Project Seminar

SleepGCN-Transformer: A Hybrid Graph Convolutional and Transformer Network for Sleep Stage Classification

Tanmay Rathod

Under the guidance: Dr. Santosh Kumar Satapathy

Assistant Professor, Department of ICT, PDPU Gandhinagar

May 21, 2025

EEG-Based Sleep Stage Classification Using Machine Learning

Problem Statement

Problem Statement

Automated Sleep Stage Classification Using EEG Signals: A Machine Learning Approach with Feature-Based Modeling and K-Fold Validation

Introduction : EEG Signal Channels from Sleep-EDF Dataset

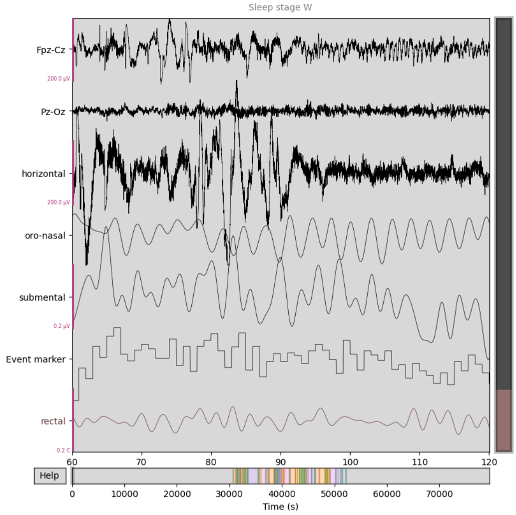


Figure: All EEG signals in the Sleep-EDF dataset

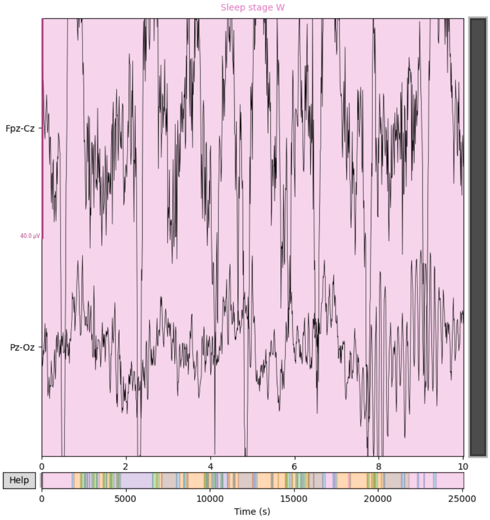


Figure: Filtered EEG – Fpz-Cz and Pz-Oz channels

Methodology : Model Architecture and Evaluation Strategy

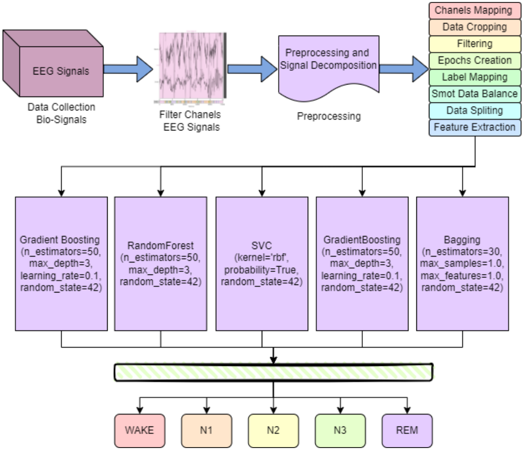


Figure: Proposed deep learning model architecture

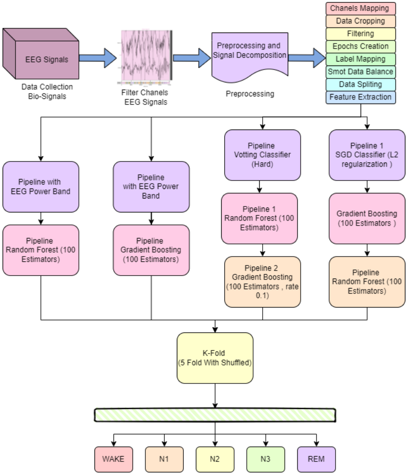


Figure: 5-Fold Cross-Validation strategy

Machine Learning Results Summary

Random Forest

Accuracy: 0.764
Precision: 0.777
Sensitivity: 0.770
F1-Score: 0.766

Ensemble Learning

Accuracy: 0.831
Precision: 0.830
Sensitivity: 0.820
F1-Score: 0.819

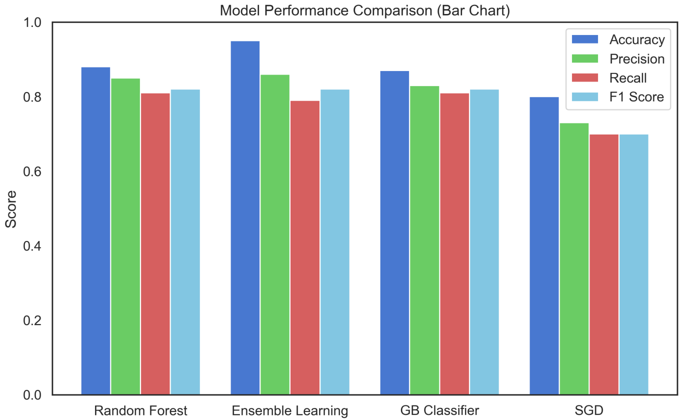
Bagging Classifier

Accuracy: 0.702
Precision: 0.687
Sensitivity: 0.689
F1-Score: 0.687

Gradient Boosting

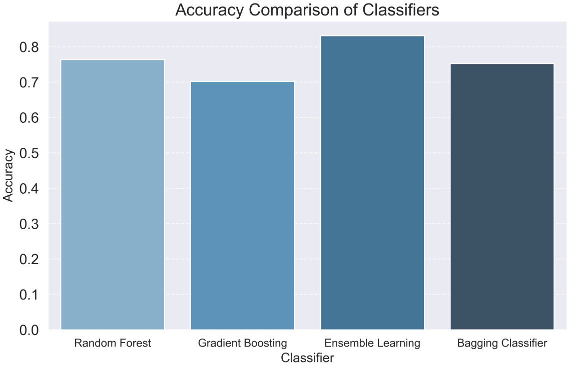
Accuracy: 0.753
Precision: 0.737
Sensitivity: 0.746
F1-Score: 0.734

