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

A Master's Thesis

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

May, 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

Correct and effective sleep stage classification is crucial in the diagnosis of sleep disorders, conventionally done through manual interpretation of polysomnography (PSG) signals. Automated sleep stage classification is investigated in this thesis by a series of increasingly sophisticated models, from traditional machine learning to sophisticated deep learning architectures. Initial trials on the SleepEDF dataset compared models like Random Forest, XGBoost, and ensemble models, which obtained maximum accuracy of 85.35%, and set the foundation for automated systems to perform consumer-level sleep monitoring. Future work introduced deep learning models like BiLSTM and RNNs, and BiLSTM reached 81.13% accuracy, which proved to have better ability to model temporal sequences.

Based on these building blocks, we introduce *SleepGCN-Transformer*, a new hybrid model that combines Graph Convolutional Networks (GCNs) to learn spatial relations between multi-channel EEG, EMG, and EOG signals, and Transformer encoders to learn temporal patterns. The model applies Focal Loss to handle class imbalance and a CosineAnnealingLR scheduler for dynamic learning rate adaptation. Trained with the AdamW optimizer on preprocessed 30-second epochs, SleepGCN-Transformer is 93.12% training and 93.04% validation accurate. Feature attribution by LIME indicates EMG and EEG Pz-Oz to be the most significant channels, demonstrating the model's interpretability.

# Contents

**Contents** v

**List of Figures** vii

**List of Tables** ix

**1 Introduction** 1

1.0.1 The Role of EEG in Sleep Analysis . . . . . 2

1.0.2 Understanding the SleepEDF Dataset . . . . . 2

1.0.3 Sleep Stages and Signal Patterns . . . . . 3

1.0.4 The Need for Automation . . . . . 3

**2 Literature Review** 5

**3 Automated Sleep Staging System with EEG Signal using Machine Learning Techniques** 7

3.1 Methodology . . . . . 7

3.2 Dataset Information . . . . . 8

3.3 Preprocessing . . . . . 9

3.4 Model Architecture and Learning Framework . . . . . 11

3.5 K-Fold Cross-Validation . . . . . 12
3.5.1 Random Forest with K-Fold Cross-Validation Pipeline . . 14
3.5.2 Ensemble Classifier with K-Fold Cross-Validation Pipeline 14
3.5.3 Gradient Boosting with K-Fold Cross-Validation Pipeline 14
3.5.4 Random Forest Classifier . . . . . 15
3.5.5 Gradient Boosting Classifier with PCA . . . . . 16
3.5.6 Ensemble Learning (Voting Classifier) . . . . . 16
3.6 Results and Evaluation With Machine Learning . . . . . 17
3.6.1 Random Forest . . . . . 17
3.6.2 Bagging Classifier . . . . . 17
3.6.3 Ensemble Learning . . . . . 18
3.6.4 Gradient Boosting . . . . . 19
3.7 Results and Evaluation with K-fold Cross Validation . . . . . 21
3.8 Comparison Sections . . . . . 22

**4 Deep Neural Model for Automated Sleep Staging System using Single-Channel EEG Signal 23**

4.1 Methodology . . . . . 23
4.1.1 Modeling Approach . . . . . 24
4.2 Dataset Information . . . . . 24
4.3 Preprocessing Techniques . . . . . 26
4.4 Model Architecture and Learning Framework . . . . . 26
4.5 Results and Evaluation . . . . . 35
4.5.1 LSTM Model . . . . . 35
4.5.2 RNN Model . . . . . 36

4.5.3 BiLSTM Model . . . . . 37
4.5.4 Neural Network (NN) Model . . . . . 38

**5 SleepGCN-Transformer: A Hybrid Graph-Convolutional and Transformer Network for Sleep Stage Classification 39**

5.1 Methodology . . . . . 39
5.2 Dataset Information . . . . . 39
5.3 Preprocessing Techniques . . . . . 39
5.4 Model Architecture and Learning Framework . . . . . 39
5.5 Results and Evaluation . . . . . 39

