National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Anif Zarus Andrew Thomas
Student ID: x23307609

School of Computing
National College of Ireland

Supervisor: Dr.Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anif Zarus Andrew Thomas
Student ID:	x23307609
Programme:	Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr.Christian Horn
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	931
Page Count:	12

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:	Anif Zarus Andrew Thomas
Date:	11th August 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

Anif Zarus Andrew Thomas
x23307609

1 Report Overview

This report provides detailed information on the code developed for the research project. It is intended to enable others to understand, replicate, and build upon the code as needed.

2 System Requirements

OS version,Hardware specs (CPU, GPU, RAM)

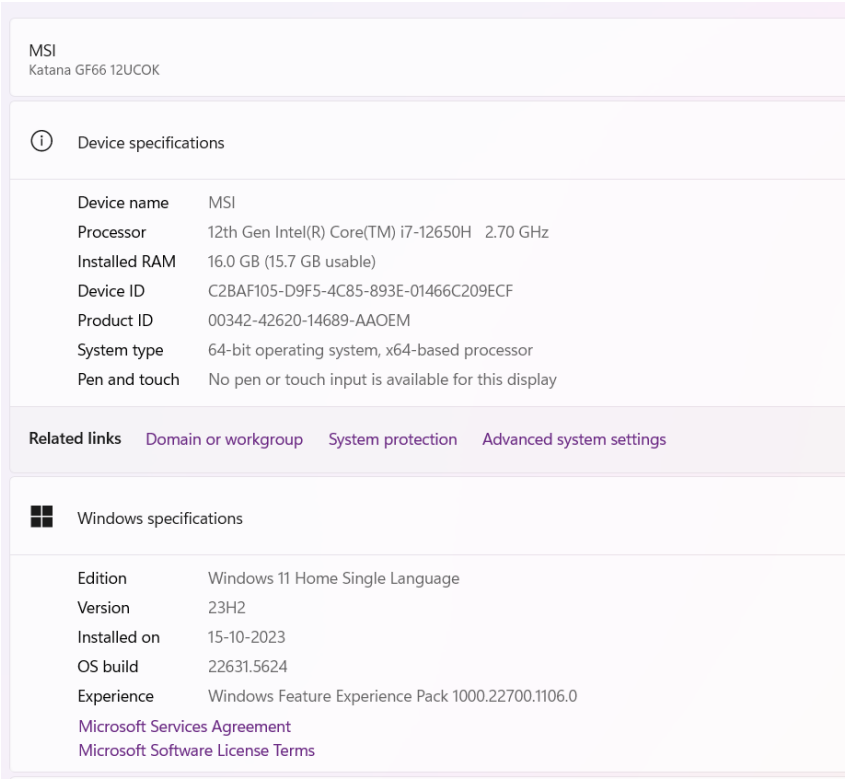


Figure 1: System Configurations used in the Experiment

Python version and environment info

The original Python version available on the desktop is Python 3.12.4, which was used for data preprocessing and initial modeling. However, due to compatibility issues with TensorFlow, a virtual environment named `tfenv` was created using Anaconda, in which Python 3.9.23 was installed for model implementation.

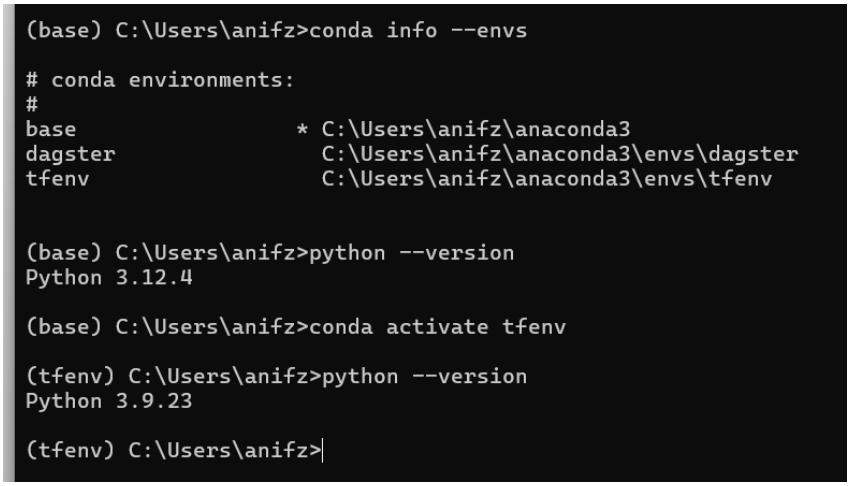


Figure 2: python versions

This setup ensured that the deep learning framework and related libraries functioned properly. You can download Python from the official website: Download Python

3 Software Dependencies

These are the packages that were used for the project.Some were pre-installed and some were added based on the requirements of the project. If you're python version is the latest one, tensorflow might not be compatible, so need to downgrade.

