Task 4 : time domain analysis

Features

1. **EEG Signal Visualization**:
   - Plots individual channel signals in subplots.
   - Combines all channels in a single plot for a comparative view.
2. **Metric Computation**:
   - **Mean**: Average value of signal amplitudes.
   - **Zero Crossing Rate**: Frequency of zero crossings in the signal.
   - **Range**: Difference between the maximum and minimum values of the signal.
   - **Energy**: Sum of squared amplitudes of the signal.
   - **Root Mean Square (RMS)**: RMS value of the signal.
   - **Variance**: Measure of signal variation.

Directory Structure

The script assumes the .npy files are organized in subdirectories according to the EEG data's classification. Below are examples of directory paths:

- Complex_Partial_Seizures/
- Electrographic_Seizures/
- Normal/
- Video_detected_Seizures_with_no_visual_change_over_EEG/

Each directory contains .npy files, where each file represents a single EEG data sample.

How to Use

1. **Mount Google Drive**:
   - Ensure the EEG data is stored in Google Drive.
   - The script mounts Google Drive using:

python
CopyEdit
```
from google.colab import drive
drive.mount('/content/drive')
```

2. **Update Directory Paths**:
   - Modify the dir variable to match your data's directory structure (for eg. Complex , video_detected, normal).
3. **Run the Script**:
   - The script processes files one by one and outputs visualizations and metrics.
4. **Output**:
   - Signal plots for each EEG sample.
   - Metrics summary displayed as a pandas DataFrame.

Code Workflow

1. **Load Data**:
   - The script loads .npy files containing EEG samples using numpy.load().
2. **Plot Signals**:
   - The plot_eeg_signals() function visualizes the EEG channels.
3. **Compute Metrics**:
   - The compute_metrics() function calculates time-domain metrics for each channel.
4. **Display Results**:
   - Visualizations and metrics are displayed interactively.

Task 5：Frequency-Domain Analysis of EEG Signals

This task (code) provides a Python implementation for analyzing EEG data using three frequency-domain techniques: Fourier Transform, Wavelet Decomposition, and Spectrogram Generation. The code is designed to process .npy files containing EEG data and visualize the frequency-based features of the signals for all EEG channels.

File Description Code Functionality

1. Load EEG Data:
   - Reads .npy files containing EEG signals. Each file should have a shape of (19, 500) representing 19 channels and 500 samples per channel.
2. Fourier Transform:
   - Computes the Fast Fourier Transform (FFT) to extract the magnitude spectrum of the signal.
   - Visualizes the frequency content for each channel.
3. Wavelet Decomposition:
   - Performs wavelet decomposition using the db4 wavelet up to 4 levels.
   - Visualizes the decomposition coefficients for each channel and compares the original signal with its approximation coefficients.
4. Spectrogram Generation:
   - Generates spectrograms to show the time-frequency representation of the signal.
   - Displays the power spectrum for each channel over time.
   -

Visualizations The code generates plots for:

1. Magnitude spectrum from the Fourier Transform.
2. Wavelet decomposition coefficients for different levels.
3. Time-frequency spectrograms.

Task 6 : EEG Signal Classification with SVM

This project classifies EEG signals into four categories using Fourier and Zero Crossing Rate (ZCR) features with an SVM model.

Pipeline

1. **Data Loading**:
    - Extracts Fourier (first 10 frequencies) and ZCR features from .npy files.
    - Supports four categories:
        - Complex Partial Seizures (label=0)
        - Electrographic Seizures (label=1)
        - Normal (label=2)
        - Video-detected Seizures with no EEG changes (label=3).
2. **Feature Normalization**:
    - Standardizes features using StandardScaler.
3. **Model Training**:
    - Trains an SVM with an RBF kernel (C=0.1, gamma=0.1).
4. **Validation & Testing**:
    - Evaluates using classification report, ROC AUC, and balanced accuracy.
    - Saves test predictions to test_predictions.csv.

Usage

1. **Dataset Structure**:
2. EEG_Data/

- train_data/
- validation_data/
- test_data/

3. **Run the Code**:
    - Execute in Python with numpy, pandas, and scikit-learn installed.
4. **Output**:
    - Validation metrics and test predictions in test_predictions.csv.

TASK 7 & 8: EEG Feature Extraction

Key Features of the Pipeline

1. **EEG Feature Extraction**:
   - Extracts time-domain, frequency-domain, time-frequency, and entropy-based features from EEG signals.
2. **Dimensionality Reduction**:
   - Selects relevant features using methods like mutual information and Principal Component Analysis (PCA).
3. **Baseline Model Building**:
   - Creates an initial predictive model using deep learning (CNN, LSTM) or machine learning algorithms (e.g., SVM).
4. **Hyperparameter Optimization**:
   - Tunes model parameters using advanced techniques like Bayesian optimization.
5. **Evaluation Metrics**:
   - Assesses model performance using classification metrics such as ROC AUC and Balanced Accuracy.

