National College of Ireland

Configuration Manual

MSc Research Project
M.Sc. AI for Business

Ajul Gopu Elayadath
Student ID: x23295872

School of Computing
National College of Ireland

Supervisor: Victor del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Ajul Gopu Elayadath

Student ID: X23295872

Programme: M.Sc. AI for Business **Year:** 2024-2025

Module: MSc (Research) Practicum/Internship

Lecturer: Victor del Rosal

Submission Due Date: **11/05/2025**

Project Title: Deepfake Detection Using a Lightweight Hybrid CNN-ViT Model for Low-Power Devices

Word Count: 1200 **Page Count:** 7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ajul Gopu Elayadath

Date: 11/02/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□

You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

AJUL GOPU ELAYADATH
Student ID: x23295872

1 Introduction

This configuration manual provides the complete technical setup required to replicate the deepfake detection project. This artefact implements a lightweight Hybrid CNN–ViT model combining MobileNetV2 for efficient local feature extraction and Tiny Vision Transformer (TinyViT) for capturing global contextual features. The system is designed for deployment in edge devices like CPU only systems or mobile phones.

This document provides a step-by-step guide setting up the hardware, software, datasets, and project environment. Including instructions for:

- Extracting facial regions from training and testing dataset using **MTCNN**.
- Preprocessing datasets with resizing, augmentation, and normalization.
- Training the model using **focal loss** to address class imbalance.
- Performing post-training dynamic quantization to reduce model size and improve inference speed.
- Performing post-training dynamic quantization to reduce model size and improve inference speed.
- Running a Gradio web interface for real/fake video classification.

2 System Configuration

This section describes the hardware and software setup required to reproduce the deep-fake detection project starting from data preprocessing to model deployment.

2.1 Hardware Requirements

Operating system: Windows 11
Processor: Equivalent to Intel Core i5 or above
RAM: 8 GB
Storage: 10 GB free disk space
GPU: NVIDIA Tesla T4 / RTX 3060 or higher. For faster training

All the preprocessing, training, and evaluation processes can be done using google colab for faster training and evaluation process and when connect to drive the result are saved automatically.

2.2 Software Requirements

All the preprocessing, training, and evaluation processes can be done using Google Colab Pro for faster training and evaluation process and when connect to drive the result are saved automatically. All the code except *app.py* is written for working in google colab.

2.2.1 Programming Language & Environment

- Python 3.10: provides compatibility with PyTorch and the selected libraries.

2.2.2 Core Deep Learning Libraries

- Torch: PyTorch framework for building, training, and deploying the deepfake detection model.
- Torchvision: Provides the CNN backbones and image transformation utilities.
- Timm: Implements TinyViT vision transformer for global feature extraction.
- scikit-learn: For model evaluation with metrics such as Accuracy, Precision, Recall, F1-score, and AUC.

2.2.3 Computer Vision & Preprocessing

- opencv-python: Enables video frame extraction.
- facenet-pytorch: Provides Multi-task Cascaded Convolutional Networks (MTCNN) for high accuracy face localization and cropping.
- tqdm: provides the visual progress bar.
- Matplotlib: provide visual analytics for evaluation.

2.2.4 Deployment & Interface

- gradio: used to build an interactive web interface for real-time video upload, face detection.

2.3 Building & Hosting Tools

The face extraction is done using VS code with python 3.10
Google Colab Pro is used for model training with access to NVIDIA Tesla T4 GPUs for faster training time with direct integration with google drive for model results and checkpoint. Hugging Face is used host the interface for making it publicly available.

2.4 Dataset Requirements

Training Dataset: FaceForensics ++ dataset is used as the training dataset for this project dataset was requested through a google form and the script for downloading (dataface.py) was mailed as a reply.

Testing dataset: Celeb-DF v2 a high quality deepfake dataset was used for evaluating the model.

