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

A Master's Report

Submitted in partial fulfillment of the requirement for the award of the degree of

Master of Technology in Artificial Intelligence

by

Tanmay Rathod

23MAI007

Under the guidance of

Dr. Santosh Kumar Satapathy

Department of Information and Communication Technology

School of Technology

Pandit Deendayal Energy University

Gandhinagar – 382426. Gujarat - India

February, 2025

Approval Sheet

This thesis entitled **"SleepGCN-Transformer: A Hybrid Graph Convolutional and Transformer Network for Sleep Stage Classification"** by **Tanmay Rathod** is recommended for the degree of **M.Tech** in **Artificial Intelligence**.

Guide : Dr Santosh Satapathy,

Assistant Professor

Dept. of ICT

Dr Nitin Singh Rajput,

Assistant Professor

Dept. of ICT

Dr. Paawan Sharma,

Head Of Department

Dept. of ICT

Date: _____

Place: _____

Acknowledgment

While my name appears as the sole contributor to the completion of this summer internship, it is important to recognize that the guidance and support of many individuals played a significant role in its success. This internship is the result of a collective effort, shaped by the contributions of numerous people, to whom I am deeply grateful.

I extend my heartfelt appreciation to my internal guide, Dr. Santosh Kumar Satapathy Author, for his consistent guidance, encouragement, and insightful suggestions throughout this internship. His belief in my abilities provided the confidence and motivation necessary to complete this project. Dr. Santosh Kumar Satapathy Author's support, along with his generous provision of time and access to resources, was instrumental in achieving the objectives of this internship.

I am also profoundly thankful for the unwavering support of my parents. Their continuous inspiration, moral backing, and blessings have been the cornerstone of my journey. Their understanding and encouragement have been invaluable, and I am truly fortunate to have had their steadfast support throughout this endeavor.

Tanmay Rathod

Student Declaration

I, **Tanmay Rathod**, hereby declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated, or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Pandit Deendayal Energy University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Tanmay Rathod

Roll No: 23MAI007

Date: _____

Abstract

This project presents *SleepGCN-Transformer*, a hybrid model combining Graph Convolutional Networks (GCN) and Transformer encoders for sleep stage classification. Using the SleepEDF dataset, it incorporates four physiological signals: EEG (Fpz-Cz, Pz-Oz), EMG (submental), and EOG (horizontal). Preprocessing includes 30-second epochs, band-pass filtering (0.3–30 Hz), and graph-based EEG channel representation.

The GCN module captures spatial relationships across EEG channels, while the Transformer encoder models temporal dependencies from graph-level embeddings. Focal Loss addresses class imbalances, and a CosineAnnealingLR scheduler optimizes learning rate decay. Training with the AdamW optimizer for 20 epochs achieves 93.12 % training and 93.04% validation accuracy.

Model performance, assessed using precision, recall, and F1-score, demonstrates high efficacy. LIME-based feature importance analysis highlights EMG and EEG Pz-Oz channels as key contributors. This hybrid model exhibits state-of-the-art performance, with future directions focused on enhancing explainability and clinical integration.

Contents

Contents v

List of Figures vi

List of Tables vii

1 Introduction 1

2 Literature Review 4

3 SleepGCN-Transformer Hybrid Graph Convolution and Transfomer Network Methodology 6

3.1 Methodology 6

3.1.1 Input Data and Preprocessing 7

3.1.2 Graph Construction 9

3.1.3 GCN Layers for Spatial Feature Extraction 10

3.1.4 Transformer Encoder for Temporal Dependencies 12

3.1.5 Loss Function and Optimization 14

4 Results and Discussion 19

4.0.1 Testing Data Distribution Analysis 19
4.0.2 Model Performance: Training vs Testing 19
4.0.3 Model Evaluation: Confusion Matrix 21
4.0.4 Gradient Analysis: Training Progression 21
4.0.5 Performance Metrics: Precision, Recall, F1-Score 22
4.0.6 Feature Importance Analysis with LIME 24

5 Conclusion **26**

6 Reference **27**

List of Figures

1.1 Sleep EDF Data Visulization 2

3.1 Proposed Architecture of GCN-Transformer Hybrid Approach . . 7
3.2 Graph Weight Visulization 10
3.3 Graph Convolution Neural Network 11
3.4 Enter Caption 13

4.1 Balanced class distribution across the test set. 20
4.2 Accuracy Curve: Training vs Testing. 20
4.3 Loss Curve: Training vs Testing. 21
4.4 Confusion Matrix showing model performance across sleep stages. 22
4.5 Gradient 3D Surface: Training vs Validation Metrics. 23
4.6 Precision Scores per Class. 23
4.7 Recall Scores per Class. 24
4.8 F1 Scores per Class. 24
4.9 Feature Importance Analysis for 4 Channels. 25

List of Tables

1.1 Frequency ranges of different sleep stages. 2

3.1 Mapping of Original Sleep Stages to Labels 8
3.2 Graph Weight Representation 9
3.3 Dataset shape Information 10
3.4 GCN Layer Architecture 11
3.5 Transformer Encoder Overview 12

Chapter 1

Introduction

Sleep is an essential component of human health, contributing significantly to cognitive function, memory consolidation, and emotional regulation. Adequate and quality sleep ensures optimal brain performance and supports overall physical and mental well-being. Disruptions in sleep patterns can lead to various health issues, including impaired concentration, mood disorders, and chronic conditions such as cardiovascular diseases.