Machine Learning Accuracy Results



Accuracy, Precision, Recall, F1 Score Comparison

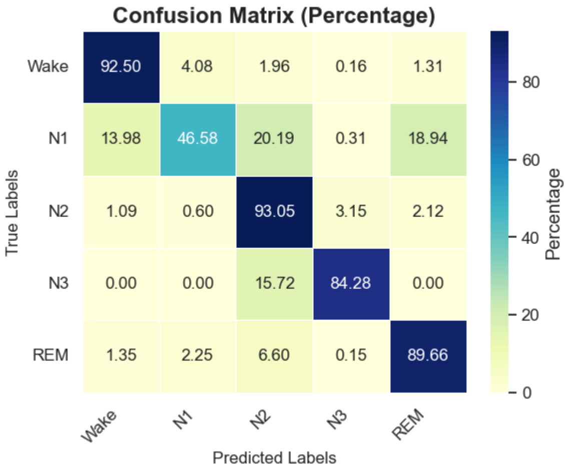
Detailed Accuracy Comparison



Classifier Accuracy Scores

Random Forest: 0.764 **Bagging:** 0.702 **Ensemble:** 0.831 **Gradient Boosting:** 0.753

Results: K-fold Random Forest

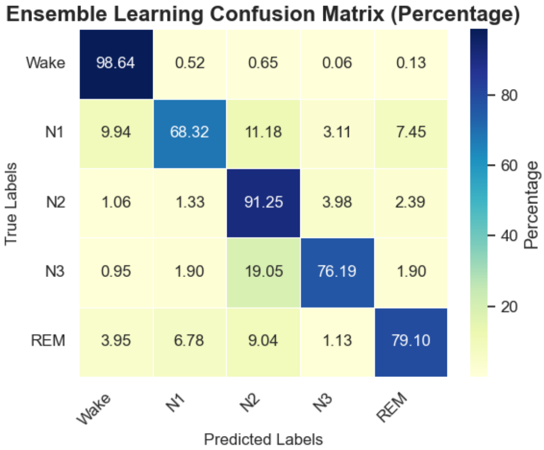


Average Accuracy: 0.88

Class-wise F1-Scores:
Wake: 90.43%
N1: 57.36%
N2: 91.36%
N3: 85.53%
REM: 87.11%

Macro Avg: Precision 85%, Recall 81%, F1 82%
Weighted Avg: Precision 87%, Recall 88%, F1 87%

Results: K-fold Ensemble Learning

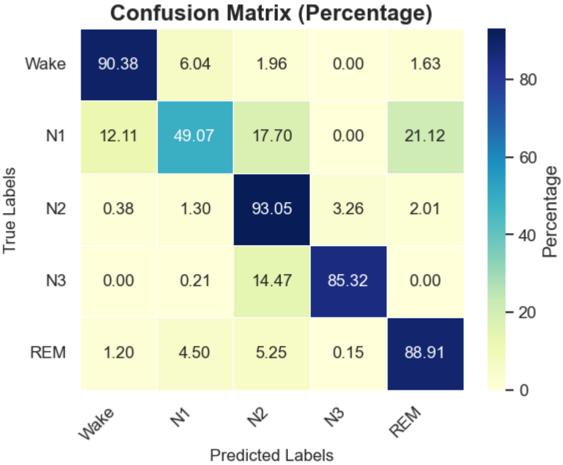


Average Accuracy: 0.95

Class-wise F1-Scores:
Wake: 99%
N1: 48%
N2: 91%
N3: 84%
REM: 86%

Macro Avg: Precision 86%, Recall 79%, F1 82%
Weighted Avg: Precision 95%, Recall 95%, F1 95%

Results: K-fold Gradient Boosting

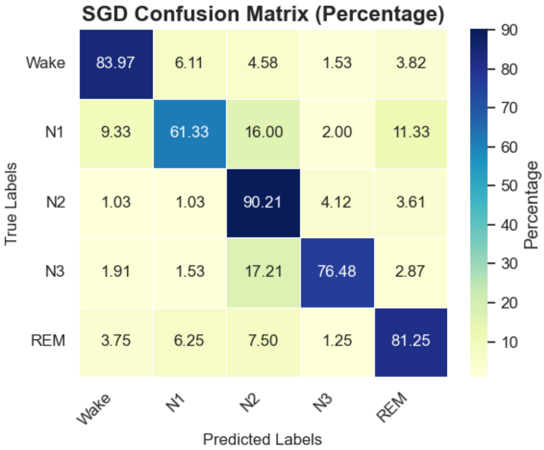


Average Accuracy: 0.87

Class-wise F1-Scores:
Wake: 90.75%
N1: 55.24%
N2: 91.93%
N3: 86.14%
REM: 86.25%

Macro Avg: Precision 83%, Recall 81%, F1 82%
Weighted Avg: Precision 87%, Recall 87%, F1 87%

Results: K-fold SGD Classifier



Average Accuracy: 0.80

Class-wise F1-Scores:
Wake: 80.42%
N1: 29.04%
N2: 88.19%
N3: 81.67%
REM: 72.83%

Macro Avg: Precision 73%, Recall 70%, F1 70%
Weighted Avg: Precision 78%, Recall 80%, F1 79%

Efficient Sleep Stage Classification Using EEG and PKL Data”

Problem Statement

Problem Statement

Deep Neural Model for Automated Sleep Staging using Single-Channel EEG Signal with Preprocessed Data for Efficient Training

Abstract

This work explores sleep stage classification using preprocessed EEG data (Fpz-Cz and Pz-Oz channels) converted into .pkl format from the Sleep-EDF dataset. The cleaned and normalized data is fed into various machine learning and deep learning models. Notably, ensemble methods and XGBoost achieved high accuracy, while Bi-LSTM demonstrated strong performance in deep learning. Despite challenges in classifying the N1 and REM stages, the system shows robust multi-class classification capabilities.

