SSVEP Ensemble-Based Classification

This repository contains the code for a project that classifies Steady-State Visually Evoked Potential (SSVEP) tasks from EEG data. It utilizes a robust machine learning pipeline featuring advanced feature extraction and an ensemble model combining XGBoost, SVM, and Logistic Regression to achieve high classification accuracy. The project is structured for clarity and reproducibility, with separate scripts for training and inference.

Table of Contents

Project Structure

The repository is organized as follows:

```
.
├── checkpoints/
│   └── (Generated model artifacts will be stored here)
├── models/
│   └── models.py           # Defines the ensemble model
├── src/
│   ├── config.py           # Central configuration for all parameters
│   ├── data_loading.py     # Functions for loading trial data
│   ├── preprocessing.py    # Preprocessing pipeline for EEG signals
│   ├── augmentation.py     # Data augmentation techniques for EEG signals
│   ├── feature_extraction.py # Comprehensive feature extraction functions
│   └── template_generation.py # Functions to generate class templates
├── utils/
│   └── training_utils.py   # Helper functions for CV, model saving/loading
├── train.py                # Main script to run the training pipeline
├── inference.py            # Script to run inference on test data
└── requirements.txt        # Python dependencies
```

Prerequisites

Before you begin, ensure you have the following installed:

- Python 3.8+
- pip (Python package installer)

Setup and Installation

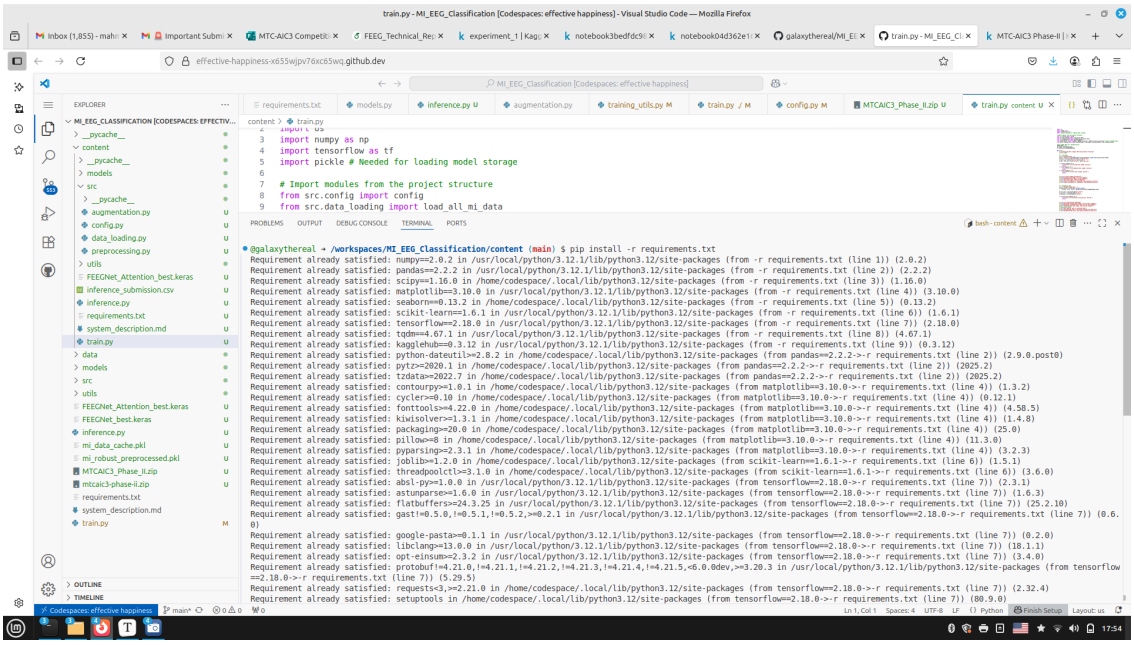
Follow these steps to set up the project environment.

1. **Install Dependencies**

   Install the required Python packages using the `requirements.txt` file. The scripts are designed to be robust and will attempt to install `pyriemann` automatically if it is missing.

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

   Screenshot of Terminal after running pip install:

2. **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):