The classification of sleep stages is fundamental in understanding sleep architecture and diagnosing sleep disorders. Experts categorize sleep into five primary stages—N1, N2, N3, REM, and Wake—based on distinct patterns observed in brain activity. Each sleep stage is characterized by specific brainwave frequencies, which can be detected using electroencephalography (EEG) signals. Accurate classification of these stages is crucial for analyzing sleep quality and identifying abnormalities such as insomnia, sleep apnea, and narcolepsy.

Table 1.1 provides an overview of the frequency ranges associated with each sleep stage.

TABLE 1.1
Frequency ranges of different sleep stages.

Sleep Stage	Frequency Range (Hz)
Wake (Beta waves)	12–30
N1 (Light Sleep)	4–8
N2 (Moderate Sleep)	4–6
N3 (Deep Sleep, Delta)	0.5–4
REM (Theta waves)	4–6

To classify sleep stages effectively, bio-signals from multiple physiological channels are analyzed. The primary sources of information for sleep stage classification are EEG (electroencephalogram), EMG (electromyogram), and EOG (electrooculogram) signals. These channels capture brain activity, muscle movement, and eye movemets, respectively, offering a comprehensive view of sleep patterns.

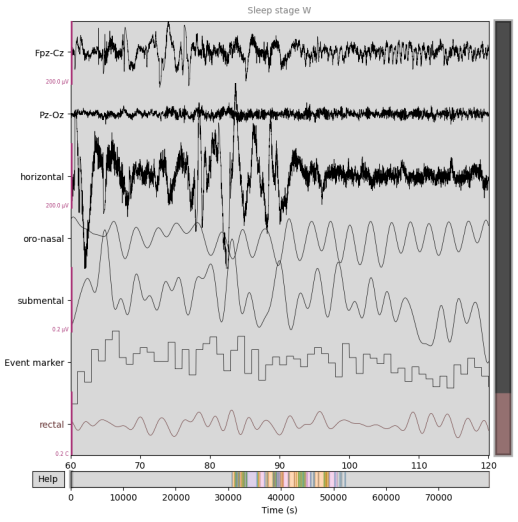

Figure 1.1. Sleep EDF Data Visulization

Despite the availability of multiple channels in the SleepEDF dataset, not all contribute equally to classification performance. To reduce model complexity

without compromising accuracy, we selectively utilize four channels from the nine standard channels provided in the dataset. The selected channels are:

- **EEG Fpz-Cz**: Captures frontal-to-central brain activity.
- **EEG Pz-Oz**: Measures parietal-to-occipital brain activity.
- **EMG submental**: Records muscle tone from the chin area.
- **EOG horizontal**: Tracks horizontal eye movements.

In this study, we adopt a graph-based modeling approach to capture spatial relationships between the selected channels. Additionally, we employ a Transformer encoder to effectively model temporal dependencies across sleep cycles. To facilitate a better understanding of the input data, we also present visualizations of bio-signals from all channels.

This approach aims to leverage the strengths of both spatial and temporal modeling techniques, resulting in a robust and efficient sleep stage classification framework.

Chapter 2

Literature Review

[1] The proposed architecture, Efficient Sleep Sequence Network (ESSN), has overcome the limitations of existing automatic sleep stage algorithms. This model addresses two main challenges. First, the model is quite complex, and often low-end systems are unable to process it; therefore, this model is designed to work on lightweight systems. The second challenge is the misclassification of the N1 stage, where models often confuse wake and REM stages. To address this, it introduces the N1 structure loss function. The ESSN model has achieved impressive metrics: 88.0% accuracy, 81.2% macro F1, and 0.831 Cohen's kappa. These results were obtained on the SHHS dataset. Additionally, it has reduced computational requirements, with only 0.27M parameters and 0.35G floating-point operations, and it claims to be faster than models like L-SeqSleepNet.

[2] The Multi-Domain View Self-Supervised Learning Framework (MV-TTFC) introduces a new approach to classify sleep stages by leveraging self-supervised learning (SSL) on unlabeled EEG data. By incorporating multi-view repre-

sentation technology, this model enhances information exchange across different views. It also introduces the multisynchrosqueezing transform, which improves the quality of the time-frequency view. Ultimately, it captures the latent features within EEG signals. It was evaluated on two datasets (SleepEDF-78 and SHHS), and MV-TTFC achieved state-of-the-art performance with accuracies of 78.64% and 81.45%, and macro F1-scores of 70.39% and 70.47%, respectively.

