ABSTRACT

The field of AI-generated deepfake video detection is dynamic and evolving, with researchers continuously exploring new methodologies, refining existing techniques, and addressing challenges to create more accurate and reliable detection systems. Collaboration between academia, industry, and policy makers is emphasised to navigate the ethical, technical, and societal implications of these technologies.

The growing computation power has made the deep learning algorithms so powerful that creating an indistinguishable human synthesised video popularly called as deep fakes has become very simple. Possible situations where these convincing face-swapped deep fakes might be employed include fabricating political events and false acts of terrorism, generating revenge porn content, and engaging in blackmail schemes against individuals. These scenarios highlight the potential misuse and harm associated with realistic deepfake technology.In this project, we outline a novel deep learning approach designed to successfully differentiate between artificially generated fake videos and authentic ones. Our aim is to leverage Artificial Intelligence (AI) as a means to counteract the deceptive capabilities of Artificial Intelligence (AI). Our system uses an XceptionNet combined with LSTM to decide whether the video is deep fake or real. To emulate the real time scenarios and make the models perform better on real time data, we evaluate our method on large amount of balanced and mixed dataset prepared by mixing the various available dataset like Face-Forensic [1], Deep Face detection challenge [2], Kaggle and Celeb-DF [3]. We also show how our system can achieve competitive results using a very simple and robust approach.

Keywords: *Artificial Intelligence(AI), Deep Learning (DL), Extreme Inception(Xception), Long-Short Term Memory (LSTM), Machine Learning(ML),.*

Table of Contents

List of Figures

List of Abbreviations/Acronyms

AI: Artificial Intelligence

CNN: Convolution Neural Network

DL: Deep Learning

ML: Machine Learning

Xception: Extreme Inception

CHAPTER 1

INTRODUCTION

An "AI Generated Fake video Detector" web application is a digital tool powered by artificial intelligence (AI) that aims to identify and flag videos that have been artificially generated, manipulated, or edited using AI-based algorithms or video editing software.

The primary purpose of such an application is to combat the spread of misleading or falsified visual content across digital platforms. It assists users, fact-checkers, journalists, and online platforms in verifying the credibility of videos, aiding in the detection of misinformation and enhancing the authenticity of visual content.

1.1 Background

AI-generated fake videos have the potential for misinformation and the erosion of trust in visual content. Here are several specific drawbacks associated with AI-generated fake videos. They are:

Misleading Information:AI-generated fake videos can mislead viewers by presenting false or manipulated visual information. This misinformation can contribute to misunderstandings, false narratives, and confusion, affecting decision-making and perceptions.

Impact on Trust:The proliferation of fake videos can erode trust in the authenticity of visual content online. Ascertaining whether a video is genuine or manipulated becomes challenging, leading to skepticism and reduced trust in online sources.

Spreading False Narratives:Fake videos, especially when shared extensively on social media or news platforms, can perpetuate false narratives. They can influence opinions, public discourse, and even policy decisions based on inaccurate information.

Legal and Regulatory Challenges:Addressing the misuse of AI-generated fake videos poses legal and regulatory challenges, as existing laws might not adequately cover these new forms of digital manipulation.

1.2 Motivation

Latest rise of manipulated and fake videos across social media and digital platforms has led to widespread misinformation. Detecting these videos helps in curbing the spread of false narratives and misleading information.Promoting the responsible and ethical use of AI technology is essential. Developing tools that detect AI-generated fake videos encourages ethical considerations and responsible usage of advanced technologies.

Overall, the motivation behind creating an AI-generated fake video detector web application is to address the challenges posed by the proliferation of manipulated visual content and to promote a more informed, trustworthy, and safe online environment.

1.3 Statement of the Problem

In an era where Advanced AI technologies facilitate the creation and dissemination of increasingly convincing fake videos, there exists a critical need for a robust and accessible web application capable of swiftly and accurately detecting AI-generated or manipulated visuals. The proliferation of misleading videos across digital platforms poses a substantial threat to information credibility, public trust, and the ability to discern authentic content. Addressing this challenge requires the development of an intelligent, user-friendly tool that empowers individuals, journalists, fact-checkers, and online platforms to identify and mitigate the impact of AI-generated fake videos, thereby fostering a more trustworthy and informed digital environment.

So, From the above we feel the need of “AI Generated Fake video Detector” Web Application.

1.4 Project objective

To develop a deepfake video detection web application.

1.5 Significance of the study

An AI-generated deepfake video detection web application holds significant importance for various beneficiaries and can benefit them in several ways:

General Public: Provides the public with a tool to verify the authenticity of videos encountered online, fostering a more informed and discerning audience. It helps individuals avoid falling victim to misinformation, enhancing their digital literacy and critical thinking.

Online Platforms and Social Media: Helps online platforms implement measures to identify and mitigate the spread of fake videos on their platforms, thereby fostering a more trustworthy and responsible online community.

Fact-Checkers and Researchers**:** Supports fact-checking organisations and researchers by streamlining the process of identifying manipulated visuals. This accelerates the verification process and contributes to more accurate analyses of information.

Individuals Targeted by Manipulated videos: Protects individuals or groups depicted in manipulated videos from potential harm, defamation, or false representation by swiftly identifying and addressing the distribution of fake visuals.

By offering a reliable means of detecting AI-generated fake videos, this web application contributes to a more trustworthy digital environment, empowers users to make informed decisions, and aids various sectors in ensuring accuracy, credibility, and security in the use of visual content online.

CHAPTER 2

LITERATURE REVIEW

In recent years there have been astonishing advances in AI-based synthetic media generation.Thanks to deep learning methods it is now possible to generate visual data with a high level of realism. This is especially true for human faces. Thus a lot of research has been conducted on detecting these AI generated videos with the help of AI itself.

One of the major neural networks used for generating synthetic videos is GAN(Generative Adversarial Network).The visual appearance of the videos generated by the latest GAN architectures is so realistic that it deceives even the experienced and attentive observer. This raises major concerns on the possible malicious use of such tools.Despite their high visual quality, GAN videos are characterized by specific artifacts left from the generation process that can be used to develop effective tools for their detection. In some cases, their synthetic origin can be identified by visual inspection due to the presence of semantic inconsistencies, such as color anomalies or lack of symmetries. More generally, these videos present invisible artifacts, closely linked to the architecture of the generative network, which can be extracted through appropriate processing steps. These artifacts represent very strong clues, which can be exploited even when synthetic videos appear perfectly realistic.[13]

D. Guera and J. Delp,[6] have used a recognition pipeline to automatically detect deep fakes. They have proposed a two step analysis. During the first stage, features are extracted at frame level using CNN. The second stage consists of RNN which will capture erratic frames introduced due to the face swapping process. The dataset contains 600 videos collected from various online sources. The accuracy achieved by their model is 94 percent.

Y. Li, MC. Chang and S. Lyu [9] have introduced a new system of exposing deep fakes based on the eye blinking which are generated using neural networks. The paper focused on analyzing the eye blinking in the video as it is a natural signal and it cannot be presented well in the synthesized media. In the method the videos have been first preprocessed to

locate face area in each frame, then a Long Term Recurrent Convolutional Network(LRCN) finds out temporal incongruity.

