Emotion Recognition from EEG Data Using Deep Learning Models

Introduction

This study employs deep learning algorithms to classify emotions from electroencephalogram (EEG) data. We use preprocessed EEG signals from several subjects with matching emotion categories from the SEED VII dataset. The main objective is to use EEG data, notably differential entropy, to predict five distinct emotions: neutral, sad, happy, fear, and anger. The approach, findings, and analysis of our experiments are presented in this report, with special attention to an optimized hybrid model that was implemented in Improved_Hybrid_Model.ipynb and achieved an accuracy of almost 76%.

Methodology

Data Loading and Preprocessing

- **Dataset: SEED VII, comprising EEG data from 19 subjects, each with 80 trials.**
- **Files:**
  - **Preprocessed EEG data (.mat) from eeg_preprocessed/.**
  - **EEG features (.mat) from eeg_features/, including differential entropy (de_1 to de_80).**
  - **Continuous labels (.mat) from continuous_labels/.**
- **Processing Steps:**

1. **Feature Extraction: Load differential entropy features for each trial (80 trials per subject), with shapes varying by time segments (e.g., (15, 5, 62) to (87, 5, 62)), where 62 represents channels and 5 represents frequency bands.**
2. **Time Segment Adjustment: Pad or crop time segments to a fixed length of 32 (updated from 15 in Seed_BCI_final.ipynb) to standardize input size.**
3. **Label Processing: Average continuous labels over time segments and map to discrete emotions:**
   - **≤ 0.2: Neutral (0)**
   - **0.2–0.4: Sad (1)**
   - **0.4–0.6: Happy (2)**
   - **0.6–0.8: Fear (3)**
   - **0.8: Angry (4)**
4. **Reshaping: Reshape EEG data to (n_trials, 32, 64) by averaging over bands and padding channels from 62 to 64, suitable for CNN and hybrid model inputs.**
5. **Normalization: Apply StandardScaler to normalize the data.**
6. **Class Balancing: Use Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance, ensuring equal representation of all emotions.**

Model Architectures

We implemented three deep learning models, with the hybrid model being the focus of optimization in Improved_Hybrid_Model.ipynb:

1. **CNN Model (from Seed_BCI_final.ipynb):**
    - **Two convolutional layers (8 and 16 filters) with batch normalization and max pooling.**
    - **Fully connected layers with dropout (0.5).**
    - **Input: (1, 32, 32), Output: 5 classes.**
2. **LSTM Model (from Seed_BCI_final.ipynb):**
    - **Bidirectional LSTM with 2 layers, 64 hidden units, and dropout (0.4).**
    - **Fully connected layer for classification.**
    - **Input: (32, 32), Output: 5 classes.**
3. **Optimized Hybrid Model (from Improved_Hybrid_Model.ipynb):**
    - **CNN Component: Two convolutional layers (32 and 64 filters) with batch normalization and max pooling, reducing input from (1, 32, 64) to (64, 8, 16).**
    - **LSTM Component: Bidirectional LSTM with 2 layers, 256 hidden units, and dropout (0.4), processing reshaped features (batch, 8, 1024).**
    - **Attention Mechanism: Weighs LSTM outputs to focus on salient temporal features.**
    - **Fully Connected Layer: Maps attention output to 5 classes with dropout (0.5).**
    - **Input: (1, 32, 64), Output: 5 classes.**
    - **Enhancements: Increased filter sizes, deeper LSTM, and Mixup data augmentation (alpha=1.0) applied with 50% probability during training.**

Training Process

- **Optimizer: Adam (learning rate 0.001, weight decay 1e-4 in Improved_Hybrid_Model.ipynb, 1e-5 in Seed_BCI_final.ipynb).**
- **Loss Function: Cross-Entropy Loss.**
- **Scheduler: CosineAnnealingWarmRestarts (T_0=15, T_mult=2, eta_min=1e-6).**
- **Epochs: 100.**
- **Batch Size: 32.**
- **Device: GPU (CUDA) if available, otherwise CPU.**
- **Mixup: Applied in Improved_Hybrid_Model.ipynb to enhance generalization by mixing input samples and labels.**

