Chapter 1.

Introduction

1.1 Introduction

In today's world, with advancement in Artificial Intelligence, we are entering a new era of automatons and technologies that were only possible in science fiction. at this age, artificially intelligent soft-wares are intelligent to even generate an image which could give a tough competition to the best and best of artists and photographers.

The Artificial Intelligence and Machine Learning Models are now sophisticated and capable enough to do this , but as widespread these technologies have become , it also have spread to the hands of malicious and mischievous actors . These mischievous and malicious actors are using these technologies for unprecedented and threat provoking activities .

These actors have a variety of malicious activities and a multitude of consequences. These activities can range from using Generative AI to generate morphed images of individuals to generate images for scamming. Morphed images of famous and common individuals could be used against them for proving if the victimised individual is per-say involved in an illegal activity and could be used to extortion. Obscene images are been morphed to harass or defame individuals, causing emotional distress and reputations damage. These activities can also extend to the creation of videos, which can be used to fabricate scenarios or events that never occurred.

> As per experts the definition of the word **Deepfake** is "*a video, image, etc. in which a person's face, body, or voice has been digitally altered so that they appear to be someone else, typically used maliciously or to spread false information*" [1]

Moreover, these malicious activities are not limited to individuals but can also target organizations. For instance, counterfeit images or documents can be created to spread misinformation or conduct fraud. This can lead to financial losses for businesses and can undermine public trust in these organizations.

In the face of these threats, it's crucial for individuals and organizations to be aware of the potential misuse of Generative AI and take appropriate measures for protection. This can include digital literacy education, and advocating for ethical guidelines and regulations in the use of AI technologies.

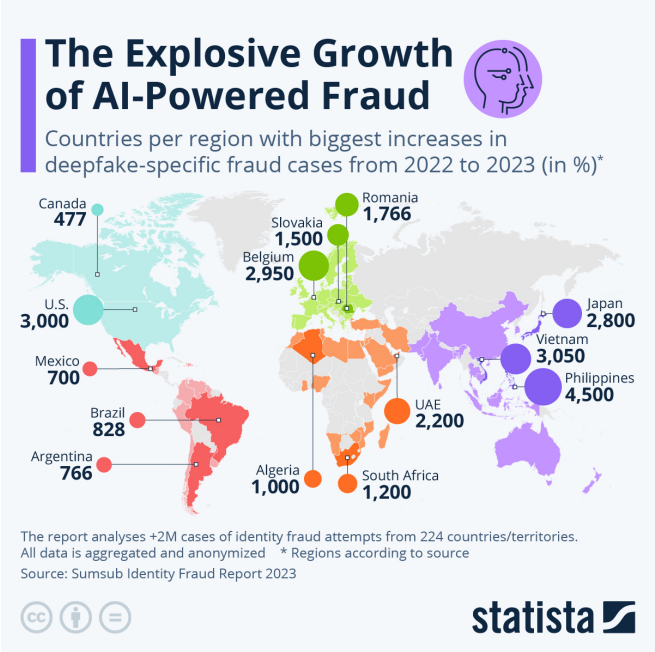


Figure 1.1: Growth in AI related scams [2]

The above image shows the growth in AI related frauds , scams and other malicious activites in recent years . As per theses study , this shows how deepfakes are used for frauds and other illegal activities .

In conclusion, while Generative AI holds immense potential for positive applications, its misuse poses significant challenges that society must address to ensure its benefits are realized without causing harm.

1.2 Motivation

The motivation of researching on and developing a technology capable of detecting AI-generated or Deepfake images from those that are real , comes from the curiosity of these technolgies and the will and wanting to tackle the threat foreshadowing .

As discussed in the above section about the threat these technologies are poses significant risks to individuals and organisation alike but also do to the Nation . The Honourable Prime Minister of India , Shri Narendra Modiji stated in his eye-opening speech at **virtual G20 summit** on

> ”We have to be careful with new technology. If these are used carefully, they can be very useful. However, if these are misused, it can create huge problems. You must be aware of deepfake videos made with the help of generative AI . ”These videos look very real and, therefore, we need to be very careful before believing the authenticity of a video or an image. India is emphasising a global framework for AI,” [3]

The PM emphasized the need for caution when dealing with new technology. While AI can be incredibly useful when used carefully, misuse can lead to substantial problems. This ignited a desire to work towards this goal for coubtering the arising problem by trying to develop a model or software to detect these deep fakes

1.3 Problem Statement and Objectives

1.3.1 Problem Statement

To develop a software with the help of artificial intelligence and deep learning such that it would be able to detect the deepfake image apart from the real image

1.3.2 Obejctives

- Dataset Collection and Preprocessing
  - Gather a diverse dataset containing both real images and deepfakes ,and ensuring its diverisity and authenticity
  - Preprocess the images by resizing, normalizing, and other image processcing techniques such that they are easy to work with .
- Model Architecture Selection
  - Choose an appropriate deep learning architecture like Convolutional Neural Networks (CNNs) .
- Feature Extraction and Representation
  - Extract relevant features from the images using the chosen model.
  - Use of intermediate layers or embeddings to capture discriminative information.
- Training and Validation
  - Split the dataset into training, validation, and test sets.
  - Train the model using real and deepfake images. and cross validate them .
- Loss Function and Optimization
  - Define an appropriate loss function
  - Optimize the model using approriate algorithms
- Fine-Tuning
  - Fine-tune hyperparameters (learning rate, batch size, etc.) to improve model convergence
  - Apply regularization techniques (dropout, weight decay) to prevent overfitting

- Evaluation
  - Evaluate the model on the test set using metrics such as accuracy, F1-score, and AUC-ROC .
  - Try to increase the metrics by various tuning techniques till a satisfactory position has been reached .
- Deployment
  - Deploy the model as a web application , thus accessible by all

1.4 Organisation of the report

Chapter 1 introduces to the need of a deepfake detection model and how it could be done . The Second Chapter covers the Literature Survey done for numerous papers and any existing systems trying to do the same . The design , architecture , software requirements etc are discussed in the Chapter 3 . Chapter 4 gives the final results and displays the outputs and explainations . The fifth chapter covers conclusion and discusses further about the future scope of this project .

Chapter 2.

Literature Survey

2.1 Survey of Existing Systems

2.1.1 Deepfake Detect

DeepFake Detect is a website dedicated to the detection of DeepFake content, providing users with tools and resources to identify manipulated media. The website leverages advanced algorithms and techniques to analyze multimedia content and detect signs of tampering or manipulation. With the proliferation of DeepFake technology and its potential for misuse, DeepFake Detect aims to empower users to distinguish between authentic and manipulated media, thereby mitigating the spread of misinformation and deception

Capabilities and Features

- DeepFake Detection Algorithms: DeepFake Detect employs state-of-the-art detection algorithms that utilize machine learning and computer vision techniques to identify anomalies indicative of DeepFake manipulation. These algorithms analyze various aspects of multimedia content, including facial features, audio characteristics, and visual artifacts, to determine the likelihood of manipulation.

