National College of Ireland

Configuration Manual

MSc Research Project
MSc Artificial Intelligence

Rajat Ranjit Amate
Student ID: x23269723

School of Computing
National College of Ireland

Supervisor: Prof Lavish Thomas

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rajat Ranjit Amate
Student ID:	X23269723
Programme:	MSC Ai **Year:** 2024-25
Module:	Practicum Research
Lecturer:	Prof. Lavish Thomas
Submission Due Date:	11/08/20225
Project Title:	Real-Time Multimodal AI for Continuous Mental Health Monitoring
Word Count:	1039 **Page Count: 6**

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:

Date: 11/08/2025

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

Rajat Ranjit Amate
Student ID: x23269723

1 Introduction

Maintaining mental well-being is a critical challenge in today's fast-paced world. Traditional mental health assessments rely on discrete, single-modality observations—often audio-based surveys, video-based analyses, or text-based self-reports. These isolated approaches can miss complementary physiological and behavioral cues, lack real-time responsiveness, and offer limited explainability.

Our Real-Time Multimodal AI pipeline addresses these gaps by concurrently analyzing:

- Facial expressions via a fine-tuned EfficientNet-B0 model on the MELD dataset
- Speech emotion through CNN–BiLSTM with self-attention on RAVDESS
- Text sentiment from Whisper transcriptions processed by TextBlob
- Fatigue levels via blink detection using MediaPipe FaceMesh

This setup manual provides comprehensive instructions to replicate the Real-Time Multimodal AI system for continuous mental health monitoring. The manual details all hardware and software requirements, installation procedures, dataset preparation, model training, and deployment steps necessary to run the complete multimodal pipeline that combines facial emotion recognition, speech emotion detection, text sentiment analysis, and fatigue monitoring through blink detection.

2 Hardware and Software Requirements

Core Software:

Python: 3.9.16 or 3.10.x
CUDA Toolkit: 11.8 or 12.1 (for GPU acceleration)
Google Colab pro

Python packages:

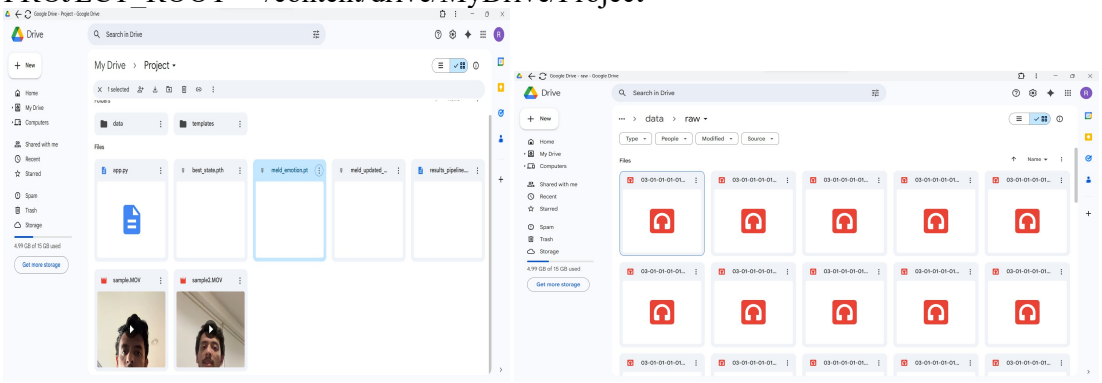
- numpy==1.26.4
- pandas==2.2.2
- scikit-learn==1.6.1
- joblib==1.5.1
- librosa==0.11.0
- soundfile==0.13.1
- ffmpeg-python (via system)
- moviepy==2.2.1
- imageio==2.37.0
- imageio-ffmpeg==0.6.0
- opencv-python-headless==4.11.0.86
- mtcnn==1.0.0

- mediapipe==0.10.21
- torch==2.6.0+cpu
- torchvision==0.21.0+cpu
- timm==1.0.19
- torchaudio (CPU) matching torch version
- tensorflow==2.19.0
- openai-whisper==2025.7.0
- transformers==4.35.0
- textblob==0.19.0
- tqdm==4.67.1
- pillow==11.3.0
- PyDub==0.25.1 (for audio I/O)
- moviepy dependencies (decorator, python-dotenv, proglog) at compatible versions

3 Installation & Environment Setup

The most important of the datasets are stored in drive

```
from google.colab import drive
drive.mount('/content/drive')
PROJECT_ROOT = '/content/drive/MyDrive/Project
```