Adil Mohammed Parayil, Ameen Masood V, Muhammed Ajas P, Tharun R, and Usha K [14] employ deep learning for deep fake detection, utilizing Convolutional Neural Network (CNN) with the Xception network and ytLSTM for Recurrent Neural Network (RNN). The algorithm recognizes spatial features and temporal inconsistencies between frames. Models are trained and validated on three standard datasets, resulting in efficient deepfake prediction with minimal computational time and nominal accuracy.

Franc¸ois Chollet [5] conducted the evaluation which employed a single Xception model with a single crop of input images, comparing its performance to Inception V3 on ImageNet and JFT datasets. Xception demonstrated marginal superiority over Inception V3 on ImageNet, with a 4.3% relative improvement on JFT. The classification performance on ImageNet revealed that Xception outperformed VGG-16, ResNet-152, and Inception V3, achieving a Top-1 accuracy of 0.790 and Top-5 accuracy of 0.945. Notably, Xception showcased a more substantial performance boost on JFT, potentially attributed to Inception V3 being tailored for ImageNet, whereas Xception was not specifically tuned for JFT. Fine-tuning Xception's hyperparameters for ImageNet could yield further enhancements.

So, From various literature surveys exploring diverse neural networks and deep learning techniques, Convolutional Neural Networks (CNNs), specifically the Xception network and Long Short-Term Memory (LSTM), emerge as standout tools. These models demonstrate robustness by employing transfer learning to detect DeepFakes with remarkable accuracy, facilitated by the ingestion of extensive training datasets.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 System Requirements

A requirement is a feature that the system must have or a constraint that must be accepted by the client. Requirement Engineering aims at defining the requirements of the system under construction. Requirement Engineering includes two main activities: requirement elicitation which results in the specification of the system that the client understands and analysis which results in an analysis model that the developer can unambiguously interpret. A requirement is a statement about what the proposed system will do.

Requirements can be divided into two major categories:

3.1.1 Functional Requirement

Video Analysis:

Objective: Detect manipulated videos.

Details: Implement algorithms to analyze videos for signs of manipulation, including inconsistencies in lighting, shadows, and textures.

Real-time Processing:

Objective: Enable real-time analysis.

Details: Minimize processing time to ensure quick identification of fake videos, suitable for real-time applications like social media.

Integration:

Objective: Integrate with various platforms.

Details: Provide API integration for seamless collaboration with different applications and services.

User Interaction:

Objective: User-friendly interface.

Details: Create an intuitive interface for users to interact with the system, view analysis results, and provide feedback

Real-time processing with low latency.

Scalable architecture for handling a large volume of videos.

API integration for collaboration with various platforms.

User-friendly interface for interaction and feedback.

3.1.2 Non-functional Requirement

Accuracy:

Objective: High precision in detection.

Details: Minimise false positives and false negatives to ensure accurate identification of manipulated videos.

Scalability:

Objective: Handle a large volume of videos.

Details: Design a scalable system architecture to accommodate a growing number of video analysis requests.

Reliability:

Objective: Consistent performance.

Details: Ensure the system performs reliably across various types of manipulated videos and under different conditions.

Security:

Objective: Protect user data.

Details: Implement security measures to safeguard user privacy during the video analysis process.

Adaptability:

Objective: Adapt to new techniques.

Details: Design the system to adapt to emerging methods of video manipulation, staying current with technological advancements.

Compliance:

Objective: Adherence to legal and ethical standards.

Details: Ensure the system complies with relevant laws and ethical standards, addressing concerns related to user privacy and data protection.

Documentation:

Objective: Comprehensive documentation.

Details: Provide thorough documentation for administrators, developers, and end-users, covering system functionality, configuration, and usage.

3.1.3 Hardware Requirements

- Processor: Intel Core i5 or higher
- RAM: 4 GB or higher
- Internet Access

3.1.4 Software Requirements

- Programming Language: Python
- Web Framework: Flask
- Machine Learning Libraries: TensorFlow, Keras
- Frontend Development: Flask,HTML