- User-Friendly Interface: The website offers a user-friendly interface that allows users to easily upload multimedia files for analysis. Users can simply upload images or videos to the platform, and the detection algorithms will promptly process the content to assess its authenticity.

- Real-Time Analysis: DeepFake Detect provides real-time analysis of uploaded content, enabling users to receive instant feedback on the authenticity of the

media. This rapid analysis facilitates timely decision-making and enables users to quickly identify potentially deceptive content.

- Educational Resources: In addition to detection tools, DeepFake Detect offers educational resources and information about DeepFake technology. These resources aim to increase awareness and understanding of the capabilities and risks associated with DeepFake technology, empowering users to make informed judgments about the media they encounter online.

Advantages

- Combatting Misinformation: By providing reliable detection tools, the website helps to mitigate the spread of false information and preserve the integrity of digital content.

- Real-Time Analysis: DeepFake Detect offers real-time analysis of multimedia content, allowing users to receive immediate feedback on the authenticity of uploaded media. This rapid analysis enhances user convenience and enables timely decision-making in situations where the authenticity of media is in question.

Disadvantages

- False Positives and Negatives: Like all detection systems, DeepFake Detect may produce false positives or false negatives in its analysis. False positives occur when authentic media is mistakenly identified as manipulated, while false negatives occur when manipulated media is erroneously classified as authentic. These errors can undermine the reliability of the detection results and potentially lead to misinterpretation of media content.

- Dependency on User Uploads: DeepFake Detect relies on users to upload media for analysis, which may limit its effectiveness in detecting widespread DeepFake dissemination. Without comprehensive monitoring of online platforms

and social media channels, the website may fail to detect DeepFake content that has already been widely circulated.[18]

2.1.2 Deepfake Detection: A Systematic Literature Review by M. S. Rana , M. N. Nobi , B. Murali, A. H. Sung

Deepfake detection methods employ various techniques to identify manipulated videos. Here are some common approaches:

Deep Learning-Based Methods

- Convolutional Neural Networks (CNNs): These models learn features from image frames and classify them as real or fake.
- Recurrent Neural Networks (RNNs): They analyze temporal dependencies in video sequences.
- Autoencoders: These neural networks learn to reconstruct input data and can detect anomalies.
- Capsule Networks: They capture hierarchical relationships between features.
- 3D Convolutional Networks: These extend CNNs to process video volumes.

Machine Learning Techniques

- Feature-Based Methods: Extract handcrafted features (e.g., color histograms, optical flow) and use classifiers.
- Spatio-Temporal Features: Combine spatial and temporal information for better detection.
- Transfer Learning: Pretrained models (e.g., VGG, ResNet) fine-tuned for Deepfake detection.

Statistical Techniques

Analyze statistical properties (e.g., noise patterns, motion consistency) to differentiate real and fake videos.

Blockchain-Based Approaches

Use blockchain to verify video provenance and prevent tampering.

Advantages

- High Accuracy: Deep learning methods achieve impressive accuracy in detecting Deepfakes.
- Automation: Once trained, models can automatically identify manipulated content.
- Generalization: Some models generalize well across different types of Deepfakes.

Disadvantages

- Adversarial Attacks: Deepfake generators can adapt to detection methods, leading to evasion.
- Resource-Intensive: Training deep learning models requires substantial computational resources.
- False Positives/Negatives: No method is perfect; false alarms and missed detections occur.
- Lack of Generalization: Some methods struggle with unseen Deepfake variations.

In summary, Deepfake detection is an ongoing research area with promising results but challenges to overcome. Researchers continue to refine existing methods and explore new avenues to enhance detection accuracy and robustness[19]

2.1.3 DeepWare

Deepware is a comprehensive online platform dedicated to the detection and analysis of DeepFake content. Designed to address the growing threat of manipulated media, Deepware leverages advanced machine learning algorithms and computer vision techniques to identify signs of tampering or manipulation in multimedia content. With its user-friendly interface and powerful detection capabilities, Deepware empowers users to verify the authenticity of digital media and combat the spread of misinformation and deception.

Capabilities and Features

1. DeepFake Detection and Analysis: Deepware offers state-of-the-art DeepFake detection algorithms that analyze multimedia content to identify signs of manipulation. By examining various visual and auditory cues, such as facial features, voice characteristics, and visual artifacts, Deepware can determine the likelihood of media tampering.

2. Real-Time Scanning: The platform provides real-time scanning of uploaded media files, allowing users to receive instant feedback on the authenticity of the content. Whether it's images, videos, or audio recordings, Deepware quickly processes the media and provides users with detailed analysis results.

3. Comprehensive Reporting: Deepware generates comprehensive reports for analyzed media, detailing the detected anomalies and providing insights into the likelihood of manipulation. These reports include visualizations and explanations to help users understand the analysis results and make informed decisions about the media's authenticity. [20]

2.1.4 Eric Ji

The "NSF REU 2023 SD 21" dataset, analyzed by Eric Ji, is a big collection of data from the National Science Foundation's Research Experience for Undergradu-

ates (REU) program in 2023. This dataset focuses on a project called ”SD 21.”. This dataset is like a big pool of information that researchers can dive into to study specific questions, test ideas, and find trends and patterns in the data. It’s hosted on Kaggle, a website where people can share datasets and do analysis. Anyone can look at this dataset and do more research with it using tools to clean up the data, make it easier to understand, and find interesting stuff in it. Even though it doesn’t say much about how we can use it, datasets on Kaggle usually have rules about how people can use them. Overall, the ”NSF REU 2023 SD 21” dataset is super helpful for scientists, data experts, and analysts who want to dig into research data related to the REU program and the project led by Eric Ji, especially since it includes images of food, people, and animals. We used this dataset to help train our deepfake model, which means we taught our model to recognize real and fake images better. [5]

Chapter 3.

Proposed System

3.1 Introduction to proposed system

In the proposed system, the primary objective is to distinguish between real images and those generated by Deepfakes or AI. After an extensive literature survey, it was decided to employ Convolutional Neural Networks (CNNs) for model development and subsequent fine-tuning. The choice of CNNs was motivated by their proven effectiveness in image recognition tasks.

For the development of the Graphical User Interface (GUI), Flask was chosen due to its lightweight nature and ease of integration with Python. The front-end design will utilize HTML, CSS, and JavaScript to ensure a user-friendly experience.

This system aims to provide a robust solution to the growing issue of Deepfake and AI-generated imagery, leveraging the power of deep learning.

3.1.1 The Dataset

In the development of any machine learning or neural network model, a dataset is crucial. A variety of datasets were surveyed and collected from various sources. However, the dataset collected by Eric Ji for Fake Image Classification [5] was found to be the most suitable. This robust dataset consists of over 30,000 high-quality images, with an equal distribution of real and AI-generated images. The AI-generated images were produced using advanced Generative AI tools like DreamBooth[4], which employs stable diffusion techniques.