[3] The proposed CNN-Transformer-ConvLSTM-CRF hybrid model presents a new integration method between local and global feature extraction to enhance the classification ability of sleep stages. The model can identify relationships among EEG features by applying a multi-scale convolutional neural network combined with a Transformer for encoding features of the EEG signal and a spatio-temporal encoder via ConvLSTM. Additionally, the adaptive feature calibration module improves the extracted features, and there is efficient learning of the transition relationships between the stages of sleep by the CRF module. Based on evaluations on three datasets, this hybrid model outperforms existing state-of-the-art methods, demonstrating its efficacy in sleep stage classification.

Chapter 3

SleepGCN-Transformer Hybrid Graph Convolution and Transfomer Network Methodology

3.1 Methodology

In this study, we propose a hybrid model combining Graph Convolutional Networks (GCNs) and Transformer encoders for sleep stage classification from EEG signals. Our approach captures both spatial relationships between EEG channels using GCNs and temporal dependencies using Transformer encoders. Below, we describe the methodology in detail.

biosignals as input and filter the four channels. The filtered signals are passed through a graph convolution layer, followed by a pooling layer to reduce their size. Next, we adjust the batch size and feed the output into a transformer encoder. Finally, we obtain the softmax classification into five stages.

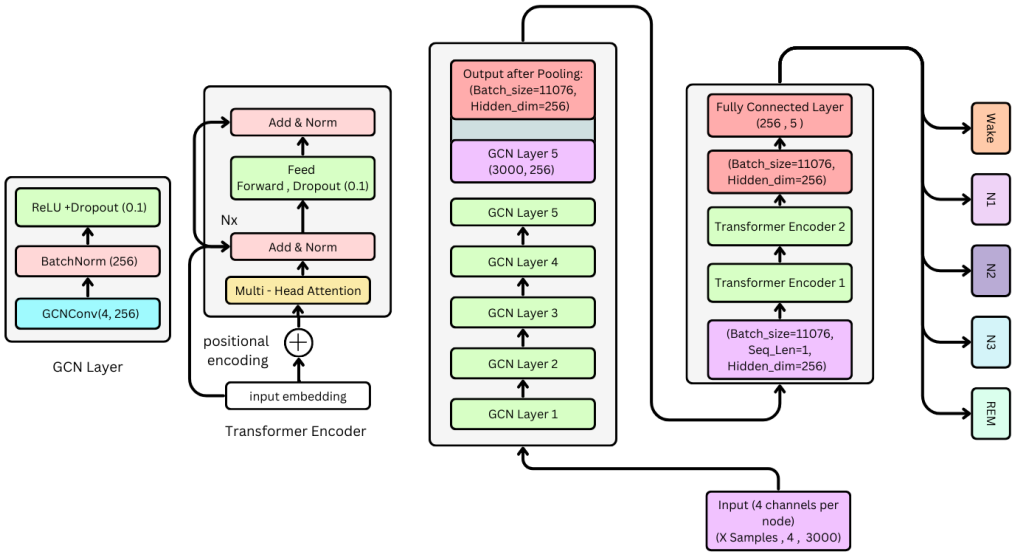

Figure 3.1. Proposed Architecture of GCN-Transformer Hybrid Approach

3.1.1 Input Data and Preprocessing

We use EEG data represented as graph-structured inputs. Each sample contains 4 EEG channels recorded at 3000 frequency points. The dataset dimensions are:

- **X_all**: (11076, 4, 3000) - Samples, Channels, Frequency Points
- **Y_all**: (11076,) - Labels for sleep stages
- Converted to PyTorch tensors:
  - **X_tensor**: torch.Size([11076, 4, 3000])
  - **Y_tensor**: torch.Size([11076])

Sleep Stage Mapping

The sleep stages are mapped as follows:

TABLE 3.1
Mapping of Original Sleep Stages to Labels

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

Channel Selection

The following EEG channels are selected for analysis: EEG Fpz-Cz, EEG Pz-Oz, EMG submental, and EOG horizontal.

Preprocessing

Epoch Segmentation: The EEG signals are segmented into 30-second epochs. Each epoch contains 3000 samples per channel, resulting in the final shape of the data as:

$$\text{Final Data Shape: } [X, 4, 3000]$$

Band-Pass Filtering: A band-pass filter with a frequency range of 0.3 - 30 Hz is applied to the signals. This removes frequencies outside this range, filtering out high-frequency noise and low-frequency drift.

3.1.2 Graph Construction

We represent each sample as a graph, where nodes correspond to EEG channels and edges represent spatial relationships. The graph is defined as:

- **x**: Node features (shape: [3000, 4]) - EEG signals per channel
- **edge_index**: Graph connectivity (shape: [2, 12])
- **y**: Sleep stage label for each sample (shape: [1])

Methodology: Graph Dataset Creation

The graph's adjacency matrix represents the spatial relationship between EEG channels. The edge weights are derived from the following graph adjacency matrix:

TABLE 3.2
Graph Weight Representation

	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

This matrix is used to define the weights of the edges between the EEG channels in the graph. The value at position (i, j) indicates the weight of the edge between node i and node j.