3.1.4.1 Programming Languages

- Python

3.1.4.2 Web Browser

- Any Chromium based Web Browser

3.1.4.3 Libraries

- Keras
- NumPy
- Pandas
- Flask

3.1.4.4 OS

- Windows,macOS

3.1.4.5 IDE

- VS Code
- Jupyter Notebook
- Google colab

CHAPTER 4

SYSTEM DESIGN AND ARCHITECTURE

4.1 System Architecture

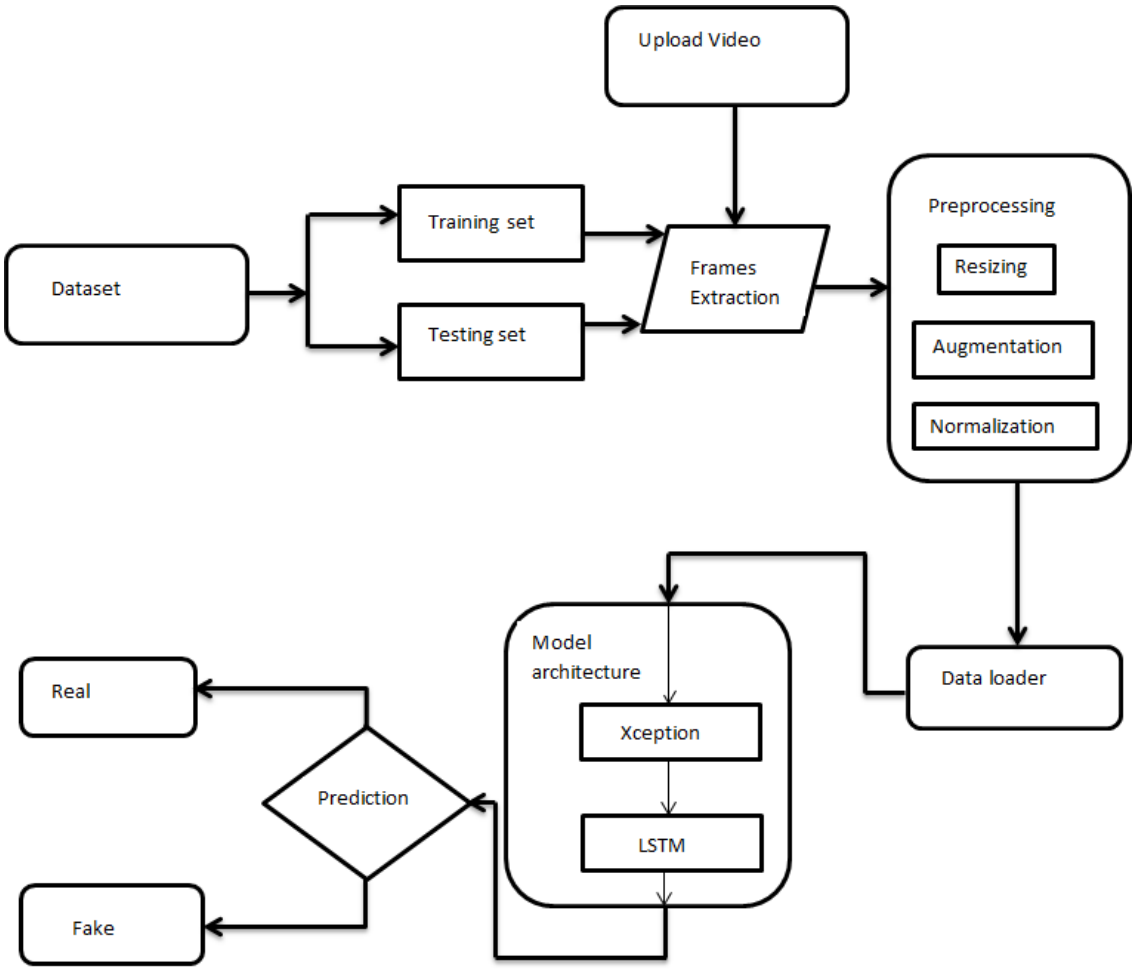

Fig 4.1:System Architecture

4.2 Flowchart

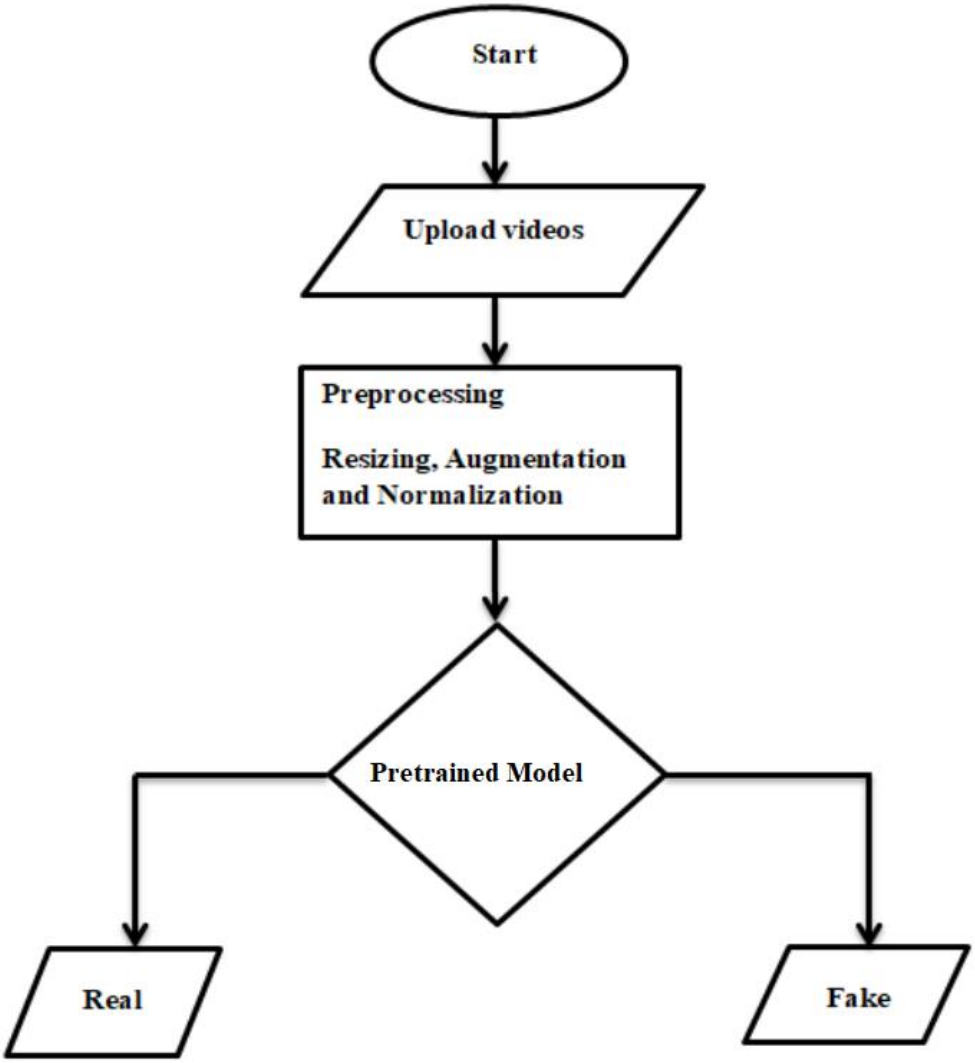

Fig 4.2:Flowchart

4.3 DFD Level 0

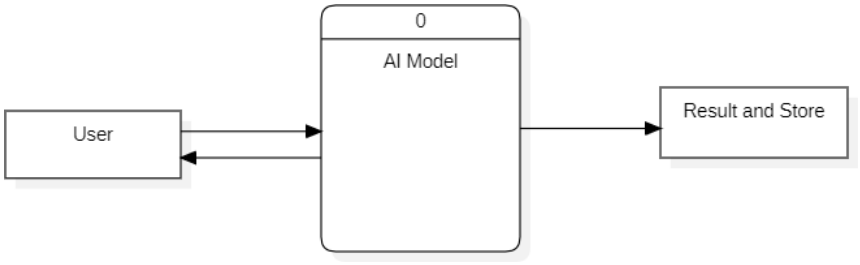

Fig 4.3:DFD Level 0

4.4 DFD Level 1

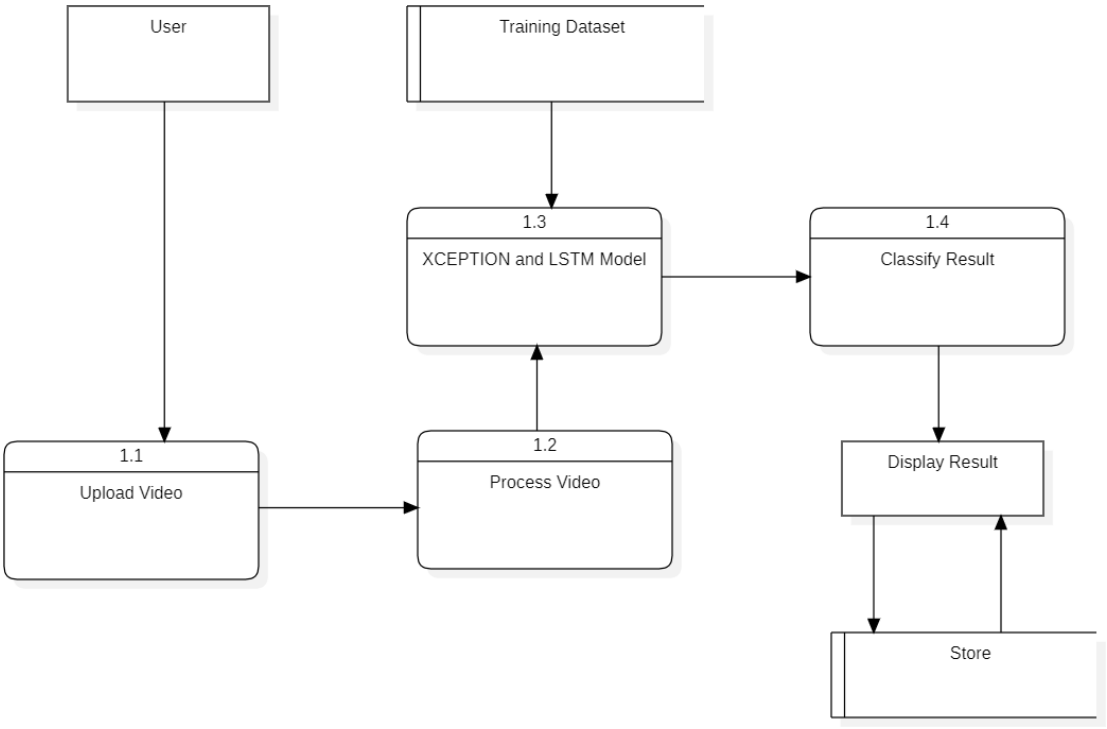

Fig 4.4:DFD Level 1

4.5 Use Case Diagram

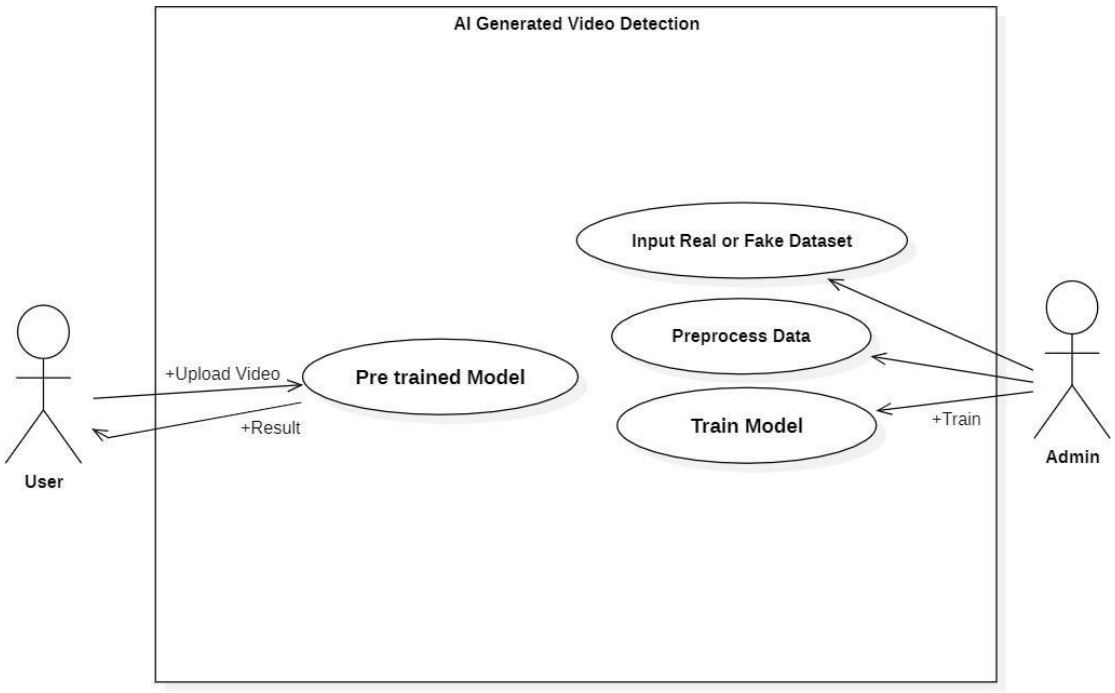

Fig 4.5 : Use Case Diagram

4.6 ER Diagram

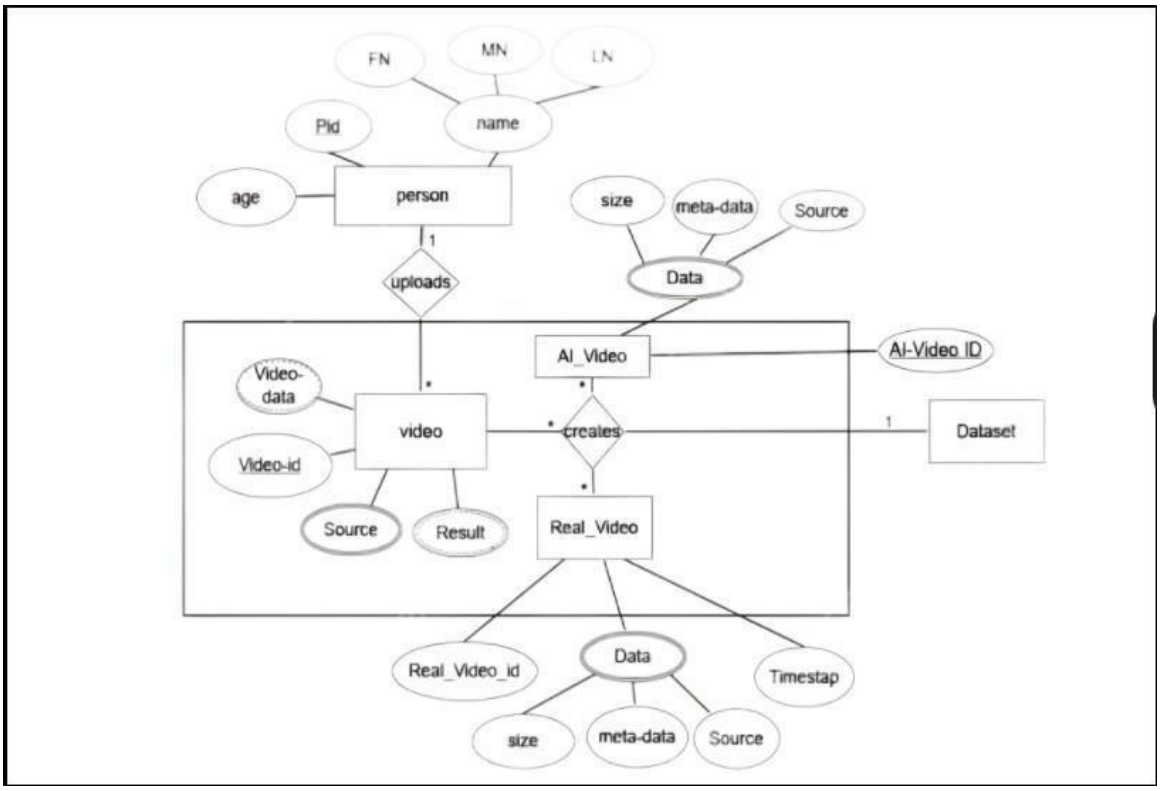

Fig 4.6: ER Diagram

CHAPTER 5

METHODOLOGY

Section 5.1 Software development model

We have chosen the Agile model as our software development model.The Agile model is an iterative and customer-centric approach to software development which involves short development cycles, frequent customer collaboration, and a focus on delivering incremental value.As any of the AI related works is of dynamic nature which requires continuous refinement and real world-feedback,This model allows to respond quickly to changing requirements, ensuring that the development process remains flexible and aligned with the evolving needs of the project.

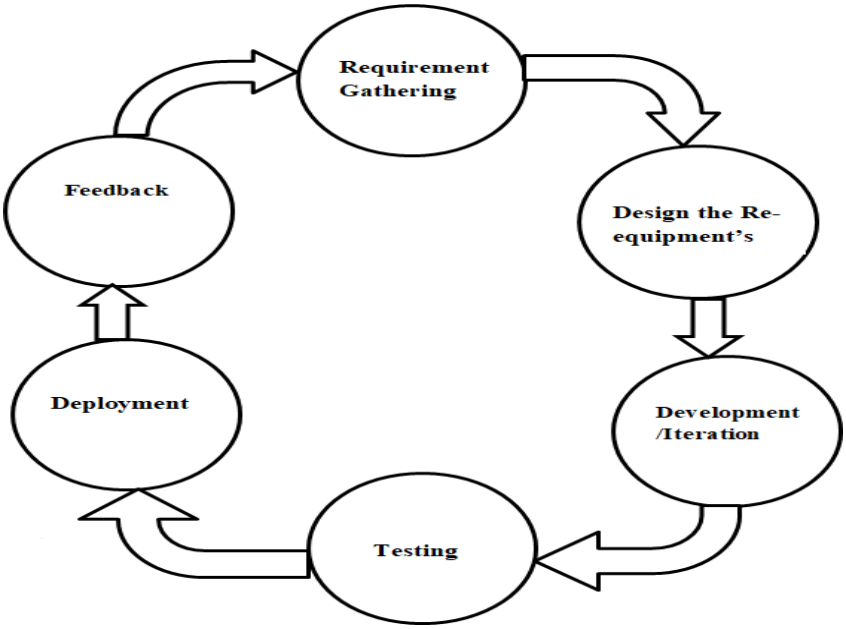

Fig 5.1: Agile Model

Step 1:Requirements Gathering

The requirements of the project are defined in this phase. The time and effort required for the project should also be discussed. By analysing this information, we will be able determine the system's economic and technical feasibility.