As we can see in the figure 3.1. , there is not any stark difference between the two images which could be distinguished by our naked eyes . Whereas the (b) image was

(a) A real image (b) A fake image

Figure 3.1: *An example of image from the dataset showing how they are hard to tell apart from a naked eye* [4]

generated using **DreamBooth** with the custom prompt as *Leonardo Di Caprio in a prison cell* [4]

Thus , the choice of this dataset was motivated by its size, quality, and balanced representation of both real and AI-generated images. This balance is critical for training a model that can accurately distinguish between the two categories.

3.1.2 The Algorithms

After looking into various algorithms for doing *"Image Classification"* , as Image Classification is required for classifying an image as fake / Ai-gen from real .Image Classification consists of following steps :-

There were various algorithms used for image classification , these are mentioned and explained in following :-

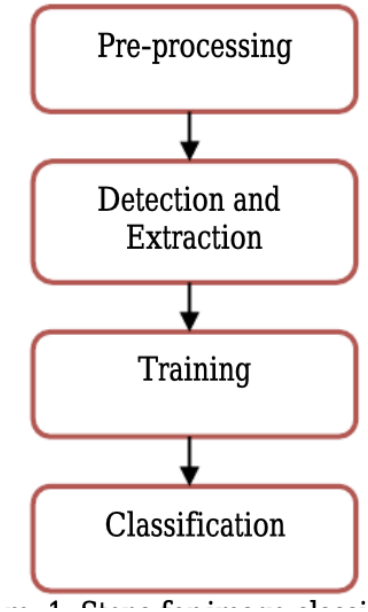


Figure 3.2: Steps of Iamge Classification [6]

1. **Artificial Neural Networks (ANN)**: ANNs are a type of artificial intelligence that mimics some functions of the human brain. They store experiential knowledge and consist of layers of neurons, each connected by weighted connections. ANNs are computational models inspired by biological neural networks. They are used in image processing, which involves image acquisition, pre-processing, feature extraction, and classification.

2. **Naive Bayes Classifier**: This classifier is based on probability representation and assigns the class with the greatest estimated subsequent probability to the feature vector extracted from the Region of Interest (ROI). It performs well even without orthogonal attributes and is robust to outliers.

3. **K-Nearest Neighbor (KNN)**: The KNN classifier partitions the space of instances into hyper spheres by conveying the majority class of the k-nearest instances according to a defined metric. However, it is sensitive to the curse of dimensionality and the choice of the metric.

4. **Multi-Layered Perceptron (MLP)**: MLPs are feed-forward neural networks inspired by the human nervous system. They consist of one input layer, one

output layer, and optional hidden layers. MLPs can generate models with arbitrary complexity by drawing infinite decision boundaries and are robust to noisy features.

5. **Kernel Support Vector Machines (SVM)**: Kernel SVMs map input feature vectors to a higher dimensional space using a kernel function. In the transformed space, a maximal separation hyperplane is built considering a two-class problem. SVMs allow training nonlinear classifiers in high-dimensional spaces using a small training set.

6. **Decision Tree (DT)**: DTs are based on a hierarchical rule-based method and use a non-parametric approach. They calculate class membership by repeatedly partitioning a dataset into uniform subsets. [6]

ANNs and CNNs can be effectively used for Deepfake Image Classification. While ANNs provide a foundation for recognizing complex patterns in data, CNNs offer a more specialized approach that is particularly suited to image analysis. By training these networks on a large dataset of real and deepfake images, they can learn to distinguish between the two with a high degree of accuracy. This makes them valuable tools in the ongoing effort to detect and combat deepfake technology. [7] Thus CNN were decided to develop the model of this project .

3.2 Hardware and Software Requirements :-

3.2.1 Hardware

- GPU with CUDA Support:
  - A CUDA-enabled NVIDIA GPU with compute capability 3.0 or higher is essential for training and inference with CNNs1.
  - GPUs significantly accelerate CNN computations due to their parallel processing capabilities.

- Memory and Storage:
  - Sufficient RAM is crucial for handling large datasets during training. At least 16 GB is recommended.
  - Storage: High-speed storage (SSD or NVMe) is necessary for quick data access during training and inference.
- Processing Power:
  - Multi-core CPUs: A powerful CPU with multiple cores is beneficial for data preprocessing and managing parallel tasks.
  - GPU Cores: More GPU cores allow faster training and inference. Consider GPUs with multiple cores (e.g., NVIDIA RTX series).

3.2.2 Software

- **Python in CNN:** Python is a popular language for Convolutional Neural Networks (CNNs) due to its simplicity and the availability of powerful libraries like TensorFlow and PyTorch. These libraries provide high-level APIs for designing and training neural networks, making Python an excellent choice for CNNs [8]
- **Jupyter vs Colab Environment:** Jupyter Notebook is an open-source web-based interactive computing environment that supports various programming languages. It's great for those who need full control over their environment and want to work locally. On the other hand, Google Colab is a cloud-based Jupyter Notebook environment that allows users to create, share, and collaborate on Jupyter Notebook documents. It's best suited for those who need to collaborate with others in real-time or who need access to a GPU. In this case it , the project is developed with a use of both the technologies . [9]
- **Python Libraires :-**
  - **TensorFlow:** TensorFlow is an open-source machine learning library developed by Google. It's used for high-level computations and is particu-

larly useful in machine learning and deep learning algorithms . [10]

  - **Keras**: Keras is an open-source high-level Neural Network library. It's user-friendly, extensible, and modular, facilitating faster experimentation with deep neural networks . [11]
  - **Pandas**: Pandas is a Python library used for data analysis and manipulation. It provides flexible high-level data structures and a variety of analysis tools. [12]
  - **NumPy**: NumPy is a Python library that provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these elements. [13]
  - **os**: The os library in Python provides a wealth of functions that allow us to interact with the underlying operating system, including accessing and manipulating the file system, environment variables, and even processes . [14]
  - **Flask**: Flask is a lightweight and flexible web framework for Python that follows the UNIX philosophy of doing one thing well. It's used to build web applications quickly and easily . [15]
  - **PIL**: PIL stands for Python Imaging Library. It adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.