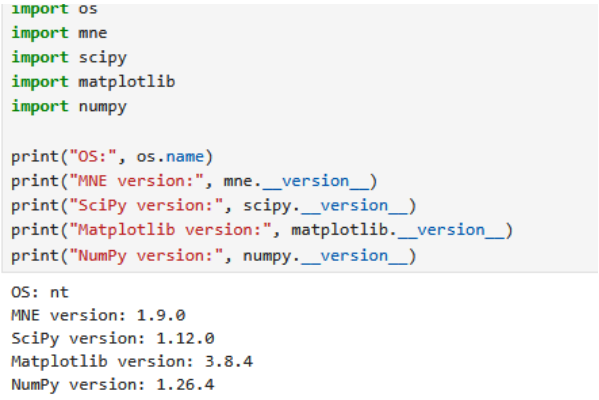


Figure 3: Packages available originally available in desktop

Software Installation

Install the required Python packages using:

```
pip install tensorflow scikit-learn matplotlib numpy mne scipy
```

For more details, visit the package pages:

- TensorFlow

```python
import os
import tensorflow as tf
import sklearn
import matplotlib
import numpy

print("OS:", os.name)
print("TensorFlow version:", tf.__version__)
print("Scikit-learn version:", sklearn.__version__)
print("Matplotlib version:", matplotlib.__version__)
print("NumPy version:", numpy.__version__)
```

```
OS: nt
TensorFlow version: 2.13.0
Scikit-learn version: 1.6.1
Matplotlib version: 3.9.4
NumPy version: 1.24.3
```

Figure 4: packages installed in virtual environment for the project

- scikit-learn
- matplotlib
- NumPy
- MNE
- SciPy

4 Dataset Preparation

The experiments utilize the CHB-MIT Scalp EEG Database, a collection of EEG recordings from pediatric subjects with intractable seizures. This dataset contains 22 subjects monitored for up to several days following withdrawal of anti-seizure medication to characterize seizures and assess their candidacy for surgical intervention.

The dataset can be downloaded from the official source here: CHB-MIT Scalp EEG Database.

After downloading, organize the EEG recordings and annotations according to the directory structure expected by the code, typically as follows:

```
/data/CHB_MIT/edf/
/data/CHB_MIT/annotations/
```

Any required preprocessing steps, such as filtering, segmentation, or normalization, are applied during the data loading phase in the code.

5 Preprocessing and project structure

The `Splittingeventandchannelcheck` folder contains code related to splitting the dataset into pre-seizure and post-seizure segments, as well as checking channel information. Non seizure files do not need any splitting and can be directly pre-preprocessed