4 Data Preprocessing:

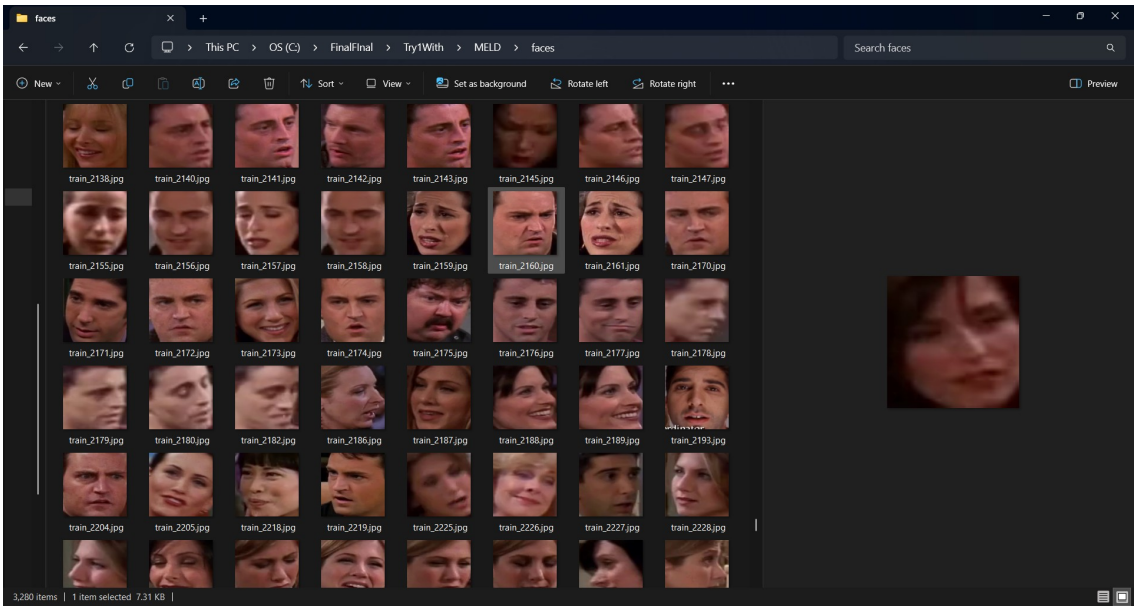
Facial Data Extraction (MELD):

- Read each split's CSV (train_sent_emo.csv, dev_sent_emo.csv, test_sent_emo.csv).
- For every row, construct the video filename (dia{Dialogue_ID}_utt{Utterance_ID}.mp4) and locate it under …/data/MELD/{split}extract/.
- Capture the first frame via OpenCV, run MTCNN face detection, and crop the bounding box.
- Resize crop to 224×224, save as {split}_{Sr No.}.jpg under …/data/MELD/faces/, and record metadata (img, emotion, dialogue_id, utterance_id) into {split}.json.
- Reorganize Face Crops:
- Create directories data/{train,dev,test}/{anger,disgust,fear,joy,neutral,sadness,surprise}.

- Load each split's JSON metadata, copy matching face images into their respective split/emotion folders.

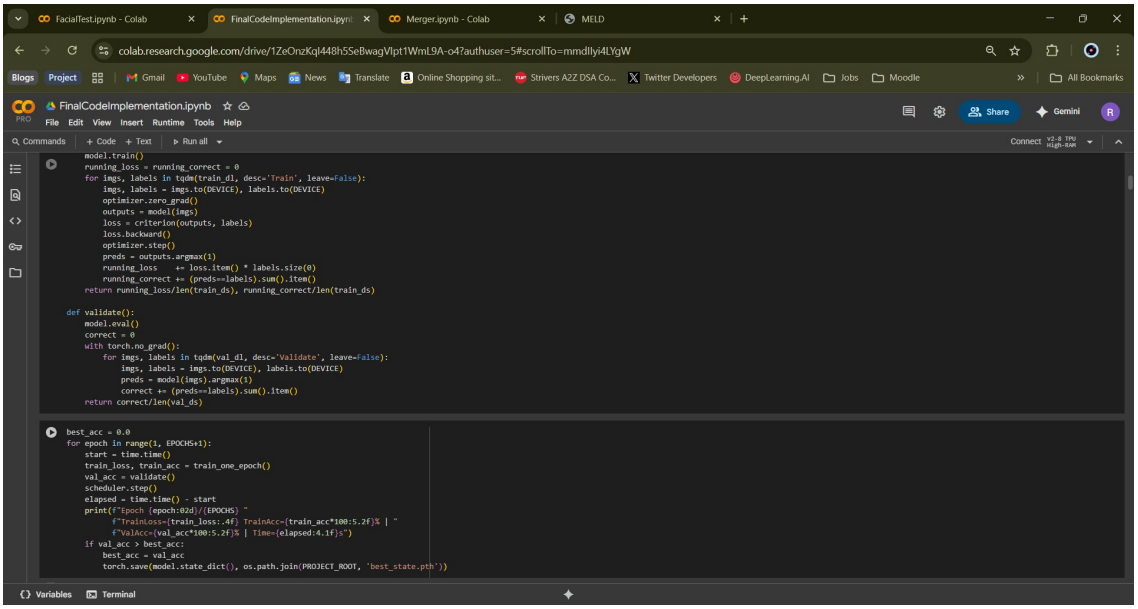
Audio Feature Extraction (RAVDESS):

- Recursively find all .wav files under data/RAVDESS/Actor_*.
- For each file, load at 16 kHz, iterate 4.32 s windows with 50% overlap.
- For every window: compute features—40 MFCCs + first/second deltas, spectral centroid, rolloff, bandwidth, tempo, RMS energy, zero-crossing rate—stack into a fixed-size 9×173 matrix, save as .npy.
- Extract emotion label from filename pattern, append {"file":..., "emotion":...} to processed/meta.json.
- Train/Validation Split for Audio:
- Load processed/meta.json into a Pandas DataFrame.
- Use StratifiedShuffleSplit(test_size=0.2) on the emotion column to split into train and val sets.
- Save train.json and val.json manifests for downstream ingestion.



5 Model Training Pipeline

Face Emotion Module: fine-tuned EfficientNet-B0 (timm pretrained) on 224×224 MTCNN-cropped face images from MELD; 7-way softmax, class-weighted CrossEntropyLoss with label smoothing, AdamW optimizer, CosineAnnealingLR schedule over 25 epochs.

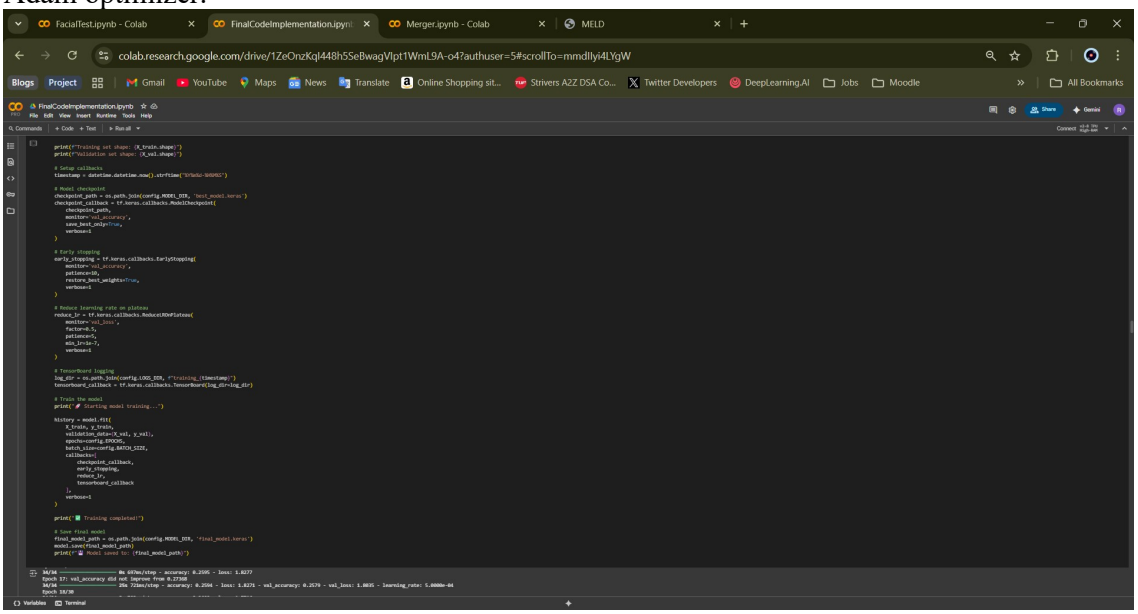


```
        model.train()
        running_loss = running_correct = 0
        for imgs, labels in tqdm(train_dl, desc='Train', leave=False):
            imgs, labels = imgs.to(DEVICE), labels.to(DEVICE)
            optimizer.zero_grad()
            outputs = model(imgs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            preds = outputs.argmax(1)
            running_loss    += loss.item() * labels.size(0)
            running_correct += (preds==labels).sum().item()
        return running_loss/len(train_ds), running_correct/len(train_ds)

    def validate():
        model.eval()
        correct = 0
        with torch.no_grad():
            for imgs, labels in tqdm(val_dl, desc='Validate', leave=False):
                imgs, labels = imgs.to(DEVICE), labels.to(DEVICE)
                preds = model(imgs).argmax(1)
                correct += (preds==labels).sum().item()
        return correct/len(val_ds)
```

```
best_acc = 0.0
for epoch in range(1, EPOCHS+1):
    start = time.time()
    train_loss, train_acc = train_one_epoch()
    val_acc = validate()
    scheduler.step()
    elapsed = time.time() - start
    print(f"Epoch {epoch:02d}/{EPOCHS} "
          f"TrainLoss={train_loss:.4f} TrainAcc={train_acc*100:5.2f}% | "
          f"ValAcc={val_acc*100:5.2f}% | Time={elapsed:4.1f}s")
    if val_acc > best_acc:
        best_acc = val_acc
        torch.save(model.state_dict(), os.path.join(PROJECT_ROOT, 'best_state.pth'))
```