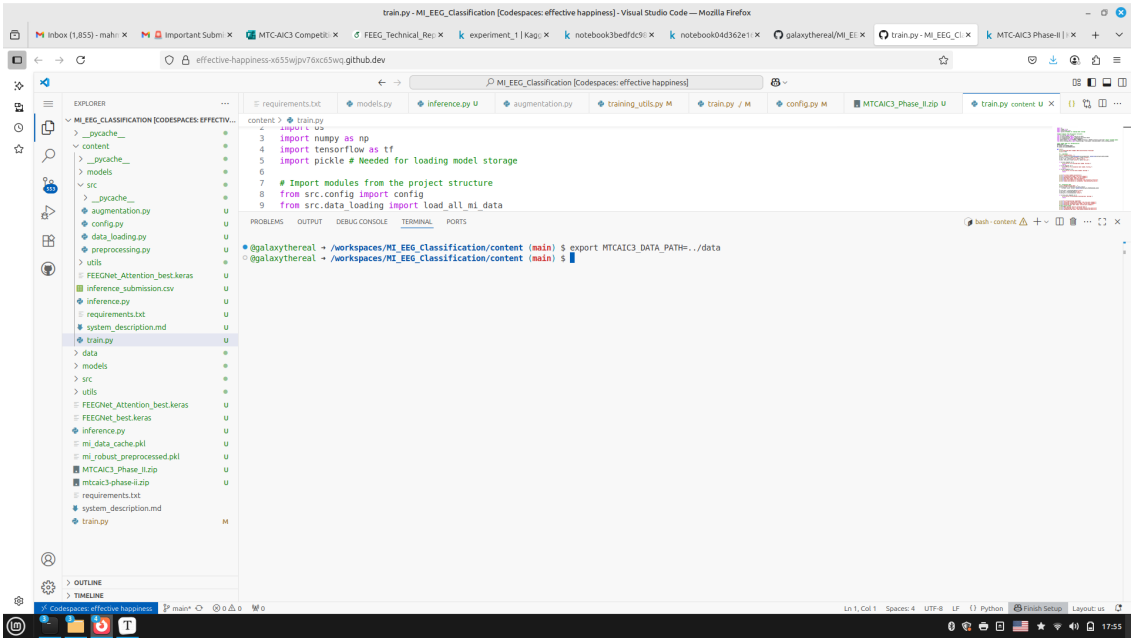
   ```
   set MTCAIC3_DATA_PATH=C:\path\to\your\mtcaic3-phase-ii
   ```

   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):



Usage

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

Training the Model

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

1. **Run the Training Script**

   Execute the following command in your terminal:

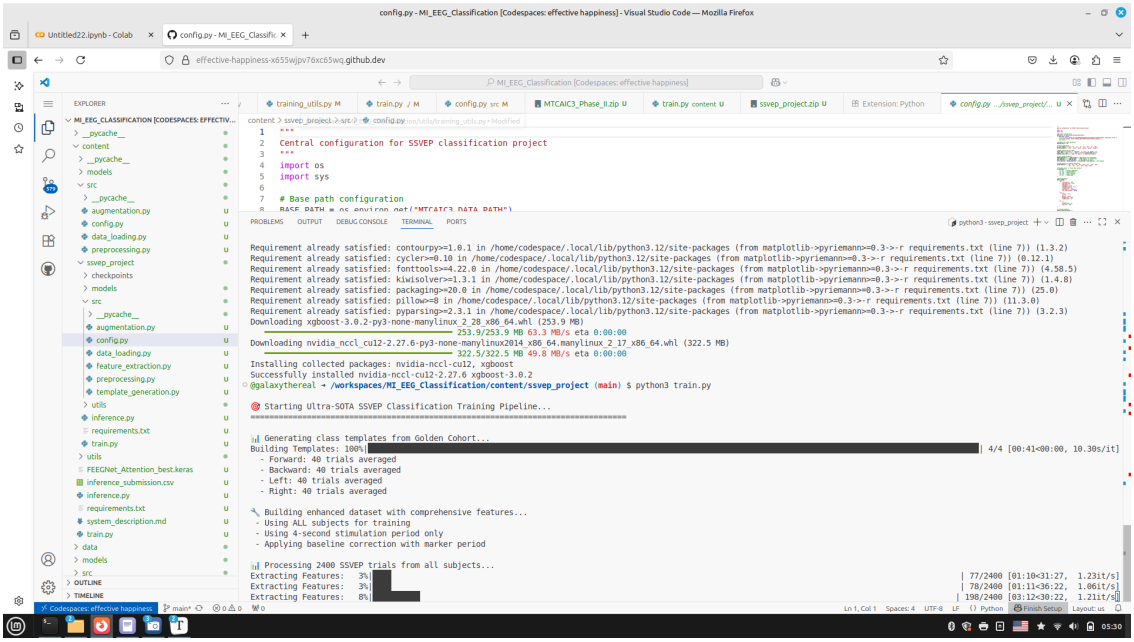
   ```
   python train.py
   ```

2. **Training Process**

   The script will perform the following steps:

   - Generate class-specific templates from the "Golden Cohort" subjects.
   - Build an enhanced dataset by processing all trials, applying data augmentation, and extracting comprehensive features.
   - Perform 5-fold cross-validation on an ensemble model to evaluate its performance.
   - Train the final ensemble model on the entire dataset.
   - Save the trained model, data scaler, and templates to the `checkpoints/` directory.

   Screenshot of the training process starting:



Running Inference

The `inference.py` script is a standalone module for generating predictions on the test set using the artifacts created during training.