3 Project Implementation

This section provides a step-by-step guideline on how to run full pipeline starting from face extraction using CPU deployment. The project was done in 2 phases. The first phase is dataset collection and face extraction, and it is done using VS code. The second phase which consists of the rest of the project is done using Google Colab.

3.1 Project Structure

The faceforensic++ dataset is downloaded using the ***dataface.py*** and Celeb-DF v2 is directly downloaded.

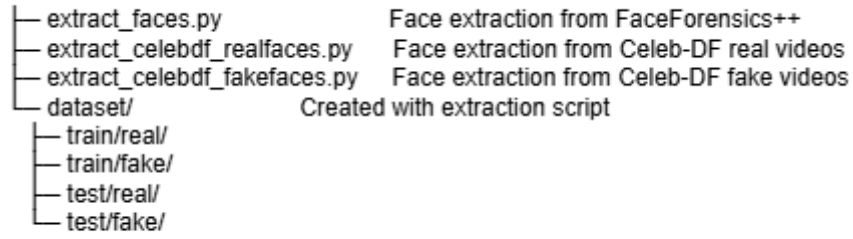

Figure 1 Local structure

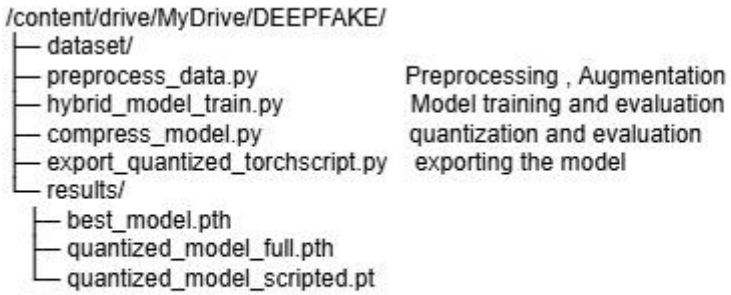

Figure 2 Colab structure

Google colab is connect to Google Drive and new Folder named Deepfake is created to store all the project files.

```
your-space/
├── app.py
├── results/quantized_model_scripted.pt
├── requirements.txt
├── runtime.txt
└── apt.txt (optional)
```

Figure 3 hugging face structure

3.2 Dataset Preparation (Frame Extraction and Face Cropping)

This process is conducted using VS code and its aim is to extract frames, and crop faces and categories and store them accordingly

Download all the dependencies required for extraction

```
pip install torch torchvision timm
            opencv-python facenet-pytorch
            scikit-learn matplotlib tqdm
```

Figure 4 Dependencies

Using ***extract_faces.py*** script the training dataset was processed, and the extracted faces were stored in a folder name dataset in dataset/train/real and dataset/train/fake format.

```
VIDEO_DIRS = {
    'real': 'ffpp_data/original_sequences/youtube/c23/videos',
    'fake': 'ffpp_data/manipulated_sequences/Deepfakes/c23/videos'
}

OUTPUT_BASE = 'dataset/train'
MAX_FRAMES = 50        # Limit per video
SKIP_FRAMES = 5        # Use every 5th frame
```

Figure 5 extract training face location

extract_celebdf_fakefaces.py and ***extract_celebdf_realfaces.py*** files were used to extract the faces for testing and were stored in the same dataset folder as dataset/test/fake format dataset/test/real.

```
VIDEO_DIRS = {
    'real': 'Celeb-DF-v2 (1)/Celeb-real',

}

OUTPUT_BASE = 'dataset/test'  # Testing dataset
MAX_FRAMES = 50               # Max faces per video
SKIP_FRAMES = 5               # Use every 5th frame to reduce redundancy
```

Figure 6 extract testing faces

While running the above-mentioned code MTCNN will detect the faces from every 5th frame and save it in a jpg format.

After the extraction the dataset folder is zipped and uploaded into /content /drive /MyDrive/DEEPFAKE in google drive.