Step 2:Design the Re-equipments

In this phase,we analyse the system architecture and flow diagrams to further define the requirements to show the work of new features and show how it will apply to the existing system.

Step 3:Development/ iteration:

When the requirements are defined, the working on the project begins which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

Step 4:Testing:

In this phase, The product's performance is examined and looked out for the bug and errors

Step 5:Deployment:

In this phase, A product for the user's work environment is issued.

Step6:Feedback:

After releasing the product, the last step is feedback. In this, feedback about the product is received and worked through the feedback.

Section 5.2 Xception

Convolutional Neural Networks (CNNs) have revolutionised the field of computer vision by delivering remarkable performances in various image-related tasks. One such innovation in the architecture of CNNs is the XceptionNet, introduced by François Chollet in 2017. XceptionNet combines the strengths of depthwise separable convolutions and residual connections, resulting in a highly efficient and effective model. In this article, we will dive deeper into the XceptionNet architecture and explore its unique characteristics and advantages.

In Xception, The data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and Separable Convolution layers are followed by batch normalization.

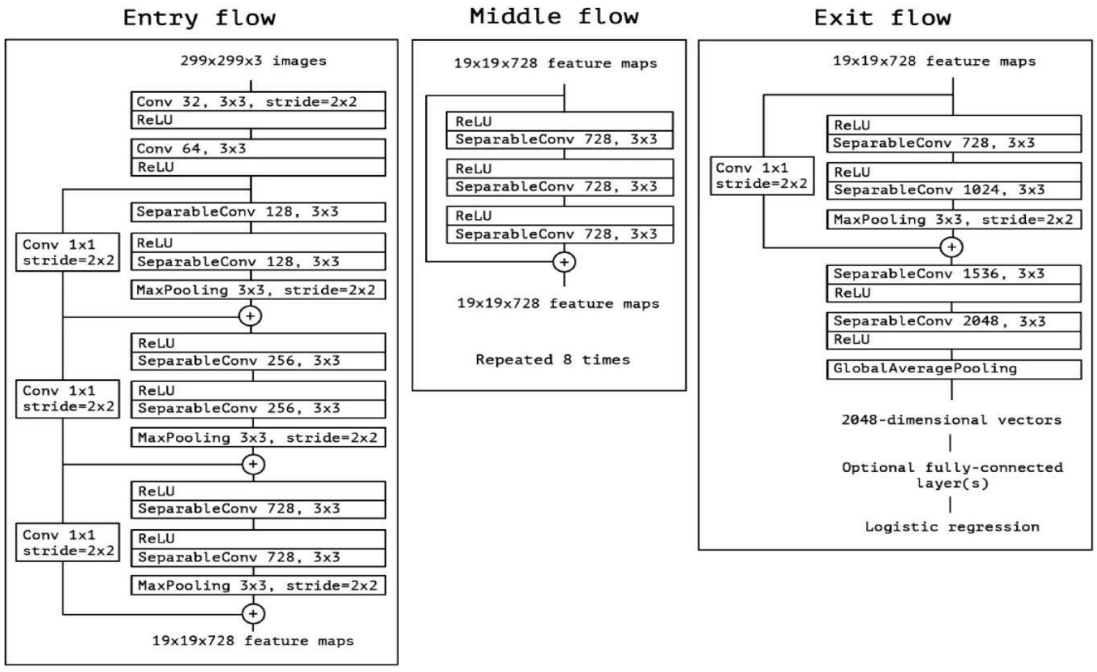

Fig 5.2 :Architecture of Xception

(Source: François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions.")

Entry Flow: The Entry Flow serves as the initial component of XceptionNet and consists of three stacked convolutional blocks. Each convolutional block contains a combination of depthwise separable convolutions, residual connections, and max pooling. This component extracts essential features from the input image and reduces its spatial dimension.

Middle Flow: The Middle Flow is responsible for processing the extracted features in a highly efficient manner. It consists of eight convolutional blocks, each having a similar structure. These blocks enhance the feature representation by incorporating residual connections, depthwise separable convolutions, and skip connections. The skip connections facilitate the direct flow of information across different blocks and foster the propagation of gradients during training.

Exit Flow: The Exit Flow is the final component of XceptionNet and focuses on the efficient classification of the input image. It comprises two convolutional blocks, followed by global average pooling and a fully connected layer. The global average pooling reduces the spatial dimensions, while the fully connected layer performs the actual classification into various classes.

Depthwise separable convolutions consist of two separate operations – depthwise convolutions and pointwise convolutions. These two operations are applied sequentially.

1) Depthwise convolution

Depthwise Convolution is a first step in which instead of applying convolution of size $d \times d \times C$, we apply a convolution of size $d \times d \times 1$. In other words, we don't make the convolution computation over all the channels, but only 1 by 1. Here is an illustration of the Depthwise convolution process :

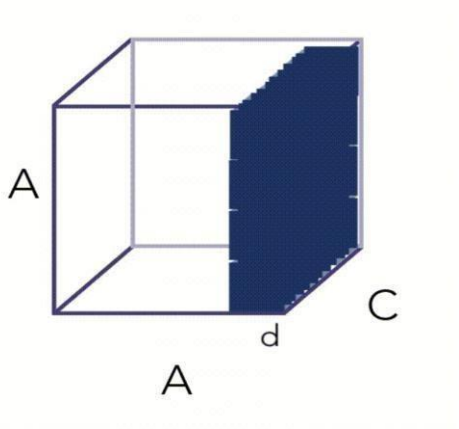

Fig 5.3:Illustration of the Depthwise convolution process

(Source:https://maelfabien.github.io/assets/images/XCeption)

This creates a first volume that has size K × K × C , and not K × K × N as before. Indeed, so far, we only made the convolution operation for 1 kernel /filter of the convolution, not for N of them. This leads us to our second step.