Best Accuracy Achieved

XGBoost: 85.3%, **Bi-LSTM: 81.1%**, **Random Forest: 84.2%**

Introduction

- Sleep stage classification is crucial for diagnosing sleep-related disorders.
- Processing raw EEG signals is computationally expensive and resource-intensive, especially with large datasets.

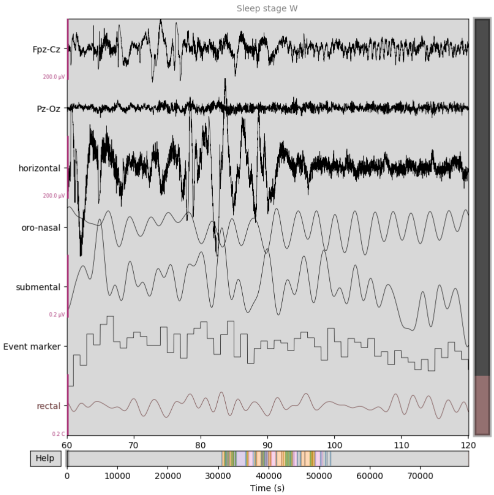


Figure: Visualization of EEG signal (PDEU)

Introduction

- This article discusses a method for preparing the Sleep-EDF dataset:
  - Extracting, segmenting, and labeling PSG data.
  - Converting data into Python pickle (`.pkl`) format for easy handling with deep learning frameworks.
- Used annotation descriptions for sleep stage labeling.

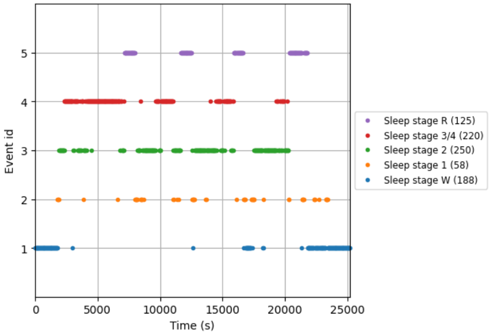


Figure: Sleep stage event plot

Proposed Architecture

The proposed system processes EEG data through cleaning, normalization, and encoding before feeding it into neural models. It supports Dense, RNN, LSTM, and Bi-LSTM architectures with dropout layers for regularization. The models classify sleep stages (W, N1, N2, N3, REM) based on processed input features.

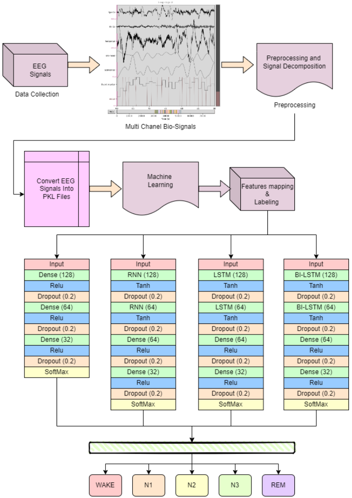


Figure: Proposed System Architecture

Methodology: Overview and Data Samples

Proposed System Architecture:

Component	Description
Input	Sleep EEG Data (.pkl format)
Preprocessing	Normalization, Label Encoding, One-hot Encoding, Reshaping
Model	Neural Network (Dense / RNN / LSTM / Bi-LSTM)
Output	Predicted Sleep Stage (W, N1, N2, N3, REM)

Sample Input Features and Labels:

Input Features (x)	Label (y)
[0.059, 0.596, -0.193, ..., -0.601, 0.201]	W
[-0.022, -0.107, -0.135, ..., 0.038, 0.103]	W
[... (more samples)]	N1

Preprocessing Workflow

Steps involved in data preparation:

- **Loading Data:** Sleep stage data is loaded from '.pkl' files in preprocessed directories.
- **Handling Test Sets:** Ensured test data availability by splitting the training set if necessary.
- **Normalization:** Standardized input features to have zero mean and unit variance.
- **Label Encoding:** Converted sleep stage labels into numerical format.
- **One-Hot Encoding:** Transformed numerical labels into one-hot vectors.
- **Reshaping:** Adjusted input dimensions for model compatibility.

Simple Neural Network Architecture

Layer Type	Neurons/Units	Activation Function
Input Layer	X_train	-
Dense Layer	128	ReLU
Dropout Layer	-	0.2
Dense Layer	64	ReLU
Dropout Layer	-	0.2
Dense Layer	32	ReLU
Dropout Layer	-	0.2
Output Layer	Number of Classes	Softmax

Model: Fully Dense Connected Neural Network

Simple RNN Architecture

Layer Type	Neurons/Units	Activation Function
Input Layer	X_train	-
RNN Layer 1	128	Tanh
Dropout Layer	-	0.2
RNN Layer 2	64	Tanh
Dropout Layer	-	0.2
Dense Layer	64	ReLU
Dropout Layer	-	0.2
Dense Layer	32	ReLU
Dropout Layer	-	0.2
Output Layer	Number of Classes	Softmax

Model: Recurrent Neural Network

LSTM Architecture

Layer Type	**Neurons/Units**	**Activation Function**
Input Layer	X_train	-
LSTM Layer 1	128	Tanh
Dropout Layer	-	0.2
LSTM Layer 2	64	Tanh
Dropout Layer	-	0.2
Dense Layer	64	ReLU
Dropout Layer	-	0.2
Dense Layer	32	ReLU
Dropout Layer	-	0.2
Output Layer	Number of Classes	Softmax