TABLE 3.3
Dataset shape Information

Component	Details
Total Samples	11,076
Example Sample Format	Data(x=[3000, 4], edge index=[2, 12], edge attr=[12], y=[1])
x (Node features)	EEG signals (shape: [3000, 4])
edge index	Connectivity between EEG channels (shape: [2, 12])
edge attr	Weights of edges between EEG channels (shape: [12])
y (Label)	Sleep stage label for the sample (shape: [1])

Graph Representation of EEG Channels

The EEG channels are represented as nodes in the graph, with edges indicating spatial relationships based on the adjacency matrix. The resulting graph structure captures both the spatial dependencies between the channels and the temporal dynamics of the EEG signals.

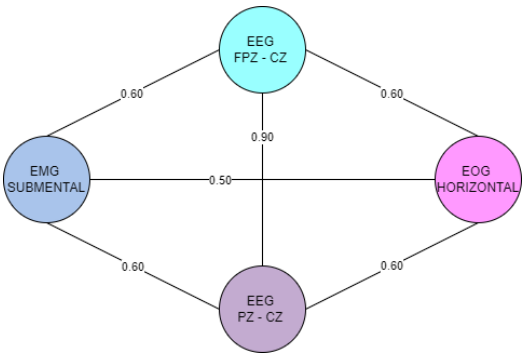

Figure 3.2. Graph Weight Visulization

3.1.3 GCN Layers for Spatial Feature Extraction

The GCN module captures spatial dependencies among EEG channels. We employ five Graph Convolutional Network (GCN) layers to learn the connectivity patterns between EEG channels, thereby enhancing feature extraction by lever-

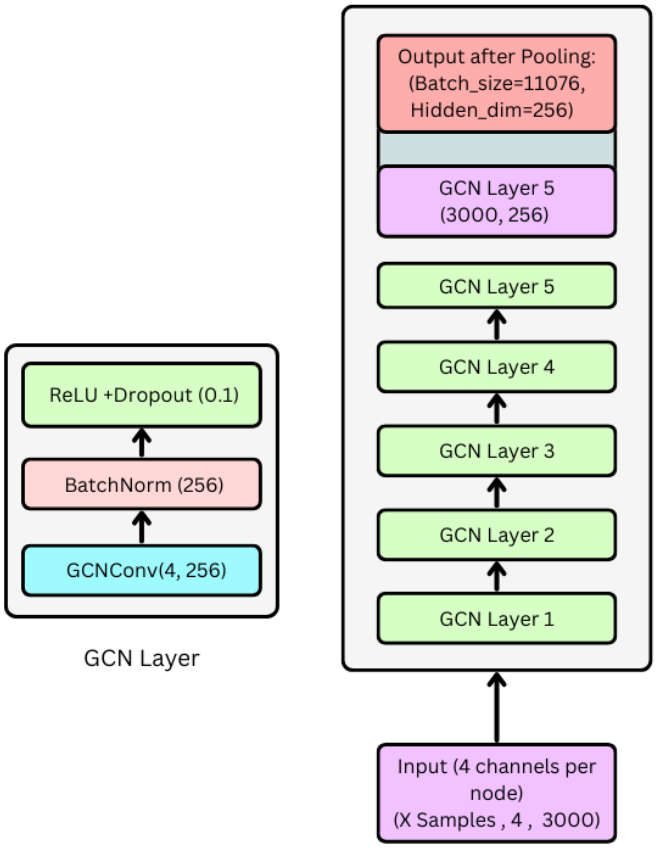

Figure 3.3. Graph Convolution Neural Network

aging the graph structure of EEG data. The architecture of each GCN layer is described as follows:

TABLE 3.4
GCN Layer Architecture

Layer	Operations
GCN Layer 1	GCNConv(4, 256) → BatchNorm(256) → ReLU → Dropout(0.1)
GCN Layer 2 to 5	GCNConv(256, 256) → BatchNorm(256) → ReLU → Dropout(0.1)

The GCN layers work by taking the input features, which are of shape (4, 256) after the first layer, then passing them through a **Batch Normalization** layer with 256 output channels. The output is then activated using the **ReLU** activation

function, formulated as:

$$\mathrm{ReLU}(x) = \max(0, x)$$

This is followed by **Dropout**(0.1) for regularization, helping to avoid overfitting by randomly setting 10% of the input units to zero.

After passing through all five GCN layers, the final output tensor has a shape of $(3000, 256)$, representing 3000 frequency points with a 256-dimensional feature representation for each frequency point across all EEG channels.