Speech Emotion Module: 1D CNN on stacked MFCC + delta + spectral features (9×173), trained on RAVDESS audio windows; softmax over 8 classes, categorical cross-entropy loss, Adam optimizer.



```
print(f"Training set shape: {X_train.shape}")
print(f"Validation set shape: {X_val.shape}")

# Setup callbacks
timestamp = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

# Model checkpoint
checkpoint_path = os.path.join(config.MODEL_DIR, 'best_model.keras')
checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    checkpoint_path,
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)

# Early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_accuracy',
    patience=10,
    restore_best_weights=True,
    verbose=1
)

# Reduce learning rate on plateau
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=5,
    min_lr=1e-7,
    verbose=1
)

# TensorBoard logging
log_dir = os.path.join(config.LOGS_DIR, f"training_{timestamp}")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)

# Train the model
print("🚀 Starting model training...")

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=config.EPOCHS,
    batch_size=config.BATCH_SIZE,
    callbacks=[
        checkpoint_callback,
        early_stopping,
        reduce_lr,
        tensorboard_callback
    ],
    verbose=1
)

print("✅ Training completed!")

# Save final model
final_model_path = os.path.join(config.MODEL_DIR, 'final_model.keras')
model.save(final_model_path)
print(f"💾 Model saved to: {final_model_path}")
```

Sentiment Module: TextBlob polarity classifier on Whisper transcripts.

Fatigue Module: Mediapipe Face Mesh EAR detector with thresholding (EAR < 0.25 over ≥3 frames) to compute blink rate and duration; rule-based state mapping.
The late-fusion module computes a unified risk score by weighting each modality's confidence or state score and averaging:

- Inputs
  – Face emotion confidence (1 − softmax_max)
  – Speech emotion confidence (1 − softmax_max)
  – Sentiment polarity (absolute TextBlob polarity)
  – Fatigue state score (rule-mapped blink rate → [0.1–0.9])

