National College of Ireland

Configuration Manual

MSc Research Project

Design & Evaluation of a Multi-Modal Deepfake Detection Framework for Advanced Phishing Threats

Gavin Corr
Student ID: x18382836

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……Gavin Corr……

Student ID: ……x18382836……..……

Programme: …… MSCCYBE_JANO24_O…… **Year:** ……2025……..

Module: …… MSc (Research) Practicum Part 2

Lecturer: ……Raza Ul Mustafa……..……

Submission Due Date: ……11/12/2025…….……

Project Title: Design & Evaluation of a Multi-Modal Deepfake Detection Framework for Advanced Phishing Threats

Word Count: …1131 (Excluding References)…… **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: …… Gavin Corr ……

Date: ……11/12/2025……

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

Gavin Corr
Student ID: x18382836

IMPORTANT!
All artefacts can be accessed from the public Hugging Face repo:
https://huggingface.co/spaces/gcorr99/DeepFakeDetection/tree/main

All training and preprocessing of models can be found in their respective notebooks (AudioTraining.ipynb and VideoTraining.ipynb) found on the hugging face repo and will not be detailed in this manual to reduce verbosity.

1 Section 1

This configuration manual specifies the environment, software dependencies and key settings required to:

- Reproduce the audio and video deepfake detection experiments, and
- Run the deployed multi-modal deepfake detection application implemented as a Hugging Face Space using Gradio.

The manual focuses on:

- Hardware and OS assumptions
- Python environment and libraries
- Repository and file structure for the Hugging Face Space
- Application-level configuration (token, logging, UI behaviour)

It does not explain how to install standard tools such as Python, Git, Jupyter or basic OS configuration, in line with project requirements.

2 Hardware & System Requirements

2.1 Hardware

The project is designed to run on CPU-only hardware, both for training and deployment, to reflect realistic constraints in many organisational environments.

Minimum recommended specification:

- CPU: Quad-core 64-bit processor (e.g. Intel Core i7 or equivalent)
- RAM: 16 GB
- GPU: Not required (GPU is explicitly disabled in the app)

Storage:

- 20–30 GB free for datasets, extracted features and models if retraining
- Additional space for logs and temporary files

2.2 Operating System

- 64-bit OS such as Windows 10/11 or equivalent.
- Python 3.9 or newer environment used for both training notebooks and the application.

3 Software Environment

3.1 Core Runtime

- Python: 3.9.x
- Jupyter Notebook / JupyterLab: for running AudioTraining.ipynb and VideoTraining.ipynb during model development.

3.2 Python Dependencies

- All library versions for the deployed application are defined in requirements.txt:
- gradio>=4.0.0 (tested via Hugging Face Space with Gradio SDK 5.49.1)
- README
- tensorflow==2.19.0
- numpy==1.26.4
- pillow
- librosa==0.10.1
- soundfile

- huggingface_hub>=0.23.0
- mtcnn==0.1.1
- opencv-python-headless==4.8.1.78
- matplotlib

For training, the same or compatible versions should be installed in the local Python environment to avoid discrepancies between training and inference behaviour.

3.3 Environment Configuration (Global Settings)

Within app.py, the following environment settings are applied to ensure CPU-only and reduce TensorFlow logging:

- os.environ["CUDA_VISIBLE_DEVICES"] = "-1" # Force CPU
- os.environ["TF_ENABLE_ONEDNN_OPTS"] = "0"
- os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2" # Suppress verbose logs

Matplotlib is configured to use a non-interactive backend suitable for headless environments (Hugging Face Spaces):

- import matplotlib
- matplotlib.use("Agg")

4 Repository & Hugging Face Space

The experimental setup and final app are hosted in a Hugging Face Space with the following core files:

- README.md – Space metadata and brief usage notes
- app.py – Gradio application entry point
- requirements.txt – Python dependencies
- AudioTraining.ipynb – Audio model training notebook
- VideoTraining.ipynb – Video model training notebook
- datasets.zip – Small sample datasets for testing the app (not full training datasets)
- .gitattributes – Git attributes file

An example repository structure:

```
/
├── app.py
├── requirements.txt
├── README.md
├── AudioTraining.ipynb
├── VideoTraining.ipynb
├── datasets.zip          # demo/testing data only
└── .gitattributes
```

The Hugging Face Space configuration (README.md) specifies:

- README
- sdk: gradio
- sdk_version: 5.49.1
- app_file: app.py
- emoji, colorFrom, colorTo and license metadata

5 Model Artefacts and Hugging Face Hub Integration

Model Artefacts and Hugging Face Hub Integration
The application does not store the trained models directly in the Space repository. Instead, it loads them at runtime from two private Hugging Face model repositories using huggingface_hub.

```
AUDIO_REPO = "gcorr99/AudioDeepfakeDetection"
AUDIO_FILE = "voice_classifier_model.keras"

VIDEO_REPO = "gcorr99/VideoDeepfakeDetection"
VIDEO_FILE = "face_deepfake_model.keras"
```

A helper function downloads and loads the models:

```
def load_from_hub(repo_id: str, filename: str):
    path = hf_hub_download(repo_id=repo_id, filename=filename, token=TOKEN)
    return tf.keras.models.load_model(path, compile=False)
```

To replicate the application:

1. Ensure both model repositories exist under your Hugging Face account.
2. Upload the trained models as:

   voice_classifier_model.keras (audio CNN)
   face_deepfake_model.keras (MobileNetV2-based visual model)

3. Grant read access to the Hugging Face Space via a personal access token (see Section 7.1).

6 Dataset Configuration

Full training datasets are not shipped with the Space due to size constraints. datasets.zip contains a small subset suitable for testing the application only.