To obtain a graph-level representation, we apply **global mean pooling** across nodes, which reduces the output to a $(\text{Batch\_size} = 11076, 256)$ tensor. This means that each sample is represented by a single 256-dimensional feature vector. The hidden dimension in this final layer is 256, which captures the most relevant features of the input data.

3.1.4 Transformer Encoder for Temporal Dependencies

TABLE 3.5
Transformer Encoder Overview

Component	Details
Preprocessing	Expand graph embedding to (Batch_size, 1, 256)
Transformer Encoder	2 Transformer Encoder Layers
d_model (Hidden dimension)	256
nhead (Number of heads)	4
Dropout	0.1
batch first	True
Postprocessing	Squeeze output to (Batch_size, 256)
Fully Connected Layer	Linear($256 \rightarrow 5$)
Output	Logits for 5-class classification

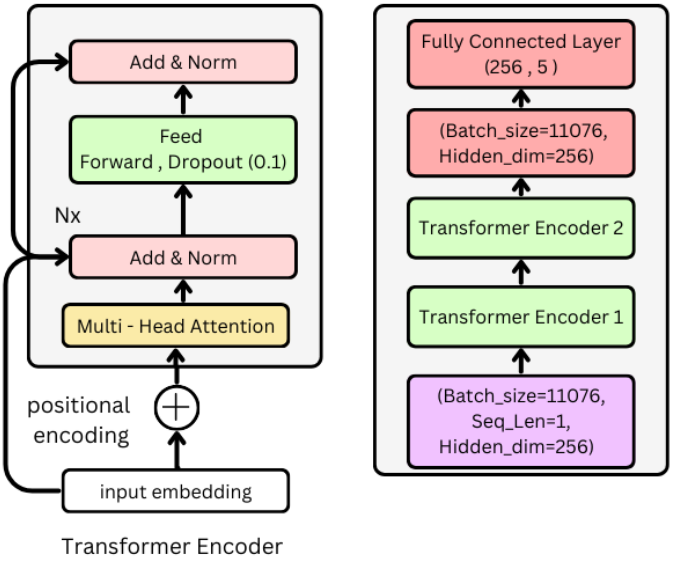

Figure 3.4. Enter Caption

The Transformer architecture follows the GCN layer pooling shapes. After pooling, the embeddings are passed through the multi-head attention mechanism, followed by Add & Norm, which normalizes the output. This is then followed by a feed-forward layer, dropout with a 0.1 probability, and another Add & Norm. The input embeddings are connected in parallel with the Add & Norm. Positional encoding is applied during input to account for the temporal order of the sequence. The batch size is set to 1, with a sequence length of 1 and a hidden dimension of 256. The input passes through two Transformer Encoder layers, followed by a fully connected linear layer with a transformation from 256 to 5, which outputs the logits for 5-class classification. The final output is obtained using the softmax function, represented by:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

3.1.5 Loss Function and Optimization

We employ Focal Loss to address class imbalance, with parameters:

- Gamma (γ): 3
- Label smoothing: 0.1
- Class weights computed from the dataset

The optimizer and learning rate scheduler are configured as follows:

- **Optimizer**: AdamW (learning rate=0.0003, weight decay=1e-4)
- **Scheduler**: Cosine Annealing Learning Rate Scheduler

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 the 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.

Focal Loss Formulation

The Focal Loss is formulated as follows:

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

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: Label smoothing is applied to prevent the log(0) issue by adding a small constant ε:

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

where C is the number of classes.

The PyTorch-based computation for Focal Loss with label smoothing becomes:

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

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? Cosine Annealing smoothly reduces the learning rate following a cosine decay. It starts with a large step size for exploration and gradually fine-tunes. This helps avoid sharp drops in the learning rate, improving generalization.

Cosine Annealing Learning Rate Decay

The learning rate decay is governed by the formula:

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

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:

- Compute class weights for imbalanced data.
- Iterate through training batches, compute loss, and update weights.
- Validate model performance on a separate validation set.
- 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, helping 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: The class weight for class c is computed as:

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

where:

- w_c is the computed weight for class c.
- Smaller classes receive higher weights.

These weights are applied to the Focal Loss during training.

Chapter 4

Results and Discussion

4.0.1 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 below 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:

4.0.2 Model Performance: Training vs Testing

Figures below demonstrate the accuracy and loss curves during training and testing, providing a clear visual comparison of model performance:

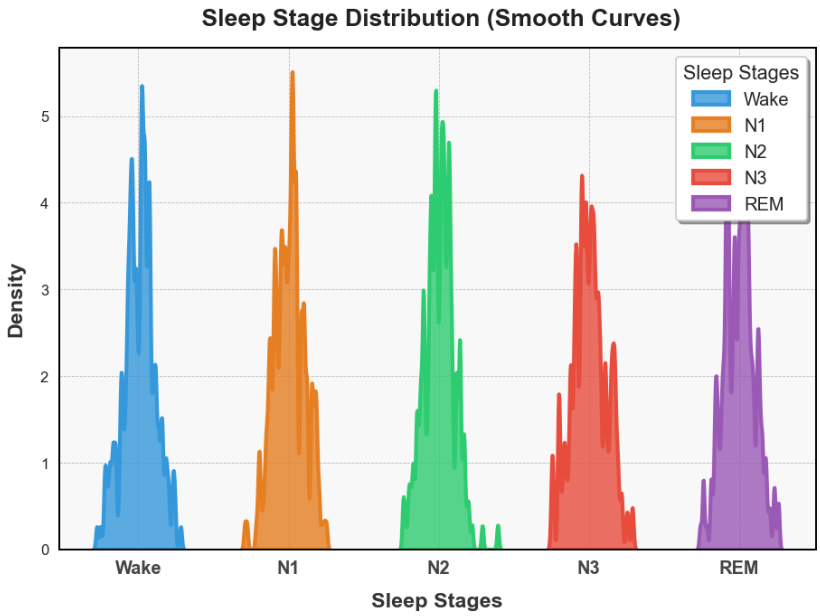

Figure 4.1. Balanced class distribution across the test set.

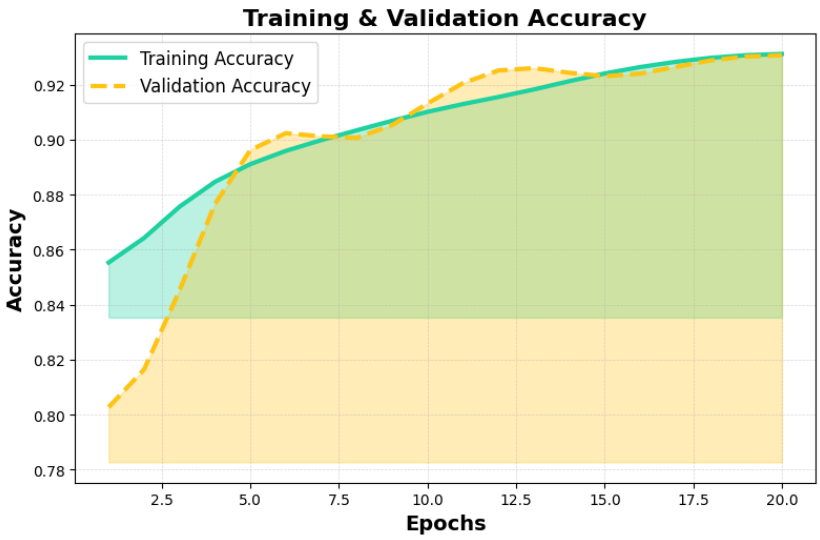

Figure 4.2. Accuracy Curve: Training vs Testing.

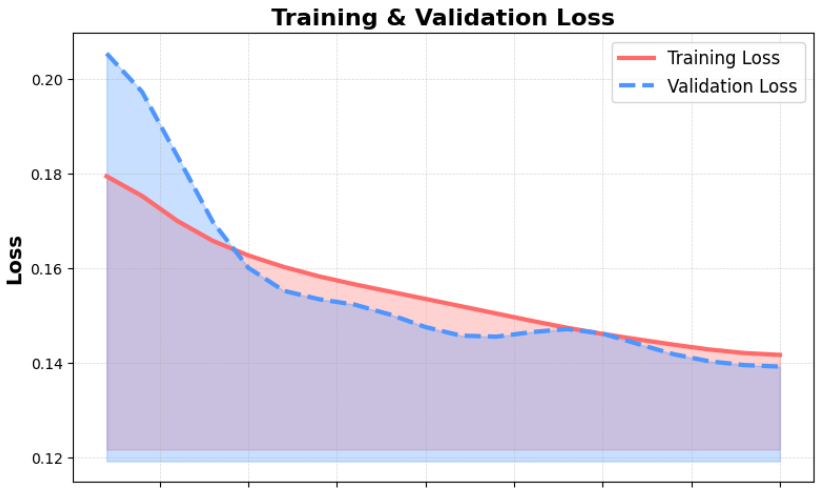

Figure 4.3. Loss Curve: Training vs Testing.

4.0.3 Model Evaluation: Confusion Matrix

Performance metrics are calculated to ensure a comprehensive evaluation of the model's performance across all classes. We use precision, recall, and F1-score to assess the model:

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

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

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

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

4.0.4 Gradient Analysis: Training Progression

Understanding model training dynamics:

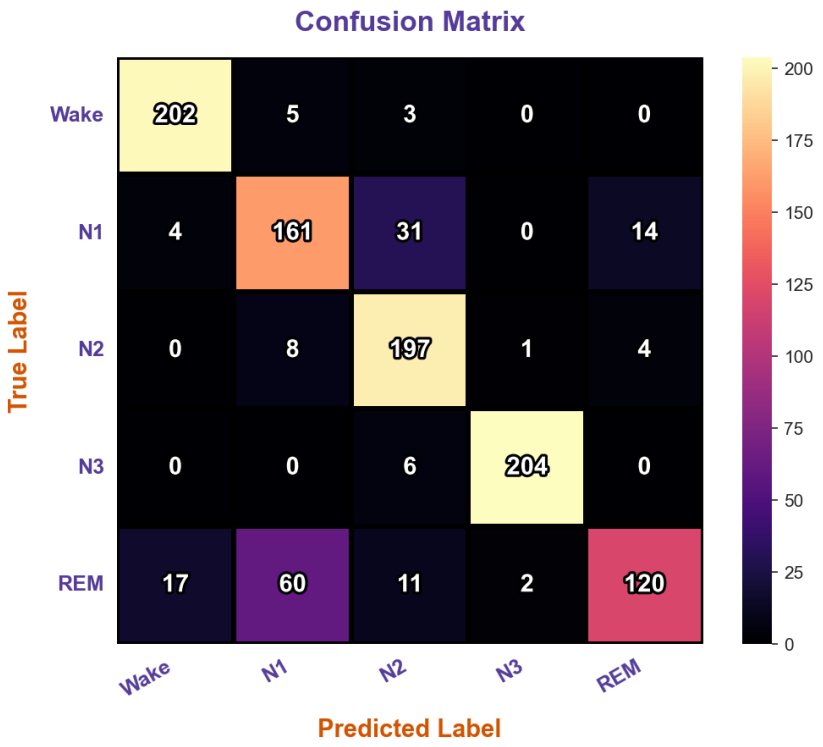

Figure 4.4. Confusion Matrix showing model performance across sleep stages.

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

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

4.0.5 Performance Metrics: Precision, Recall, F1-Score

Evaluating model performance across all classes using key metrics:

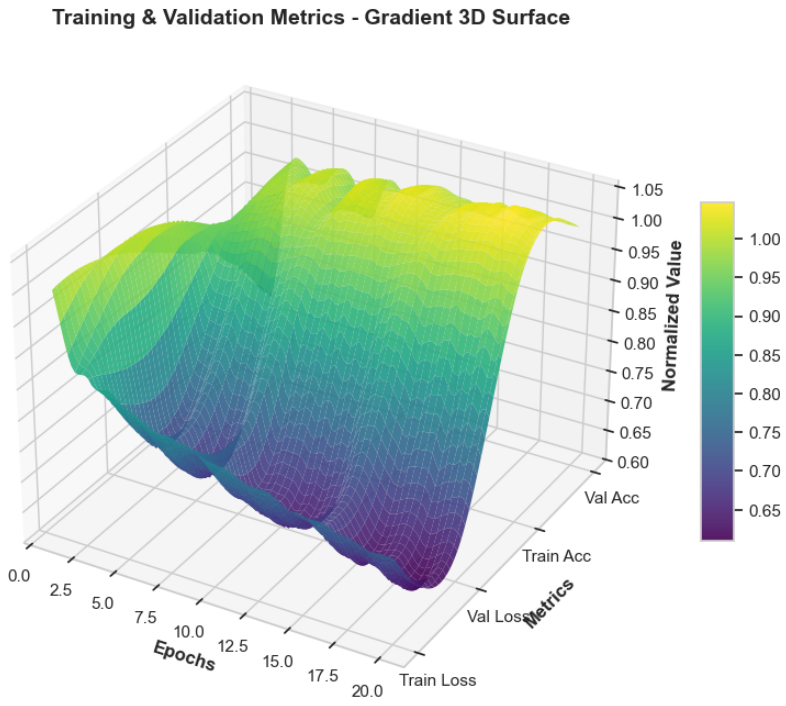

Figure 4.5. Gradient 3D Surface: Training vs Validation Metrics.

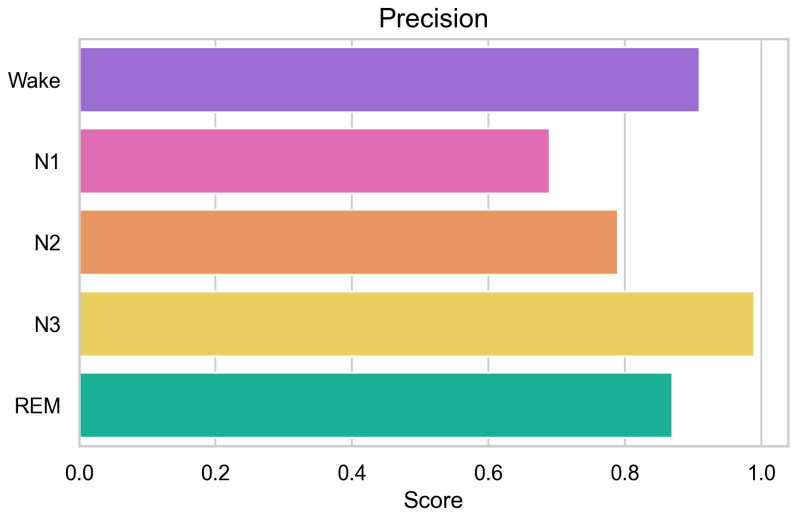

Figure 4.6. Precision Scores per Class.