Folder Structure

Organize the EEG data in the following directory format before running the script:

scss

CopyEdit

```
EEG_Data/
  train_data/
    Complex_Partial_Seizures/
    Electrographic_Seizures/
    Normal/
    Video_detected_Seizures_with_no_visual_change_over_EEG/
  validation_data/
    Complex_Partial_Seizures/
    Electrographic_Seizures/
    Normal/
    Video_detected_Seizures_with_no_visual_change_over_EEG/
  test_data/
    (contains .npy files for testing)
```

Each subfolder corresponds to a specific class of EEG data, where each .npy file contains the EEG signals for a sample.

Feature Extraction

Extracted Features

1. **Time-Domain Features**:
   - Mean, Variance, Skewness, Kurtosis, Zero Crossings, Energy, RMS.
2. **Frequency-Domain Features**:
   - Fourier Transform: Mean, Variance, Entropy of FFT components.
   - Power Spectral Density (PSD): Mean, Variance.
3. **Time-Frequency Features**:
   - Wavelet Transform: Mean and Energy of wavelet coefficients.
4. **Entropy-Based Features**:
   - Shannon Entropy of the signal histogram.

How It Works

1. Extracts features for each EEG channel.
2. Aggregates extracted features across all channels.
3. Labels the samples based on folder names.
4. Saves extracted features into CSV files:
    - train_all_features.csv
    - validation_all_features.csv

Baseline Model Building

Workflow

1. **Load Data**:
    - Load the CSV files containing extracted features and labels.
2. **Split Data**:
    - Train-validation split for model training and evaluation.
3. **Normalize Features**:
    - Standardize features using StandardScaler for consistency.
4. **Train Model**:
    - Use Support Vector Machine (SVM) or deep learning models (e.g., CNN, LSTM) to create a baseline.
5. **Evaluate Model**:
    - Use metrics such as ROC AUC, Balanced Accuracy, and a classification report.

Hyperparameter Optimization

Key Steps

1. Use **Bayesian Optimization** to find optimal hyperparameters:
    - Learning rate, batch size, kernel type, etc.
2. Evaluate optimized hyperparameters on the validation set to ensure performance improvement.

Results and Evaluation

Metrics

- **Classification Report**: Includes Precision, Recall, and F1-Score for each class.
- **Balanced Accuracy**: Accounts for class imbalance.
- **ROC AUC Score**: Evaluates model's ability to distinguish between classes.

Output

- Test predictions are saved as test_predictions.csv, containing:
    - Filename.
    - Predicted label for each test sample.

Requirements

Python Libraries

Install the required dependencies using pip:

bash

CopyEdit

```
pip install numpy pandas scipy pywt scikit-learn
```

Dependencies

- numpy: For numerical computations and handling .npy files.
- pandas: For data manipulation and saving results to CSV.
- scipy: For statistical computations.
- pywt: For Wavelet Transform.
- scikit-learn: For model training, evaluation, and feature scaling.

Usage Instructions

1. **Prepare EEG Data**:
   - Organize EEG signals in the specified folder structure.
2. **Run the Feature Extraction Script**:
   - Execute the feature extraction script to generate train_all_features.csv and validation_all_features.csv.
3. **Train and Evaluate Model**:
   - Use the training and validation CSV files to train a machine learning or deep learning model.
   - Tune hyperparameters and evaluate performance on the validation set.
4. **Test Model**:
   - Use the test data to generate predictions and save them to test_predictions.csv.

References

This project incorporates techniques from the following papers:

1. **Feature Extraction**:
   - "A Comprehensive Review on Feature Extraction Techniques in EEG Signal Analysis."
   - "Mutual Information-Based Feature Selection for Epileptic Seizure Detection Using EEG."
2. **Deep Learning**:
   - "Deep Learning Approaches for Epileptic Seizure Prediction Using EEG Signals."
3. **Hyperparameter Tuning**:
   - "Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks."

Overview

This analysis focuses on identifying the key EEG channels that contribute most significantly to the prediction of different classes in a classification task. Using explainable AI (xAI) techniques like SHAP (SHapley Additive exPlanations) and saliency maps, channels 2, 3, and 4 were identified as the top contributors. The evaluation further explores the impact of removing these channels on model performance.

Key Steps

1. Identifying Important Channels

- **Techniques Used:**
  - SHAP values: Quantified the contribution of each EEG channel to the model's predictions.
  - Saliency Maps: Highlighted the most critical areas in the input that influenced the model's decisions.
- **Findings:**
  - Channels 2, 3, and 4 consistently showed higher importance scores across multiple runs.
  - These channels were critical to differentiating between classes effectively.

