conceptual understanding of convolutional neural network-a deep learning approach. Procedia computer science, 132, pp.679-688.

[5] Ralf C. Staudemeyer, “Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks”, arXiv:1909.09586v1 [cs.NE] 12 Sep 2019

[6] Güera, D. and Delp, E.J., 2018, November. Deepfake video detection using recurrent neural networks. In 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS) (pp. 1-6). IEEE.

[7] Mallet, J., Dave, R., Seliya, N. and Vanamala, M., 2022, November. Using deep learning to detecting deepfakes. In 2022 9th International Conference on Soft Computing & Machine Intelligence (ISCMI) (pp. 1-5). IEEE.

[8] Abir, W.H., Khanam, F.R., Alam, K.N., Hadjouni, M., Elmannai, H., Bourouis, S., Dey, R. and Khan, M.M., 2023. Detecting Deepfake Images Using Deep Learning Techniques and Explainable AI Methods. Intelligent Automation & Soft Computing., pp.2151-2169.

[9] Gong, D., Kumar, Y.J., Goh, O.S., Ye, Z. and Chi, W., 2021. DeepfakeNet, an efficient deepfake detection method. International Journal of Advanced Computer Science and Applications, 12(6).

[10] Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Nießner, M., 2019. Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1-11).

[11] DFDC data from Kaggle:- https://www.kaggle.com/competitions/deepfake-detection-challenge (Accessed on 13/09/2023)

[12] Li, Y., Yang, X., Sun, P., Qi, H. and Lyu, S., 2020. Celeb-df: A large-scale challenging dataset for deepfake forensics. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3207-3216).

**APPENDIX**

**PYTHON**

**1.1 Introduction**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* **Readability:** Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* **Easy to Learn:** Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* **Interpreted Language:** Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* **Cross-platform Compatibility:** Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* **Large Standard Library:** Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* **Extensible and Modular:** Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* **Wide Range of Libraries and Frameworks:** Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* **Object-Oriented:** Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* **Dynamic Typing:** Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**1.2 Installation**

To install Python on your computer, follow these basic steps:

* **Step 1:** Visit the Python website Go to the official Python website at <https://www.python.org/>.
* **Step 2:** Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* **Step 3:** Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* **Step 4:** Install Python 3.7.0 using Cmd.
* **Step 5:** Install the all libraries that required to run the project
* **Step 6:** Run

**1.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language; therefore the type of the value will be determined at runtime.

**1.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

**1)** Install the Tkinter module in place.

**2)** The GUI applicatioMakeske the primary window

**3)** Include one or more of the widgets mentioned above in the GUI application.

**4)** Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**1.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* **Editor:** IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* **Interactive Shell:** The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* **Debugging:** IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* **Python Help:** IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* **Script Execution:** In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* **Customization:** IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* **Python IDLE** serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics.

However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**1.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries.

It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning.

It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open-source platform for machine learning.

TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python.

It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics