Motor Imagery EEG Classification

This repository contains the code for a project that classifies motor imagery tasks (left vs. right hand movement) from EEG data. It utilizes deep learning models, specifically variants of EEGNet and FEEGNet, to achieve high accuracy in classification. The project is structured for reproducibility, with separate scripts for training and inference, as well as clear configuration and utility modules.

Table of Contents

Project Structure

The repository is organized as follows:

```
.
├── models/
│   └── models.py           # Defines the FEEGNet, FEEGNet_Attention, and
FEEGNet_Attention_Improved architectures
├── src/
│   ├── config.py           # Central configuration for all parameters
│   ├── data_loading.py     # Functions for loading and caching data
│   ├── preprocessing.py    # Preprocessing pipeline for EEG data
│   ├── augmentation.py     # Data augmentation techniques for EEG signals
│   └── training_utils.py   # Helper functions for training, evaluation, and
model storage
├── utils/
│   ├── training_utils.py
├── train.py                # Main script to run the training pipeline
├── inference.py            # Script to run inference on test data
├── FEEGNet_best.keras      # Model checkpoint
└── requirements.txt        # Python dependencies
```

Prerequisites

Before you begin, ensure you have the following installed:

- Python 3.8+
- pip (Python package installer)

Setup and Installation

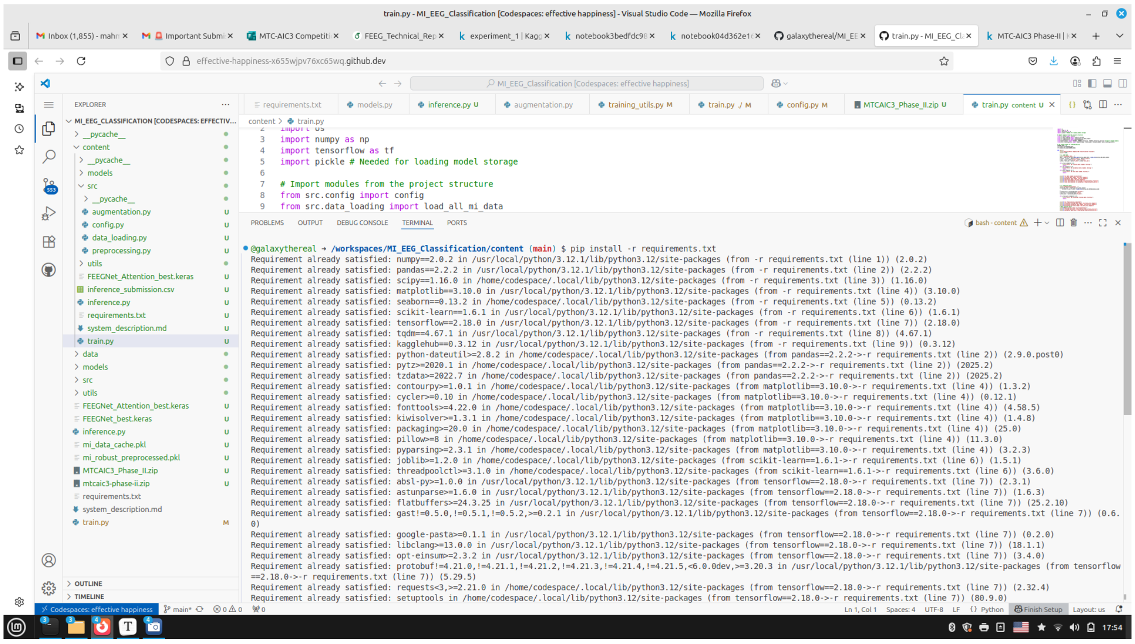
Follow these steps to set up the project environment.

1. **Install Dependencies**

   Install the required Python packages using the `requirements.txt` file.