2. Evaluating the Impact of Key Channels

To understand the role of these channels in model performance:

a. Masking/Removing Channels

- Channels 2, 3, and 4 were either masked or removed entirely from the dataset.

b. Re-training and Validation

- The model was retrained on the modified dataset without the top three channels.
- Performance was re-evaluated using the validation set.

3. Observations

Key Impacts:

- **Accuracy Drop:**
  - Removal of the top three channels caused a significant decline in overall model accuracy.
- **Decline in Metrics:**
  - Classification metrics such as ROC AUC and Balanced Accuracy showed notable reductions.

Class-Specific Impacts:

- Certain classes that relied heavily on features from channels 2, 3, and 4 experienced higher misclassification rates.
- The absence of these channels disrupted the model's ability to effectively differentiate between classes.

Conclusion
The analysis highlights the critical role of EEG channels 2, 3, and 4 in the classification task. Their consistent importance across runs and the significant drop in performance upon their removal demonstrate their strong contribution to class separability and overall model performance. These insights can guide future model refinement and data collection strategies.

TASK 9A : EEG Wavelet Denoising and Signal Plotting

This tasl (code) performs wavelet denoising on noisy EEG signals, evaluates the denoised signals using PSNR, and visualizes the noisy, denoised, and ground truth signals for comparison.

Requirements

- numpy
- matplotlib
- pywt (PyWavelets)
- sklearn
- random

Install dependencies:
pip install numpy matplotlib pywt scikit-learn

Folder Structure
Ensure your EEG data is organized in the following structure:
EEG_Data/

```
noisy_train_data/
    Complex_Partial_Seizures/
    Electrographic_Seizures/
    Normal/
    Video_detected_Seizures_with_no_visual_change_over_EEG/
train_data/
    Complex_Partial_Seizures/
    Electrographic_Seizures/
    Normal/
    Video_detected_Seizures_with_no_visual_change_over_EEG/
denoised_data/
    Complex_Partial_Seizures/
    Electrographic_Seizures/
    Normal/
    Video_detected_Seizures_with_no_visual_change_over_EEG/
```

Usage

1. Organize your EEG data in the specified folder structure.
2. Run the script to denoise signals and compute PSNR.
3. The script will randomly select and plot up to 3 files from each class, displaying the noisy, denoised, and ground truth signals for visual inspection.
4. Denoised signals are saved in the denoised_data folder.

Plot Details

For each selected file, the following three plots are shown:

- **Noisy Signal** (in red)
- **Denoised Signal** (in blue)
- **Ground Truth Signal** (in green)

This allows for a visual comparison between the noisy input, the processed output, and the expected ground truth.

Task 9B : from feature selection and model evaluation

EEG Seizure Detection Using Machine Learning: Feature Selection and CNN-LSTM Evaluation

Overview

This project involves detecting EEG seizure events using machine learning. The workflow is broken down into three main parts:

1. **Feature Selection**: Extract features from the EEG data and select the most relevant ones for training.
2. **Model Training**: Use a CNN-LSTM model for classification of EEG data into different seizure categories.
3. **Model Evaluation**: Evaluate the trained model using metrics such as classification report, ROC-AUC score, and confusion matrix.

Workflow

Step 1: Feature Selection

This step extracts relevant features from the EEG dataset and selects the most important ones.

1.1 Extracting Features

- **Time-domain features**: Includes mean, variance, skewness, kurtosis, zero crossings, energy, and RMS for each channel.

- **Frequency-domain features**: Includes FFT-based mean, variance, and entropy; Power Spectral Density (PSD) mean and variance.
- **Wavelet features**: Calculated using a Discrete Wavelet Transform (DWT) to capture time-frequency information at different levels.
- **Entropy features**: Based on Shannon entropy to measure signal unpredictability.

1.2 Selecting Uncorrelated Features

The uncorrelated features are selected by dropping highly correlated or irrelevant features. These selected features are saved for model training.

```
features_to_drop = { 'channel_18_energy', 'channel_4_energy', 'channel_1_wavelet_level_4_energy', ... }

# Drop the features
X_uncorrelated = X.drop(columns=features_to_drop)
```

The selected features are then combined with the label and saved in train_selected_features.csv:

```
selected_features_df = X_uncorrelated
selected_features_df["label"] = y
selected_features_df.to_csv("train_selected_features.csv", index=False)
```

Step 2: Model Training with CNN-LSTM

After the feature selection, we use the CNN-LSTM model for classification. The model is trained using the processed training data, with features normalized and reshaped for input into the model.

```
X_train = (X_train - X_train.mean()) / X_train.std()  # Normalization
X_train = np.expand_dims(X_train.values, axis=2)  # Reshape: (samples, time-steps, 1)
```

The labels are converted into one-hot encoding format:

```
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
```

The model, optimized via Bayesian Optimization, is loaded and trained on the processed features.