**6 Conclusion 40**

**7 Reference 41**

# List of Figures


3.1 Filter EEG Chanels . . . . . 9

3.2 Architecture of Machine Learning Classifiers for Sleep Stage Classificatio . . . . . 12

3.3 Architecture of Machine Learning Classifiers for Sleep Stage Classificatio . . . . . 13

3.4 Random Forest Confusion Matrix . . . . . 17

3.5 Random Forest Accuracy Curve . . . . . 17

3.6 Random Forest Loss Curve . . . . . 17

3.7 Random Forest Precision Per Class . . . . . 17

3.8 Bagging Classifier Confusion Matrix . . . . . 18

3.9 Bagging Classifier Accuracy Curve . . . . . 18

3.10 Bagging Classifier Loss Curve . . . . . 18

3.11 Bagging Classifier Precision Per Class . . . . . 18

3.12 Ensemble Learning Confusion Matrix . . . . . 19

3.13 Ensemble Learning Accuracy Curve . . . . . 19

3.14 Ensemble Learning Loss Curve . . . . . 19

3.15 Ensemble Learning Precision Per Class . . . . . 19

3.16 Gradient Boosting Confusion Matrix . . . . . 20
3.17 Gradient Boosting Accuracy Curve . . . . . 20
3.18 Gradient Boosting Loss Curve . . . . . 20
3.19 Gradient Boosting Precision Per Class . . . . . 20
3.20 Ensemble Learning Confusion Matrix . . . . . 21
3.21 Random Forest Confusion Matrix . . . . . 21
3.22 GB Classifier Confusion Matrix . . . . . 21
3.23 SGD Confusion Matrix . . . . . 21
3.24 Accuracy Comparison . . . . . 22
3.25 Improved Bar Chart . . . . . 22
3.26 Box Plot Comparison . . . . . 22
3.27 Improved Violin Plot . . . . . 22

4.1 LSTM Accuracy and Loss Plot . . . . . 35
4.2 LSTM Confusion Matrix . . . . . 35
4.3 RNN Accuracy Plot . . . . . 36
4.4 RNN Confusion Matrix . . . . . 36
4.5 BiLSTM Accuracy Plot . . . . . 37
4.6 BiLSTM Confusion Matrix . . . . . 37
4.7 Neural Network Accuracy Comparison . . . . . 38
4.8 Neural Network Confusion Matrix . . . . . 38

# List of Tables

1.1 Brainwave Types and Their Characteristics in Sleep Staging . . . 3

4.1 Architecture of the Deep Neural Network . . . . . 27

4.2 Recurrent Neural Network Architecture for Sleep Stage Classification . . . . . 30

4.3 LSTM Neural Network Architecture for Sleep Stage Classification 31

4.4 Bidirectional LSTM Neural Network Architecture for Sleep Stage Classification . . . . . 34

# Chapter 1

# Introduction

Sleep is a fundamental aspect of human health that greatly contributes to cognitive function, memory consolidation, and emotional regulation. Proper and quality sleep guarantees proper brain function and overall physical and mental health. Sleep disturbances result in several diseases, such as impaired concentration, mood disorders, and chronic conditions like cardiovascular diseases.

Sleep plays a critical function in sustaining physical and mental well-being. It is not a passive resting state but a sophisticated biological process crucial for memory consolidation, emotional homeostasis, metabolic health, and immune system function. Alterations in sleep quality or sleep patterns can reflect or lead to many health problems, such as insomnia, depression, cardiovascular disease, and neurodegenerative disorders. It is essential for clinicians and scientists to precisely assess sleep stages for the diagnosis of such conditions and the prescription of proper treatment plans.

Staging of the sleep stages is the basis of understanding the architecture of sleep and the diagnosis of sleep disorders. Sleep is divided into five major stages—N1,

N2, N3, REM, and Wake—based on unique patterns seen in brain function. Every stage of sleep is associated with particular frequencies of brainwaves, which can be picked up using the technique of electroencephalography (EEG) signals. The precise staging is important to study the quality of sleep and to detect abnormalities like insomnia, sleep apnea, and narcolepsy.

### 1.0.1 The Role of EEG in Sleep Analysis

Electroencephalography (EEG) is among the major methods of tracking brain activity while asleep. EEG captures electrical activity created by neurons as they fire within the brain, recorded with electrodes applied to the scalp. These electrical signals are critical in distinguishing various stages of sleep since each stage has different patterns of brain wave activity. Whereas other physiological signals are indirect, EEG is a direct view into the electrical activity of the brain and hence the basis for research and diagnosis into sleep.

### 1.0.2 Understanding the SleepEDF Dataset

The Sleep-EDF (Sleep European Data Format) corpus, popular among researchers, comprises polysomnography (PSG) recordings taken from healthy subjects and mildly disordered sleep patients. Recordings are taken non-invasively and usually overnight in a laboratory setting. In the course of a session, several sensors are applied to the subject's body to record a number of different physiological signals such as EEG (electroencephalogram), EOG (electrooculogram), and EMG (electromyogram).

Specifically, the database contains two EEG channels, namely Fpz-Cz and

Pz-Oz, which record frontal and parietal brain activity. Additionally, it contains EOG signals to record eye movements and EMG signals to record muscle activity, particularly around the chin region. The recordings are manually annotated by experienced sleep technicians using visual patterns and set guidelines to mark sleep stages. The last annotation is retained in a hypnogram — a time series plot that shows the changes between various stages of sleep throughout the night.

### 1.0.3 Sleep Stages and Signal Patterns

Human sleep is commonly divided into five stages: Wake (W), Non-Rapid Eye Movement (NREM) stages N1, N2, and N3, and Rapid Eye Movement (REM). Each stage is characterized by specific frequency patterns in EEG signals:

TABLE 1.1
Brainwave Types and Their Characteristics in Sleep Staging

| Wave Type | Frequency Range | Associated Sleep Stage / Behavior |
|---|---|---|
| Alpha waves | 8–13 Hz | Relaxed wakefulness, especially with eyes closed |
| Beta waves | 13–30 Hz | Alertness and active thinking; decrease during sleep |
| Theta waves | 4–8 Hz | Light sleep (Stages N1 and N2) |
| Delta waves | 0.5–4 Hz | Deep sleep (Stage N3); synchronized neuronal firing |
| Sleep spindles & K-complexes | 12–15 Hz (spindles) | Characteristic of Stage N2; used for stage classification |

These patterns are extracted from raw EEG signals through filtering and segmentation. Experts use these patterns, along with eye movement and muscle tone data, to classify each 30-second segment (epoch) into one of the five stages.

### 1.0.4 The Need for Automation

Manual scoring of sleep stages, while precise, is time-consuming and subject to inter-rater reliability. It takes hours of professional scoring for one night's worth

of recording. Automated sleep stage scoring provides a quicker, more scalable, and more possibly consistent solution. Using machine learning and deep learning models, particularly those that can address intricate spatial and temporal patterns in physiological data, we can develop systems that emulate expert opinion and aid in clinical decision-making. This thesis suggests just such a system — *SleepGCN-Transformer* — that unifies graph-based representation of EEG sensor relationships with temporal learning from transformers. Our strategy not only targets high classification performance but also contributes to the increasing body of research in interpretable AI in medicine, moving us closer to clinically integrated, explainable, and fully automated sleep diagnostics.

# 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

# Automated Sleep Staging System with EEG Signal using Machine Learning Techniques

## 3.1 Methodology

We begin by importing the Sleep-EDF data provided in EDF (European Data Format). The raw EEG recordings are loaded using MNE's `read_raw_edf` function, and corresponding annotations are aligned using `read_annotations`. We optionally exclude all non-EEG channels unless explicitly required, focusing on EEG signals for our analysis. The dataset includes two EEG channels: `Fpz-Cz` and `Pz-Oz`, which are the primary input sources in our work.

During loading, we crop the signal to reduce unnecessary wake-time data, retaining 30 minutes before the first sleep stage and 30 minutes after the last. This ensures that our training data remains focused around relevant sleep activity. The

channel names are cleaned by stripping prefixes for standardization.

After loading and cropping, we segment the data into fixed-length 30-second epochs using the known sampling frequency of the recordings. Only epochs labeled with valid sleep stages — specifically stages W, N1, N2, N3, and REM — are retained. For each epoch, the raw data is extracted and structured into a format suitable for training.

Once all epochs are collected, we flatten the 3D EEG data (epochs $\times$ channels $\times$ time points) into 2D arrays (epochs $\times$ features). This is necessary for traditional machine learning classifiers. To address class imbalance in sleep stages, we apply SMOTE (Synthetic Minority Oversampling Technique) to generate balanced training samples.

The balanced dataset is then split into training and testing sets in an 80-20 ratio. We train a Machine Learning on the training data. After training, predictions are made on the test set, and the model is evaluated using standard metrics including classification report and confusion matrix. The confusion matrix is visualized using a heatmap to give a clear view of the stage-wise performance of the classifier.

