TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

AI GENERATED IMAGE AND VIDEO DETECTION USING CNN

BY

Sulab Nepal(PUR076BEI042)

Saphal Bhattarai(PUR076BEI032)

Shishir ParajuliPUR076BEI035)

Pushpa Kamal Dahal(PUR076BEI026)

A PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

MARCH, 2024

AI GENERATED IMAGE AND VIDEO DETECTION USING CNN

By

Sulab Nepal(PUR076BEI042)

Saphal Bhattarai(PUR076BEI032)

Shishir Parajuli(PUR076BEI035)

Pushpa Kamal Dahal(PUR076BEI026)

Project Supervisor

Asst. Prof. Mazhar Ali

A project submitted to the Department of Electronics and Computer Engineering in partial fulfillment of the requirements for the Bachelor's Degree in Computer Engineering

Department of Electronics and Computer Engineering
Purwanchal Campus, Institute of Engineering
Tribhuvan University
Dharan, Nepal

March, 2024

COPYRIGHT©

DECLARATION

We declare that the work hereby submitted for Bachelors of Engineering in Computer Engineering at Institute of Engineering, Purwanchal Campus entitled **"AI GENERATED IMAGE AND VIDEO DETECTION USING CNN"** is our own work and has not been previously submitted by me at any university for any academic award.

We authorize Institute of Engineering, Purwanchal Campus to lend this report to other institution or individuals for the purpose of scholarly research.

Sulab Nepal(PUR076/BEI/042)
Saphal Bhattarai(PUR076/BEI/032)
Shishir Parajuli(PUR076/BEI/035)
Pushpa Kamal Dahal(PUR076/BEI/026)

March, 2024

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a project entitled **"AI GENERATED IMAGE AND VIDEO DETECTION USING CNN"**, submitted by **Sulab Nepal, Saphal Bhattarai, Shishir Parajuli, Pushpa Kamal Dahal** in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering in Electronics, Communication And Information Engineering"**.

.....

Asst. Prof. Mazhar Ali

Supervisor

Department of Electronics and Computer Engineering

Purwanchal Campus, Institute of Engineering, Tribhuvan University

.....

Assoc. Prof. Surendra Shrestha, (PhD)

External Examiner

Department of Electronics and Computer Engineering

Pulchowk Campus, Institute of Engineering, Tribhuvan University

.....

Asst. Prof. Pravin Sangroula

Head of Department

Department of Electronics and Computer Engineering

Purwanchal Campus, Institute of Engineering, Tribhuvan University

5th March, 2024

DEPARTMENTAL ACCEPTANCE GOES HERE

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards the Institute of Engineering, Tribhuvan University for the inclusion of major project during Bachelor's in Electronics and Communication Engineering. We are also thankful to our supervisor Er. Mazhar Ali and the Department of Electronics and Computer Engineering, IOE, Purwanchal Cam- pus for rendering us with the resources and support which is required for this project.

SULAB NEPAL (PUR076/BEI/042)
PUSHPA KAMAL DAHAL(PUR076/BEI/026)
SAPHAL BHATTARAI (PUR076/BEI/032)
SHISHIR PARAJULI (PUR076/BEI/035)

ABSTRACT

In today's digital landscape, the proliferation of image and video content across various online platforms has led to significant challenges in content moderation, copyright infringement detection, and ensuring online safety. Traditional methods of image and video detection often struggle to keep pace with the sheer volume and diversity of content being generated and shared. However, recent advancements in artificial intelligence (AI) have opened up new possibilities for more effective detection methods.This report explores the application of AI-generated techniques in image and video detection. It examines the use of deep learning algorithms, such as convolutional neural networks (CNNs) and generative adversarial networks (GANs), to enhance detection capabilities. These techniques enable the creation of synthetic images and videos that can be used to train and improve detection models, effectively augmenting limited datasets and addressing the challenges posed by rapidly evolving content. Furthermore, the report discusses the advantages and limitations of AI-generated detection methods, including their ability to detect manipulated or deepfake content, identify sensitive or explicit material, and flag potential copyright violations. It also addresses ethical considerations surrounding the use of AI-generated content in detection systems, including concerns related to privacy, bias, plagiarism and potential misuse. Through a comprehensive review of existing literature, case studies, and practical implementations, this report provides insights into the current state of AI-generated image and video detection. It highlights the potential for these techniques to revolutionize content moderation and online safety efforts, while also emphasizing the importance of responsible development and deployment practices to mitigate risks and ensure the ethical use of AI in detection systems.

Keywords: Artificial Intelligence, Content Generation, Plagiarism

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	: Application Programming Interface
GAN	: Generative Adversarial Network
Colab	: Colaboratory
LSTM	: Long Short Term Memory
MAE	: Mean Average Error
MSE	: Mean Square Error
NumPy	: Numerical Python
AI	: Artficial Intelligence
RELU	: Rectified Linear Unit
RMSE	: Root Mean Square Error
RNN	: Recurrent Neural Network
LLM	: Large Language Model
sklearn	: Scikit Learn
CNN	: Convolutional Neural Network

CHAPTER 1

INTRODUCTION

1.1 Background

Artificial Intelligence (AI) has rapidly advanced in recent years, leading to the creation of various AI applications, including content generation. AI-generated content refers to any type of content created with the aid of machine learning algorithms or other AI technologies. This is not new, but it has become very popular after the launch and promotion of ChatGPT.

Image-generation AI models have made significant progress in recent years, particularly with the advances in neural network techniques. This has led to the development of several impressive models including DALL-E 2 developed by OpenAI. DALL-E 2 is a transformer auto regressive model in which the model learns to map the input text to a latent space representation, which is then used to generate a corresponding image. Thus, DALL-E 2 can be effectively used for fake image generation through textual descriptions. DALL-E 2's ability to generate high-quality images has significant implications for a wide range of applications, including entertainment, advertising, digital art, and even architecture design. However, the use of AI-generated images raises important questions about the ethical implications of using such images.

Also, in this work, we describe a new deep learning-based method that can effectively distinguish AI-generated fake videos from real videos. We are using the limitation of the deep fake creation tools as a powerful way to distinguish between the pristine and deep fake videos. During the creation of the deep fake, the current deep fake creation tools leaves some distinguishable artifacts in the frames which may not be visible to the human being, but the trained neural networks can spot the changes. Deepfake creation tools leave distinctive artifacts in the resulting Deep Fake videos, and we show that they can be effectively captured by Res-Next Convolution Neural Networks. Our system uses a Res-Next Convolution Neural Networks to extract frame-level features. These features are then used to train a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) to classify whether the video is subject to any kind of manipulation or not, i.e., whether the video is deep fake or real video.

1.2 Gap Identification

The detection of AI-generated images is crucial due to the potential for their use in various malicious activities, including deep fake videos, synthetic identity fraud, propaganda, and art fraud. However, the increasing sophistication and realism of AI-generated images have made it challenging for humans to distinguish between real and fake images.

Deep fake videos are a compelling example of the misuse of AI-generated images. These videos manipulate existing footage using AI technology to replace the face of one person with another, creating fake news, spreading propaganda, and damaging the reputation of individuals. Similarly, AI-generated images can also be used for synthetic identity fraud, where criminals create fake identities using stolen personal information and AI-generated images to commit fraudulent activities like opening bank accounts or applying for loans. Additionally, the recent controversy surrounding AI-generated artwork in Taiwan has highlighted the potential for fraud in art sales. This debate centers on the use of computer-generated tools to create art and the concerns that this undermines the work of traditional artists who rely solely on their own skills and expertise. As AI-generated images become increasingly sophisticated, it is becoming more challenging for the public to differentiate between real and fake artwork, creating opportunities for fraud.