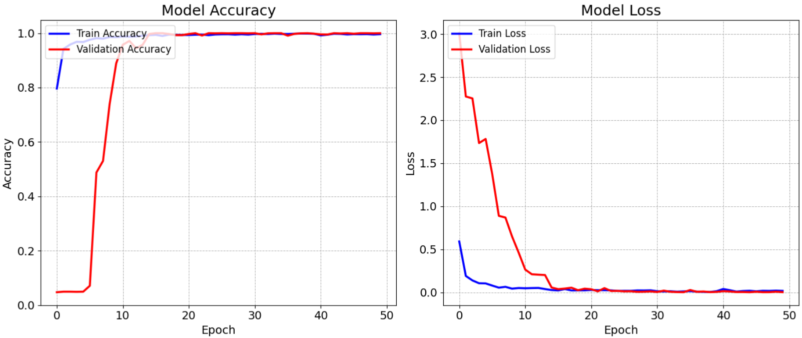
Model: Long Short-Term Memory Network

Bidirectional LSTM Architecture

Layer Type	Neurons/Units	Activation Function
Input Layer	X_train	-
Bi-LSTM Layer 1	128	Tanh
Dropout Layer	-	0.2
Bi-LSTM Layer 2	64	Tanh
Dropout Layer	-	0.2
Dense Layer	64	ReLU
Dropout Layer	-	0.2
Dense Layer	32	ReLU
Dropout Layer	-	0.2
Output Layer	Number of Classes	Softmax

Model: Bidirectional LSTM Network

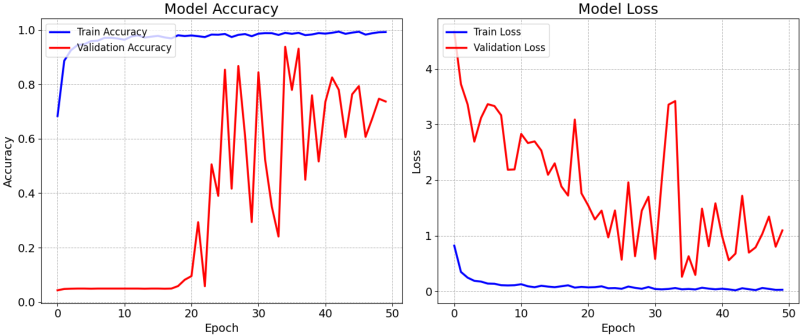
Simple Neural Network: Evaluation Metrics



Model Training Results

Accuracy: 77.72% **Precision:** 74.44% **Recall:** 77.72%
F1 Score: 74.99% **Macro Precision:** 56.69% **Macro Recall:** 45.15% **Macro F1 Score:** 48.74%

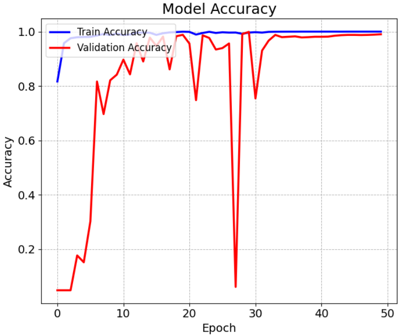
RNN Model: Evaluation Metrics

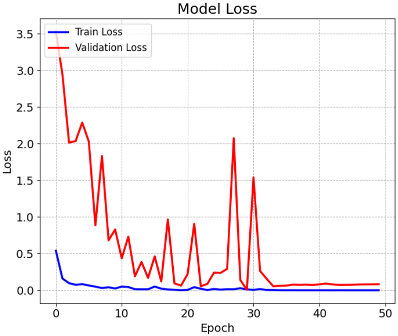


RNN Evaluation Results

Accuracy: 70.60% **Weighted Precision:** 69.79% **Macro Precision:** 38.09%
Weighted Recall: 70.60% **Macro Recall:** 34.52% **Weighted F1 Score:** 68.98%
Macro F1 Score: 33.17%

LSTM Model: Evaluation Metrics

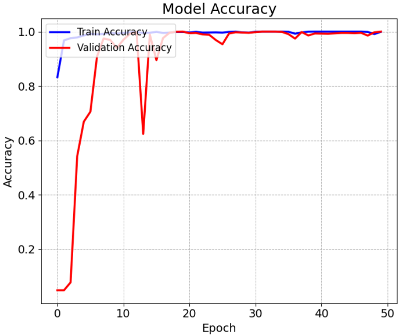


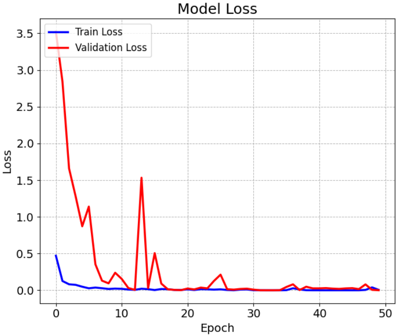


LSTM Evaluation Results

Accuracy: 79.97% **Weighted Precision:** 78.03% **Macro Precision:** 49.86%
Weighted Recall: 79.97% **Macro Recall:** 43.58% **Weighted F1 Score:** 78.51%
Macro F1 Score: 46.05%

BiLSTM Model: Evaluation Metrics





BiLSTM Evaluation Results

Accuracy: 81.13% **Weighted Precision:** 79.68% **Macro Precision:** 49.39%
Weighted Recall: 81.13% **Macro Recall:** 45.65% **Weighted F1 Score:** 80.30%
Macro F1 Score: 47.19%