## 3.2 Dataset Information

We have used the Sleep-EDF Dataset available at `https://www.physionet.org/content/sleep-edfx/1.0.0/`. For our experiments, we selected recordings from approximately 30 patients, resulting in 60 files — 30 EDF (European Data Format) recordings and their corresponding hypnogram annotation files. The EDF files include various bio-signals captured through multiple channels. The

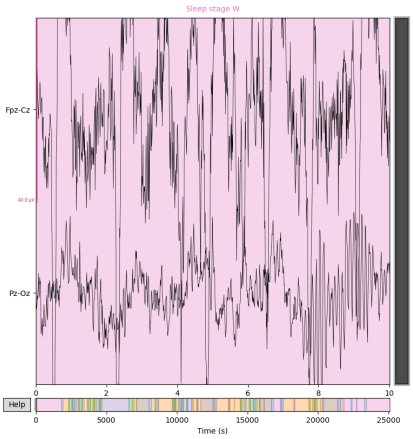


Figure 3.1. Filter EEG Chanels

hypnogram files contain sleep stage annotations, specifying the start time and type of each stage (Wake, N1, N2, N3, N4, REM) with timestamps in the HH:MM:SS format.

The EDF recordings consist of a range of physiological signals, including EEG (`Fpz-Cz` and `Pz-Oz`), EOG, EMG (submental), rectal temperature, and oro-nasal respiration. Each channel provides continuous signal recordings at a specific sampling frequency (Hz), which are used as input features for our downstream machine learning pipeline. From these signals, we derive two essential inputs: spatial features, used to construct graph-based representations, and temporal features, which capture the sequence and timing dynamics necessary for modeling the sleep process.

## 3.3 Preprocessing

This study applied machine learning techniques to classify sleep stages using EEG signals. The preprocessing pipeline included several crucial steps: channel map-

ping, data cropping, signal filtering, epoch creation, label mapping, and class balancing using SMOTE. These steps ensured clean, well-structured input for feature extraction and model training. The extracted features spanned time-domain, frequency-domain, and time-frequency representations. Multiple machine learning models—Gradient Boosting, Random Forest, Support Vector Machine (SVC), and Bagging—were trained and evaluated using metrics such as accuracy, precision, recall, and F1-score with cross-validation to assess generalization. The trained models were finally used to classify unseen EEG data into five sleep stages: Wake, N1, N2, N3, and REM.

**A. Dataset Acquisition** EEG signals were recorded using non-invasive electrodes placed on the scalp of human volunteers, capturing brain electrical activity during overnight sleep studies. Additional demographic data such as age, gender, medical history, and sleep habits were collected to provide contextual understanding. The data collection took place in controlled conditions to minimize noise and interference. Manual inspection of the raw EEG was performed to reject or correct artifacts where necessary, ensuring the quality of the input data. The final recordings were saved in standard EDF format for compatibility with downstream analysis. The dataset used in this study was sourced from publicly available Sleep-EDF recordings, which include EEG data and associated hypnogram annotations reflecting sleep stages.

**B. Data Preprocessing** Preprocessing was applied to prepare EEG data for modeling. Channel mapping was done to identify relevant EEG channels, particularly `Fpz-Cz` and `Pz-Oz`, and irrelevant or noisy channels were excluded. Data

cropping was applied to isolate the sleep period from wake periods based on annotation markers. The signals were then filtered to remove noise and isolate specific frequency bands important for sleep analysis.

Epochs of 30-second duration were created from continuous EEG signals to structure the data for machine learning. Sleep stage labels corresponding to these epochs were derived from hypnogram annotations. Given the naturally imbalanced class distribution in sleep stage data (e.g., fewer REM and N1 stages), the SMOTE algorithm was employed to oversample the minority classes, thereby enhancing model robustness. After preprocessing, the dataset was split into training and testing subsets. This enabled reliable training of models and unbiased performance evaluation. The output from preprocessing was then used in the feature extraction and modeling stages.