   ```
   pip install -r requirements.txt
   ```

   Screenshot of Terminal after running pip install:



```
import os
import numpy as np
import tensorflow as tf
import pickle # Needed for loading model storage

# Import modules from the project structure
from src.config import config
from src.data_loading import load_all_mi_data
```

```
@galaxythereal → /workspaces/MI_EEG_Classification/content (main) $ pip install -r requirements.txt
Requirement already satisfied: numpy==2.0.2 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 1)) (2.0.2)
Requirement already satisfied: pandas==2.2.2 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 2)) (2.2.2)
Requirement already satisfied: scipy==1.16.0 in /home/codespace/.local/lib/python3.12/site-packages (from -r requirements.txt (line 3)) (1.16.0)
Requirement already satisfied: matplotlib==3.10.0 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 4)) (3.10.0)
Requirement already satisfied: seaborn==0.13.2 in /home/codespace/.local/lib/python3.12/site-packages (from -r requirements.txt (line 5)) (0.13.2)
Requirement already satisfied: scikit-learn==1.6.1 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 6)) (1.6.1)
Requirement already satisfied: tensorflow==2.18.0 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 7)) (2.18.0)
Requirement already satisfied: tqdm==4.67.1 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 8)) (4.67.1)
Requirement already satisfied: kagglehub==0.3.12 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from -r requirements.txt (line 9)) (0.3.12)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/codespace/.local/lib/python3.12/site-packages (from pandas==2.2.2->-r requirements.txt (line 2)) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /home/codespace/.local/lib/python3.12/site-packages (from pandas==2.2.2->-r requirements.txt (line 2)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /home/codespace/.local/lib/python3.12/site-packages (from pandas==2.2.2->-r requirements.txt (line 2)) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (4.58.5)
Requirement already satisfied: kiwisolver>=1.3.1 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (25.0)
Requirement already satisfied: pillow>=8 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /home/codespace/.local/lib/python3.12/site-packages (from matplotlib==3.10.0->-r requirements.txt (line 4)) (3.2.3)
Requirement already satisfied: joblib>=1.2.0 in /home/codespace/.local/lib/python3.12/site-packages (from scikit-learn==1.6.1->-r requirements.txt (line 6)) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/codespace/.local/lib/python3.12/site-packages (from scikit-learn==1.6.1->-r requirements.txt (line 6)) (3.6.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (2.3.1)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (0.6.
0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (3.4.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/python/3.12.1/lib/python3.12/site-packages (from tensorflow
==2.18.0->-r requirements.txt (line 7)) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in /home/codespace/.local/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (2.32.4)
Requirement already satisfied: setuptools in /home/codespace/.local/lib/python3.12/site-packages (from tensorflow==2.18.0->-r requirements.txt (line 7)) (80.9.0)
```

3. **Set Environment Variable for Data Path**

   The scripts require an environment variable `MTCAIC3_DATA_PATH` to be set to the root directory of the dataset.

   On macOS/Linux:

   ```
   export MTCAIC3_DATA_PATH=/path/to/your/mtcaic3-phase-ii
   ```

   On Windows (Command Prompt):

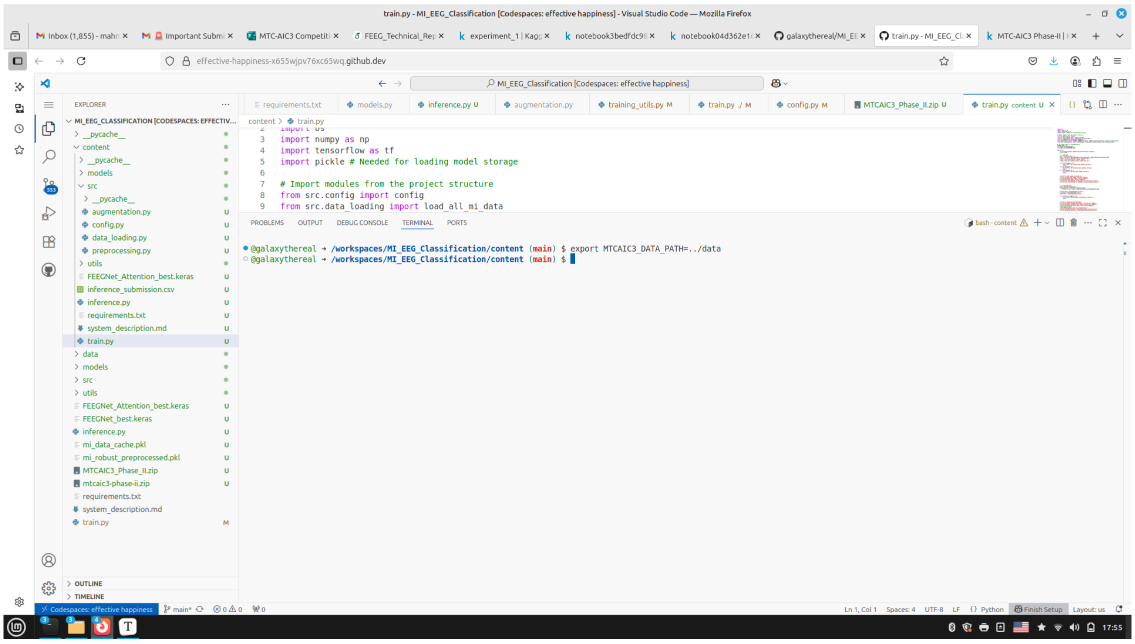
   ```
   set MTCAIC3_DATA_PATH=C:\path\to\your\mtcaic3-phase-ii
   ```

   On Windows (PowerShell):

   ```
   $env:MTCAIC3_DATA_PATH="C:\path\to\your\mtcaic3-phase-ii"
   ```

   To make this permanent, add it to your shell's profile file (e.g., `.bashrc`, `.zshrc`) or your system's environment variables.

   Screenshot of setting the environment variable (macOS/Linux):



```
import os
import numpy as np
import tensorflow as tf
import pickle # Needed for loading model storage

# Import modules from the project structure
from src.config import config
from src.data_loading import load_all_mi_data
```

```
@galaxythereal → /workspaces/MI_EEG_Classification/content (main) $ export MTCAIC3_DATA_PATH=../data
@galaxythereal → /workspaces/MI_EEG_Classification/content (main) $ 
```

Usage

This project has two main workflows: training the models from scratch and running inference with a pre-trained model.

Training the Models

The `train.py` script handles the entire training pipeline, including data loading, preprocessing, augmentation, model training, and evaluation.

1. **Run the Training Script**

   Execute the following command in your terminal:

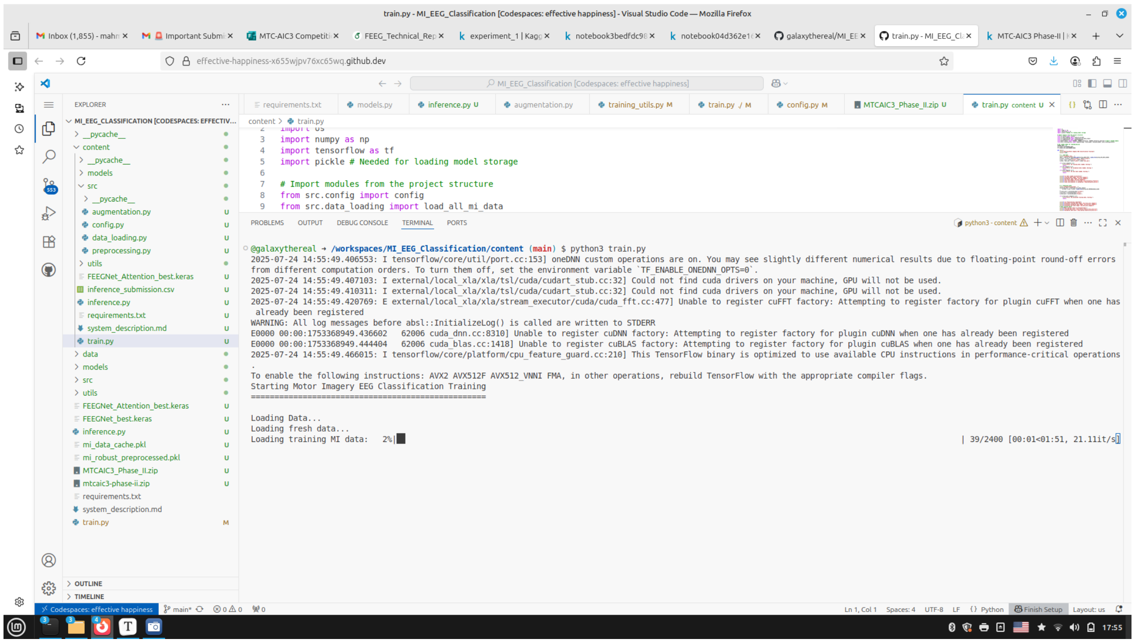
   ```
   python train.py
   ```

2. **Training Process**

   The script will perform the following steps:

   - Load and cache the raw MI data.
   - Preprocess and cache the data.
   - Apply data augmentation.
   - Train three different models: `FEEGNet`, `FEEGNet_Attention`, and `FEEGNet_Attention_Improved`.
   - Save the best weights for each model (`.keras` files).
   - Generate and save training history plots.
   - Create an ensemble prediction from the top-performing models.
   - Save individual and ensemble submission files.

