Problem 1

**Title - Challenging Fake Image Detection using GAN Models**

Detecting fake or manipulated images in today's digital age has become increasingly challenging due to the advancements in Generative Adversarial Networks (GANs). These AI-powered tools have made it easier than ever to create convincing fake images that can deceive both human observers and traditional image analysis techniques. The problem at hand is to develop a robust and effective fake image detection system that can differentiate between genuine and manipulated images generated by GAN models.

Objective:

Develop a fake image detection system that can distinguish between real and AI generated images.

**Solution:**

Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating synthetic data by learning the underlying distribution of training data. These networks consist of two components, a generator and a discriminator, that engage in a competitive learning process to create increasingly realistic synthetic data.

In a previous internship, I had the opportunity to work with GANs and explored how to create fake or synthetic data using these models. The experience gained during that internship will be invaluable in developing the fake image detection system required for this project.

For the codebase and project documentation, you can refer to the following GitHub repository: <https://github.com/BassammaKorvi/SRIP_IIITB>

**Prerequisites:**

**Python:** Ensure you have Python installed on your system. You can download Python from the official website: <https://www.python.org/downloads/>

**Required Libraries:** You need to have the following Python libraries installed:

**numpy**: For numerical operations and array handling.

**opencv-python (OpenCV)**: For image loading and manipulation.

**scikit-learn**: For machine learning utilities.

**tensorflow and keras**: For building and training neural networks.

You can install these libraries using pip:

pip install numpy opencv-python scikit-learn tensorflow

**Dataset: Organise your dataset in the following structure:**

dataset/

├── test/

│   ├── real/

│   ├── fake/

│

└── train/

    ├── real/

    ├── fake/

Ensure that the "real" folders contain real images, and the "fake" folders contain fake or GAN-generated images.

**Usage Guidelines for the Fake Image Detection System**

This guide provides step-by-step instructions for running the fake image detection system using the provided Google Colab notebook. Before proceeding, please make sure you have access to the following resources:

**Google Chrome Browser:** To ensure compatibility and a seamless experience, it's recommended to use Google Chrome as your web browser.

**Google Colab Notebook:** The main tool for running the fake image detection system is theGoogle Colab notebook. You can access the notebook using the following link: [**https://colab.research.google.com/drive/1ii5BVZULRcVZf9ElHyVVRlUdpsjz0Ffk?usp=sharing**](https://colab.research.google.com/drive/1ii5BVZULRcVZf9ElHyVVRlUdpsjz0Ffk?usp=sharing)

**Dataset**: The dataset required for testing the system can be downloaded from Kaggle using this link: CIFake Real and AI-generated Synthetic Images Dataset [**https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images**](https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images)Please note that only a portion of the dataset has been used in this notebook due to its large size.

**Now, let's get started with the usage instructions:**

**Step 1:** Open the Google Colab Notebook Click on the provided link to access the Google Colab notebook. You will be redirected to a Google Colab environment where you can execute the code.

**Step 2:** Authenticate and Mount Google Drive (if necessary) Depending on the notebook's requirements, you might need to authenticate your Google account and mount Google Drive. Follow the on-screen prompts if this step is necessary.

**Step 3:** Upload the Dataset Download the dataset from the provided Kaggle link. If you are instructed to upload the dataset to Google Colab, use the Colab's file upload feature to do so. This typically involves running a code cell to select and upload the dataset from your local machine to the Colab environment.

**Step 4:** Run the Notebook Cells The notebook is divided into cells, each containing a portion of the code and explanations. Start from the beginning and execute each cell sequentially. You can run a cell by clicking the "Run" button or using the Shift + Enter keyboard shortcut. Follow any additional instructions provided within the notebook for configuring parameters, paths, or settings.

**Step 5:** Monitor Progress As you run the cells, the notebook will display output, including progress updates and results. Pay attention to any error messages or prompts that may require your input. Step 6: Review Results

After running all the necessary cells, you should be able to review the results and outputs generated by the fake image detection system. Analyse the system's performance and any visualisations or metrics provided in the notebook.

**Step 7:** Save and Share Results (if needed) If you want to save the results or share them with others, you can export the notebook or save specific sections as needed.

**Step 8:** Terminate the Session (if needed) Once you have completed your analysis, you can terminate the Colab session by closing the browser window or selecting "File" -> "Close" in the Colab environment.

These usage guidelines are designed to help you navigate and utilize the fake image detection system within the Google Colab environment. Make sure to follow the instructions carefully and refer to the notebook's documentation for any specific details related to the project.

**Steps:**

**1. Data Collection and Preprocessing:**

In this step, the code loads and preprocesses images from the provided dataset folders. It reads all .jpg files in the specified folders.

Images are loaded, converted to RGB colour format, and resize to 128x128 pixels.

Real and fake images are separated and labelled accordingly.

**2. Data Splitting:**

The code splits the data into training and testing sets.

Training data consists of both real and fake images, while testing data is loaded from the respective "test" folders.

Labels are assigned: 1 for real images and 0 for fake images.

**3. Building and Training a CNN Model:**

A Convolutional Neural Network (CNN) model is constructed using Keras Sequential API.

The model architecture includes three convolutional layers with max-pooling, followed by fully connected layers.

A dropout layer is added for regularisation to prevent overfitting.

The model is compiled with the Adam optimizer and binary cross-entropy loss for binary classification.

It is trained for 10 epochs with a batch size of 32 on the training data.

**4. Evaluation:**

After training, the model is used to predict labels for the test dataset.

The predicted labels are compared to the true labels to calculate accuracy.

Classification metrics, including precision, recall, and F1-score, are also computed and displayed.

The code will load, preprocess, train, and evaluate a CNN model for fake image detection, providing accuracy and classification metrics as output.