ML Models: Key Evaluation Metrics

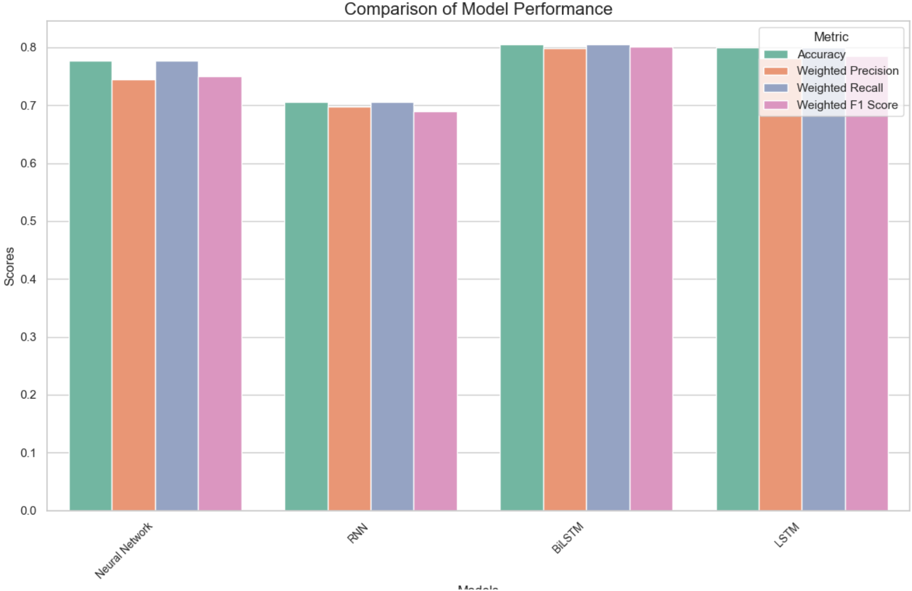
Random Forest

Accuracy:	84.29%
F1 Score:	82.29%
Macro Precision:	90.05%
Macro Recall:	55.81%
Macro F1 Score:	64.12%

XGBoost

Accuracy:	85.35%
F1 Score:	84.33%
Macro Precision:	77.64%
Macro Recall:	59.88%
Macro F1 Score:	64.44%

Neural Network Models: Performance Comparison



SleepGCN-Transformer

Problem Statement

Problem Statement

Using SleepGCN-Transformer: A Hybrid Graph Convolutional and Transformer Network for Sleep Stage Classification

Abstract

Dataset: SleepEDF dataset.

Preprocessing: Using 4 selected channels:

- EEG Fpz-Cz
- EEG Pz-Oz
- EMG submental
- EOG horizontal

Methodology:

- **Graph Convolutional Neural Network (GCN)**
- **Transformer Encoder**

Results:

- **Epoch 20/20:** Train Loss: **0.1413**, Train Acc: **93.12%**
- Validation Loss: **0.1390**, Validation Accuracy: **93.04%**

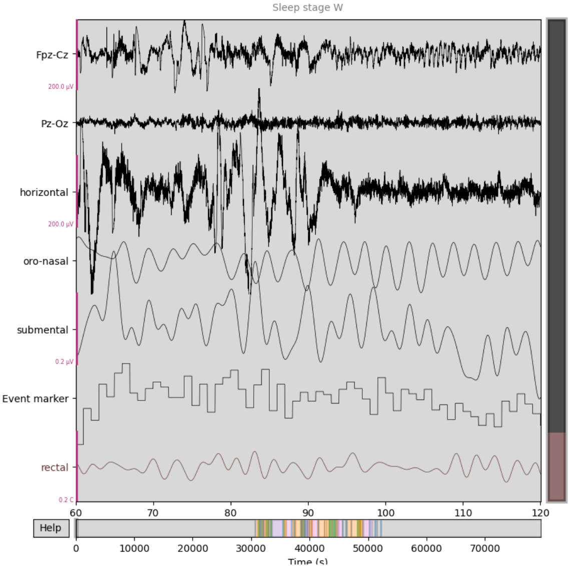
Introduction

SleepEDF Channels:

- EEG Fpz-Cz
- EEG Pz-Oz
- EMG submental
- EOG horizontal

Sleep Stages and Frequency Ranges:

Sleep Stage	Frequency (Hz)
Wake (Beta)	12-30
N1 (Light Sleep)	4-8
N2 (Moderate Sleep)	4-6
N3 (Deep Sleep)	0.5-4
REM (Theta)	4-6



Methodology

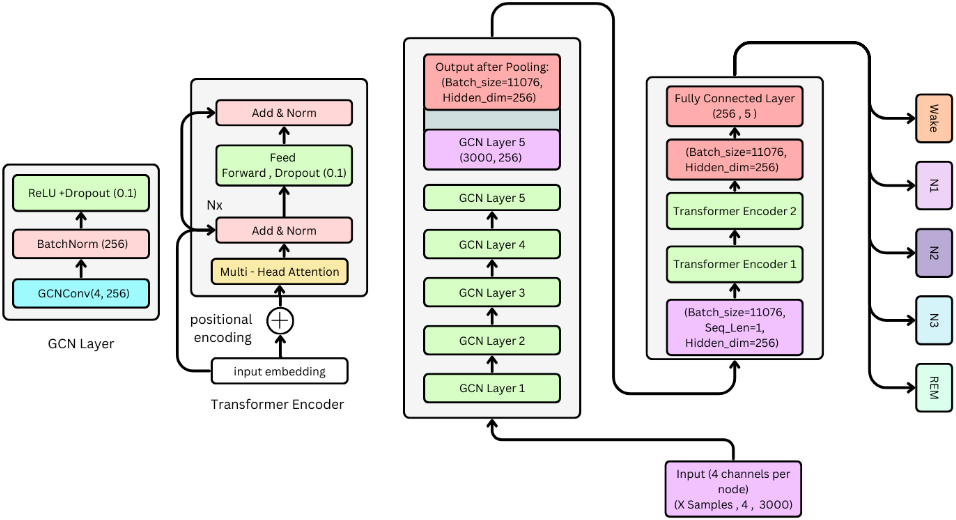


Figure: Proposed SleepGCN-Transformer Architecture

Methodology: Preprocessing

Channel Selection:

- Extracting four relevant EEG channels:
  - EEG Fpz-Cz
  - EEG Pz-Oz
  - EMG submental
  - EOG horizontal

Sleep Stage Mapping:

Original Stage	Mapped Label
Sleep stage W	0
Sleep stage 1	1
Sleep stage 2	2
Sleep stage 3	3
Sleep stage 4	3
Sleep stage R	4

Methodology: Preprocessing

Epoch Segmentation:

- EEG signals are segmented into 30-second epochs.
- Each epoch contains 3000 samples per channel.