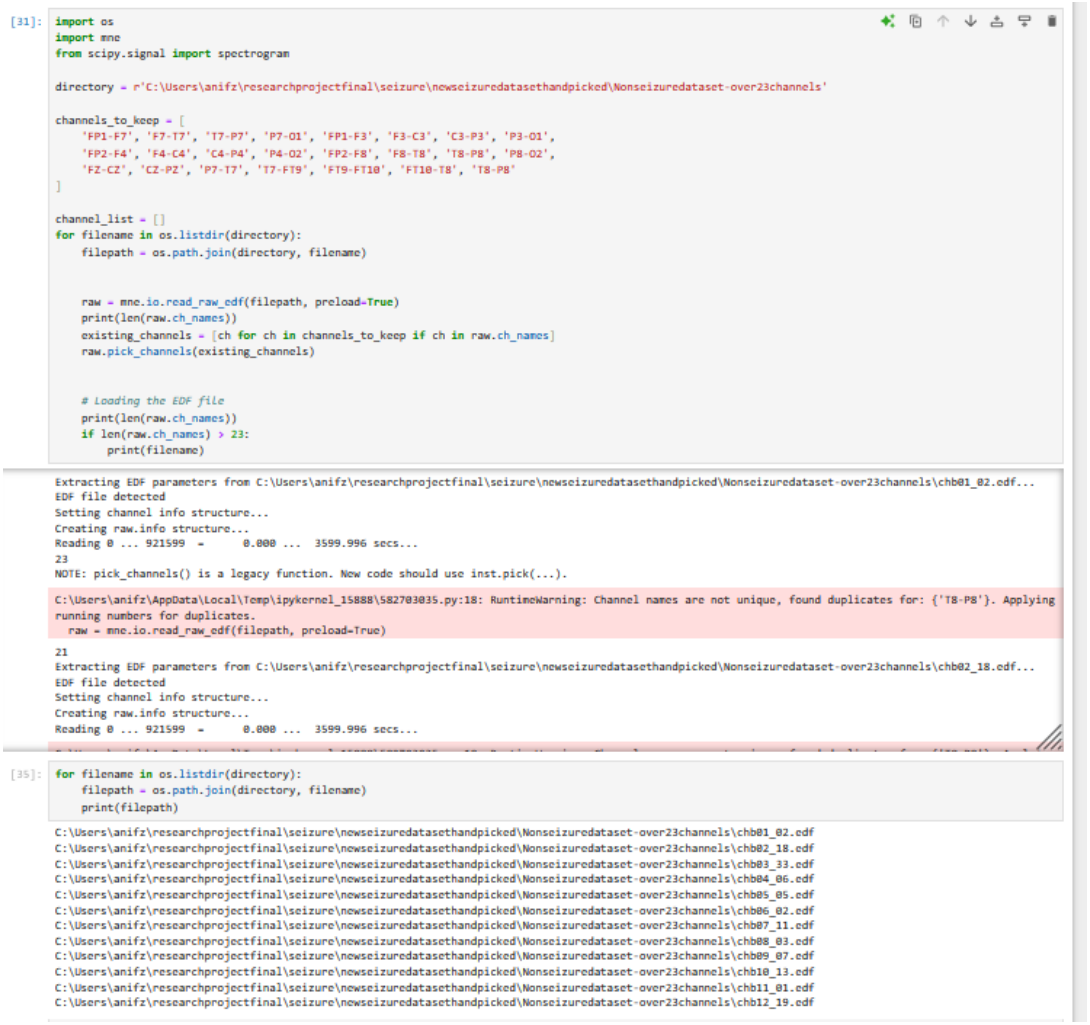
- `SeizureSplit.ipynb`: A Jupyter Notebook that handles the splitting of the raw EEG dataset into separate files for the pre-seizure and post-seizure phases. This notebook performs the segmentation based on seizure event annotations to facilitate phase-specific analysis.

This notebook forms a crucial preprocessing step by organizing the dataset for downstream analysis and model training.

The project is organized into three main folders, each corresponding to a different seizure phase in the dataset:

- `Nonseizure-code-and-dataset`: Contains three Jupyter Notebook files:
  - `channelCheck.ipynb` — notebook to check the number of channels in the EDF files.
  - `preprocess.ipynb` — notebook to preprocess the dataset for the non-seizure phase.
  - `visualize.ipynb` — notebook to visualize the preprocessed data.
- `Post-seizureCodeandDataset`: Contains three notebooks:
  - `channelCheck.ipynb`
  - `preprocess.ipynb`
  - `visualize.ipynb`
- `Pre-seizureCodeandDataset`: Contains two notebooks:
  - `channelCheck.ipynb`
  - `preprocess.ipynb`