2) Pointwise Convolution

Pointwise convolution operates a classical convolution, with size 1 × 1 × N over the K × K × C volume. This allows creating a volume of shape K × K × N. Here is an illustration of the Pointwise Convolution :

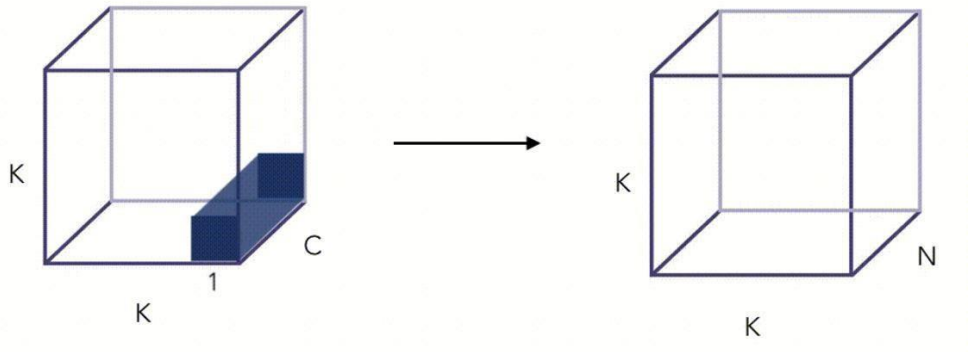

Fig 5.4:Illustration of the Point Convolution

(Source:https://maelfabien.github.io/assets/images/XCeption2)

The number of operations is reduced by a factor proportional to 1/N.

Implementation

Xception offers an architecture that is made of Depthwise Separable Convolution blocks + Max Pooling, all linked with shortcuts as in ResNet implementations.

The specificity of Xception is that the Depthwise Convolution is not followed by a Pointwise Convolution, but the order is reversed, as in this example :

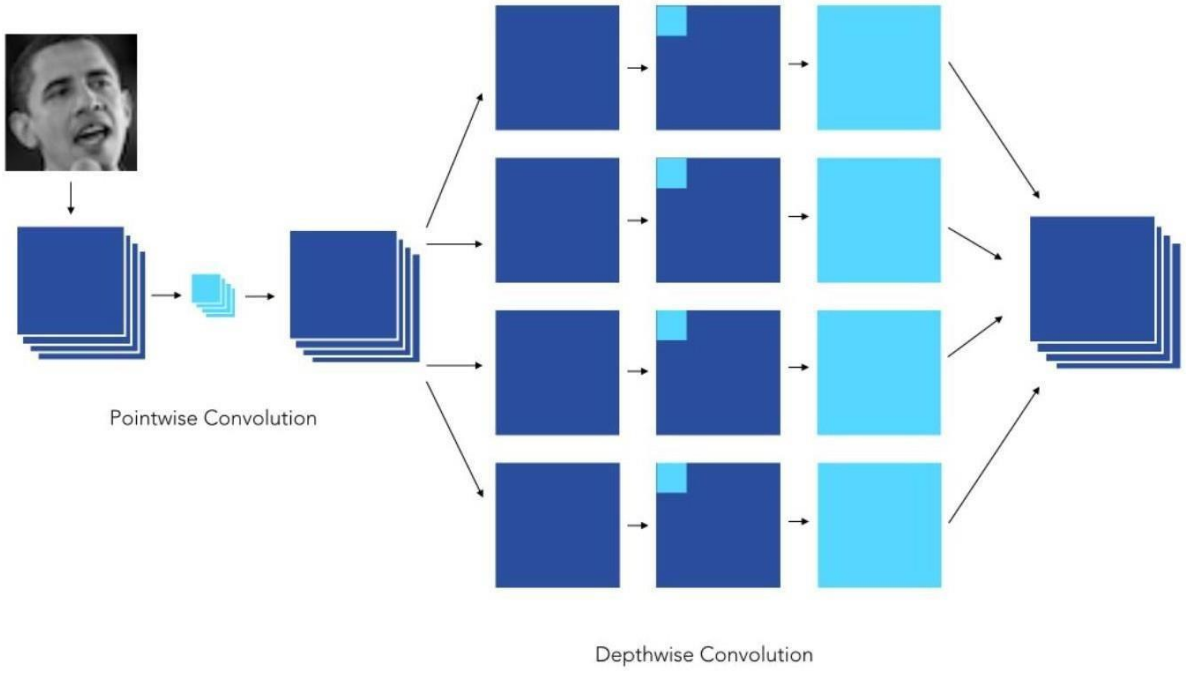

Fig 5.5: Depthwise Separable Convolution blocks + Maxpooling

(Source:https://maelfabien.github.io/assets/images/XCeption3)

ReLU

The rectified linear activation function, or ReLU for short, is a piecewise linear function that, if the input is positive, outputs the input directly; else, it outputs zero. Because a model that utilizes it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks.At the end of CNN, there is a Fully connected layer of neurons. As in conventional Neural Networks, neurons in a fully connected layer have full connections to all activations in the previous layer and work similarly. After training, the feature vector from the fully connected layer is used to classify images into distinct categories. Every activation unit in the next layer is coupled to all of the inputs from this layer. Overfitting occurs because all of the parameters are occupied in the fully-connected layer. Overfitting can be reduced using a variety of strategies, including dropout.Soft-Max is an activation layer that is typically applied to the network's last layer, which serves as a classifier. This layer is responsible for categorizing provided input into distinct types. A network's non-normalized output is mapped to a probability distribution using the softmax function.

Section 5.3: LSTM(Long short-term memory)

LSTM networks are an extension of recurrent neural networks (RNNs) mainly introduced to handle situations where RNNs fail.The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got only a single neural net layer of tanh, LSTMs comprise three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the

information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where ‘0’ means ‘reject all’ and ‘1’ means ‘include all’.

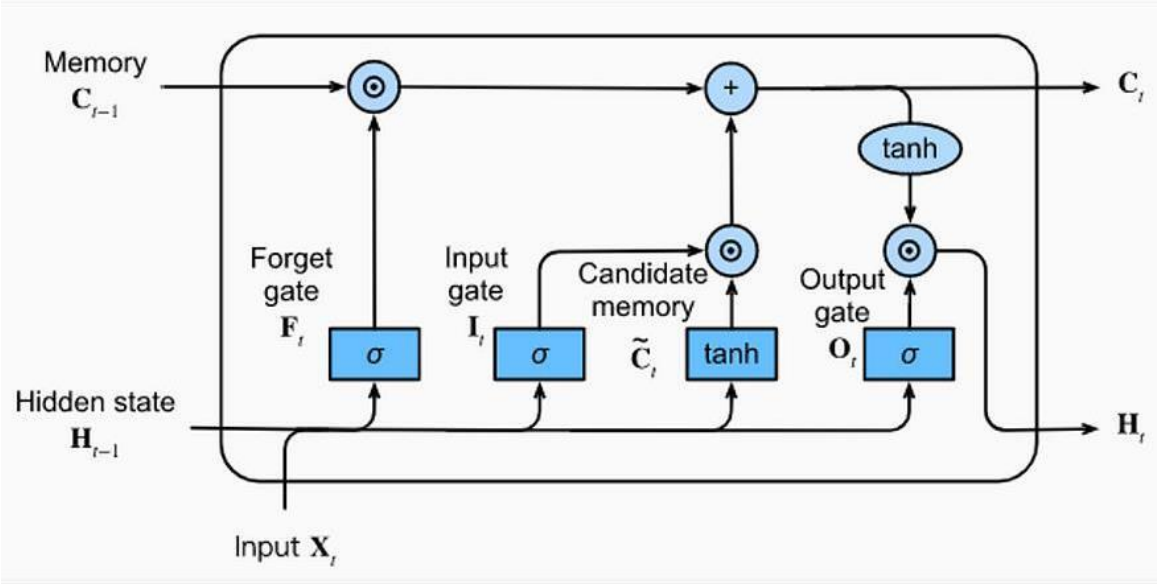

Fig 5.5: LSTM architecture

(Source: https://miro.medium.com/v2/resize:fit:875/1*Mb_L_slY9rjMr8-IADHvwg)

Forget Gate

The forget gate in a recurrent neural network helps decide which information from the past and present should be kept and which should be discarded. It does this by using weights and biases to transform inputs, followed by a simple decision: keep (output 1) or discard (output 0) the information.