Band-Pass Filtering:

- A band-pass filter (0.3 - 30 Hz) is applied.
- Signals above 30 Hz are removed to eliminate noise.

Final Data Shape:

[X, 4, 3000]

Methodology: Graph Dataset Creation

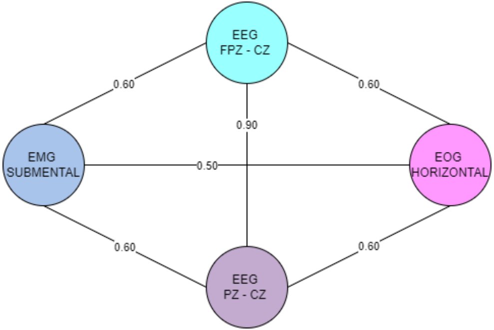
Graph Adjacency Matrix (Edge Weights):

	Fpz-Cz	Pz-Oz	EMG	EOG
Fpz-Cz	0	0.9	0.6	0.6
Pz-Oz	0.9	0	0.6	0.6
EMG	0.6	0.6	0	0.5
EOG	0.6	0.6	0.5	0

Dataset Information:

- Total Samples: 11,076
- Example Sample Format:

Data(x=[3000, 4], edge_index=[2, 12], edge_attr=[12], y=[1])



Graph Representation of EEG Channels

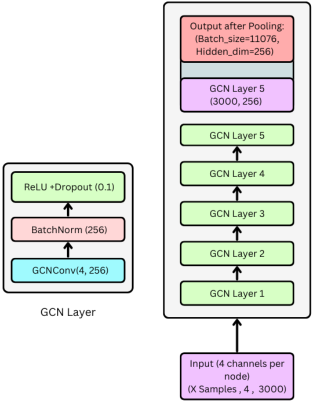
Methodology: Graph Convolutional Layer

Graph Convolutional Layer (GCL)

- Captures spatial relationships in EEG signals.
- Learns connectivity patterns between EEG channels.
- Enhances feature extraction by leveraging graph structures.

Tensor Shapes for GCL Input:

- **X_all**: (11076, 4, 3000)
- **Y_all**: (11076,)
- **X_tensor**: `torch.Size([11076, 4, 3000])`
- **Y_tensor**: `torch.Size([11076])`



Graph Convolutional Layer Representation

Methodology: GCN Tensor Details and Global Pooling

Additional Tensor Shapes for GCL:

- **Sample x**: `torch.Size([3000, 4])`
- **Sample edge_index**: `torch.Size([2, 12])`
- **Sample y**: `torch.Size([1])`

Global Mean Pooling:

- **Input:** Node embeddings from GCN layers (e.g., (3000, 256))
- **Operation:** Mean pooling over nodes based on batch indices
- **Output:** Graph-level embedding (e.g., (Batch_size=11076, 256))

Methodology: Transformer Encoder

Transformer Encoder Overview

- **Preprocessing:** Expand graph embedding to `(Batch, 1, 256)`
- **Transformer Encoder:**
  - **2 Transformer Encoder Layers** with:
    - **d_model = 256**
    - **nhead = 4**
    - **dropout = 0.1**
    - `batch_first=True`
- **Postprocessing:** Squeeze output to `(Batch, 256)`

Fully Connected Layer

- **Linear Layer:** `Linear(256 → 5)`
- **Output:** Logits for 5-class classification

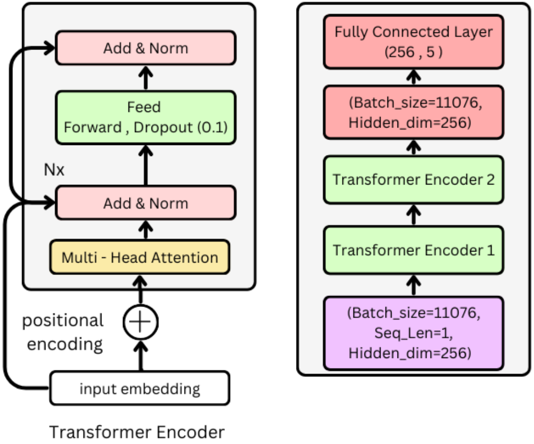


Figure: Transformer Encoder Architecture

Methodology: Why Focal Loss Instead of Standard Cross-Entropy?

Motivation for Focal Loss

- Standard Cross-Entropy treats all samples equally, leading to bias towards majority classes.
- In imbalanced datasets, minority class predictions get suppressed.
- **Focal Loss** dynamically adjusts loss contribution based on prediction confidence.
- It reduces the importance of well-classified samples and focuses more on hard-to-classify ones.

Key Features of Focal Loss

- Introduces a focusing parameter γ to adjust class weighting.
- Includes class weighting factor α to handle imbalance.
- Works well for highly imbalanced datasets in classification tasks.

Methodology: Focal Loss Formulation

Mathematical Formulation

$$\mathrm{FL}(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t)$$

where:

- p_t is the predicted probability for the target class.
- α is the weighting factor for class imbalance.
- γ is the focusing parameter (higher values focus more on hard examples).