## 3.4 Model Architecture and Learning Framework

The core modeling pipeline begins with transforming the raw EEG data into structured epochs suitable for machine learning classification. Each EEG recording is segmented into 30-second epochs using the sampling frequency and annotated sleep stages. Only valid sleep stages (Wake, N1, N2, N3, and REM) are retained for downstream processing. After epoching, each segment is extracted using the `mne.Epochs` function, resulting in structured EEG data with consistent time windows. These segments are then reshaped into two-dimensional feature vectors where each row corresponds to an epoch and each column to a time-series sample across all EEG channels.

To address the class imbalance inherent in sleep stage data, the SMOTE (Syn-

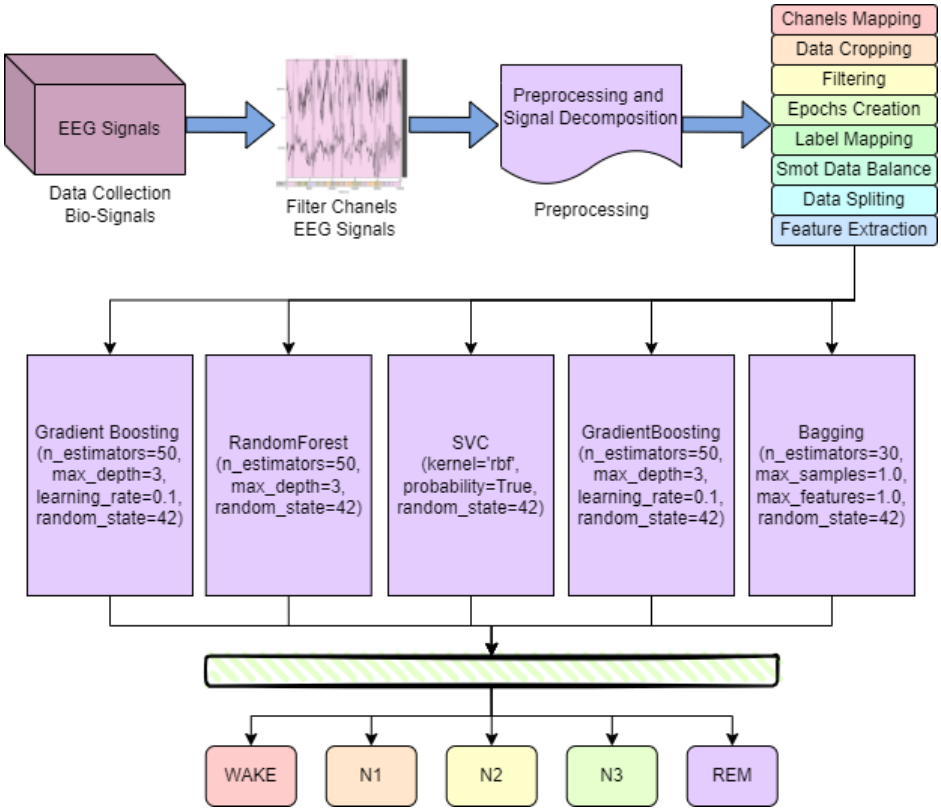


Figure 3.2. Architecture of Machine Learning Classifiers for Sleep Stage Classificatio

thetic Minority Over-sampling Technique) algorithm is applied. This balances the dataset by synthetically generating new examples in underrepresented classes. The balanced dataset is then split into training and testing sets in an 80:20 ratio.

## 3.5 K-Fold Cross-Validation

K-Fold Cross Validation 5-fold cross-validation approach was employed to evaluate the model's performance. The 5-fold cross-validation procedure divides the dataset into 5 equal subsets (or folds). In each iteration, the model is trained on 80% of the data (four folds) and tested on the remaining 20% (the fifth fold). This

process is repeated 5 times, with each fold serving as the test set once, ensuring that each data point is tested exactly once.