- **Deployment**

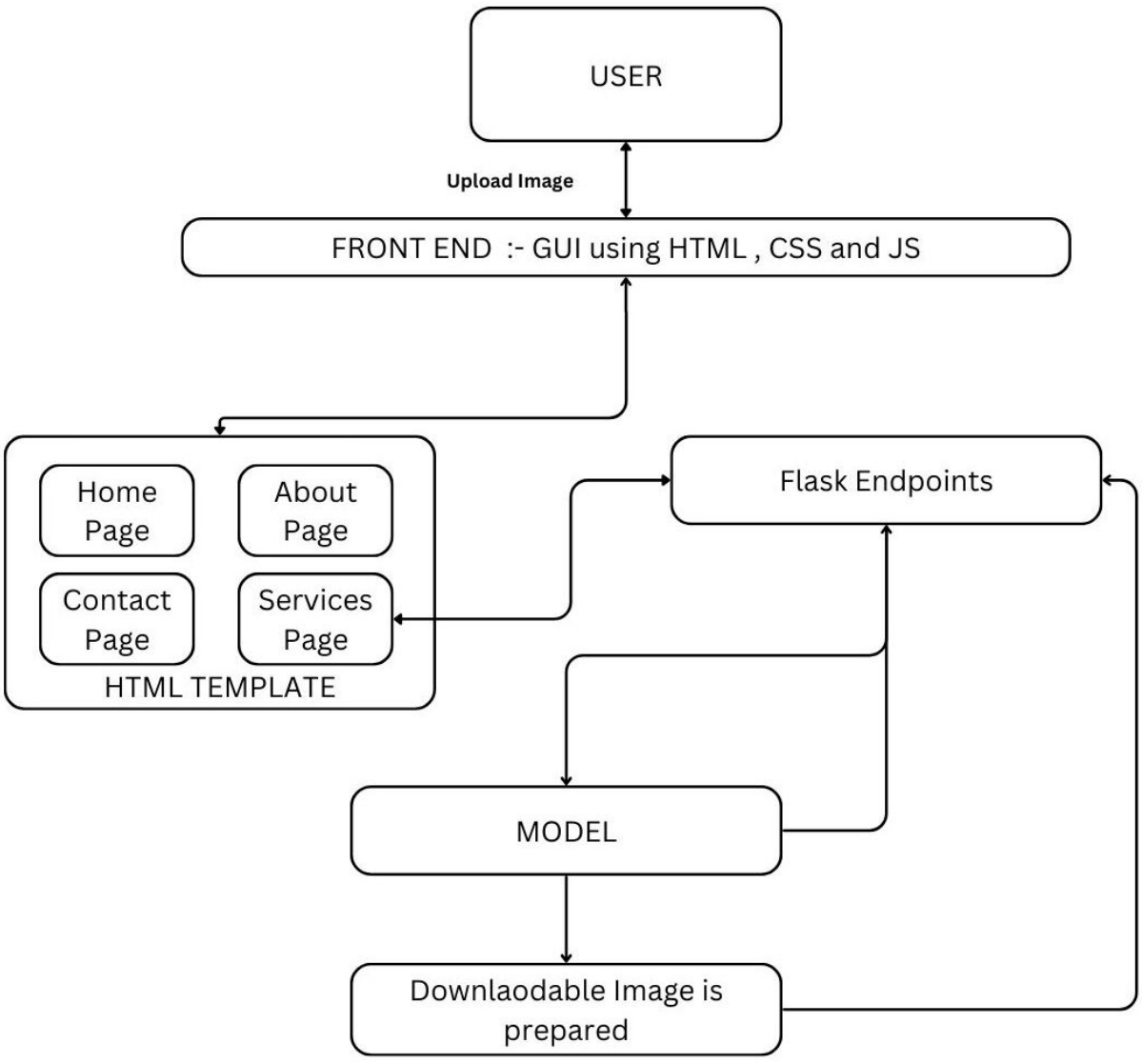
  - **Docker**: Docker is an open-source platform that automates the deployment, scaling, and management of applications. It uses containerization technology to package an application and its dependencies into a standardized unit for software development.
  - **Azure**: Azure is a cloud computing service created by Microsoft. It provides a range of cloud services, including those for computing, analytics, storage, and networking. Users can pick and choose from these services

to develop and scale new applications, or run existing applications in the public cloud.

- **Frontend**
  - **HTML**: HTML (HyperText Markup Language) is the standard markup language for creating web pages. It describes the structure of a web page and it can be used with CSS and JavaScript to create interactive websites.
  - **CSS**: CSS (Cascading Style Sheets) is a style sheet language used for describing the look and formatting of a document written in HTML. It provides powerful control over the visual aspect of an HTML document.
  - **JS**: JS stands for JavaScript. It's a high-level, interpreted programming language that conforms to the ECMAScript specification. JavaScript is widely used in web development to add interactivity to web pages.

3.3 Flowchart and Architecture

3.3.1 Overall Workflow



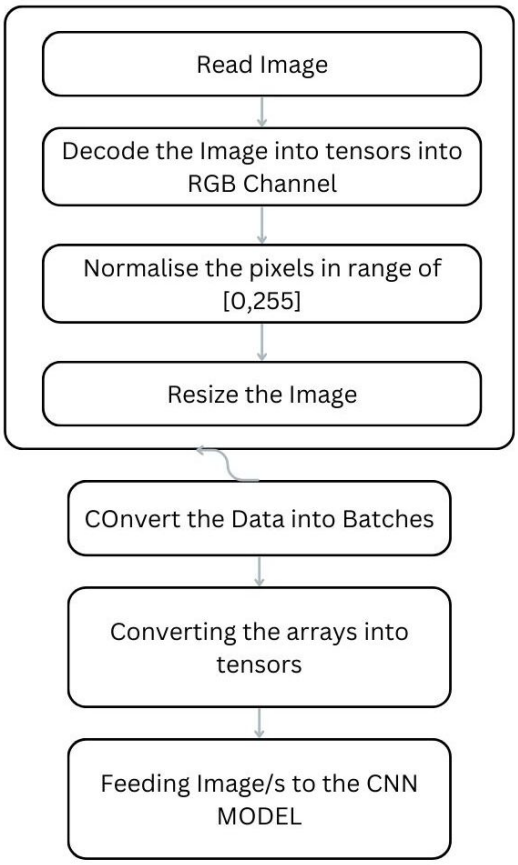
..

Figure 3.3: Workflow of the whole project

This flowchart outlines the entire project. When a user wants to determine if an image is a deepfake, they upload the image via a GUI developed using HTML, CSS, and JS. There are various web pages that provide different information, but the main functionality of this project is facilitated by the services page.

All these pages interact with the H5/Keras model through Flask endpoints provided by a Python library. With the help of these endpoints, the image is sent to the model where it is preprocessed and analyzed to classify it as real or a deepfake. The classification result is then sent back to the user through Flask endpoints. Additionally, an image is created in the back-end with a watermark indicating its classification. This image is made downloadable for the user.

3.3.2 Image Processing



..

Figure 3.4: The process of image preprocessing done for the model

The flowchart highlights essential steps like reading, decoding, normalizing, resizing, and batching. This systematic process is crucial in preparing and processing images to be fed into a CNN model. So first the iamge-file is read from the given path , then with the help of tensorflow library [10] the image-file's pixels are converted into tensors .

'a tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space. Tensors may map between different objects such as vectors, scalars, and even other tensors.' [10]

Then the image is Normalised to a range of 0 to 255.

$$\text{Normalized image} = \frac{\text{Original image} - \min(\text{Original image})}{\max(\text{Original image}) - \min(\text{Original image})} \times 255$$

and resized all the images to the same size. Convert the data into images and labels in batches thus lowering the load on the RAM. Thus after finally converting these batches to tensors, the tensors are fed to the CNN model.