  Note that the visualization notebook is not included in the pre-seizure folder.



```
[31]:
import os
import mne
from scipy.signal import spectrogram

directory = r'C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels'

channels_to_keep = [
    'FP1-F7', 'F7-T7', 'T7-P7', 'P7-O1', 'FP1-F3', 'F3-C3', 'C3-P3', 'P3-O1',
    'FP2-F4', 'F4-C4', 'C4-P4', 'P4-O2', 'FP2-F8', 'F8-T8', 'T8-P8', 'P8-O2',
    'FZ-CZ', 'CZ-PZ', 'P7-T7', 'T7-FT9', 'FT9-FT10', 'FT10-T8', 'T8-P8'
]

channel_list = []
for filename in os.listdir(directory):
    filepath = os.path.join(directory, filename)


    raw = mne.io.read_raw_edf(filepath, preload=True)
    print(len(raw.ch_names))
    existing_channels = [ch for ch in channels_to_keep if ch in raw.ch_names]
    raw.pick_channels(existing_channels)


    # Loading the EDF file
    print(len(raw.ch_names))
    if len(raw.ch_names) > 23:
        print(filename)
```

```
Extracting EDF parameters from C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb01_02.edf...
EDF file detected
Setting channel info structure...
Creating raw.info structure...
Reading 0 ... 921599  =      0.000 ...  3599.996 secs...
23
NOTE: pick_channels() is a legacy function. New code should use inst.pick(...).
C:\Users\anifz\AppData\Local\Temp\ipykernel_15888\582703035.py:18: RuntimeWarning: Channel names are not unique, found duplicates for: {'T8-P8'}. Applying
running numbers for duplicates.
  raw = mne.io.read_raw_edf(filepath, preload=True)
21
Extracting EDF parameters from C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb02_18.edf...
EDF file detected
Setting channel info structure...
Creating raw.info structure...
Reading 0 ... 921599  =      0.000 ...  3599.996 secs...
```

```
[35]:
for filename in os.listdir(directory):
    filepath = os.path.join(directory, filename)
    print(filepath)
```

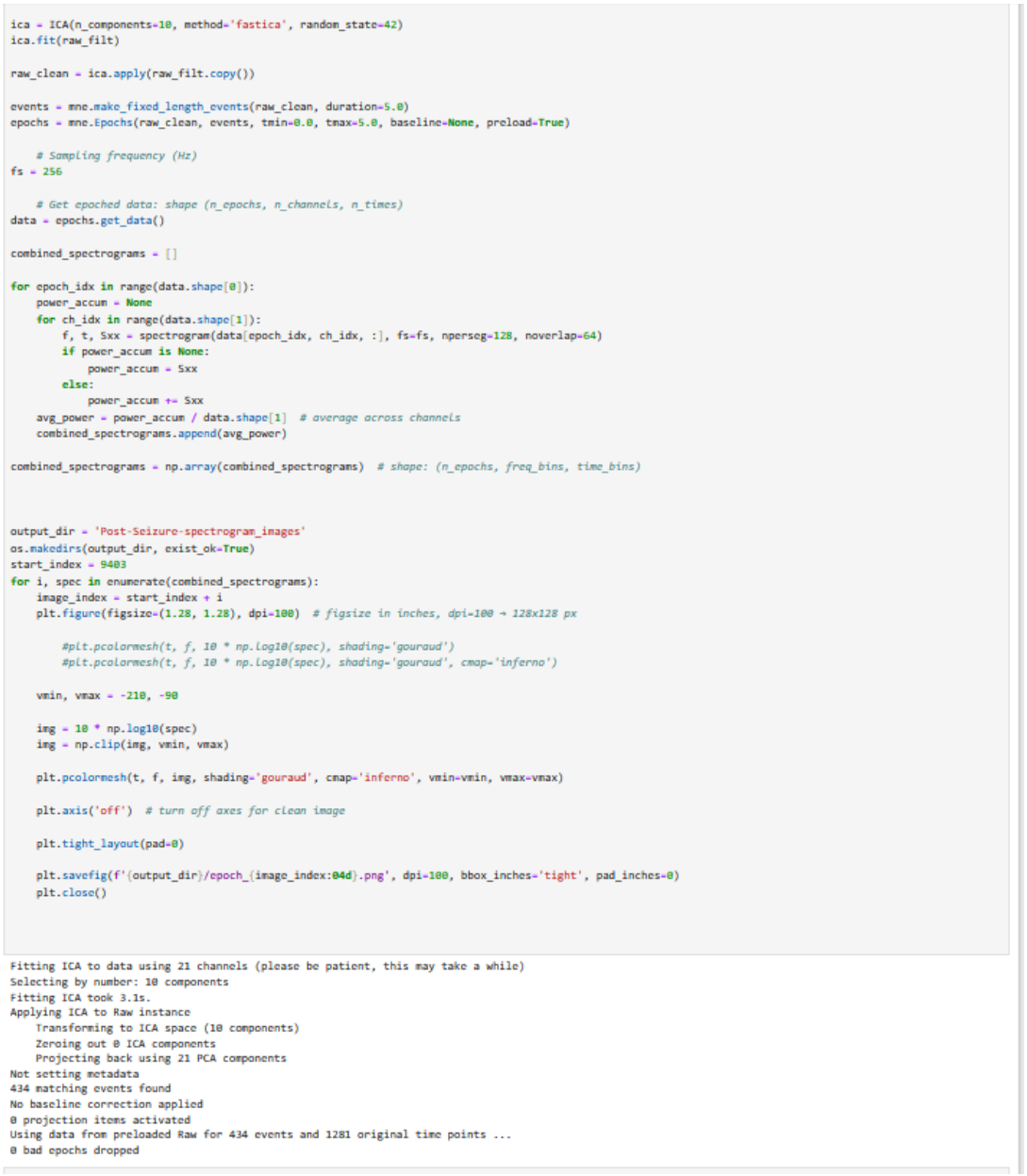
```
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb01_02.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb02_18.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb03_33.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb04_06.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb05_05.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb06_02.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb07_11.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb08_03.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb09_07.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb10_13.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb11_01.edf
C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Nonseizuredataset-over23channels\chb12_19.edf
```