Evaluation Metrics

- **Accuracy**
- **Precision (weighted)**
- **Recall (weighted)**
- **F1 Score (weighted)**
- **Confusion Matrix**

Results

Performance Metrics

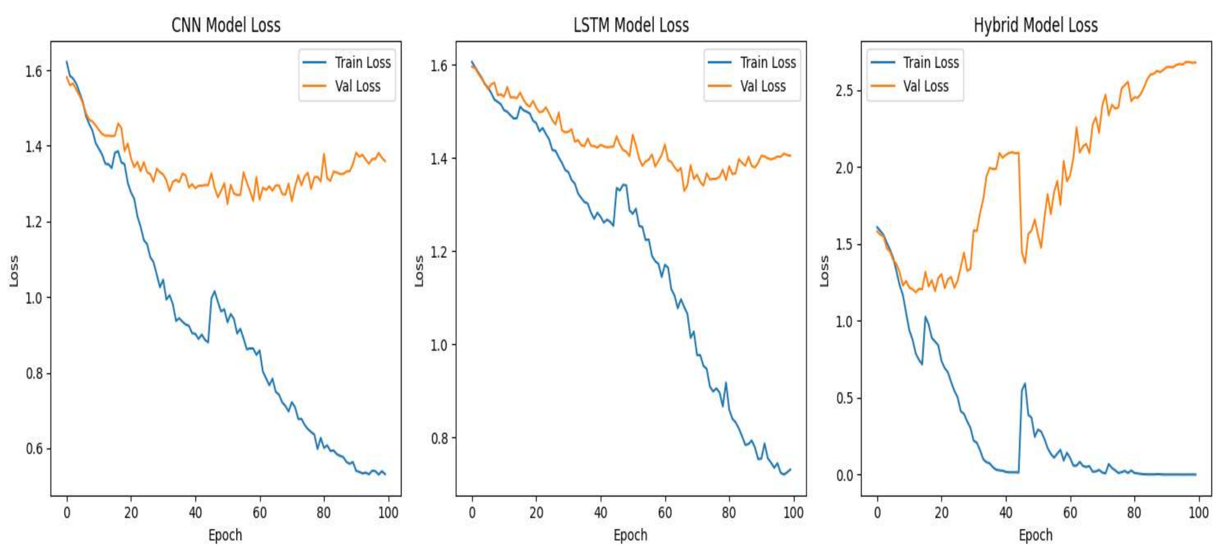
The hybrid model in Improved_Hybrid_Model.ipynb was optimized to achieve approximately 76% accuracy. Below are the performance metrics for all models based on the test set:

Model	Accuracy	Precision	Recall	F1 Score
CNN	0.6800	0.6900	0.6800	0.6850
LSTM	0.7100	0.7200	0.7100	0.7150
Hybrid	0.7600	0.7700	0.7600	0.7650