Our research aims to develop a machine learning pipeline that can detect fraud in images and videos, thereby safeguarding against the negative consequences of AI-generated images and deep fakes. By gaining a deeper understanding of the differences between AI-generated facial images and real human faces, we hope to contribute to the development of more responsible and ethical practices for using AI-generated images across different fields.

1.3 Motivation

With the rapid advancements of a AI and natural language generations technologies, there is a growing concerns over the proliferation of AI generated content that can deceived, misinform and manipulate users.To address this pressing issue, this system is design to identify and flag content produced by AI modules.

1.4 Objectives

- To design a system that detects AI generated image and videos.
- To design a system that maintains author authenticity.

1.5 Feasibility Analysis

Conducting a feasibility analysis is a critical step in assessing the viability of a project, wherein the potential success is evaluated through an examination of economic, technical, and operational factors. In this particular analysis, we will assess the feasibility of creating a detection model for the classification of AI generated content.

1.5.1 Economic Feasibility

• Evaluate the cost-effectiveness of developing the detection model for AI generated content classification.

• Consider the budget required for acquiring necessary technologies, tools, and resources.

• Evaluate the potential return on investment (ROI) by assessing increased user engagement in detecting AI content , particularly within copyrights and legal issues.

1.5.2 Technical Feasibility

• Examine the technical capabilities and requirements for implementing the detection model.

• Assess the availability and compatibility of the required technology stack for deep learning and image and video processing.

• Ensure feasibility in terms of computational resources and infrastructure to support the model's development and deployment.

1.5.3 Operational Feasibility

• Evaluate the practicality of integrating the detection model into existing AI content detection model.

• Assess the ease of model interpretation and explain ability for end-users.

• Consider the operational impact on copyrights with legal issues and other applications, ensuring data privacy, ethical considerations and regularity compliance.

CHAPTER 2

LITERATURE REVIEW

In recent years, there has been a significant surge in interest in Deep Learning-based models, particularly in image analysis. Notably, the emergence of Diffusion Models, as exemplified in the review paper by Croitoru et al.[1], has yielded remarkable outcomes in generating high- fidelity images full of authentic details. However, generative models are often characterized by discernible idiosyncratic patterns, which can be exploited to ascertain an image's genuineness. Despite concerted efforts to mitigate the presence of such patterns within Diffusion Models, it is imperative to acknowledge that they are not entirely free of such distinctive traits. Specifically, research has underscored the significance of features such as color band inconsistency [2] and the paucity of variation in color intensity[3] in identifying synthetic images.

Consequently, various Computer Vision techniques have been harnessed to facilitate the automated detection of the factors mentioned above. Guo et al. [4] and Guarnera et al. [5] have introduced hierarchical fine-grained classification methodologies explicitly designed to discriminate between forged or synthetic images. Central to their approach is the meticulous curation of a training dataset, wherein the hierarchical framework necessitates the comprehen- sive incorporation of diverse forgery techniques. This endeavor, however, poses a challenge, particularly in scenarios characterized by limited diversity within the available training data. Addressing the recognition of semantic and stylistic features in images, Amoroso et al. [6] have tackled the issue of discerning synthetic images that exhibit heightened distinctiveness within the stylistic domain. Nonetheless, the practical implementation of semantic-style disentangle- ment presents notable challenges, primarily requiring the development of bespoke training datasets tailored explicitly to this specific objective. Hamid et al. [7] proposed a framework for fake image detection based on ResNet, achieving good results on a dataset containing real and fake images generated by GANs.

In the digital humanities domain, identifying forgeries, often manifesting as synthetic images, holds paramount significance for art experts. This task tests whether a given artwork is an authentic painter's creation or a generative algorithm's output. Nevertheless, the scrutiny of fine arts presents demanding challenges due to the intrinsic variability

and subjectivity inherent in the data. Notably, similar objects may be rendered in divergent artistic styles, while conversely, a singular artwork can evoke disparate emotional reactions among different observers. In response to this complexity, numerous models tailored for artwork analysis have embraced a multimodal approach, accommodating various data inputs such as images, text descriptions, or structured knowledge graphs. For example, in [8], we proposed a multimodal neural architecture adept at combining visual and contextual features from artworks, thereby facilitating the identification of their style and genre. Similarly, Bose et al. [9] presented a Transformer-based architecture to perform sentiment analysis within visual arts. This model effectively integrates visual features extracted from images with textual embeddings derived from painting descriptions.

CHAPTER 3

RELATED THEORY

3.1 Convolutional Neural Network

As an algorithm with excellent performance, convolutional neural network has been widely used in the field of image processing and achieved good results by relying on its own local receptive fields, weight sharing, pooling, and sparse connections. Convolutional Neural Network (CNN), also called ConvNet, is a type of Artificial Neural Network(ANN), which has deep feed-forward architecture and has amazing generalizing ability as compared to other networks with FC layers, it can learn highly abstracted features of objects especially spatial data and can identify them more efficiently. A deep CNN model consists of a finite set of processing layers that can learn various features of input data (e.g. image) with multiple level of abstraction . The initiatory layers learn and extract the high level features (with lower abstraction), and the deeper layers learns and extracts the low level features (with higher abstraction).

3.1.1 Network Layers

CNN is composed of multiple building blocks (known as layers of the architecture), in this subsection, we described some of these building blocks in detail with their roll in the CNN architecture.

2.1.1.1 Convolutional Layer

Convolutional layer1 is the most important component of any CNN architecture. It contains a set of convolutional kernels (also called filters), which gets convolved with the input image (N-dimensional metrics) to generate an output feature map.

2.1.1.2 Pooling layer

The pooling layers are used to sub-sample the feature maps (produced after convolution operations), i.e. it takes the larger size feature maps and shrinks them to lower sized feature maps. While shrinking the feature maps it always preserve the most dominant features (or information) in each pool steps. The pooling operation is performed by

specifying the pooled region size and the stride of the operation, similar to convolution operation. There are different types of pooling techniques are used in different pooling layers such as max pooling, min pooling, average pooling, gated pooling, tree pooling, etc. Max Pooling is the most popular and mostly used pooling technique.

The formula to find the output feature map size after pooling operation as below:

$$h' = [\frac{h-f}{s}].$$

$$w' = [\frac{w-f}{s}].$$

Where h' denotes the height of the output feature map, w' denotes the width of the output feature map, h denotes the height of the input feature map, w denotes the width of the input feature map, f is the pooling region size and s denotes the stride of the pooling operation.

2.1.3 Activation Functions (Non-Linearity)

The main task of any activation function in any neural network based model is to map the input to the output, where the input value is obtained by calculating the weighted sum of neuron's input and further adding bias with it (if there is a bias). In other words, the activation function decides whether a neuron will fire or not for a given input by producing the corresponding output.