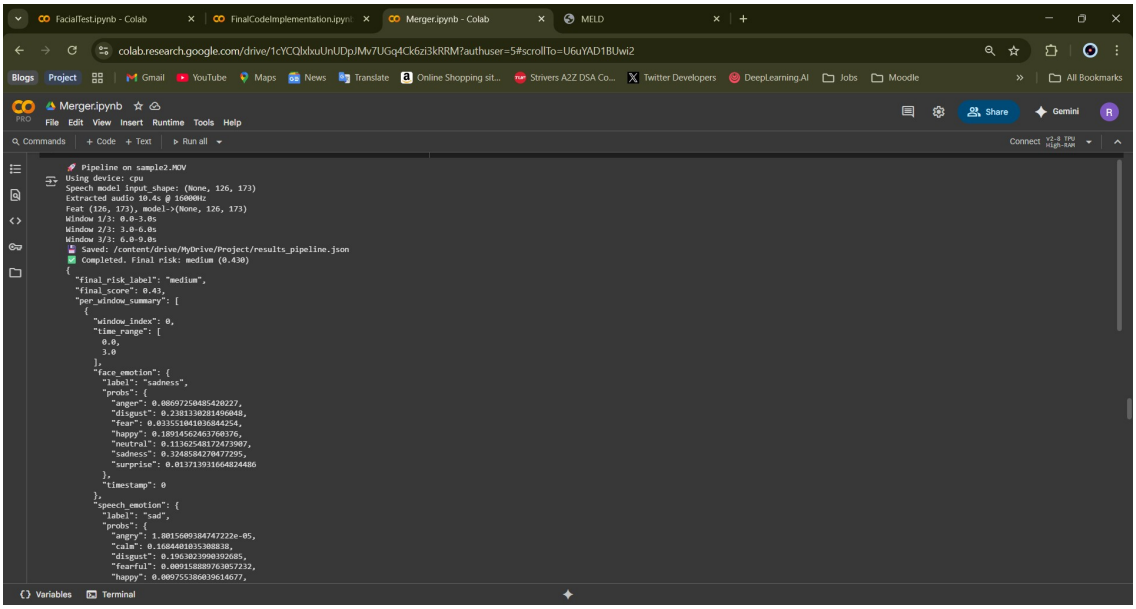
- Weights
  – face: 0.3
  – speech: 0.3
  – sentiment: 0.2
  – fatigue: 0.2
- Scoring
  fusion_score = (face_score*0.3* + *speech_score*0.3 + sentiment_score*0.2* + *fatigue_score*0.2)
- Category
  – ≥0.6 → HIGH risk
  – 0.3–0.6 → MEDIUM risk
  – <0.3 → LOW risk

6 Troubleshooting & Performance

Mostly the errors arise with conflicting versions of numpy and moviepy specifically for the moviepy.editor we need an exact version of moviepy which is moviepy==1.0.3.

7 How to run

First, download and extract the MELD multimodal dataset from the official website, then run face_extract_fixed.py to crop and save all face images and reorganize them by emotion with reorganize_faces.py; next, download and unpack the RAVDESS speech dataset from Kaggle, perform cleaning and exploratory data analysis (e.g., class distributions, audio waveforms, spectrograms), then train a 1D-CNN on MFCC and spectral features to produce emotion_model.keras and its LabelEncoder. Separately, fine-tune an EfficientNet B0 model on the extracted MELD face images to generate a TorchScript file meld_emotion.pt, and implement a MediaPipe-based blink detector to compute real-time blink rate and fatigue state. Use OpenAI's Whisper model to transcribe video audio and TextBlob for sentiment polarity. Finally, execute Merger.ipynb to load all pretrained weights, extract and fuse face emotion, speech emotion, transcript, sentiment, and blink-based fatigue metrics on any input video, producing a unified JSON report and visualizations.


FacialTest.ipynb - Colab
FinalCodeImplementation.ipynb
Merger.ipynb - Colab
MELD
colab.research.google.com/drive/1cYCQlxlxuUnUDpJMv7UGq4Ck6zi3kRRM?authuser=5#scrollTo=U6uYAD1BUwi2
Merger.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
Pipeline on sample2.MOV
Using device: cpu
Speech model input_shape: (None, 126, 173)
Extracted audio 10.4s @ 16000Hz
Feat (126, 173), model->(None, 126, 173)
Window 1/3: 0.0-3.0s
Window 2/3: 3.0-6.0s
Window 3/3: 6.0-9.0s
Saved: /content/drive/MyDrive/Project/results_pipeline.json
Completed. Final risk: medium (0.430)
{
"final_risk_label": "medium",
"final_score": 0.43,
"per_window_summary": [
{
"window_index": 0,
"time_range": [
0.0,
3.0
],
"face_emotion": {
"label": "sadness",
"probs": {
"anger": 0.08697250485420227,
"disgust": 0.2381330201490048,
"fear": 0.03355104103684254,
"happy": 0.18914562463760376,
"neutral": 0.11362548172473907,
"sadness": 0.32485842704772795,
"surprise": 0.013713931664824486
},
"timestamp": 0
},
"speech_emotion": {
"label": "sad",
"probs": {
"angry": 1.8015609384747222e-05,
"calm": 0.16844010353088838,
"disgust": 0.19630239903926085,
"fearful": 0.009158888763857232,
"happy": 0.009755386039614677,
Variables
Terminal