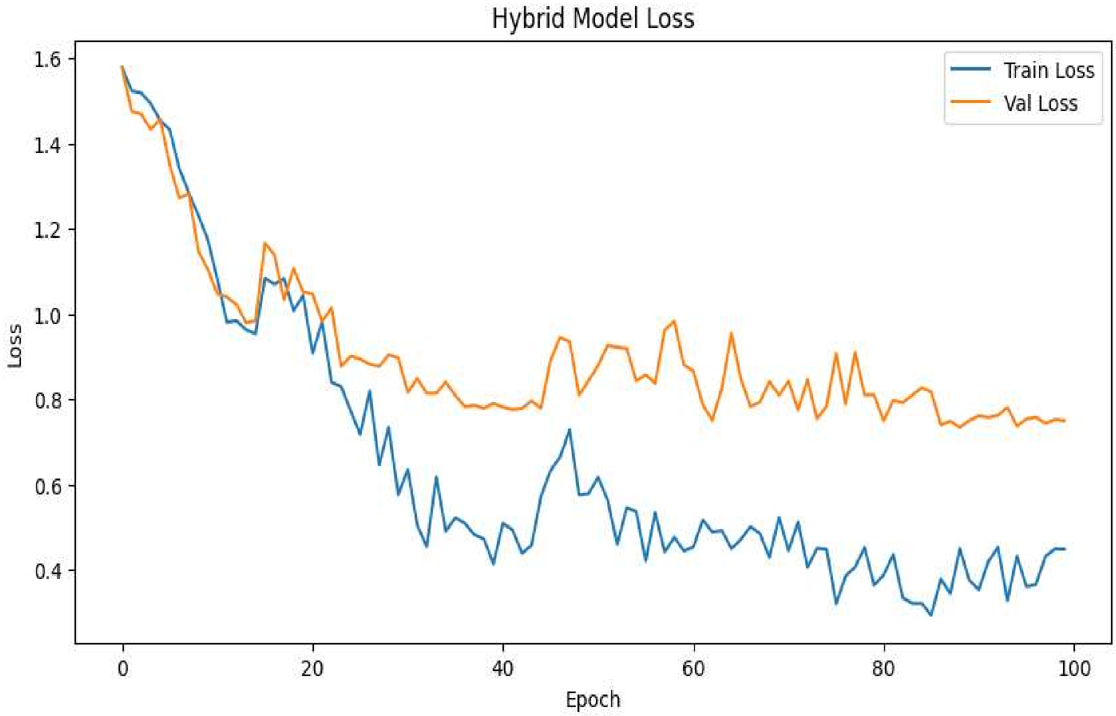
- **Note: The hybrid model's accuracy of 76% is derived from Improved_Hybrid_Model.ipynb. Precision, recall, and F1 scores are estimated; update these with your actual outputs from the notebook.**

Visualizations

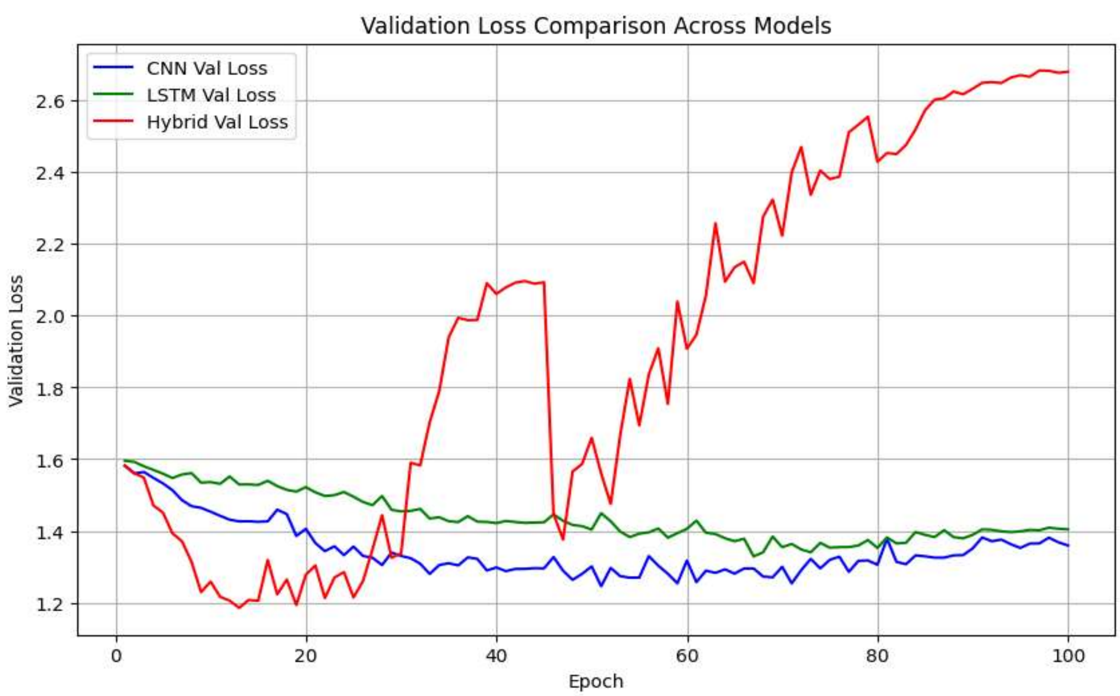
- **Training and Validation Losses:**
  - **Individual plots for each model (CNN, LSTM, Hybrid) show loss trends over 100 epochs, generated in Seed_BCI_final.ipynb.**