Input gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered

similar to the forget gate using inputs h_t-1 and x_t. Then, a vector is created using the tanh function that gives an output from -1 to +1, which contains all the possible values from h_t-1 and x_t. At last, the values of the vector and the regulated values are multipliedto obtain useful information.

Output gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying the tanh function on the cell. Then, the information is regulated using the sigmoid function and filtered by the values to be remembered using inputs h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

CHAPTER 6

RESULT AND ANALYSIS

By harnessing the power of convolutional neural networks (CNNs) like Xception and recurrent neural networks (RNNs) like LSTM, our web application aims to accurately identify manipulated facial images and distinguish them from genuine ones.

Throughout this chapter, we will delve into the output produced by our detection system through screenshots of ,

1. Model Total Loss vs Total Validation Loss

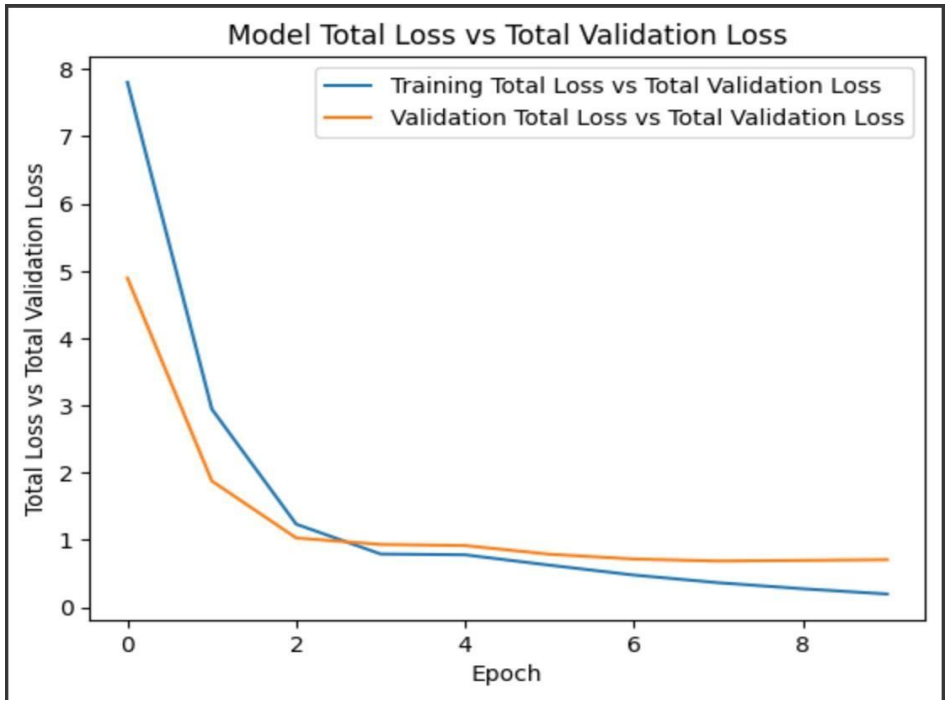

Fig: Model Total Loss vs Total Validation Loss

2. Model Total Accuracy vs Total Validation Loss

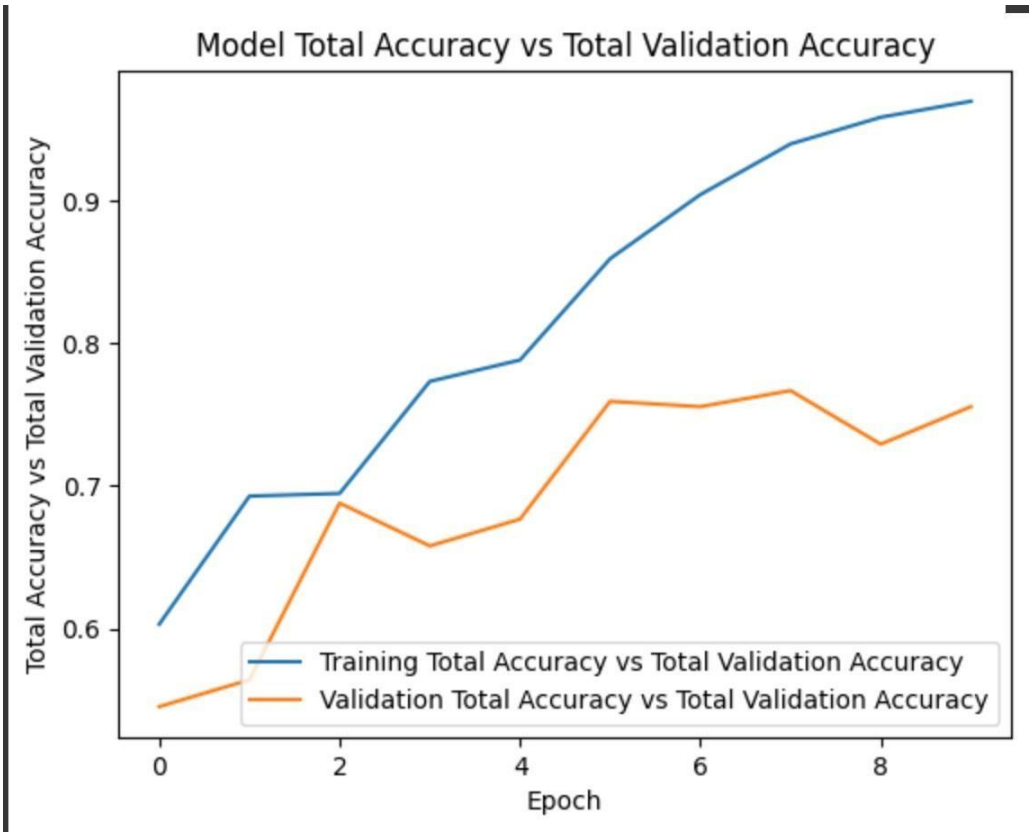

Fig: Model Total Accuracy vs Total Validation Loss

3. Model Accuracy

```
# Evaluate the trained model.
model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)
```

```
8/8 [==============================] - 16s 906ms/step - loss: 0.4630 - accuracy: 0.8625
```

Fig: Model Accuracy

4. Testing Loss and Testing Accuracy

```
Extracting Data of Class: Real
Extracting Data of Class: Fake
2/2 [==============================] - 5s 2s/step - loss: 0.4945 - accuracy: 0.8387
Testing Loss: 0.49448615312576294
Testing Accuracy: 0.8387096524238586
```

Fig: Testing Loss and Testing Accuracy

Output from the user side:

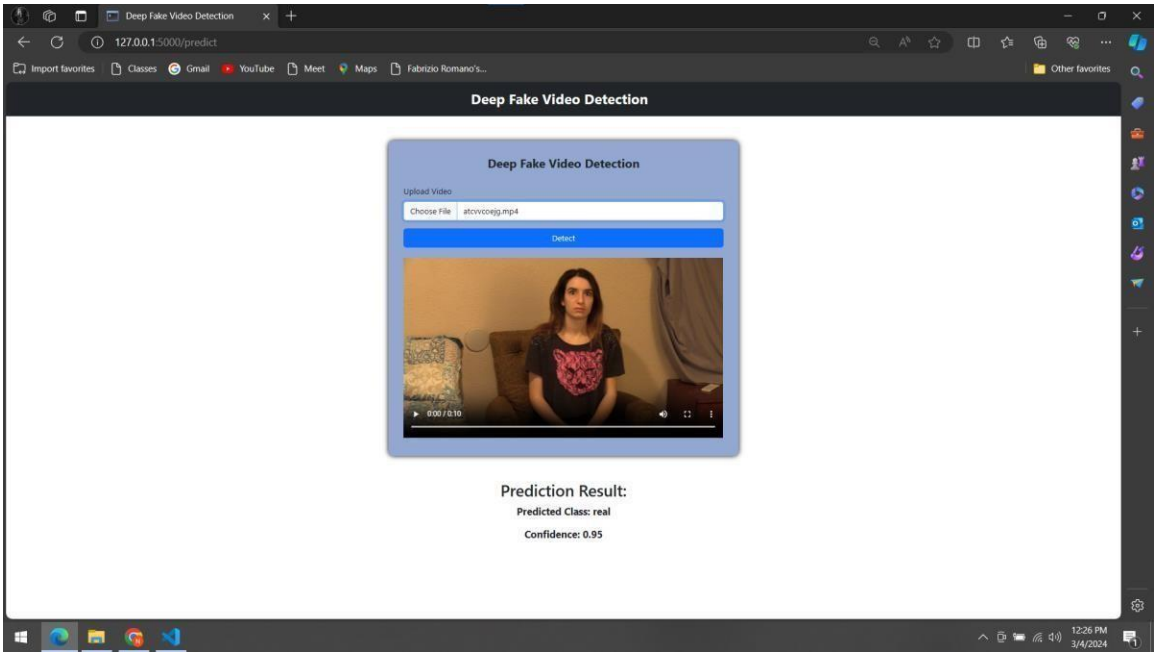

Fig: Output(Real)

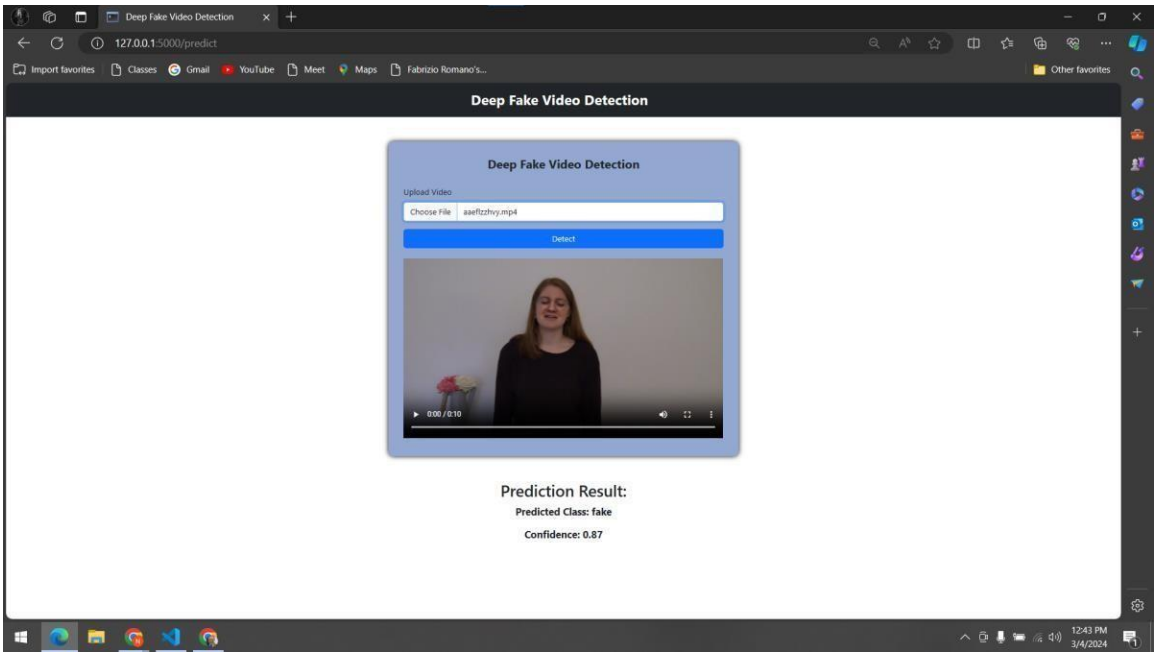

Fig: Output(Fake)

CHAPTER 7

CONCLUSION, LIMITATIONS AND FUTURE ENHANCEMENT

Conclusion

Finally, We would like to conclude that the result of our "AI GENERATED VIDEO DETECTOR" is better. Using XceptionNet and LSTM, we achieved overall accuracy of 86.25% , along with testing accuracy of 83.87%. And, using our system for deepfake detection system is one effective approach. Also, our system is useful in the real world.

Limitation

Some of the limitations that we see are listed below as:

1. Generalization to New Deepfake Variants are bit challenging.
2. Incompatible with mobile devices.

FUTURE ENHANCEMENT

1. Using the latest release of ResNet 152 could be the best for deepfake video detection.
2. Responsive GUI for all the devices.
3. Continuous monitoring and updating.

REFERENCES

[1] https://cs230.stanford.edu/projects_spring_2020/reports/38857501

[2] Deepfake detection challenge dataset: https://www.kaggle.com/c/deepfake-detection challenge/data Accessed on 26 March 2020

[3] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi and Siwei Lyu “Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics” in arXiv:1909.12962

[4] Nagothu D, Xu R, Chen Y, Blasch E, Aved A. Deterring Deepfake Attacks with an Electrical Network Frequency Fingerprints Approach. *Future Internet*. 2022; 14(5):125. https://doi.org/10.3390/fi14050125

[5] Chollet, François. "Xception: Deep Learning with Depthwise Separable Convolutions." arXiv preprint arXiv:1610.02357 (2016).

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

[7] Aarti Karandikar.et al., International Journal of Advanced Trends in Computer Science and Engineering, 9(2), March - April 2020, 1311 – 1315

[8] Zhiming Xie et al 2019 J. Phys.: Conf. Ser. 1395 012006

[9] Yuezun Li, Ming-Ching Chang and Siwei Lyu “Exposing AI Created Fake Videos by Detecting Eye Blinking” in arXiv:1806.02877v2.

[10] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen “Using capsule networks to detect forged videos and videos” in arXiv:1810.11215

[11] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.

[12] Umur Aybars Ciftci, ˙Ilke Demir, Lijun Yin “Detection of Synthetic Portrait Videos using Biological Signals” in arXiv:1901.02212v2

[13] Gragnaniello, D., Marra, F., Verdoliva, L. (2022). Detection of AI-Generated Synthetic Faces. In: Rathgeb, C., Tolosana, R., Vera-Rodriguez, R., Busch, C. (eds) Handbook of Digital Face Manipulation and Detection. Advances in Computer Vision and Pattern Recognition. Springer, Cham.

[14] Adil Mohammed Parayil, Ameen Masood V, Muhammed Ajas P, Tharun R, Usha K. "Deepfake Detection Using Xception and LSTM." International Research Journal of Modernization in Engineering Technology and Science, Volume 05, Issue 05, May 2023, DOI: https://www.doi.org/10.56726/IRJMETS41123