3.4 Algorithm and Process Design

3.4.1 Convolutional Neural Network

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex [13, 14] and is designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, while the third, a fully connected layer, maps the extracted features to the final output, such as classification [16].

Convolution is an important analytical operation in mathematics. It is a mathematical operator that generates a third function from two functions f and g, representing the area of overlap between function f and function

g that has been flipped or translated.[7]

$$z(t)^{def} = f(t) * g(t) = \sum r = -\infty^{+\infty} f(r)g(t-r)$$

$$z(x,y) = f(x,y) * g(x,y) = \sum t \sum h f(t,h)g(x-t,y-h) \tag{3.3}$$

$$z(x,y) = f(x,y) * g(x,y) = \int \int f(t,h)g(x-t,y-h)dtdh \tag{3.4}$$

$$z(x,y) = f(x,y) * g(x,y) = \sum t = 0^{m} \sum h = 0^{n} f(t,h)g(x-t,y-h) \tag{3.5}$$

The above equations express that :-

- Let f represent the input image G.
- The size of the convolution kernel is m x n.
- The size of an image is M x M.
- The convolution kernel multiplies with each image region of n x n size of the image.
- This is equivalent to extracting the image region of n x n and expressing it as a column vector of n x n length.
- In a zero-zero padding operation with a step of 1, a total of (M-n+1)*(M-n+1) calculation results can be obtained.
- If the number of convolution kernels is K, the output of the original image obtained by the convolution operation is K*(M-n+1)*(M-n+1). The output is the number of convolution kernels times the image width after convolution times the image length after convolution.

Therefore , the above quotes and equations the basic definition and mathematical workings of CNNs . This project uses CNNs model . Thee Steps in modelling for binary image classification with CNNs

1. Becoming one with the data
2. Preparing data for modelling
3. Creating a CNN model (starting with a baseline)
4. Fitting a model (getting it to find patterns in our data)
5. Evaluating a model
6. Improving a model
7. Making a prediction with a trained model

Hyperparameter/Layer type	What does it do?	Typical values
Input image(s)	Target images you'd like to discover patterns in	Whatever you can take a photo (or video) of
Input layer	Takes in target images and preprocesses them for further layers	`input_shape = [batch_size, image_height, image_width, color_channels]`
Convolution layer	Extracts/learns the most important features from target images	Multiple, can create with `tf.keras.layers.ConvXD` (X can be multiple values)
Hidden activation	Adds non-linearity to learned features (non-straight lines)	Usually ReLU (`tf.keras.activations.relu`)
Pooling layer	Reduces the dimensionality of learned image features	Average (`tf.keras.layers.AvgPool2D`) or Max (`tf.keras.layers.MaxPool2D`)
Fully connected layer	Further refines learned features from convolution layers	`tf.keras.layers.Dense`
Output layer	Takes learned features and outputs them in shape of target labels	`output_shape = [number_of_classes]` (e.g. 3 for pizza, steak or sushi)
Output activation	Adds non-linearities to output layer	`tf.keras.activations.sigmoid` (binary classification) or `tf.keras.activations.softmax`

..

Figure 3.5: NN layers and its functionalities

The above figure , *figure 3.5* shows the functions of the various Neural Network Layers which are used in the model . The CNN layers used in this project are :-

1. Conv2D Layer: This is a 2D convolution layer. It creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The first layer has 10 filters, a kernel size of 3, uses the ReLU activation function, and expects input images of shape (224, 224, 3).

2. BatchNormalization Layer: This layer normalizes the activations of the previous layer at each batch, i.e., applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

3. Conv2D Layer: Another convolution layer with 10 filters and a kernel size of 3. It also uses the ReLU activation function.

4. BatchNormalization Layer: Normalizes the activations of the previous layer.

5. MaxPool2D Layer: This layer down-samples its input by taking the maximum value over the window defined by poolsize for each dimension along the features axis. The window is shifted by strides along each dimension

6. BatchNormalization Layer: Normalizes the activations of the previous layer.
7. Conv2D Layer: Another convolution layer with 10 filters and a kernel size of 3. It also uses the ReLU activation function.
8. BatchNormalization Layer: Normalizes the activations of the previous layer.
9. Conv2D Layer: Another convolution layer with 10 filters and a kernel size of 3. It also uses the ReLU activation function.
10. BatchNormalization Layer: Normalizes the activations of the previous layer.
11. Dense Layer: A densely connected (fully connected) layer with 100 neurons. It uses the ReLU activation function.
12. Conv2D Layer: Another convolution layer with 10 filters and a kernel size of 3. It also uses the ReLU activation function.
13. BatchNormalization Layer: Normalizes the activations of the previous layer.
14. MaxPool2D Layer: Another max pooling layer.
15. Flatten Layer: This layer flattens the input. It does not affect the batch size.
16. Dropout Layer: This layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent over-fitting. The rate is set to 0.5.
17. Dense Layer: The final layer is a dense layer with a single neuron and a sigmoid activation function. This is typically used for binary classification problems.

The model is compiled with these 17 layers using loss function as binary cross-entropy .

> Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.[17]

and the model is fitted for 25 epochs which resulted in 97% accuracy score , then this model is tested against a a test data of 10,000 images containing 5000 of real and deepfakes . [5] . After this the model is saved in H1 and keras format , thus to be loaded and used in the flask application for a further GUI .

3.4.2 Flask application

- **Application Configuration:** In the process of setting up a Flask application, an instance of Flask is first created and assigned to a variable named app. To handle image uploads, a folder named static/uploads is created, and its path is stored in a variable named **UPLOAD FOLDER**. The application also utilizes a pre-trained deep learning model, the path to which is stored in a variable named **MODEL PATH**. For security purposes, a secret key is set for the Flask application. Lastly, the pre-trained deep learning model is loaded into the application using the **tf.keras.models.load mode**l function.

- **Route Definitions:** These establishes multiple routes for a web application by utilizing the `@app.route` decorator. These routes serve as a bridge between URLs and specific functions within the application. The root of the application, denoted by "/", is managed by the `index` function, which presumably renders the homepage template. The routes "/about", "/services", "/portfolio", and "/contact" are likely associated with their respective HTML templates, representing different sections of the website. Additionally, there's a "/upload" route dedicated to handling file uploads. When a file is submitted via a POST request to this route, it triggers the execution of the `upload_file` function.

– **Upload Functionality :** The function in question initiates by creating an upload folder if it doesn't already exist. It then retrieves the file uploaded by the user from the request object and secures the filename to mitigate any potential security risks. The function checks whether the filename is empty. If the filename isn't empty, the function proceeds to save the uploaded file in two distinct locations: the upload folder (`UPLOAD_FOLDER`) and a folder named

`static/images`. Upon successful upload, a flash message is displayed. The function then calls the `process_image` function to process the uploaded image and obtain a prediction. Subsequently, the `apply_watermark` function is invoked to add a watermark to the image based on the prediction. The path of the watermarked image is modified to be relative to the static folder, and this path is assigned to the `wm_image_path` variable. Finally, the function renders the `uploaded_file.html` template, passing the filename, prediction, and watermarked image path as variables. If the filename is found to be empty, the function displays a flash message indicating an invalid file format and renders the `index.html` template.