Figure 5: Channel Check and File used

```python
import os
import mne
from mne.preprocessing import ICA
from scipy.signal import spectrogram
import matplotlib.pyplot as plt
import numpy as np
import os

directory = r"C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Post-Seizuredataset\chb12_08.edf"
raw = mne.io.read_raw_edf(directory, preload=True)

channels_to_keep = [
    'FP1-F7', 'F7-T7', 'T7-P7', 'P7-O1', 'FP1-F3', 'F3-C3', 'C3-P3', 'P3-O1',
    'FP2-F4', 'F4-C4', 'C4-P4', 'P4-O2', 'FP2-F8', 'F8-T8', 'T8-P8', 'P8-O2',
    'FZ-CZ', 'CZ-PZ', 'P7-T7', 'T7-FT9', 'FT9-FT10', 'FT10-T8', 'T8-P8'
]
existing_channels = [ch for ch in channels_to_keep if ch in raw.ch_names]
raw.pick_channels(existing_channels)
low_cut = 0.1
hi_cut  = 50

raw_filt = raw.copy().filter(low_cut, hi_cut)
```

```
Extracting EDF parameters from C:\Users\anifz\researchprojectfinal\seizure\newseizuredatasethandpicked\Post-Seizuredataset\chb12_08.edf...
EDF file detected
Setting channel info structure...
Creating raw.info structure...
Reading 0 ... 556543  =      0.000 ...  2173.996 secs...
NOTE: pick_channels() is a legacy function. New code should use inst.pick(...).
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 0.1 - 50 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.10
- Lower transition bandwidth: 0.10 Hz (-6 dB cutoff frequency: 0.05 Hz)
- Upper passband edge: 50.00 Hz
- Upper transition bandwidth: 12.50 Hz (-6 dB cutoff frequency: 56.25 Hz)
- Filter length: 8449 samples (33.004 s)
```

Figure 6: Preprocess Image 1



```
ica = ICA(n_components=10, method='fastica', random_state=42)
ica.fit(raw_filt)

raw_clean = ica.apply(raw_filt.copy())

events = mne.make_fixed_length_events(raw_clean, duration=5.0)
epochs = mne.Epochs(raw_clean, events, tmin=0.0, tmax=5.0, baseline=None, preload=True)

    # Sampling frequency (Hz)
fs = 256

    # Get epoched data: shape (n_epochs, n_channels, n_times)
data = epochs.get_data()

combined_spectrograms = []

for epoch_idx in range(data.shape[0]):
    power_accum = None
    for ch_idx in range(data.shape[1]):
        f, t, Sxx = spectrogram(data[epoch_idx, ch_idx, :], fs=fs, nperseg=128, noverlap=64)
        if power_accum is None:
            power_accum = Sxx
        else:
            power_accum += Sxx
    avg_power = power_accum / data.shape[1]  # average across channels
    combined_spectrograms.append(avg_power)

combined_spectrograms = np.array(combined_spectrograms)  # shape: (n_epochs, freq_bins, time_bins)



output_dir = 'Post-Seizure-spectrogram_images'
os.makedirs(output_dir, exist_ok=True)
start_index = 9403
for i, spec in enumerate(combined_spectrograms):
    image_index = start_index + i
    plt.figure(figsize=(1.28, 1.28), dpi=100)  # figsize in inches, dpi=100 → 128x128 px

        #plt.pcolormesh(t, f, 10 * np.log10(spec), shading='gouraud')
        #plt.pcolormesh(t, f, 10 * np.log10(spec), shading='gouraud', cmap='inferno')

    vmin, vmax = -210, -90

    img = 10 * np.log10(spec)
    img = np.clip(img, vmin, vmax)

    plt.pcolormesh(t, f, img, shading='gouraud', cmap='inferno', vmin=vmin, vmax=vmax)

    plt.axis('off')  # turn off axes for clean image

    plt.tight_layout(pad=0)

    plt.savefig(f'{output_dir}/epoch_{image_index:04d}.png', dpi=100, bbox_inches='tight', pad_inches=0)
    plt.close()
```

```
Fitting ICA to data using 21 channels (please be patient, this may take a while)
Selecting by number: 10 components
Fitting ICA took 3.1s.
Applying ICA to Raw instance
    Transforming to ICA space (10 components)
    Zeroing out 0 ICA components
    Projecting back using 21 PCA components
Not setting metadata
434 matching events found
No baseline correction applied
0 projection items activated
Using data from preloaded Raw for 434 events and 1281 original time points ...
0 bad epochs dropped
```

