National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity (Part-Time)

Alana Griffiths Kelly
Student ID: x23302003

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Alana Griffiths Kelly
Student ID:	X23302003
Programme:	MSc Cybersecurity (Part Time) **Year:** 2025
Module:	MSc (Research) Practicum/Internshio
Lecturer:	Supervisor – Michael Pantridge
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	223 **Page Count:** 5

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:	Alana Griffiths Kelly
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

Alana Griffiths Kelly

Student ID: x23302003

1. System Requirements and Environment Setup

1.1. Software Requirements

- Python Version: Python 3.9 or later
- Package manager: pip
- Required Libraries:
  - PyTorch
  - torchvision
  - facenet-pytorch
  - MTCNN
  - PIL
  - glob

1.2. Setup Steps

Step 1: Install Python

Step 2: Install required packages

"pip install facenet-pytorch torchvision torch"

Step 3: Install FFmpeg

Download from https://ffmpeg.org/download.html

"pip install ffmpeg-python"

2. System Configuration and Pipeline Setup

2.1. Preprocessing Pipeline Configuration

2.1.1. Video Frame Extraction (FFmpeg)

Extract frames at 5 FPS

```
# Extract Frames from Video
ffmpeg
.input(video_path) # Load Video
.output('frames/frame_%05d.jpg', r=5) # Extract 5FPS to seperate images neamed 'frame_00001.jpg' etc.
.run() # Execute's FFmpeg Command
```

2.1.2. Face Detection Using MTCNN

MTCNN is used to detect and crop face regions from each frame. The cropped faces are automatically resized to 224x224.

```
mtcnn = MTCNN(image_size=224) # Rezise face crops to 224x224 pixels
```

2.1.3. *Image Normalisation*

```
# Prepare Images for PyTorch Model
transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor()
])
```

2.2. Model Architecture Confgiration

```
class CNN_ViT_Fusion(nn.Module):
    def __init__(self, num_classes=2):
        super().__init__()

        # 1. CNN Backbone (EfficientNet-B0)
        self.cnn = models.efficientnet_b0(weights="IMAGENET1K_V1")
        cnn_out = self.cnn.classifier[1].in_features
        # Remove final classification layer
        self.cnn.classifier[1] = nn.Identity()

        # 2. Vision Transformer Backbone (ViT-B/16)
        self.vit = models.vit_b_16(weights="IMAGENET1K_V1")
        vit_out = self.vit.heads.head.in_features
        self.vit.heads.head = nn.Identity()

        # 3. Fusion Layer
        fusion_dim = cnn_out + vit_out
        self.classifier = nn.Linear(fusion_dim, num_classes)

    def forward(self, x):
        # CNN Feature Vector
        cnn_feats = self.cnn(x)

        # ViT Feature Vector
        vit_feats = self.vit(x)

        # Concatenate Features
        fused = torch.cat([cnn_feats, vit_feats], dim=1)

        # Final Classifier
        return self.classifier(fused)
```

2.2.1. *EfficientNet-B0 (CNN Backbone)*

```
# 1. CNN Backbone (EfficientNet-B0)
self.cnn = models.efficientnet_b0(weights="IMAGENET1K_V1")
cnn_out = self.cnn.classifier[1].in_features
# Remove final classification layer
self.cnn.classifier[1] = nn.Identity()
# CNN Feature Vector
cnn_feats = self.cnn(x)
```

2.2.2. ViT-B/16 (Transformer Backbone)

```
# 2. Vision Transformer Backbone (ViT-B/16)
self.vit = models.vit_b_16(weights="IMAGENET1K_V1")
vit_out = self.vit.heads.head.in_features
self.vit.heads.head = nn.Identity()

# ViT Feature Vector
vit_feats = self.vit(x)
```

2.2.3. Feature Fusion

```
# 3. Fusion Layer
fusion_dim = cnn_out + vit_out
self.classifier = nn.Linear(fusion_dim, num_classes)

# Concatenate Features
fused = torch.cat([cnn_feats, vit_feats], dim=1)
```

2.2.4. Classification head

```
# Final Classifier
return self.classifier(fused)
```

2.2.5. Softmax Interference

```
#Run Detection on Cropped Faces
for face_file in glob.glob("faces/*.jpg"):
    face_img = Image.open(face_file)
    tensor = transform(face_img).unsqueeze(0) # Unsqueeze adds batch dimension as expects 4D input
    with torch.no_grad(): # Disables gradient tracking to speed up inference
        output = model(tensor) # Runs through DF model
        prob_fake = F.softmax(output, dim=1)[0,1].item() # Gets probability of "fake"
        scores.append(prob_fake)
```

2.3. Video-Level Classification

The final probability that a video is fake is computed by averaging frame predictions. A threshold of 0.5 classifies a video as manipulated.

```
scores = []

#Run Detection on Cropped Faces
for face_file in glob.glob("faces/*.jpg"):
    face_img = Image.open(face_file)
    tensor = transform(face_img).unsqueeze(0) # Unsqueeze adds batch dimension as expects 4D input
    with torch.no_grad(): # Disables gradient tracking to speed up inference
        output = model(tensor) # Runs through DF model
        prob_fake = F.softmax(output, dim=1)[0,1].item() # Gets probability of "fake"
        scores.append(prob_fake)

video_score = sum(scores) / len(scores) # Avg fake probability across frames
```

3. Running the System and Reproducing Results

3.1. Running the Full Pipeline

Run preprocessing:

Extract frames

```
os.makedirs("frames", exist_ok=True) # Create Folder 'Frames'
(
    # Extract Frames from Video
    ffmpeg
    .input(video_path) # Load Video
    .output('frames/frame_%05d.jpg', r=5) # Extract 5FPS to seperate images neamed 'frame_00001.jpg' etc.
    .run() # Execute's FFmpeg Command
)
```

Detect and crop faces

```
os.makedirs("faces", exist_ok=True)
# Detect and Crop Faces from Frames
for frame_file in glob.glob("frames/*.jpg"): # Gets all frame filenames from folder 'Frames'
    frame_img = Image.open(frame_file) # Open Frame as Image
    face = mtcnn(frame_img) # Detect face & return cropped tensor
    if face is not None: # Ensure face found in frame
        face_img = to_pil_image(face) # Convert Tensor to PIL
        face_img.save(f"faces/{os.path.basename(frame_file)}") # Save crops to new folder
```

Run model

```
# Load the Hybrid Model
model = CNN_ViT_Fusion()
model.eval()
```

Get probability

```
video_score = sum(scores) / len(scores) # Avg fake probability across frames
print(f"\nDeepfake Probability: {video_score:.3f}")
```

3.2. Output Files

/frames/ – extracted frames
/faces/ – cropped face images