– **Image Processing** : The function begins by reading the uploaded image from the file path. It then preprocesses the image to prepare it for the deep learning model, which may involve resizing the image, converting it to a specific format, or normalizing the pixel values. Once the image is preprocessed, the function uses the loaded deep learning model to make predictions on the image. These predictions are then converted into a human-readable format, such as "Real Image" or "AI-generated Image". Finally, the function returns the prediction label.

– **Watermark Function** The function initiates by opening the uploaded image and obtaining its original dimensions. It then calculates a new height that preserves the image's aspect ratio when resized to a new width, such as 500 pixels. The image is subsequently resized to these new dimensions. Next, a drawing object is created to facilitate the addition of text to the image. The position and size of the rectangle, which will house the watermark text, are defined. A semi-transparent black rectangle is then drawn on the image at the specified location. The font and size for the watermark text are defined, and the watermark text is added to the image. The watermarked image is saved with a filename that includes "watermarked" before the original filename. The function concludes by returning the path to the watermarked image. In terms of download functionality, this function handles the downloading of watermarked images and can be

accessed via a GET or POST request. For a GET request, it retrieves the filename from the request arguments. For a POST request, it retrieves the file type and aspect ratio from the form data, allowing users to specify the desired format (e.g., PNG) and whether to maintain the original aspect ratio when downloading. The function then constructs the complete file path to the watermarked image.

Thus, here the proposed system for deepfake detection, which leverages the power of Convolutional Neural Networks (CNN) and the Flask application, has been thoroughly discussed. The CNN's ability to extract and learn intricate patterns from input data makes it a robust choice for detecting deepfakes. Meanwhile, the Flask application provides a user-friendly interface for interacting with the detection system .

Chapter 4.

Implementation and Results

4.1 System Implementation

The proposed system for deepfake detection leverages the power of Convolutional Neural Networks (CNN) and a Flask application. The CNN's ability to extract and learn intricate patterns from input data makes it a robust choice for detecting deepfakes. Meanwhile, the Flask application provides a user-friendly interface for interacting with the detection system.

The system begins with a process that highlights essential steps like reading, decoding, normalizing, resizing, and batching. This systematic process is crucial in preparing and processing images to be fed into a CNN model. The image file is first read from the given path, then with the help of the TensorFlow library, the image file's pixels are converted into tensors. A tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space. Tensors may map between different objects such as vectors, scalars, and even other tensors. The image is then normalized to a range of 0 to 255 and resized. All the images are converted into the same size, and the labels are converted into batches. This process lowers the load on the RAM. After finally converting these batches to tensors, the tensors are fed to the CNN model.

The CNN model used in this project is composed of three types of layers: convolution, pooling, and fully connected layers. The first two layers, convolution and pooling, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into the final output, such as classification. The convolution operation generates a third function from two functions, representing the area of overlap between the two functions that have been flipped or translated.

The model is compiled with these 17 layers using the loss function as binary cross-entropy and is fitted for 25 epochs, which resulted in a 97% accuracy score. Then, this model is tested against a test data of 10,000 images containing 5000 of real and deepfakes. After this, the model is saved in H1 and Keras format, thus to be loaded and used in the Flask application for a further GUI.

The Flask application is set up by creating an instance of Flask and assigning it to a variable named app. To handle image uploads, a folder named static/uploads is created, and its path is stored in a variable named UPLOAD FOLDER. The application also utilizes a pre-trained deep learning model, the path to which is stored in a variable named MODEL PATH. For security purposes, a secret key is set for the Flask application. Lastly, the pre-trained deep learning model is loaded into the application using the tf.keras.models.load model function.

The application establishes multiple routes for a web application by utilizing the @app.route decorator. These routes serve as a bridge between URLs and specific functions within the application. The root of the application, denoted by "/", is managed by the index function, which presumably renders the homepage template. The routes "/about", "/services", "/portfolio", and "/contact" are likely associated with their respective HTML templates, representing different sections of the website. Additionally, there's a "/upload" route dedicated to handling file uploads. When a file is submitted via a POST request to this route, it triggers the execution of the upload file function.

Thus , the proposed system for deepfake detection, which leverages the power of Convolutional Neural Networks (CNN) and the Flask application, has been thoroughly discussed. The CNN's ability to extract and learn intricate patterns from input data makes it a robust choice for detecting deepfakes. Meanwhile, the Flask application provides a user-friendly interface for interacting with the detection system. This combination of advanced machine learning techniques and user-friendly web application design makes the system a powerful tool for deepfake detection.

4.2 Results

4.2.1 Home Page

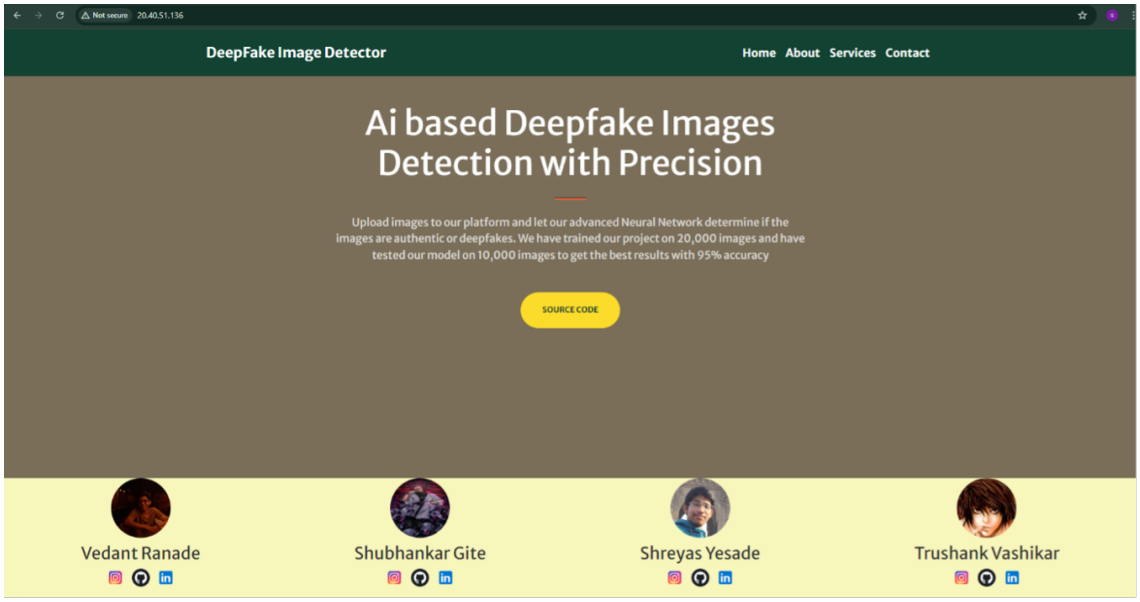


Figure 4.1: Home Page

The figure 4.1 is a screenshot of the Home Page of website offering an AI-based Deepfake Image Detection platform. This service uses CNN to determine if uploaded images are authentic or manipulated, claiming 97% accuracy. The website also offers access to the source code.