Figure 7: Preprocess Image 2

The figure 6 and figure 7 , includes the pre processing steps. It follows all the steps mentioned in the report, and stores the spectrogram image in the same folder structure as the code

6 ModelStructure

The `DeepLearningModelCodeandSplit` folder contains the core deep learning models and dataset splitting scripts used in the project:

- `CustomCNNmodel.ipynb`: Implementation of a custom convolutional neural network tailored for EEG data classification.
- `Resnet_50_Model.ipynb`: Notebook implementing the ResNet-50 architecture adapted for the seizure detection task.

- `vgg16_model.ipynb`: Notebook containing the VGG16 model architecture and training routines.
- `splitting-test-train-val.ipynb`: Notebook that handles splitting the dataset into training, validation, and test sets, ensuring proper data partitioning for model evaluation.

These notebooks contain both the model definitions and the data preparation code critical for training and evaluating the deep learning models.

7 Model

7.1 CustomCNNmodel

```
# Paths
DATA_DIR = r"C:\Users\anifz\ResearchFinal28julyfinal\CNNdataset_sp

# Parameters
BATCH_SIZE = 32
IMG_SIZE = (128, 128)
NUM_CLASSES = 3
EPOCHS = 20
SEED = 42

# Load datasets
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    os.path.join(DATA_DIR, "train"),
    label_mode='int',
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=SEED,
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    os.path.join(DATA_DIR, "val"),
    label_mode='int',
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    shuffle=False,
    seed=SEED,
)

test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    os.path.join(DATA_DIR, "test"),
    label_mode='int',
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    shuffle=False,
    seed=SEED,
)
```

Figure 8: readingthemodel

```
)

# Normalize pixel values (0-1)
normalization_layer = layers.Rescaling(1./255)
train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
val_ds = val_ds.map(lambda x, y: (normalization_layer(x), y))
test_ds = test_ds.map(lambda x, y: (normalization_layer(x), y))

# Prefetch to improve speed
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.prefetch(buffer_size=AUTOTUNE)

# Build a CNN model
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=IMG_SIZE + (3,)),
    layers.MaxPooling2D(),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()

# Training the model
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS
)

# Evaluating on the test set
test_loss, test_acc = model.evaluate(test_ds)
print(f"\nTest accuracy: {test_acc:.4f}")
```

Figure 9: CustomModel

```
]: from sklearn.metrics import confusion_matrix, classification_report
import numpy as np

y_pred = model.predict(test_ds)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.concatenate([label for _, label in test_ds])

print(confusion_matrix(y_true, y_pred_classes))
print(classification_report(y_true, y_pred_classes))
```

```
66/66 [==============================] - 5s 82ms/step
[[700   0   0]
 [  0 604  96]
 [  0 116 584]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       700
           1       0.84      0.86      0.85       700
           2       0.86      0.83      0.85       700

    accuracy                           0.90      2100
   macro avg       0.90      0.90      0.90      2100
weighted avg       0.90      0.90      0.90      2100
```