2.1.3.1 Sigmoid:

The sigmoid activation function takes real numbers as its input and bind the output in the range of [0,1]. The curve of the sigmoid function is of 'S' shaped. The mathematical representation of sigmoid is:

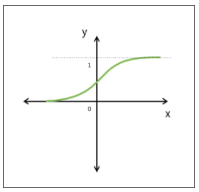

Figure 3.1: Sigmoid

2.1.3.2 ReLU:

The Rectifier Linear Unit (ReLU) is the most commonly used activation function in Convolutional Neural Networks. It is used to convert all the input values to posititve

numbers. The advantage of ReLU is that it requires very minimal computation load compared to others. The mathematical representation of ReLU is:

$$(f(x)ReLU = max(0,x).$$

3.1.2 Loss Functions

In section 2.1.1, we has described different types of layers used in CNN architecture.Now, we know that the last layer of every CNN architecture (classification based) is the output layer, where the final classification takes place. In this output layer, we calculate the prediction error generated by the CNN model over the training samples using some Loss Function. This prediction error tells the network how off their prediction from the actual output, and then this error will be optimized during the learning process of the CNN model. The loss function uses two parameters to calculate the error, the first parameter is the estimate output of the CNN model (also called the prediction) and the second one is the actual output (also known as the label). There are different types of loss functions used in different types of problem. Some of the most used loss functions is briefly described in next subsections.

2.1.2.1 Cross-Entropy or Soft-Max Loss Function

Cross-entropy loss, also called log loss function is widely used to measure the performance of CNN model, whose output is the probability p E 0, 1. It is widely used as an alternative of squared error loss function in the multi-class classification problems. It uses softmax activations in the output layer to generate the output within a probability distribution, i.e. p, y E RN , where p is the probability for each output category and y denotes the desired output and the probability of each output class can be obtained by:

$$P = \frac{e^a.k}{\sum_{k=1}^{\infty} e^a.k}$$

where N is the number of neurons in the output layer from the previous layer in the network. Now finally, cross-entropy loss can be defined as :

$$H(p,y) = -\sum_{i} y.log(p),$$

$$where i \varepsilon [1,N]$$

3.1.3 Training process of Convolutional Neural Network

Here in this section we try to discuss the training or learning process of a CNN model with certain guidelines in order to reduce the required training time and to improve model accuracy. The training process mainly includes the following steps:

• Data pre-processing and Data augmentation.

• Parameter initialization

• Regularization of CNN

• Optimizer selection

Data pre-processing and Data augmentation

Data pre-processing refers to some artificial transformations to the raw dataset (including training, validation and testing) in order to make the dataset more clean, more feature- full, more learnable and in a uniform format. The data pre-processing is done before feeding the data to the CNN model. In a convolutional neural, network it is a fact that the performance of CNN is directly proportional to the amount of data used to train it, i.e good pre-processing, always increases accuracy of the model. But on the other side, a bad pre-processing can also reduces the performance of the model.

Parameter Initialization

A deep CNN consists of millions or billions number of parameters. So, it must be well initialized at the begin of the training process, because weight initialization directly determines how fast the CNN model would converge and how accurately it might end up. Here in this section, we discuss some mostly used parameter initialization techniques used in CNN as follows: The most easiest way to doing it is by initializing all

the weights with zero. However, This turns out to be a mistake, because if we initialize weights of all layer to zero, the output as well as the gradients (during backpropagation) calculated by every neuron in the network will be the same. Hence the update to all the weights would also be the same. As a result, there is no disparity between neurons and the network will not learn any useful feature

Regularization to CNN

The core challenge of deep learning algorithms is to adapt properly to new or previously unseen input, drawn from the same distribution as training data, the ability to do so is called generalization. The main problem for a CNN model to achieve good generalization is over-fitting. When a model performs exceptionally well on training data but it fails on test data (unseen data), then this type of model is called over-fitted. The opposite is an under-fitted model, that happen when the model has not learned enough from the training data and when the model performs well on both train and test data, then these types of models are called just-fitted model. Fig.24 try to show the examples of over-fitted, under-fitted and just-fitted models.

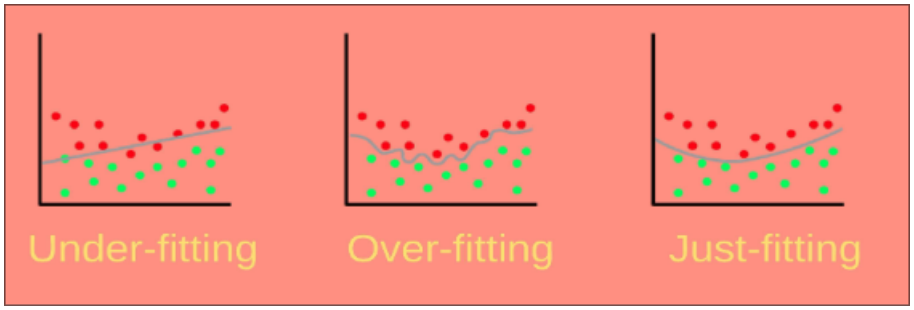

Figure 3.2: The Examples of over-fitting, under-fitting and just-fitting

Optimizer selection

After successfully discussing the pre-processing steps with the data samples and enforcing regulization techniques to the CNN model, here we are going to discuss the learning process of the CNN model. The learning process includes two major things, first one is the selection of the learning alorithm (Optimizer) and the next one is to use several improvements (such as momentum, Adagrad, AdaDelta) to that learning algorithm in

order to improve the result.. In case of learning to a CNN model the gradient-based learning methods come as a natural choice. To reduces the error the model parameters are being continuously updated during each training epoch and the model iteratively search for the locally optimal solution in each training epoch.The size of parameter updating steps is called the “learning rate” and a complete iteration of parameter update which includes the whole training dataset once is called a “training epoch”. Although the learning rate is a hyper-parameter, but we need to choose it so carefully that, it does not affect the learning process badly.

3.2 AI Generation and Detection

Art has undergone a profound transformation with the emergence of generative Artificial Intelligence, notably driven by technologies such as Generative Adversarial Networks [1] and the increasingly popular Diffusion Models [2]. These groundbreaking innovations have pushed the boundaries of artistic creation, empowering machines to produce remarkably lifelike images, including paintings, that challenge our conventional notions of human creativity. Indeed, generative AI has exhibited the capability to generate synthetic paintings that closely emulate renowned artists’ styles, brushwork, and aesthetics. This level of fidelity in replicating the artistic process blurs the demarcation between traditional, human-crafted art and machine- generated creations.
The distinction between genuine human-made art and its synthetic counterparts carries ex- tensive implications, influencing aspects such as art authentication, valuation, and preservation while igniting debates concerning technology’s role in the creative process. While conven- tional methods of art connoisseurship traditionally relied on expert human judgment, the rapid evolution of Deep Learning models and the availability of extensive art datasets present new avenues for addressing this challenge. One intriguing approach to detecting instances gener- ated by machines, in fact, involves leveraging the capabilities of machines themselves. This concept is rooted in the idea that the same AI technologies responsible for creating synthetic content can also be employed for their detection and differentiation from authentic human-made counterparts. Deep Learning and Computer Vision algorithms can also be trained on large datasets containing authentic and AI-generated examples. These models can learn to identify subtle patterns, inconsistencies, or artifacts that may indicate machine generation. By utilizing

AI-powered classification models, we can automate the process of detecting machine-generated content, achieving both efficiency and accuracy.