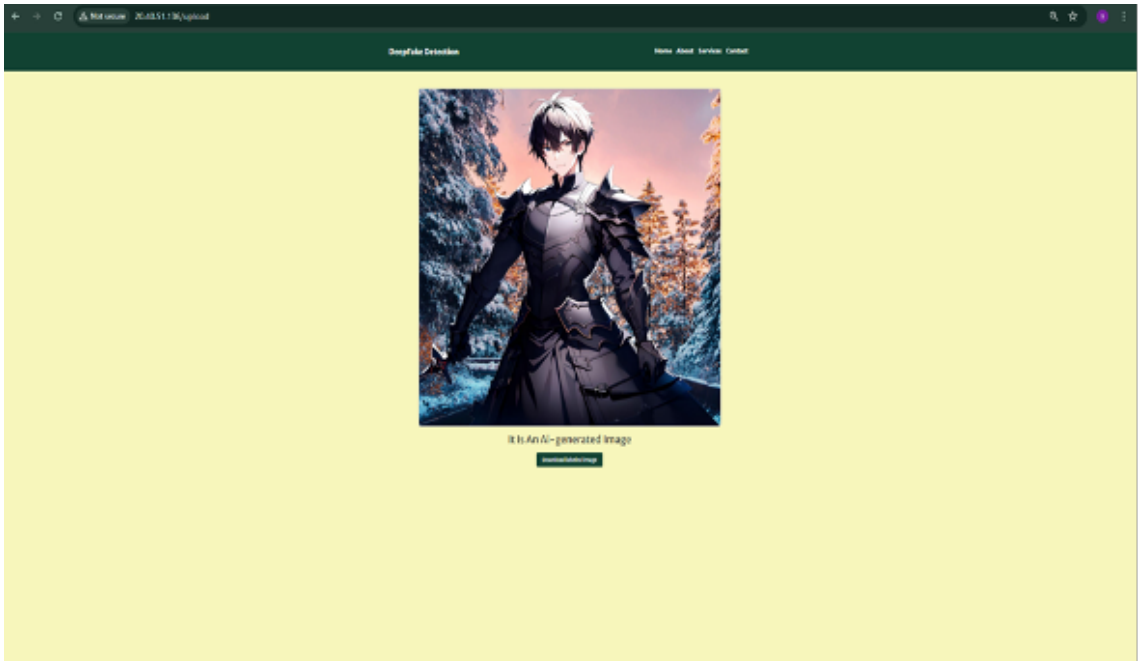


Figure 4.2: Result of Ai generated image Detection

The above figure 4.2 shows us the result of the model detecting the inputted image to be AI generated / deepfake .

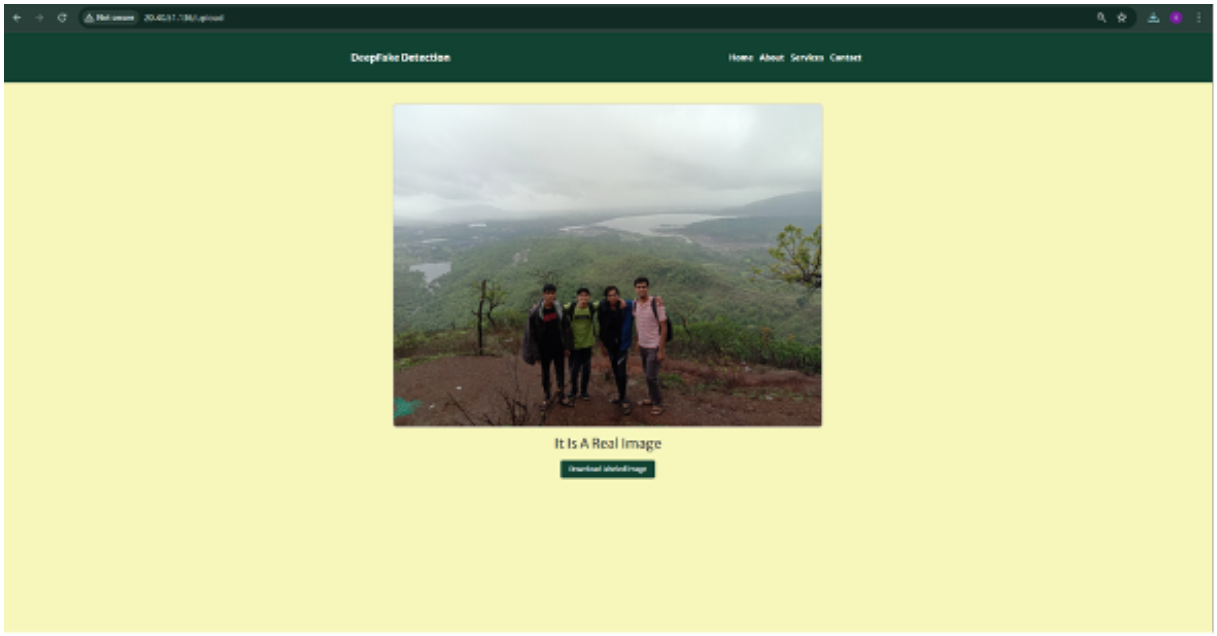


Figure 4.3: Result of Real image Detection

The above figure 4.3 shows us the result of the model detecting the inputted image to be as a real image asd also presensts us with the option to dowload the file as it is tagged with the type of image it is i.e a deepfake or real one .