Implementation Details

- Label smoothing:

$$y_{\text{smooth}} = y(1 - \epsilon) + \frac{\epsilon}{C}$$

- Prevents log(0) issue by adding a small constant ϵ.
- PyTorch-based computation:

$$\mathcal{L} = \alpha(1 - p)^\gamma(-y_{\text{smooth}} \log p)$$

Why Use a Learning Rate Scheduler?

Importance of Learning Rate Scheduling

- The learning rate is crucial for training deep models efficiently.
- A high learning rate can lead to divergence, while a low one may cause slow convergence.
- Adaptive learning rate schedules help balance stability and speed.

Why CosineAnnealingLR?

- Smoothly reduces the learning rate following a cosine decay.
- Starts with a large step size for exploration and gradually fine-tunes.
- Helps avoid sharp drops in the learning rate, improving generalization.

Cosine Annealing Learning Rate Decay

Cosine Annealing Formula:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right)$$

where:

- η_t is the learning rate at epoch t.
- $\eta_{\max}$ and $\eta_{\min}$ are the max/min learning rates.
- T_{cur} is the current epoch.
- T_{max} is the total number of epochs.

Key Benefits:

- Encourages large updates early in training.
- Smoothly transitions into finer updates as training progresses.
- Helps the model avoid getting stuck in poor local minima.

Training Methodology: Overview

SleepTrainer Class: Key Features

- Handles model training, validation, and optimization.
- Uses **Focal Loss** to address class imbalance.
- Applies **CosineAnnealingLR** scheduler for smooth learning rate decay.

Training Process

1. Compute class weights for imbalanced data.
2. Iterate through training batches, compute loss and update weights.
3. Validate model performance on a separate validation set.
4. Adjust learning rate dynamically using a scheduler.

Training Methodology: Hyperparameters

Key Hyperparameters

- **Batch Size:** 32
- **Learning Rate:** 0.0003
- **Weight Decay:** $1e^{-4}$
- **Epochs:** 20
- **Optimizer:** AdamW

Learning Rate Scheduler: CosineAnnealingLR

- Gradually reduces learning rate over time for smooth convergence.
- Helps prevent sudden drops in performance.

Training Methodology: Handling Class Imbalance

Why Compute Class Weights?

- EEG sleep data is imbalanced, with some sleep stages appearing more frequently.
- Without weighting, the model may favor majority classes.
- Weights ensure rare classes contribute more to the loss.

Class Weight Computation

$$w_c = \left(\frac{\text{Total Samples}}{\text{Class Count} + 1} \right)^{0.5}$$

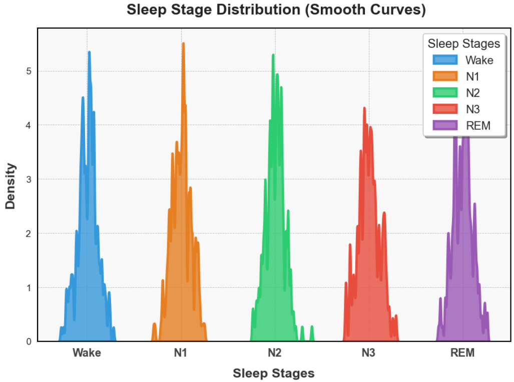
where:

- w_c is the computed weight for class c.
- Small classes receive higher weights.
- Weights are applied to Focal Loss for training.

Testing Data Distribution Analysis

Why Ensure Balanced Testing Data?

- Prevents bias toward majority classes.
- Ensures the model's performance is fairly evaluated.
- Helps achieve reliable generalization across all sleep stages.
- The figure shows the normalized class distribution during testing.
- Each class maintains an equalized density, avoiding class imbalance.
- This confirms that the model's evaluation is not biased toward any specific sleep stage.



Sampling Density Plot Showing Balanced Class Distribution

Model Performance: Training vs Testing

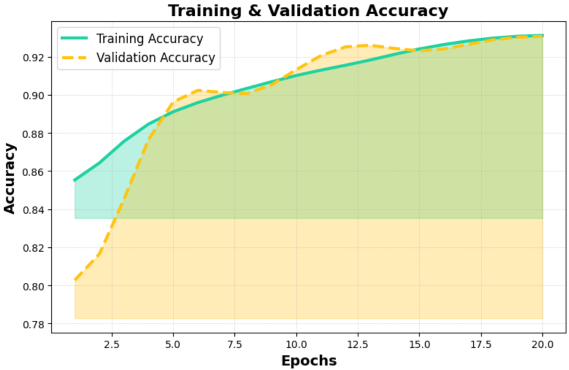


Figure: Accuracy Curve

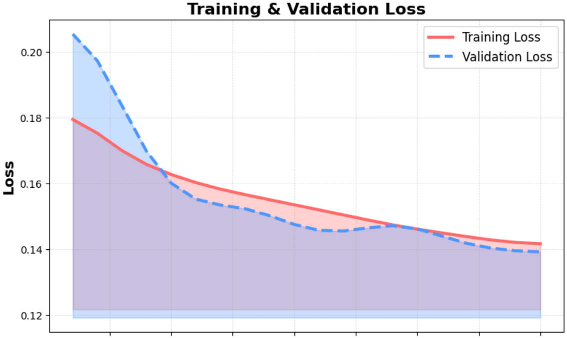


Figure: Loss Curve

Model Evaluation: Confusion Matrix

Performance Metrics

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics ensure a balanced evaluation of model performance across all classes.