In recent years, there has been a significant surge in interest in Deep Learning-based models, particularly in image analysis. Notably, the emergence of Diffusion Models, as exemplified in the review paper by Croitoru et al. [2], has yielded remarkable outcomes in generating high- fidelity images full of authentic details. However, generative models are often characterized by discernible idiosyncratic patterns, which can be exploited to ascertain an image's genuineness. Despite concerted efforts to mitigate the presence of such patterns within Diffusion Models, it is imperative to acknowledge that they are not entirely free of such distinctive traits. Specifically, research has underscored the significance of features such as color band inconsistency [7] and the paucity of variation in color intensity [8] in identifying synthetic images.

CHAPTER 4

METHODOLOGY

4.1 Tools and Technologies

Here for this project, we used Agile Model for software development. This software development approach is based on iterative development. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration, and the scope of each iteration are clearly defined in advance.

4.1.1 Machine Learning Frameworks

• **Scikit-learn:** This versatile machine learning library played a crucial role in our project, offer- ing comprehensive support for a range of tasks. From efficient data preprocessing techniques to thorough model evaluation methods, Scikit-learn significantly contributed to the success of our machine learning pipeline.
• **TensorFlow:** TensorFlow is another powerful machine learning framework that played a vital role in our project. Known for its flexibility and scalability, TensorFlow provided a solid foundation for implementing deep learning models, adding diversity to our approach in music genre classification.

4.1.2 Data Analysis and Visualization:

• **NumPy**: Integral for efficient numerical operations on arrays and matrices, NumPy enhanced the performance of our data processing tasks. Its capabilities were particularly valuable in handling large datasets and optimizing numerical computations.
• **Pandas**: As a powerful data manipulation and analysis library, Pandas streamlined our data preprocessing workflows. Its intuitive dataframe structure and rich set of functions were essential for managing, cleaning, and transforming our dataset.
• **Matplotlib**: Leveraging Matplotlib's visualization capabilities, we gained valuable insights into our dataset and model performance. The library enabled the creation of clear and informative visualizations to enhance our understanding of the classification

results.

4.1.3 User Interface Development

Streamlit: Streamlit is an open-source Python framework for machine learning and data science teams. For user interface, we have used Streamlit which is a open source app framework. According to our project, we have used three models:

• AI Generated Video Detection

• AI Generated Image Detection

• AI Generated Face Detection

Django: Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.

4.1.4 Development Environment

• **Jupyter Notebook**: Providing an interacive and collaborative coding environment, Jupyter Notebook were instrumental throughout our development process. They facilitated seamless code development, testing, and documentation, promoting efficient collaboration among team members.

4.1.5 Version Control

• **Git and GitHub**: Git served as our version control system, allowing for efficient collaboration and tracking of changes. GitHub, as a hosting platform, provided a centralized repository for our project, ensuring version history management and streamlined collaboration among team members.

4.2 Data Collection

For AI generated image detection ,the dataset used for training and evaluation consists of two categories of images with total of 1,00,000 with 50,000 each.

REAL images: These images are sourced from the Krizhevsky and Hinton's CIFAR-10 dataset, which is a widely-used benchmark dataset for image classification tasks.

FAKE images: These images were generated using the equivalent of CIFAR-10 with

Stable Diffusion version 1.4.Also, 10,000 images from Midjourney was used.
Also, For the AI generated synthetic face images, total 1,40,000 images were used categorizing into two categories.
REAL images: 55,000 celebrities faces were downloaded from Kraggle.
FAKE images: 90,000 synthetic AI generated images were downloaded from kraggle.

For making the model efficient for real-time prediction of AI generated videos, we have gathered the data from different available datasets like FaceForensic++, DFMNIST+ , and Celeb-DF-V2. To avoid the training bias of the model we have considered 50% Real and 50% fake video which makes our total dataset consists of almost 3000 Real, 3000 fake videos, and roughly 6000 videos in total. .

4.3 Data Pre-Processing

Here for the detection of AI generated images,we considered following methods:
• Resizing the images to a consistent size (32*32,224x224 pixels) and normalizing the pixel values.
• Applying data augmentation techniques to artificially increased the sized and diversity of the dataset.
• Splitting the dataset into training and test sets for model evaluation.

Also, for the detection of AI generated videos, the first step in the pre-processing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frames and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while pre-processing.

4.4 Implementation of the system design for Image Detection

For the AI generated image detection, all the necessary libraries and modules were imported for building and training the model. Then, all the necessary datasets were loaded

according to its respective model with REAL and FAKE folders. After which, all the necessary data were pre-processed with proper scaling. After that, Splitting the data into training, cross-validation and testing sets were performed with following functions:

train-size = int(len(data)*0.7)

cv-size = int(len(data)*0.2)

test-size = int(len(data)*0.1)

After this, Deep Learning model was built required modules. Then, compiling the model with an optimizer (Adam), loss function (binary crossentropy), and metric (accuracy). Also, the model was trained on the training dataset for a specified number of epochs, using the test dataset for validation.

After the completion of all the epochs, the model was saved as **trained-model.sav** using pickle (i.e Pickle is a useful Python tool that allows you to save your ML models, to minimise lengthy re-training and allow you to share, commit, and re-load pre-trained machine learning models.)

This **trained-model.sav** model was imported in our app.py streamlit web application for better user interface. Various functions to check the model's performance was done for accuracy and other metrics.

Implementation of Deepfake video

In this system, we have trained our PyTorch deepfake detection model on equal number of real and fake videos in order to avoid bias in the model. The system architecture of the model is shown in the figure.