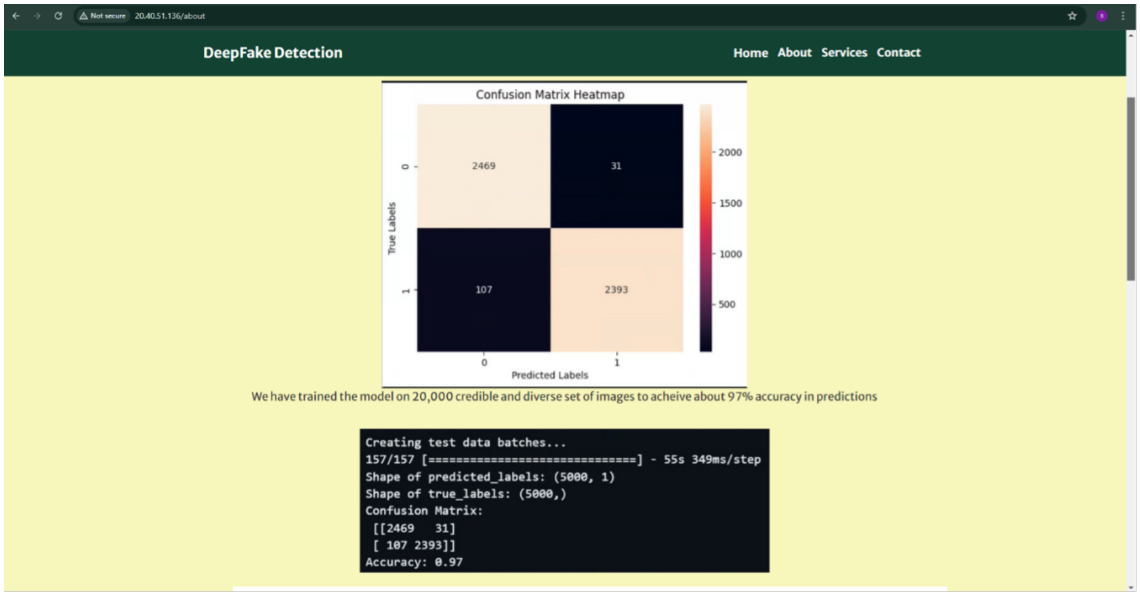


Figure 4.4: Information Page

Figure 4.4 is the Screenshot of the About page of the website , here the whole methodology and the partial results about the model are explained to prove the reliability of the model .

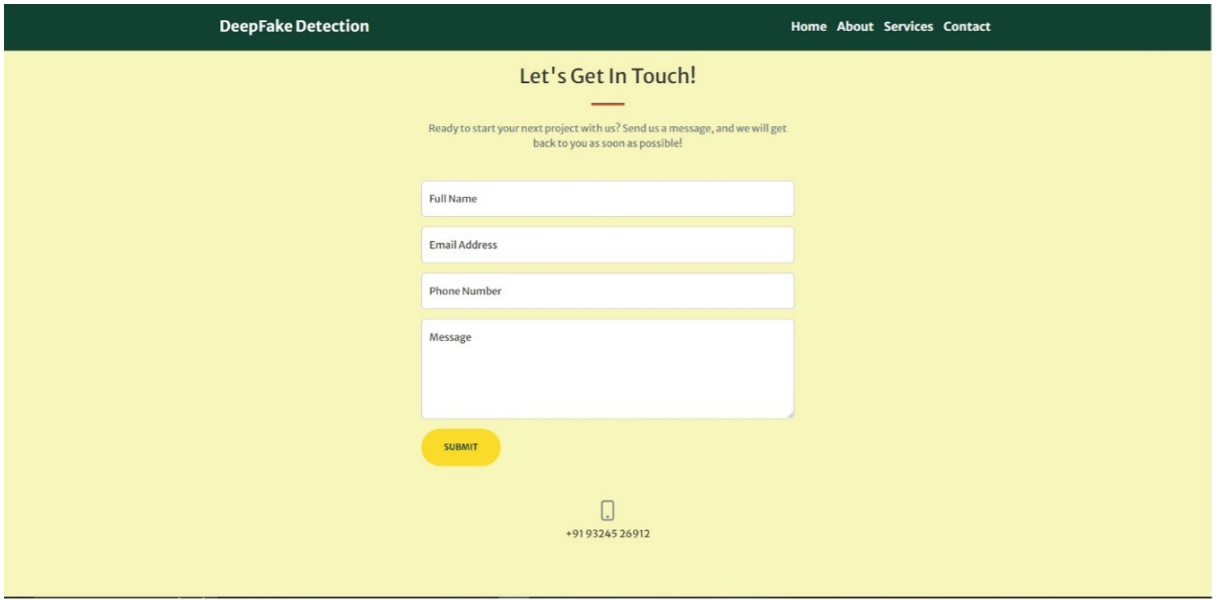


Figure 4.5: The Contact Page

The above Screenshot Figure 4.5 shows us the contact page , this is a basic form asking for the name and email of the user to fire an email to the team for enquiry , suggestion or complaint .

Chapter 5.

Conclusion and Future Work

Conclusion Our project focused on developing a deepfake detection system capable of discerning between AI-generated and real images. By leveraging the comprehensive ”NSF REU 2023 SD 21” dataset, meticulously analyzed by Eric Ji, we trained our model to achieve an impressive accuracy of 97%. Utilizing this dataset, which contains diverse images including food, people, and animals, enabled our model to learn and distinguish between various types of content effectively. Furthermore, we operationalized our model by creating a user-friendly website using Flask and hosting it on the Azure platform. This deployment facilitates easy access and utilization of our deepfake detection system, empowering users to verify the authenticity of images with confidence.

Future Scope Looking towards the future, there are several exciting avenues for further exploration and enhancement of our project. Firstly, we can expand our dataset by incorporating additional diverse and representative images, thereby enhancing the robustness and generalization capabilities of our model. Moreover, integrating real-time detection capabilities into our system would enable users to identify and mitigate the spread of deepfake content more effectively. Additionally, exploring advanced machine learning techniques and algorithms could further improve the accuracy and efficiency of our deepfake detection model. Furthermore, collaborating with experts in related fields such as cybersecurity and digital forensics could provide valuable insights and perspectives for refining and optimizing our detection system. Overall, our project lays a solid foundation for ongoing research and development efforts in the critical area of deepfake detection, with the potential to make significant contributions towards combating the proliferation of manipulated media in the digital landscape.

Bibliography

[1] bab.la :- the online dictionary

[2] Statisca - How Dangerous are Deepfakes and Other AI-Powered Fraud?

[3] India Today - 'Be careful with new technology': PM Modi's fresh warning on deepfake videos

[4] Ji, E., Dong, B., Samanthula, B., Zhou, N. (2024). *2D-FACT: Dual-Domain Fake Image Detection Against Text-to-Image Generative Models*. Champaign, IL, USA: University of Illinois Urbana-Champaign, Montclair State University, Montclair, NJ, USA. Emails: `ericji3@illinois.edu`, `dongb@samanthulab@zhoun@montclair.edu`

[5] Eric Ji. (2023). 2D-FACT - Fake Image Classification Dataset [Data set]. Kaggle

[6] R., Ponnusamy and Sathiamoorthy, S. and Kaliyamoorthi, Manikandan , *A Review of Image Classification Approaches and Techniques*, 2020.

[7] Mingyuan Xin1 and Yong Wang (2020) , *Research on image classification model based on deep convolution neural network*

[8] Building a Convolutional Neural Network (CNN) in Keras Jaz Allibhai

[9] GeekforGeeks :- Goofle Colab vs. Jupyter

[10] Tensorflow - Docs

[11] JavaTpoint :- Keras

[12] Pandas:- Docs

[13] numpy :- docs

[14] A Deep Dive into the os Library in Python: Functions, Features, and Best Practices Saeed Mohajeryami, PhD

[15] Flask - Docs

[16] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018)

[17] Binary Cross Entropy/Log Loss for Binary Classification

[18] Deepfake Deetect :- Website

[19] M. S. Rana, M. N. Nobi, B. Murali and A. H. Sung, "Deepfake Detection: A Systematic Literature Review," in IEEE Access, vol. 10, pp. 25494-25513, 2022,

[20] Deepware :- Website