1. **Ensure Model Artifacts are Present**

   Make sure the `ensemble_model.joblib`, `scaler.joblib`, and `templates.joblib` files are present in the `checkpoints/` directory. These files are generated by the `train.py` script.

2. **Run the Inference Script**

   Execute the inference script from your terminal:

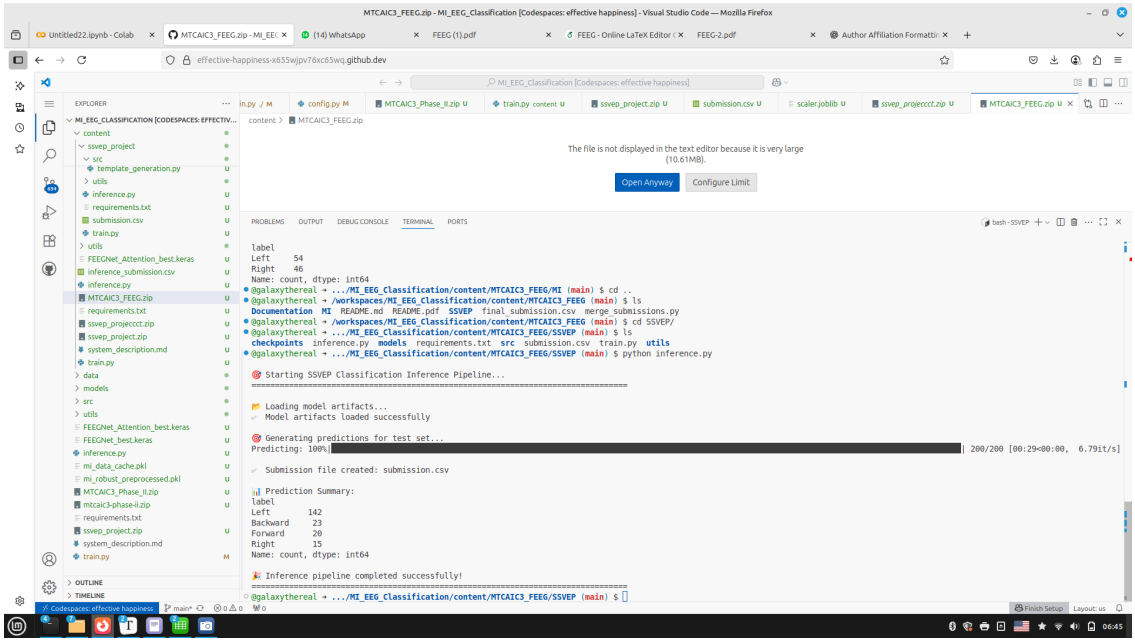
   ```
   python inference.py
   ```

3. **Inference Process**

   The script will:

   - Load the model, scaler, and templates from the `checkpoints/` directory.
   - Load the test data metadata.
   - For each test trial, it will load the EEG data, apply the same preprocessing and feature extraction steps used in training.
   - Use the trained model to generate a prediction.
   - Save the final predictions to a `submission.csv` file.

   Screenshot of the inference process:



Model and Feature Engineering

The project's success relies on a combination of sophisticated feature engineering and a powerful ensemble model.

1. **Ensemble Model**: The classifier is a `VotingClassifier` that combines the predictions of three distinct models using soft voting (averaging probabilities) for a more robust and accurate result. The ensemble consists of:
    - **XGBoost Classifier**: A highly efficient gradient-boosting algorithm.
    - **Support Vector Classifier (SVC)**: A powerful classifier using an RBF kernel.
    - **Logistic Regression**: A reliable linear model.
2. **Comprehensive Feature Extraction**: Instead of feeding raw signals to a model, the pipeline extracts a rich set of features designed to capture the unique characteristics of SSVEP responses. Key feature sets include:
    - **Task-Related Component Analysis (TRCA)**: Features measuring the correlation between a trial and class-specific templates.
    - **Multi-band CCA Analysis**: Canonical Correlation Analysis performed across multiple frequency bands to capture harmonic relationships.
    - **Riemannian Features**: Features extracted from the covariance matrices of the EEG signals, which are powerful for capturing spatial patterns.
    - **Spectral Features**: Power spectral density (PSD) in specific frequency bands and spectral entropy.
    - **Statistical Features**: Standard statistical measures from both EEG and motion sensor data.

Expected Outputs

After running the scripts, the following files will be generated:

- **During Training (`train.py`):**
    - `checkpoints/ensemble_model.joblib`: The saved final ensemble model.
    - `checkpoints/scaler.joblib`: The saved `StandardScaler` object.
    - `checkpoints/templates.joblib`: The saved class templates for feature extraction.
- **During Inference (`inference.py`):**

- `submission.csv`: A submission file containing the `id` and predicted `label` for each test trial.

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