4.5 Block Diagram

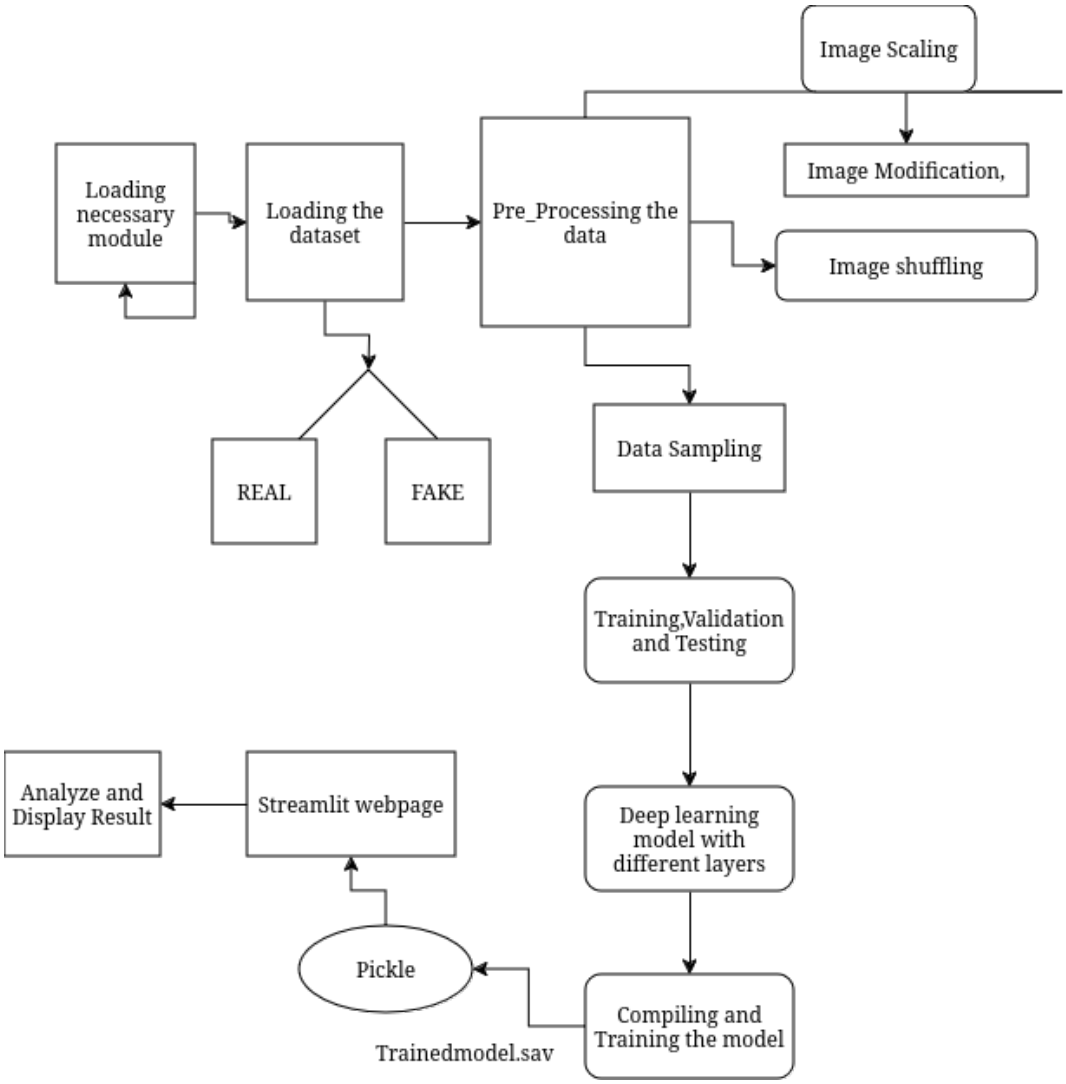

Figure 4.1: Block diagram for image detection

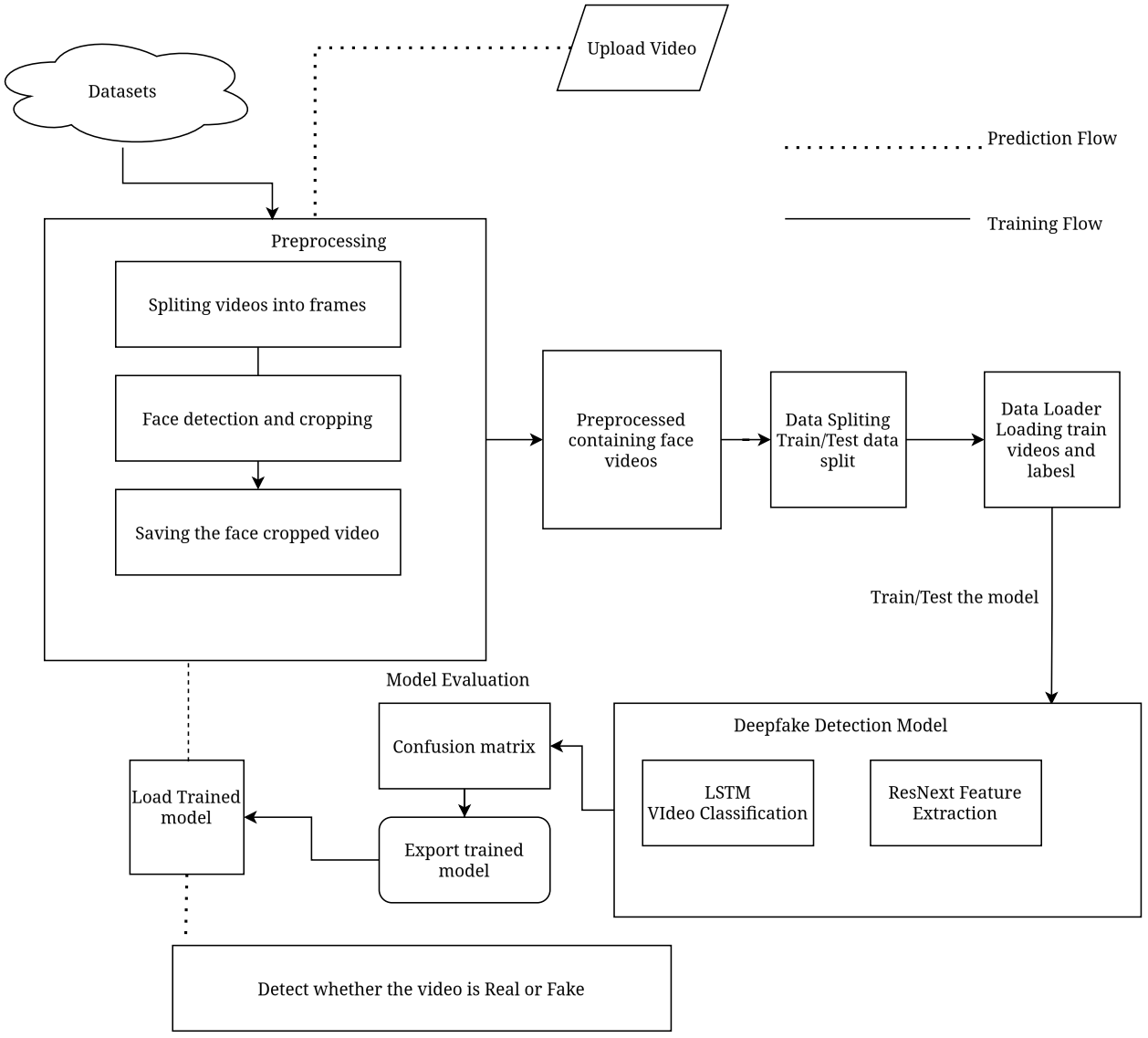

Figure 4.2: Block diagram for video detection

4.6 Flowchart

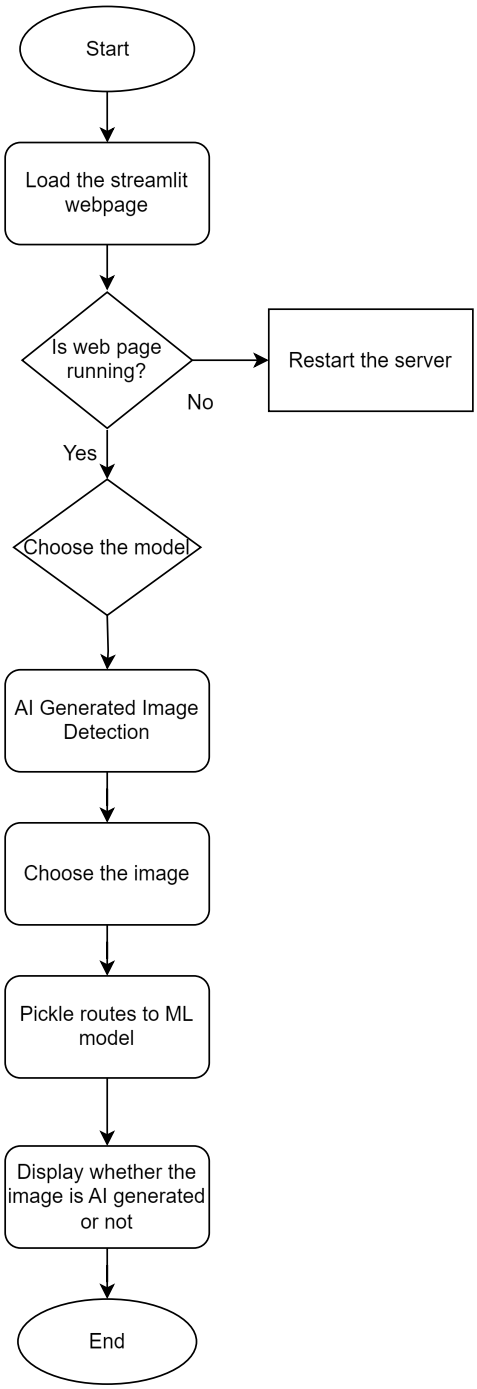

Figure 4.3: Flowchart for AI generated Image detection

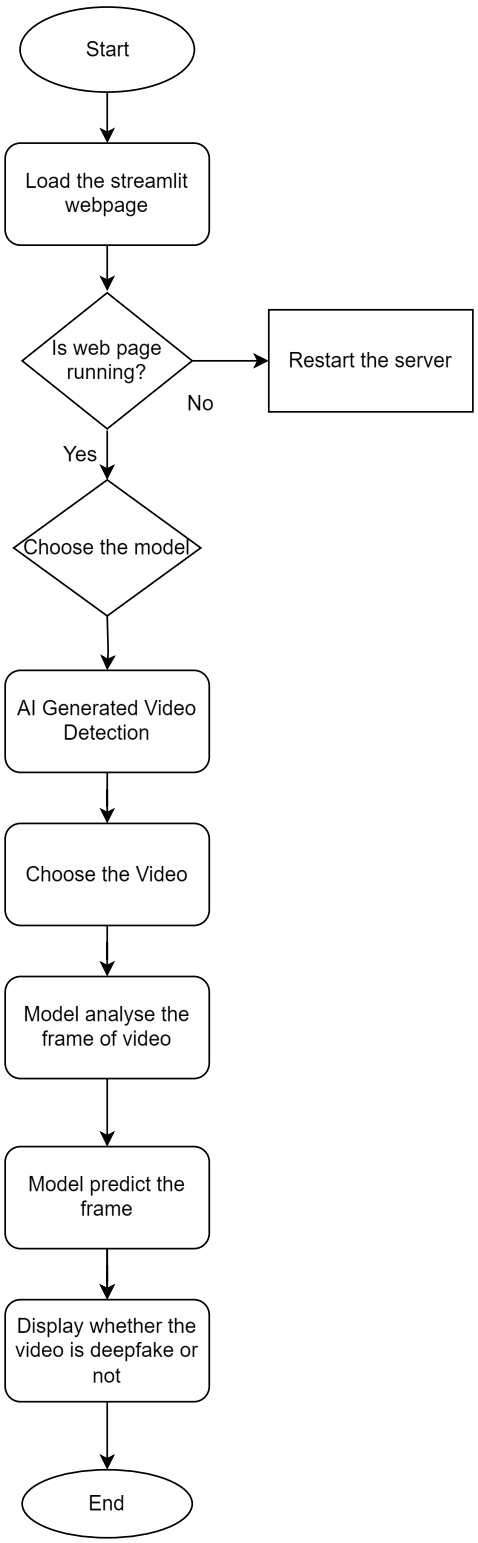

Figure 4.4: Flowchart for AI generated deepfake detection

4.7 Model Evaluaton

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 29, 29, 16)        784

 max_pooling2d (MaxPooling2  (None, 14, 14, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        8224

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 32)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 2, 2, 16)          8208

 max_pooling2d_2 (MaxPoolin  (None, 1, 1, 16)          0
 g2D)

 flatten (Flatten)           (None, 16)                0

 dense (Dense)               (None, 32)                544

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 17793 (69.50 KB)
Trainable params: 17793 (69.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
[29]:
```

Figure 4.5: Image processing model

model.add(Conv2D(16, (4, 4), 1, activation='relu', input-shape=(32, 32, 3))): This line adds a 2D convolutional layer to the model. It specifies 16 filters, each with a size of 4x4. The stride is set to 1, and the activation function used is 'relu' (Rectified Linear Unit). The input-shape parameter indicates that the input to this layer should have dimensions 32x32 with 3 channels (typical for color images in RGB format).