   Screenshot of the training process starting:


```python
import os
import numpy as np
import tensorflow as tf
import pickle # Needed for loading model storage

# Import modules from the project structure
from src.config import config
from src.data_loading import load_all_mi_data
```

```
@galaxythereal → /workspaces/MI_EEG_Classification/content (main) $ python3 train.py
2025-07-24 14:55:49.406553: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors
from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-07-24 14:55:49.407103: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2025-07-24 14:55:49.410311: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2025-07-24 14:55:49.420769: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has
 already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1753368949.436602   62006 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1753368949.444404   62006 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2025-07-24 14:55:49.466015: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations
.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Starting Motor Imagery EEG Classification Training
==================================================

Loading Data...
Loading fresh data...
Loading training MI data:   2%|█                                                                                       | 39/2400 [00:01<01:51, 21.11it/s]
```

Running Inference

The `inference.py` script is designed to be a standalone module for generating predictions on the test set using a pre-trained model.

1. **Ensure Model File is Present**

   Make sure the `EEGNet_Attention_best.keras` file (or the model you wish to use) is in the root directory of the project. This file is generated during the training process.

2. **Run the Inference Script**

   Execute the inference script from your terminal:

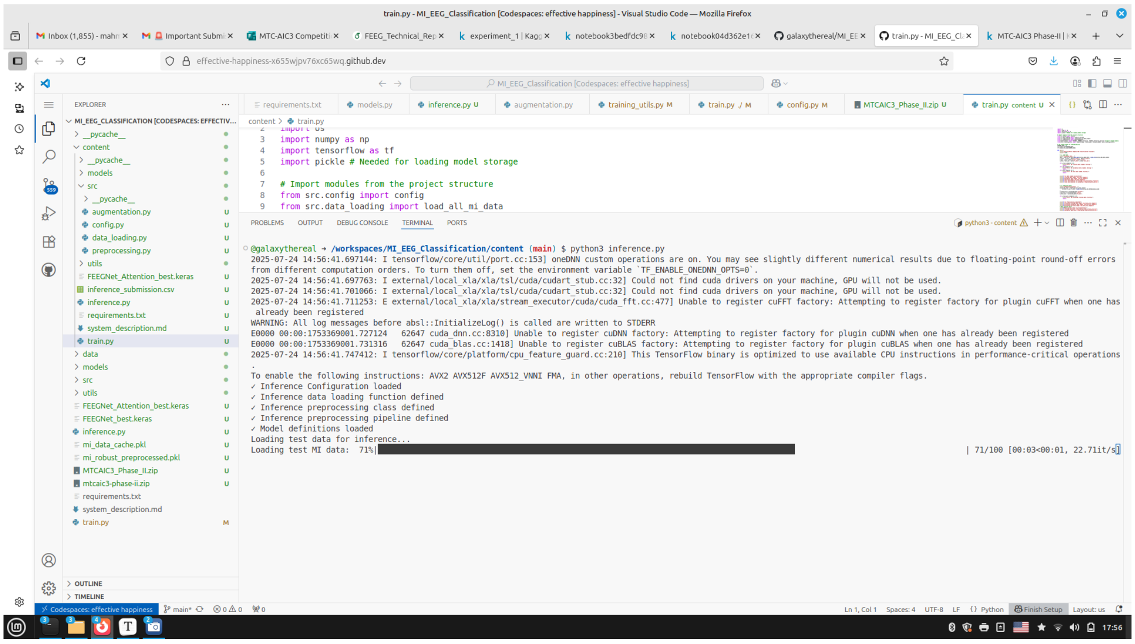
   ```
   python inference.py
   ```

3. **Inference Process**

   The script will:

   - Load the test data.
   - Apply the same preprocessing steps used during training.
   - Load the specified pre-trained model (`EEGNet_Attention_best.keras`).
   - Generate predictions on the preprocessed test data.
   - Save the predictions to a `inference_submission.csv` file.

   Screenshot of the inference process:



```python
import os
import numpy as np
import tensorflow as tf
import pickle # Needed for loading model storage

# Import modules from the project structure
from src.config import config
from src.data_loading import load_all_mi_data
```

```
@galaxythereal → /workspaces/MI_EEG_Classification/content (main) $ python3 inference.py
2025-07-24 14:56:41.697144: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors
from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-07-24 14:56:41.697763: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2025-07-24 14:56:41.701066: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2025-07-24 14:56:41.711253: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has
 already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1753369001.727124   62647 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1753369001.731316   62647 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2025-07-24 14:56:41.747412: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations
.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
✓ Inference Configuration loaded
✓ Inference data loading function defined
✓ Inference preprocessing class defined
✓ Inference preprocessing pipeline defined
✓ Model definitions loaded
Loading test data for inference...
Loading test MI data:  71%|█████████████████████████████████████████████                  | 71/100 [00:03<00:01, 22.71it/s]
```

Model Architecture

The project explores three deep learning architectures based on the FEEGNet (formerly EEGNet) paper.