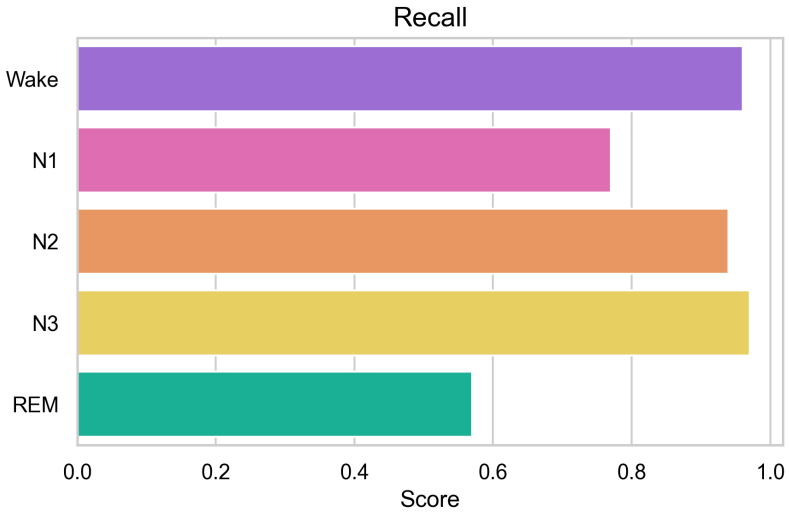

Figure 4.7. Recall Scores per Class.

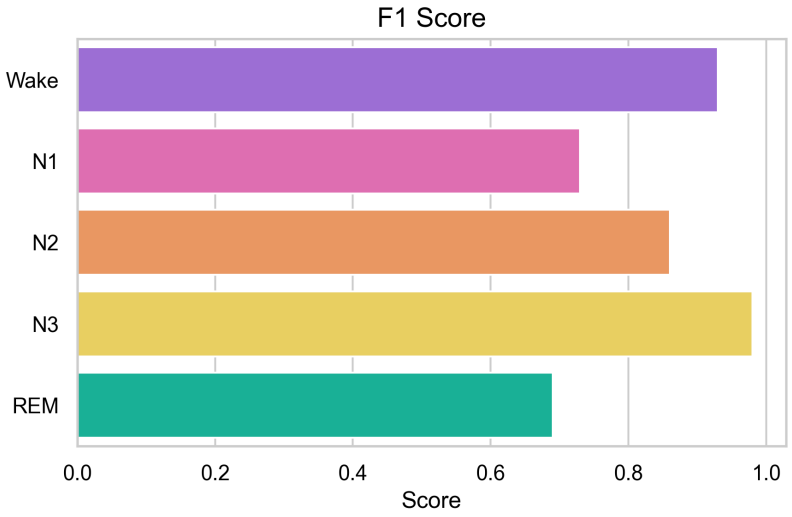

Figure 4.8. F1 Scores per Class.

4.0.6 Feature Importance Analysis with LIME

Understanding the contribution of different channels to model predictions, we used LIME (Local Interpretable Model-agnostic Explanations) to analyze feature

importance. The analysis showed that the **EMG submental** and **EEG Pz-Oz** channels contribute the most to predictions, while the **EOG horizontal** channel has minimal importance, indicating lower relevance for classification.

This insight helps optimize feature selection and improve model efficiency.

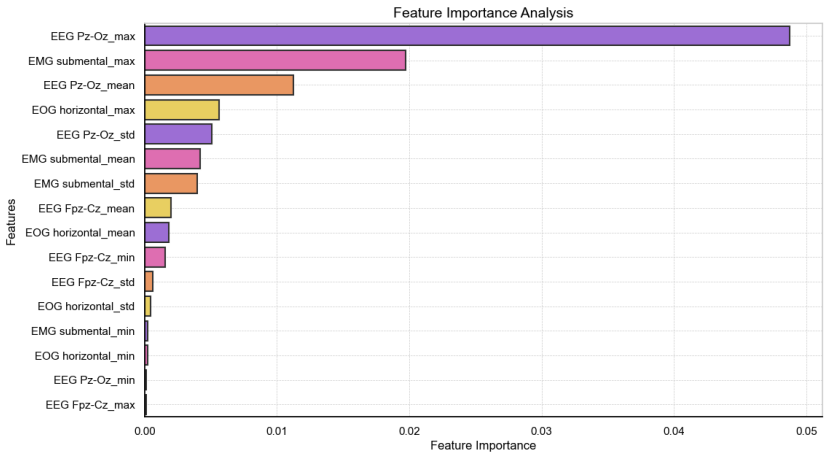

Figure 4.9. Feature Importance Analysis for 4 Channels.

Chapter 5

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.

Chapter 6

Reference