Figure 10: Evaluation

7.2 VGG16

Reading the model is the same as figure8, so it was not added to this model

```
# --- Load base VGG16 ---
base_model = VGG16(weights='imagenet', include_top=False, input_shape=IMG_SIZE + (3,))
base_model.trainable = False  # Freeze for transfer learning

# --- Build model ---
model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation='softmax')
])

# --- Compile and train initial ---
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

print("Initial Training...")
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=INITIAL_EPOCHS
)

#  Fine-tune: Unfreeze last few layers of base model
base_model.trainable = True
for layer in base_model.layers[:-4]:  # Freeze all but last 4 conv layers
    layer.trainable = False

# --- Recompile with lower LR ---
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

print("Fine-Tuning...")
history_fine = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=INITIAL_EPOCHS + FINE_TUNE_EPOCHS,
    initial_epoch=INITIAL_EPOCHS
)

# Evaluation
test_loss, test_acc = model.evaluate(test_ds)
print(f"\nTest Accuracy: {test_acc:.4f}")
```

Figure 11: model

```
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np

y_pred = model.predict(test_ds)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.concatenate([label for _, label in test_ds])

print(confusion_matrix(y_true, y_pred_classes))
print(classification_report(y_true, y_pred_classes))
```

```
66/66 [==============================] - 126s 2s/step
[[700   0   0]
 [  0 612  88]
 [  0 122 578]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       700
           1       0.83      0.87      0.85       700
           2       0.87      0.83      0.85       700

    accuracy                           0.90      2100
   macro avg       0.90      0.90      0.90      2100
weighted avg       0.90      0.90      0.90      2100
```

Figure 12: Evaluation

7.3 Resnet50

Reading the model is the same as figure8, so it was not added to this model

```
[7]: base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG_SIZE + (3,))
base_model.trainable = False  # Freeze base model

# Build model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

print("Training with frozen base model...")
history_initial = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS_INITIAL
)

# Unfreeze some layers of ResNet for fine-tuning
base_model.trainable = True

# fine-tune from this layer onwards
fine_tune_at = 140

for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

# Recompile with lower learning rate for fine-tuning
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

print("Fine-tuning the model...")
history_fine = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS_FINE_TUNE
)

# Evaluate on test set
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.4f}")
```

Figure 13: RestNetmodel

```
: from sklearn.metrics import confusion_matrix, classification_report
import numpy as np

y_pred = model.predict(test_ds)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.concatenate([label for _, label in test_ds])

print(confusion_matrix(y_true, y_pred_classes))
print(classification_report(y_true, y_pred_classes))
```

```
66/66 [==============================] - 139s 2s/step
[[700   0   0]
 [  0 665  35]
 [  0 401 299]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       700
           1       0.62      0.95      0.75       700
           2       0.90      0.43      0.58       700

    accuracy                           0.79      2100
   macro avg       0.84      0.79      0.78      2100
weighted avg       0.84      0.79      0.78      2100
```

Figure 14: ResnetEval

8 Running the Project

Follow these steps to reproduce the experiments and results:

1. Download the CHB-MIT dataset as specified in the configuration report. The dataset is also included in the provided code zip file.
2. Install all required Python packages listed in the software requirements section.
3. Download and extract the entire code base.
4. For the **Non-Seizure** data:
   - There is no need to split the dataset.
   - Navigate to the `Nonseizure-code-and-dataset` folder.
   - Run the `channelCheck.ipynb` notebook to verify the number of EEG channels.
   - Perform preprocessing by running the `preprocess.ipynb` notebook.
   - Visualize the processed data using the `visualize.ipynb` notebook.
5. For the **Seizure** data:
   - First, split the dataset into *pre-seizure* and *post-seizure* segments using the `SeizureSplit.ipynb` notebook in the `Splittingeventandchannelcheck` folder.
   - Then, navigate to the respective folders (`Pre-seizureCodeandDataset` and `Post-seizureCodeandDataset`).
   - Run the preprocessing notebooks for each folder.
   - The preprocessing step generates spectrogram images saved in the corresponding code folders.
   - To avoid class imbalance, select an equal number of files from each class.
6. After preprocessing, move to the `DeepLearningModelCodeandSplit` folder:
   - Update the data path in the notebook to point to the folder containing the spectrogram images.
   - Use the `splitting-test-train-val.ipynb` notebook to randomly split the spectrogram dataset into training, validation, and test sets, minimizing bias.
   - Run the model training notebooks (`CustomCNNmodel.ipynb`, `Resnet_50_Model.ipynb`, `vgg16_model.ipynb`) to train and evaluate the models.