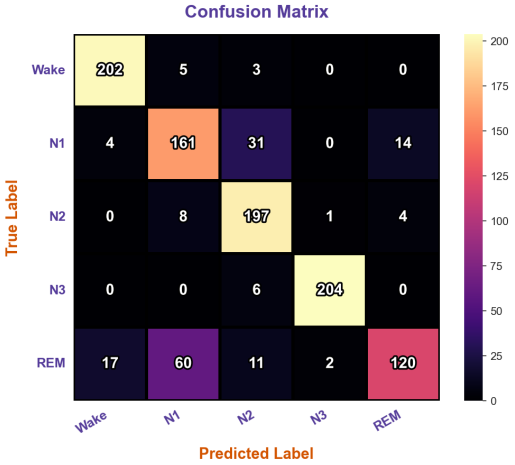


Figure: Confusion Matrix

Gradient Analysis: Training Progression

Understanding Model Training Dynamics

- **Early Training (Epochs 0-5):** High loss, accuracy starts improving.
- **Mid Training (Epochs 5-15):** Loss steadily decreases, stable gradient flow.
- **Late Training (Epochs 15-20):** Accuracy plateaus, no severe overfitting.

Conclusion: The training process remains stable, with no vanishing or exploding gradients.

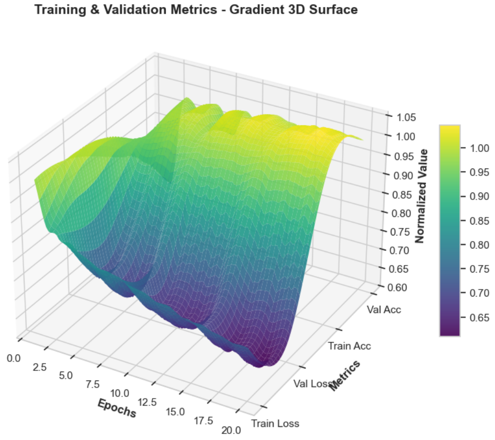


Figure: Gradient 3D Surface: Training vs Validation Metrics

Performance Metrics: Precision, Recall, F1-Score

Evaluating model performance across all classes using key metrics.

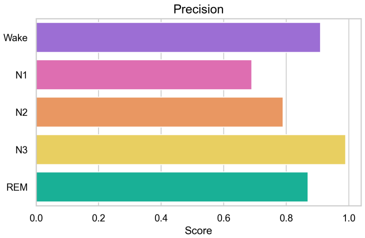


Figure: Precision Scores per Class

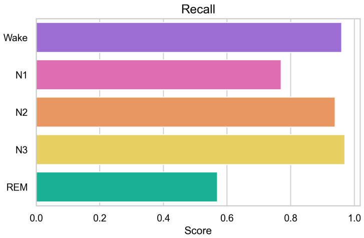


Figure: Recall Scores per Class

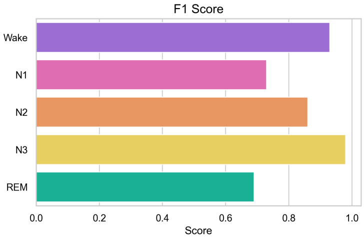


Figure: F1 Scores per Class

Feature Importance Analysis with LIME

w **Understanding the contribution of different channels to model predictions.**

- We used **LIME** (Local Interpretable Model-agnostic Explanations) to analyze feature importance.
- The **EMG submental** and **EEG Pz-Oz** channels contribute the most to predictions.
- **EOG horizontal** has minimal importance, indicating lower relevance for classification.
- This insight helps optimize feature selection and improve model efficiency.

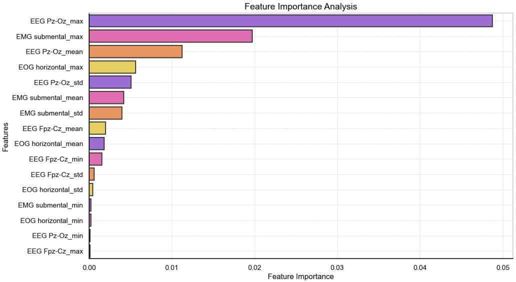


Figure: Feature Importance Analysis for 4 Channels

XAI: Enhancing Model Explainability

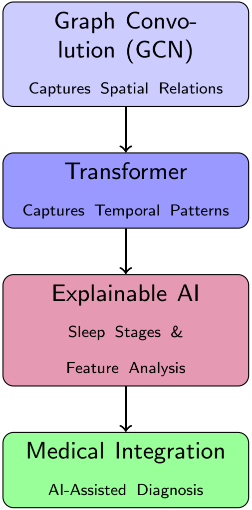
Moving Towards Explainable AI for Sleep Staging

- **Why Explainability?** - Medical experts need transparency in AI decisions for trust and adoption. - Understanding how features influence sleep stage transitions is crucial.
- **Current Achievements:** - **GCN:** Captures spatial relationships between EEG channels. - **Transformer:** Captures temporal dependencies in sleep data. - Achieved state-of-the-art accuracy using both approaches.
- **Next Steps:** - Implement AI-driven methods to highlight critical sleep stage transition points. - Develop feature attribution methods to understand the importance of each signal. - Improve model interpretability to align with clinical expectations.

Future Plan: AI for Sleep Science and Clinical Use

Bridging AI and Healthcare

- **Feature Importance:** Identify which EEG channels contribute most to predictions.
- **Clinical Relevance:** Provide insights that can be validated by sleep specialists.
- **Graph + Transformer Insights:**
  - **GCN:** Capturing inter-channel spatial dependencies.
  - **Transformer:** Learning sequential patterns across sleep cycles.



Conclusion

Our proposed SleepGCN-Transformer model achieves **93.12% training accuracy** and **93.04% validation accuracy**, demonstrating its effectiveness in sleep stage classification. The integration of **Graph Convolution Networks (GCN)** captures spatial dependencies across EEG, EOG, and EMG channels, while the **Transformer** extracts temporal patterns. The use of **Focal Loss** enhances class balancing, improving performance on underrepresented sleep stages. Feature importance analysis highlights **EMG and EEG Pz-Oz** as key predictors. This robust approach lays the foundation for future work in **Explainable AI**, enabling medical professionals to interpret AI-driven sleep diagnostics effectively.