1. **FEEGNet**: A compact convolutional neural network designed for EEG-based brain-computer interfaces. It includes temporal, depthwise, and separable convolutions to effectively capture spatial and temporal features from EEG signals.
2. **FEEGNet with Attention**: This model enhances FEEGNet by incorporating a Squeeze-and-Excitation (SE) attention block. The SE block adaptively recalibrates channel-wise feature responses, allowing the model to focus on more informative channels.
3. **Improved FEEGNet with Attention**: This version builds upon the attention model with several improvements:
    - Increased filter sizes (`F1=16`, `F2=32`) to capture more complex features.
    - Higher dropout rate (`0.6`) for better regularization.
    - L2 regularization on convolutional and dense layers to prevent overfitting.
    - An additional separable convolutional block to increase model depth.

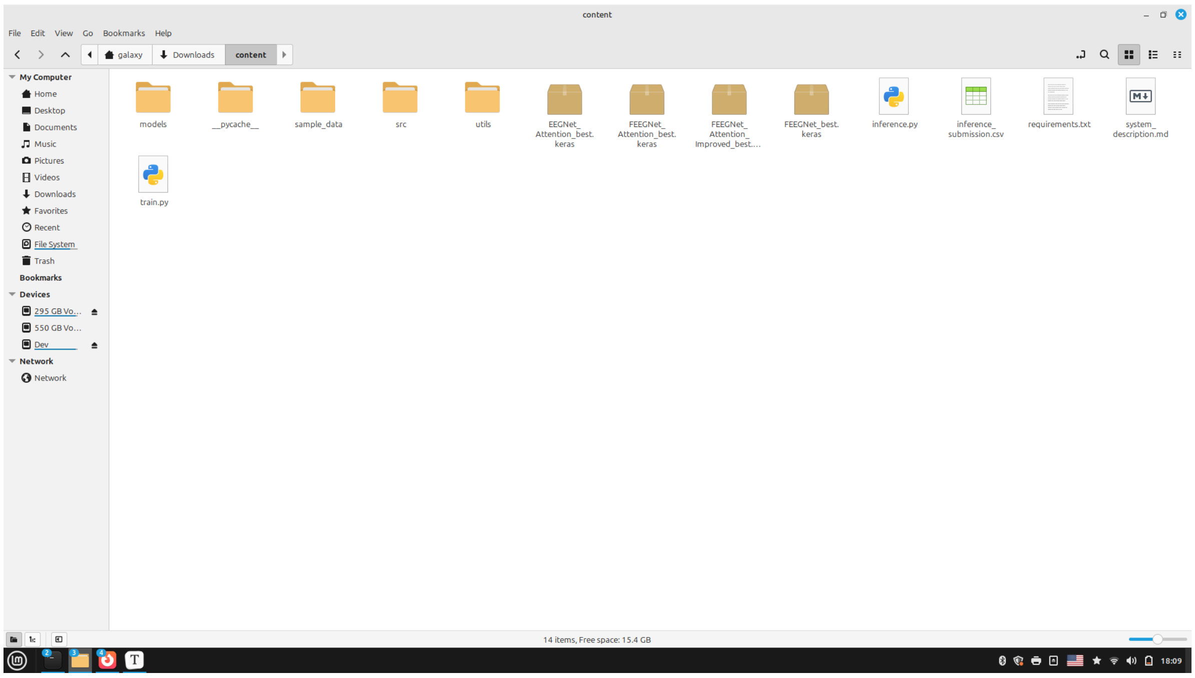
Expected Outputs

After running the scripts, the following files will be generated in the project's root directory:

- **During Training (`train.py`):**
    - `mi_data_cache.pkl`: Cached raw data.
    - `mi_robust_preprocessed.pkl`: Cached preprocessed data.
    - `model_storage.pkl`: A file containing accuracies, histories, and predictions for all trained models.
    - `FEEGNet_best.keras`, `FEEGNet_Attention_best.keras`, `FEEGNet_Attention_Improved_best.keras`: The saved weights of the best performing models.
    - `FEEGNet_training_history.png`, `FEEGNet_Attention_training_history.png`, `FEEGNet_Attention_Improved_training_history.png`: Plots of training/validation accuracy and loss.
    - `ensemble_submission.csv`: The final submission file from the ensembled model.

  - `FEEGNet_submission.csv`, `FEEGNet_Attention_submission.csv`, `FEEGNet_Attention_Improved_submission.csv`: Submission files from individual models.
- **During Inference (`inference.py`):**
  - `inference_submission.csv`: A submission file with predictions from the loaded model.

Screenshot of the generated files after running `train.py`:



Thanks