- Audio: DEEP-VOICE: DeepFake Voice Recognition dataset found on Kaggle (birdy654, n.d.) for real vs AI-generated speech.

- Video: FaceForensics++ dataset (Rössler et al. 2019) for real vs manipulated facial videos.

7 Application-Level Configuration

7.1 Hugging Face Token and Authentication

app.py requires an access token with permission to read the private model repositories.

```
TOKEN = os.getenv("HF_TOKEN") or os.getenv("HFTOKEN")
if not TOKEN:
  raise RuntimeError("No Hugging Face token found. Add HF_TOKEN under Settings → Secrets.")
```

To replicate:

1. Create a Hugging Face personal access token with read access.
2. In the Space settings, add a secret:

   Name: HF_TOKEN
   Value: <your token>

3. Ensure the Space has permission to access:

   gcorr99/AudioDeepfakeDetection
   gcorr99/VideoDeepfakeDetection

On app startup, the token is validated and available files in both repos are listed (mainly for debugging/verification).

7.2 Gradio Interface Configuration

The UI is built using gr.Blocks with three tabs: Audio, Image and Video.
app
Key components:

- Audio tab

  Input: gr.Audio(type="filepath")
  Button: “Analyze Audio”
  Outputs:

  gr.Image – Mel spectrogram preview
  gr.Textbox – Prediction (e.g. FAKE (97.23%))
  gr.File – Session log CSV

- Image tab:

  Input: gr.Image(type="pil")
  Button: “Analyze Image”
  Outputs:

  gr.Textbox – Prediction
  gr.File – Session log CSV

- Video tab

  Input: gr.Video
  Button: “Analyze Video”
  Outputs:

  gr.Gallery – Up to first 5 detected faces for preview
  gr.Textbox – Prediction
  gr.File – Session log CSV

The Blocks app is launched with:

```
if __name__ == "__main__":
    demo.launch(ssr_mode=False)
```

8 Logging and Output Files

The application logs all predictions to a CSV file stored on the container filesystem:

SESSION_LOG_PATH = "/tmp/session_log.csv"

Each new prediction appends a row with:
- timestamp
- type (audio, image, video)
- filename
- prediction (FAKE / REAL)
- confidence (float, 0–1)
- num_faces (for video entries only)

The log is made available to the user via Gradio’s gr.File component so it can be downloaded after one or more predictions.

9 Reproducibility Checklist

To successfully replicate the experimental setup and application:
1. Create Environment
   - Use Python 3.9.
   - Install libraries from requirements.txt.

2. Prepare Training Data (Local)
    - Download DEEP-VOICE dataset for audio and FaceForensics++ for video.
    - Organise files under data/audio and data/video as described in Section 6.1.
    - Update paths inside AudioTraining.ipynb and VideoTraining.ipynb if needed.

3. Train Models
    - Run AudioTraining.ipynb to train the CNN audio model with the preprocessing and architecture described in Sections 7.2 and 8.2.
    - Run VideoTraining.ipynb to train/fine-tune the MobileNetV2 visual model as in Sections 7.4 and 8.3.
    - Save final models as:
        - voice_classifier_model.keras
        - face_deepfake_model.keras

4. Upload Models to Hugging Face Hub
    - Create private model repos: AudioDeepfakeDetection and VideoDeepfakeDetection.
    - Upload the .keras files.
    - Confirm repo IDs and filenames match AUDIO_REPO, VIDEO_REPO, AUDIO_FILE, VIDEO_FILE constants in app.py.

5. Configure the Space
    - Create a new Gradio Space.
    - Add:
    - app.py
    - requirements.txt
    - README.md
    - (optionally) AudioTraining.ipynb, VideoTraining.ipynb, datasets.zip
    - Set app_file to app.py in the Space metadata.

6. Set Hugging Face Token
    - Add HF_TOKEN as a secret in the Space with read access to the model repos.

7. Run and Verify
    - Launch the Space; confirm startup succeeds (no token or access errors).
    - Upload audio, image and video files (from datasets.zip or other sources) to verify predictions and logging.
    - Compare performance (qualitatively, and quantitatively if re-evaluating) with metrics reported in the thesis.

References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing
You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

birdy654 (n.d.) DEEP-VOICE: DeepFake Voice Recognition [Dataset]. Kaggle. Available at: https://www.kaggle.com/datasets/birdy654/deep-voice-deepfake-voice-recognition (Accessed: September 10th 2025).

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Nießner, M., 2019. Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1-11).