Step 3: Model Evaluation

Once the model is trained, it is evaluated using the validation dataset. The evaluation steps include:

- **Classification Report**: Provides precision, recall, f1-score, and support for each class.
- **ROC-AUC Score**: Measures the performance of the model using the One-vs-Rest approach for multi-class classification.
- **Confusion Matrix**: Displays the confusion matrix to visualize true vs predicted class distribution.

```
# Classification Report
print("Classification Report:")
print(classification_report(y_true, y_pred))

# ROC-AUC Score
roc_auc = roc_auc_score(y_val, best_model.predict(X_val), multi_class="ovr")
print(f"ROC-AUC Score: {roc_auc}")

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=range(num_classes),
yticklabels=range(num_classes))
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

Final Output

After running the workflow, the following outputs are produced:

- train_selected_features.csv: The dataset with selected features and labels.
- Model evaluation metrics:
  - Classification Report
  - ROC-AUC Score
  - Confusion Matrix (visualized using seaborn)

TASK 10.A Synthetic Data Generation Using Variational Autoencoder (VAE) for EEG Data

Features:

- **VAE Model**: A Variational Autoencoder is used to learn the distribution of the input EEG data and generate new, synthetic samples.
- **Data Generation**: The model is trained on real EEG data, and synthetic samples are generated using the learned latent space representation.
- **Similarity Metrics**: The synthetic data is evaluated using multiple metrics such as MSE, Cosine Similarity, and Pearson Correlation to ensure its quality.

Directory Structure:

- train_data/: Contains the real EEG data stored as .npy files.
- synthetic_data/: This folder stores the generated synthetic EEG data as .npy files, categorized by the folder names in train_data.

Steps:

1. **Prepare your EEG data**: Ensure your real EEG data is in .npy format and organized into folders representing different classes.
2. **Run the script**: The script will load the data, train a VAE model on it, and generate synthetic data.
3. **Evaluate synthetic data**: Similarity metrics (MSE, Cosine Similarity, Pearson Correlation) are printed for each synthetic dataset generated.

Requirements:

- tensorflow
- numpy
- scikit-learn
- scipy

Install the dependencies with:
pip install tensorflow numpy scikit-learn scipy

TASK 10. EEG Seizure Classification using CNN-LSTM with Feature Extraction

Dataset
The dataset used in this project consists of synthetic EEG data from different seizure types. Each EEG sample corresponds to one of the following labels:

- Complex_Partial_Seizures: Label 0
- Electrographic_Seizures: Label 1
- Normal: Label 2
- Video_detected_Seizures_with_no_visual_change_over_EEG: Label 3

The EEG data is stored as .npy files, and each file contains data from multiple EEG channels over time.

Feature Extraction
The pipeline extracts various features from the EEG data, including:

- **Time-domain features**: Mean, variance, skewness, kurtosis, zero crossings, energy, RMS.
- **Frequency-domain features**: FFT mean, FFT variance, FFT entropy, PSD mean, PSD variance.
- **Wavelet Transform features**: Decomposition coefficients at multiple levels, with mean and energy calculations.
- **Entropy features**: Shannon entropy calculated from the histogram of each channel's data.

Feature Selection
After extracting features, unimportant or redundant features are removed based on a predefined list (features_to_drop). The remaining features are then saved for model training.
Data Preparation and Normalization

The dataset is split into training and validation sets. The features are normalized by subtracting the mean and dividing by the standard deviation for each feature.

Model
A **CNN-LSTM model** is used for classification. This model combines Convolutional Neural Networks (CNN) for feature extraction and Long Short-Term Memory (LSTM) networks for sequential learning. The model is optimized using Bayesian optimization to select the best hyperparameters.

Model Evaluation
After training, the model's performance is evaluated on the validation set using the following metrics:

- **Classification Report**: Precision, recall, F1-score for each class.
- **ROC-AUC Score**: AUC score for multi-class classification.

- **Confusion Matrix**: Heatmap representation of model predictions versus true labels.

Usage

1. **Install Dependencies**:
2. pip install numpy pandas scipy scikit-learn tensorflow seaborn matplotlib pywt
3. **Run Feature Extraction**:
   - Place your EEG data in the synthetic_data folder.
   - Run the feature extraction scripts to extract time-domain, frequency-domain, and wavelet features from the EEG data.
4. **Model Training**:
   - Ensure the best model is saved as optimized_cnn_lstm_model.h5.
   - Train the model using the prepared dataset, including the feature selection process.
5. **Model Evaluation**:
   - Run the evaluation script to check the classification performance on the validation set.