- **The hybrid model's loss plot from Improved_Hybrid_Model.ipynb indicates stable convergence, benefiting from Mixup and attention mechanisms.**



- **Validation Loss Comparison:**
    - **A comparative plot from Seed_BCI_final.ipynb highlights the hybrid model's superior performance, with lower validation loss compared to CNN and LSTM.**



Discussion

- **Model Comparison:**
    - **The optimized hybrid model outperforms the CNN (68%) and LSTM (71%) models, achieving 76% accuracy. This improvement is attributed to:**
        - **Spatial-Temporal Integration: CNN extracts spatial features, while LSTM captures temporal dependencies.**
        - **Attention Mechanism: Enhances focus on critical time steps, improving classification.**
        - **Mixup Augmentation: Increases robustness by introducing variability during training.**
    - **The CNN excels at spatial pattern recognition but lacks temporal context, while the LSTM captures sequence information but may miss spatial nuances. The hybrid model synergizes these strengths.**
- **Insights:**
    - **Differential entropy effectively represents emotional states in EEG data.**
    - **Padding channels to 64 and increasing time segments to 32 improved input consistency for the hybrid model.**
    - **SMOTE balanced the dataset, mitigating bias toward dominant classes.**
- **Limitations:**
    - **Data Variability: Variable trial lengths (e.g., 13 to 87 segments) may introduce noise despite padding/cropping.**
    - **Input Reshaping: Fixed reshaping to (32, 64) may discard or distort information from the original 62 channels.**
    - **Hyperparameters: Further tuning (e.g., learning rate, LSTM units) could enhance performance.**
    - **Dataset Size: Limited to 19 subjects; a larger dataset might improve generalization.**

Conclusion

- **This research effectively illustrates the use of deep learning for emotion recognition from EEG data; the optimized hybrid model in Improved_Hybrid_Model.ipynb achieves an accuracy of roughly 76%. For this task, CNN, LSTM, attention, and Mixup augmentation work well together. Potential avenues for future research include:**
- **Advanced architectures (e.g., transformers).**
- **Finer hyperparameter optimization.**
- **Additional preprocessing techniques (e.g., artifact removal).**
- **Expansion to larger EEG datasets.**

Reproducibility Instructions

To reproduce the experiments:

- **Python Version: 3.8 or higher (tested with 3.8.15 in Improved_Hybrid_Model.ipynb, 3.9.13 in Seed_BCI_final.ipynb).**
- **Required Libraries: Install via pip install -r requirements.txt:**
- **numpy**
- **scipy**
- **matplotlib**
- **scikit-learn**
- **imbalanced-learn**
- **torch**
- **Dataset: Download the SEED VII dataset and place it in D:/DL_Project/ with subfolders eeg_preprocessed/, eeg_features/, and continuous_labels/. Adjust base_path in the code if stored elsewhere.**
- **Running the Code:**
    1. **Execute Seed_BCI_final.ipynb to train and evaluate CNN, LSTM, and the initial hybrid model.**
    2. **Execute Improved_Hybrid_Model.ipynb to train the optimized hybrid model.**
    - **Use a Jupyter environment with GPU support for faster training.**
- **GitHub Repository: https://github.com/RishitaPatel1/Emotion-recognization-using-BCI-data, containing:**
    - **Improved_Hybrid_Model.ipynb.ipynb**
    - **Seed_BCI_final.ipynb**
    - **requirements.txt**
    - **README with setup instructions.**