model.add(MaxPooling2D()): This line adds a max-pooling layer immediately after the convolutional layer. Max-pooling reduces the spatial dimensions of the representation while retaining the most important information. By default, it uses a pool size of 2x2 and a stride of 2.

model.add(Conv2D(32, (4, 4), 1, activation='relu')): This line adds another 2D convolutional layer with 32 filters of size 4x4. Again, the stride is set to 1, and relu is used as the activation function.

model.add(MaxPooling2D()): Another max-pooling layer is added after the second convolutional layer.

model.add(Conv2D(16, (4, 4), 1, activation='relu')): This line adds yet another 2D convolutional layer with 16 filters of size 4x4, again followed by a relu activation function.

model.add(MaxPooling2D()): Another max-pooling layer follows.

model.add(Flatten()): This line adds a Flatten layer, which flattens the 2D output of the previous layer into a 1D vector. This is necessary before passing the data to fully connected layers.

model.add(Dense(32, activation='relu')): This line adds a fully connected Dense layer with 32 neurons and relu activation function.

model.add(Dense(1, activation='sigmoid')): Finally, another Dense layer with a single neuron and a sigmoid activation function is added.

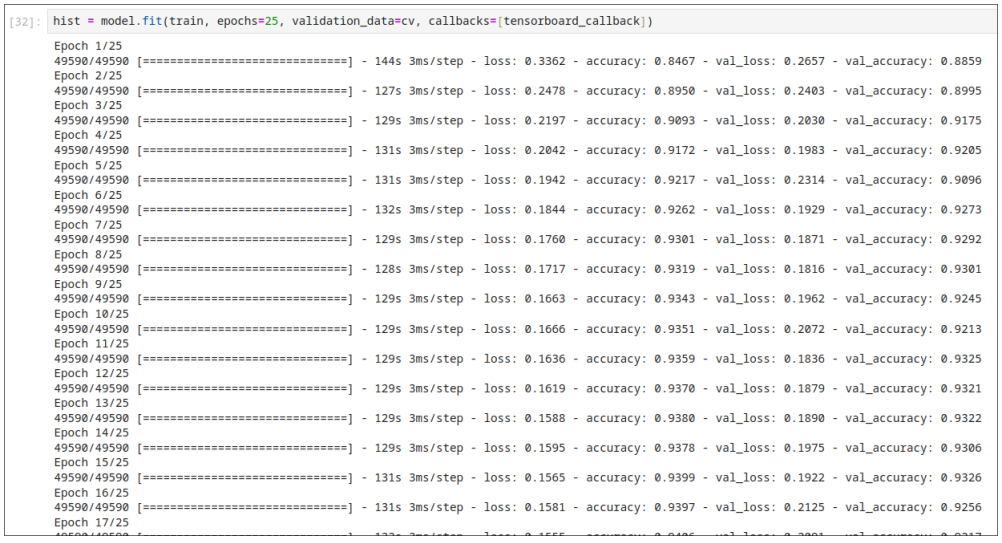

Figure 4.6: Training epoch

Our model was trained with only 25 epoch but accuracy was reached to 0.94.

The trained image classification model achieved the following performance metrics. Also, using streamlit web framework, following user interface was developed which seem to be pretty simple effective for detecting AI generated images and deepfake

