Emotion Detection Using EEG: Autoencoder + CNN-BiLSTM Model

1. Introduction

This research focuses on building an emotion detection system using EEG (Electroencephalography) signals. We utilize a hybrid deep learning model combining an autoencoder for dimensionality reduction and feature extraction, followed by a CNN-BiLSTM model for emotion classification. The autoencoder compresses high-dimensional EEG data, while the CNN-BiLSTM leverages both spatial (CNN) and temporal (LSTM) characteristics to detect emotions effectively.

2. Data Preprocessing

The input data consists of EEG brainwave signals with high dimensionality. To make the data manageable and extract meaningful features, we preprocess the data as follows:

- **Load the Dataset:** The EEG signals are loaded from CSV files where the features represent different channels of EEG data.
- **Normalization:** The EEG features are normalized using a `StandardScaler` to ensure all data is scaled properly.
- **Reshaping:** The 1D EEG signals are reshaped into 2D grids to prepare them for convolutional layers.

3. Autoencoder Model

The autoencoder is used to reduce the high-dimensional EEG data into a lower-dimensional representation. The model is composed of an encoder and a decoder:

- **Encoder:** The encoder uses several convolutional layers to compress the data into a smaller feature representation.
- **Decoder:** The decoder reconstructs the data from the encoded representation, ensuring the key features are preserved.

```
def build_autoencoder(input_shape):
    input_layer = Input(shape=input_shape)

    # Encoder
    x = Conv2D(16, (3, 3), padding='same')(input_layer)
    x = ReLU()(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), padding='same')(x)
```

```
    x = Conv2D(8, (3, 3), padding='same')(x)
    x = ReLU()(x)
    x = BatchNormalization()(x)
    encoded = MaxPooling2D((2, 2), padding='same')(x)

    # Decoder
    x = Conv2D(8, (3, 3), padding='same')(encoded)
    x = ReLU()(x)
    x = BatchNormalization()(x)
    x = UpSampling2D((2, 2))(x)

    x = Conv2D(16, (3, 3), padding='same')(x)
    x = ReLU()(x)
    x = BatchNormalization()(x)
    x = UpSampling2D((2, 2))(x)

    decoded = Conv2D(input_shape[2], (3, 3), activation='sigmoid',
padding='same')(x)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer=Adam(learning_rate=1e-3),
loss='mse')

    return autoencoder
```

4. CNN-BiLSTM Model

The core emotion detection model uses a combination of Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) layers. This hybrid model captures both the spatial and temporal dynamics of EEG signals:

- **CNN Layers:** The CNN layers extract spatial features from the EEG signals.
- **BiLSTM Layers:** The BiLSTM layers capture temporal dependencies and emotion patterns in EEG data over time.

```
def build_cnn_bilstm(input_shape):
    input_layer = Input(shape=input_shape)

    # CNN Layers for Spatial Feature Extraction
    x = TimeDistributed(Conv2D(32, (3, 3), activation='relu',
padding='same'))(input_layer)
    x = TimeDistributed(BatchNormalization())(x)
    x = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(x)

    # BiLSTM Layers for Temporal Dependencies
    x = Bidirectional(LSTM(64, return_sequences=False))(x)
    x = BatchNormalization()(x)
    x = Dropout(0.4)(x)

    # Fully Connected Layer
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.4)(x)
    output_layer = Dense(3, activation='softmax')(x)

    model = Model(input_layer, output_layer)
```

```
    model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

5. Training and Evaluation

The training process for both the autoencoder and CNN-BiLSTM models involves the following steps:

- **Autoencoder Training:** The autoencoder is trained to reconstruct EEG data with minimized loss using the `Mean Squared Error (MSE)` loss function.
- **Feature Extraction:** The trained autoencoder extracts key features from the EEG data.
- **CNN-BiLSTM Training:** The extracted features are used to train the CNN-BiLSTM model for emotion classification. Early stopping and learning rate scheduling are employed to optimize training and prevent overfitting.