3.3 Google drive mounting to Google Colab.

This step is essential for making the file and available to google colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

Figure 7 Mounting Code

```
%cd /content/drive/MyDrive/DEEPFAKE
```

Figure 8 Code to access the file in DEEPFAKE folder

Create a new Colab notebook and write the code in Figure 7 and 8 to establish connection between Google Colab and Google Drive.

3.4 Data preprocessing

The objective of this step is to prepare the dataset for model training.

The first step is to unzip the dataset folder inside DEEPFAKE. And download same dependencies in Figure 4 to Google Colab also.

```python
import zipfile
import os

# Path to your ZIP file
zip_path = "/content/drive/MyDrive/DEEPFAKE/dataset.zip"

# Destination folder
extract_path = "/content/drive/MyDrive/DEEPFAKE/dataset"

# Make sure destination exists
os.makedirs(extract_path, exist_ok=True)

# Extract
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(f"✅ Dataset extracted to: {extract_path}")
```

Figure 9 Unzip code.

After the unzipping process. *The* ***preprocess_data.py*** *script* was applied to do make the following transformations.

- Resizing all images to 224x224.
- Data augmentations like random horizontal flips, color jitter, rotation, and Gaussian blur to simulate real work artifacts.
- Normalized to the range [-1 ,1]
- Calculate class weights to address the imbalance between real and fake.

The script splits the training dataset into 85% training, and 15% validation sets and create data loaders (train_loader, val_loader, test_loader) for importing the data to training script.

Run the script in Google Colab using the code [! python preprocess_data.py]

3.5 Model training and Evaluation

The ***hybrid_model_train.py*** script defines and trains the HybridCNNViT model, which combines MobileNetV2 and TinyViT. During the initial phases of the training the backbones are frozen and gradually unfreeze after fifth epoch. After running the file .py file in a cell using the code [! python hybrid_model_train.py] the script will give back the best performing model checkpoint saved as ***best_model.pth*** *in* a new folder result along with accuracy and ROC plots.

3.6 Model Quantization and Evaluation

Once model is trained and evaluated the nest step is compressing the model to be more efficient for deploying in edge devices. This process is done using the ***compress_model.py*** script which performs post- training dynamic quantization. The script begins by loading the model from ***results/best_model.pth*** .

After quantization the compressed model is saved in the ***results*** directory as ***quantized_model_full.pth.*** The script also conducts an evaluation of the compressed model to give back to give back accuracy precision along with AUC.

3.7 TorchScript Export

After the compression the next step is to convert the compressed model into a format that is lightweight, portable, and optimized for edge devices.This is done by using ***export_quantized_torchscript.py.*** The script takes the ***results/quantized_model_full.pth*** as input and gives the exported Torchscript model saved ***results/quantized_model_scripted.pt*** as outputs.

3.8 CPU-Only Deployment (Gradio App on Hugging Face)

And the final stage is the deployment using ***app.py***. This script loads the Torchscript model and provides a Gardio web interface. Which accept videos as input and extract faces and preprocess the images to match the training data and run prediction for each frame.

A New space is created in hugging face and the ***app.py*** file along with ***quantized_model_scripted.pt*** and ***requirements.txt*** is uploaded into the hugging face for hosting the application.

Live Demo :- https://huggingface.co/spaces/ajulgopu/Deepfake_detector

References

FaceForensics++ Dataset, 2019. *FaceForensics++*. [online] Available at: https://github.com/ondyari/FaceForensics [Accessed 11 April 2025].

Celeb-DF Dataset, 2020. *Celeb-DF (v2)*. [online] Available at: https://github.com/yuezunli/celeb-deepfakeforensics?tab=readme-ov-file [Accessed 11 April 2025].

Google Colab, n.d. *Google Colaboratory*. [online] Available at: https://colab.google [Accessed 11 june 2025].

Hugging Face, n.d. *Hugging Face Spaces*. [online] Available at: https://huggingface.co/spaces/ajulgopu/Deepfake_detector[Accessed 30 june2025].