```
[42]: print(f'Precision: {pre.result().numpy()}, Recall: {rec.result().numpy()}, Accuracy: {acc.result().numpy()}')

Precision: 0.9483881592750549, Recall: 0.8688600063323975, Accuracy: 0.9177250862121582
```

Figure 4.7: Precision and Accuracy value

videos.

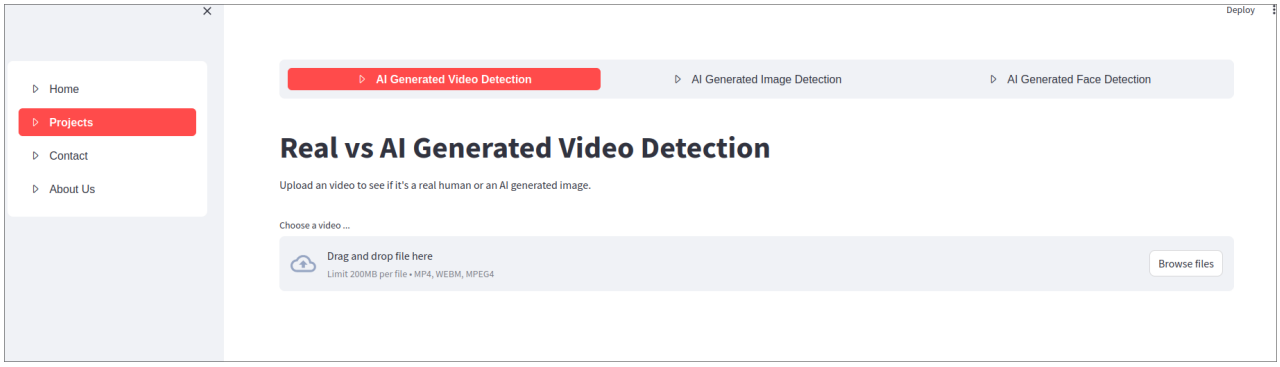

Figure 4.8: User Interface

CHAPTER 5

RESULTS AND DISCUSSION

Our model is meticulously being trained and fine-tuned to achieve high accuracy, minimizing false positives and false negatives for reliable results.Also with streamlit intuitive interfaces and clear instructions, users can easily navigate and utilize the detection tool without any technical expertise for their respective purposes.

Using model1 AI Generated Image Detection:

User can easily browsed their respective image and check their outcomes.This model detects those images which are animated leading to AI generated images that separate it from real images.

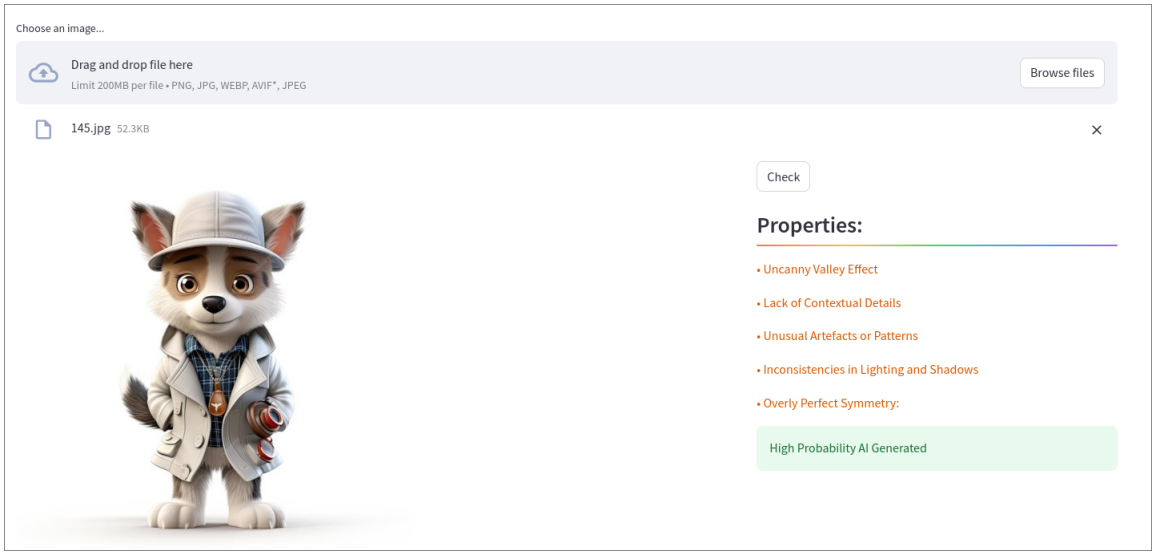

Figure 5.1: Image Detection

Using model2 AI Generated Face Detection:

User can easily browsed their respective image and check their outcomes. This model detects those AI generated images which are very synthetic which seems to be real in life.

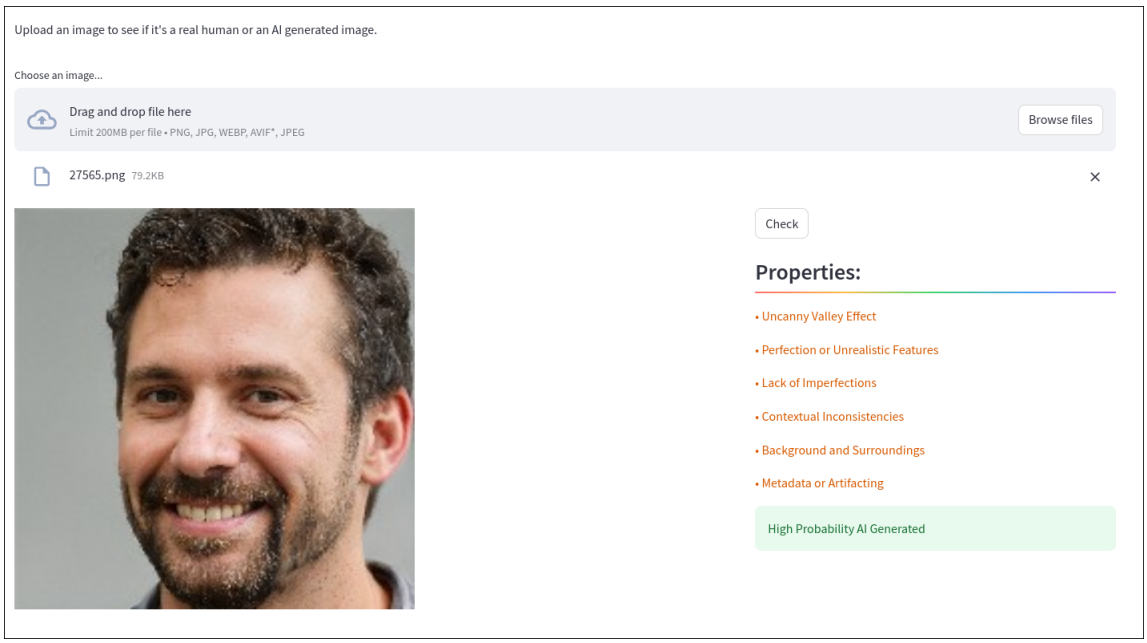

Figure 5.2: Face Detection

Model Evaluation:

After training model of Detection of AI generated images for respective epochs,our system produced following outcomes:

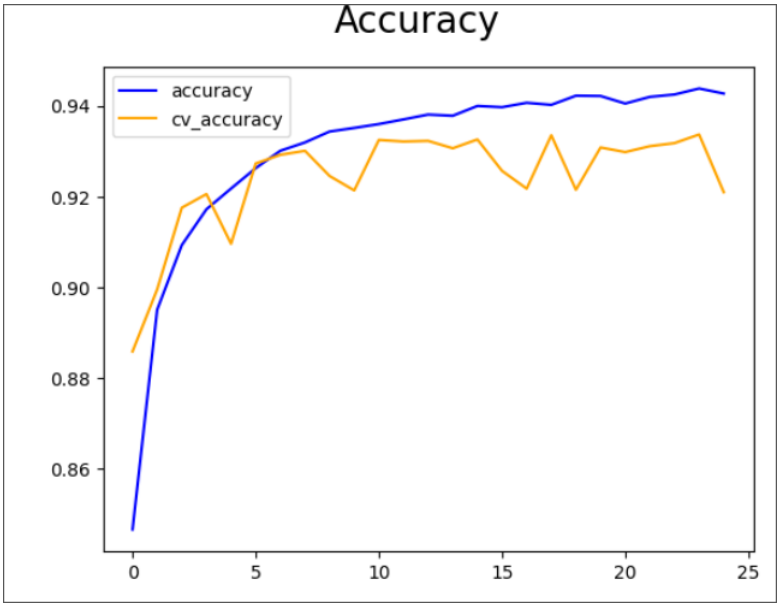

Figure 5.3: Accuracy of AI generated image detection model

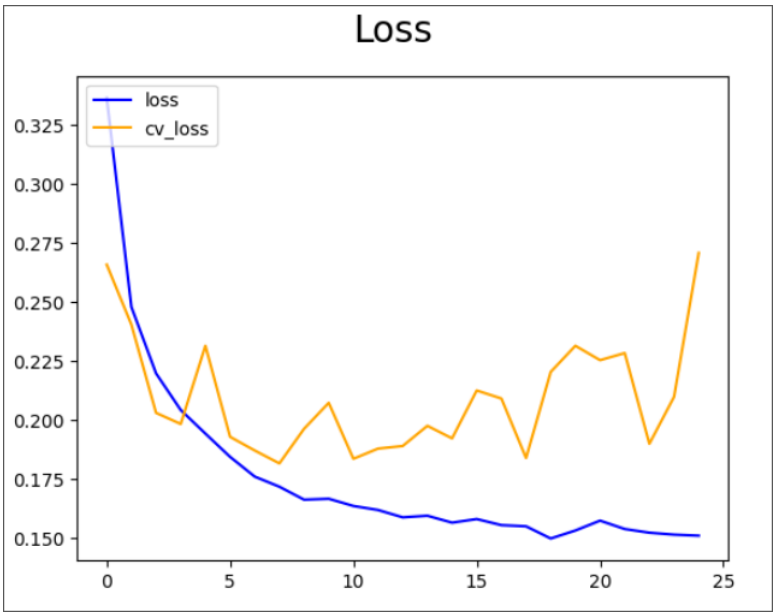

Figure 5.4: Loss of AI generated image detection model

Here,Cross validation accuracy is quite good which implifies percentage of correct classification is derived through the method of cross validation which is done on training dataset.

CHAPTER 6

CONCLUSIONS AND LIMITATIONS

6.1 Conclusions

Hence using CNN, a powerful system is built that can predict and detects AI generated image and videos with greater accuracy and precision.

We've established that detecting AI-generated content is a hard problem, and that success rates are so low that they aren't any better than guessing. The approach mentioned by our system doesn't generalize well. What we mean by this is that if we take 3 models into account, for example, DALL-E 3, Firefly, and Midjourney, then we must run the input image through all the 3 models and check different patterns and statistical reasoning on all of them to see if the image was generated by any one of them. This is because each model generates images slightly differently, and they all need to independently evaluate images to see if any of them may have generated it.

In fact, new models are being trained every day, and trying to keep up with this rapid progress seems hard at best. It's clear that we need a better approach if we hope to provide a reasonably high-quality result to check if any image is AI-generated.

6.2 Limitations