```
cnn_bilstm_model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2,
                     class_weight=class_weights_dict,
callbacks=[early_stopping, checkpoint])
```

6. Results

The CNN-BiLSTM model achieves excellent results in classifying emotions based on EEG signals, with a test accuracy of 98%. The model generalizes well to unseen data and shows balanced performance across all classes as confirmed by the confusion matrix and classification report.

7. Conclusion

The Autoencoder + CNN-BiLSTM model is highly effective for emotion detection using EEG signals. It successfully reduces the dimensionality of EEG data and captures both spatial and temporal features, making it ideal for this task.

Project Documentation for Research Paper: EEG-Based Emotion Classification Using CNN-BiLSTM

Introduction

The objective of this project is to develop a deep learning model that classifies emotions using EEG (Electroencephalography) signals. EEG signals are widely used in brain-computer interfaces to decode brain states, and emotion detection is a key application for fields like mental health, human-computer interaction, and entertainment.

Our approach combines feature extraction using an **Autoencoder** followed by a **CNN-BiLSTM** (Convolutional Neural Network + Bidirectional Long Short-Term Memory) model for emotion classification. The model successfully achieves a **98% accuracy** on the dataset provided.

Methodology

1. Dataset:

The EEG dataset used for the project contains brainwave signals corresponding to three emotion classes: *positive, neutral*, and *negative*.

Each data point represents a series of EEG signals captured across different sensors over time. Initially, the dataset was structured in CSV format, where features (EEG signals) were stored in columns, and the last column represented the emotion class labels.

2. Data Preprocessing:

Feature Scaling: All features (EEG signals) were scaled using a `StandardScaler`. This step is crucial for CNNs and LSTMs, which are sensitive to the input data scale.

```
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

3. Autoencoder for Feature Extraction:

Purpose: The autoencoder serves as an unsupervised feature extraction tool, compressing the EEG signals while retaining the most critical features.

Architecture:

- The encoder consists of convolutional layers with a reduced number of filters (32, 16) to compress the data.
- The decoder reconstructs the original data from the compressed feature set.

The output of the encoder was later used as input to the CNN-BiLSTM model.

```
autoencoder.fit(features_reshaped, features_reshaped, epochs=50,
batch_size=32)
```

Result: The autoencoder successfully reduced the dimensionality of the input data, creating a feature set of shape `(2132, 26, 26, 16)`.

4. CNN-BiLSTM for Emotion Classification:

CNN Layer:

- **Purpose**: Extract spatial features from EEG data (i.e., how different brain regions interact).
- **TimeDistributed Layers**: CNN layers are wrapped in TimeDistributed layers to apply convolution across time steps (for EEG signals).

BiLSTM Layer:

- **Purpose**: Capture temporal dependencies in EEG signals. The Bidirectional LSTM can capture both forward and backward relationships in the time-series data, making it more effective than a unidirectional LSTM.

Fully Connected Layer:

- Dense layers are added with ReLU activations and regularization (L2) to prevent overfitting.

Softmax Output: The model ends with a softmax activation to classify EEG signals into one of the three emotional states.

```
model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy', metrics=['accuracy'])
```

5. Training and Evaluation:

Training: The CNN-BiLSTM model was trained for 50 epochs with a batch size of 32. A learning rate of 0.001 was used with the Adam optimizer. Early stopping was applied to prevent overfitting.

Validation: Validation accuracy during training consistently reached over **97%**, indicating strong generalization on unseen data.

6. Results:

The final model achieved **98% accuracy** on the test set. A detailed classification report showed strong performance across all three emotion classes (*positive*, *neutral*, *negative*), with high precision, recall, and F1-scores.

Confusion Matrix: The confusion matrix showed minimal misclassifications, confirming the robustness of the model.

Visualization

We performed two key visualizations during the project:

1. **Data Visualization (Before and After Preprocessing)**:
    - **Before Preprocessing**: EEG signals in raw format had varying scales and ranges, making it unsuitable for neural networks.
    - **After Preprocessing**: The scaled and reshaped features were visualized, showing normalized and structured data suitable for the CNN-BiLSTM architecture.
2. **Training Performance Visualization**:
    - **Training and Validation Accuracy**: A plot showed consistent improvement in both training and validation accuracy, reaching 98% by epoch 12.
    - **Loss Curves**: Both training and validation loss decreased steadily, demonstrating good model convergence without overfitting.

Conclusion

The model's 98% accuracy on EEG-based emotion detection showcases the power of combining feature extraction (Autoencoder) with deep learning architectures (CNN-BiLSTM). The model can effectively classify emotions into *positive*, *negative*, and *neutral* categories using raw EEG data.