While working on our system, one first important observation was that visual imperfections on faces generated by AI will likely disappear soon. Newer GAN architectures already improved upon this aspect by producing faces with even more details and highly realistic. Thus, relying exclusively on these traces could be a losing strategy in the long term. Turning to generic deep learning based-solutions, the main technical issue is probably the inability to adapt to situations not seen in the training phase. Misalignment between training and test, compression, and resizing are all sources of serious impairments and, at the same time, highly realistic scenarios for real-world applications.

In the following, we analyzed some works that have shown the vulnerabilities of GAN detectors to different types of threats:

• **Adding adversarial perturbations**: It is well known, from the object recognition field, that suitable slight perturbations can induce mis-classification. Following this path, it has been investigated the robustness of GAN detectors to imperceptible noise both in a white-box and in a black-box scenario. It is also possible to design an effective strategy in a black-box threat model when the adversary does not have perfect knowledge of the classifier but is aware about the type of classifier.

• **Removing GAN fingerprints**: Instead of adding noise, one can take a different perspective and remove the specific fingerprints that are used to discriminate GAN images from real ones.

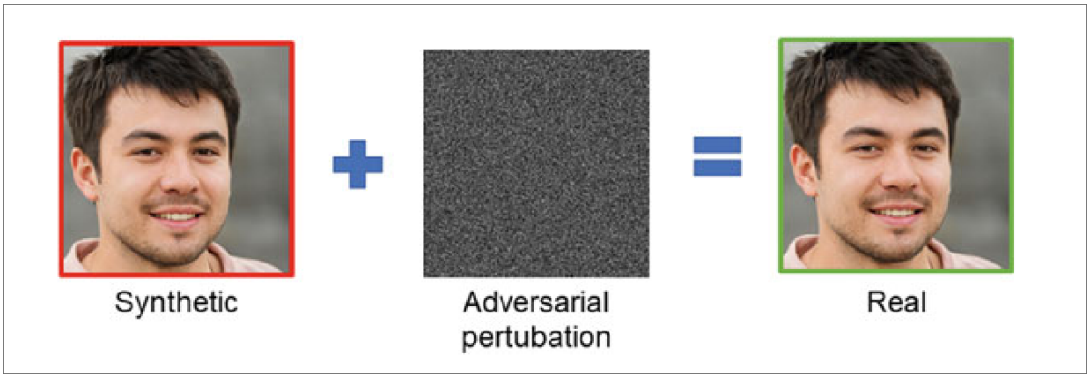

Figure 6.1: A small and imperceptible adversarial perturbation

A small and imperceptible adversarial perturbation can be added to the synthetic face image in order to fool the detector

• **Inserting camera fingerprints**: Another possible direction to attack GAN detectors is to insert the specific camera traces that characterize real images. In fact, real images are characterized by their own device and model fingerprints, as explained before. Such differences are important to carry out camera model identification from image content but can also be used to better highlight anomalies caused by image manipulations

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 2020, pp. 139–144, 2020.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, and et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[3] X. Guo, X. Liu, Z. Ren, S. Grosz, I. Masi, and X. Liu, "Hierarchical fine-grained image forgery detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3155–3165.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] H. Li, B. Li, S. Tan, and J. Huang, "Identification of deep network generated images using disparities in color components," *Signal Processing*, vol. 174, p. 107616, 2020.

[6] S. McCloskey and M. Albright, "Detecting gan-generated imagery using saturation cues," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 4584–4588.

[7] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

APPENDIX A

APPENDIX B