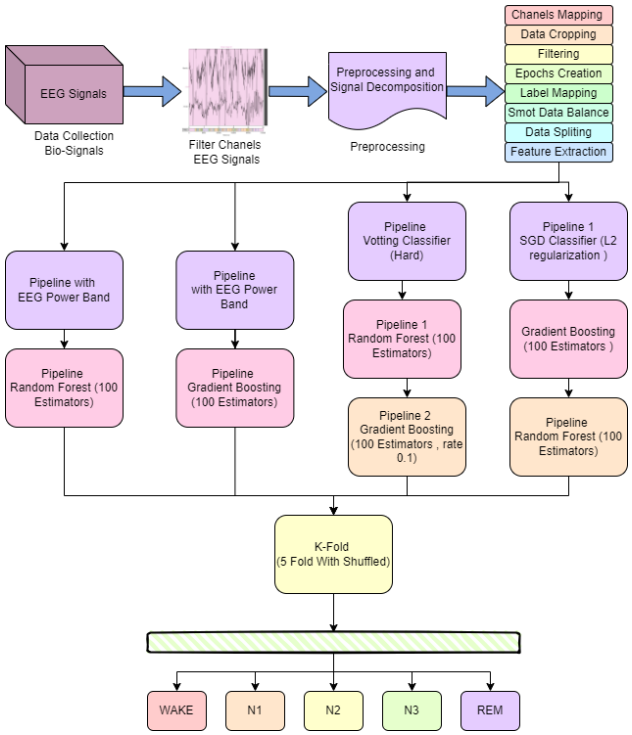


Figure 3.3. Architecture of Machine Learning Classifiers for Sleep Stage Classificatio

This approach provides a more reliable estimate of model performance by reducing the potential bias that can arise from using a single train-test split. The final performance of the model is obtained by averaging the results from each fold, offering a comprehensive evaluation of its ability to generalize to unseen data.

The advantage of K-fold cross-validation lies in its ability to provide a more robust and generalized estimate of model performance by using different subsets for training and testing in each iteration. This process reduces the risk of overfitting and ensures that the model's evaluation is based on multiple different test sets.

### 3.5.1 Random Forest with K-Fold Cross-Validation Pipeline

The model pipeline was created using `make_pipeline` from `sklearn.pipeline`, which integrates two main components: a function transformer and a random forest classifier. The function transformer is applied to extract the EEG power bands from the data, which are then used as features for training the model. The `RandomForestClassifier` was chosen for its ability to perform well with high-dimensional data, and it was set with 100 estimators to ensure robust performance. The random forest classifier was trained and tested on each fold using the 5-fold cross-validation setup, providing an unbiased evaluation of the model's performance.

### 3.5.2 Ensemble Classifier with K-Fold Cross-Validation Pipeline

The pipeline was constructed using `make_pipeline` from `sklearn.pipeline`, which integrates a function transformer for extracting EEG power bands as features and an ensemble classifier for prediction. The function transformer applies the `eeg_power_band` function to preprocess the data, and the `VotingClassifier` combines the predictions from a `RandomForestClassifier` and The ensemble classifier was trained and tested on each fold using the 5-fold cross-validation setup, offering a more robust evaluation by leveraging the strengths of both classifiers.

### 3.5.3 Gradient Boosting with K-Fold Cross-Validation Pipeline

In this section, we employed Gradient Boosting with 5-fold cross-validation to classify the EEG data. Gradient Boosting is an ensemble learning technique that

builds models sequentially, where each model attempts to correct the errors made by the previous ones. This method can improve predictive performance by focusing on difficult-to-predict instances. The dataset was split into five subsets, and each subset was used as the test set once, while the remaining four were used for training. The model's generalization ability was thus thoroughly evaluated.

A pipeline was constructed using `make_pipeline` from `sklearn.pipeline`. It incorporates a function transformer that preprocesses the EEG data by extracting power band features through the `eeg_power_band` function. The processed data is then passed to a `GradientBoostingClassifier`, which is used for prediction. The `KFold` cross-validation technique with 5 splits ensures that the model is trained and validated on different portions of the dataset, offering a reliable estimate of its performance.

### 3.5.4 Random Forest Classifier

We implemented a Random Forest classifier to establish a strong baseline model for sleep stage classification. The raw EEG recordings were segmented into 30-second epochs, and features were extracted by flattening the multidimensional epoch data. To handle class imbalance, the SMOTE algorithm was applied prior to training. The model was initialized with **100 decision trees** (`n_estimators=100`) and a fixed **random seed** (`random_state=42`) to ensure reproducibility. This configuration allowed the model to generalize well while maintaining robust performance across varying sleep stages.

### 3.5.5 Gradient Boosting Classifier with PCA

To improve computational efficiency and potentially boost model accuracy, we introduced Principal Component Analysis (PCA), retaining **95% of the explained variance** (`n_components=0.95`). The reduced feature set was passed to a **Gradient Boosting classifier** configured with **30 estimators** (`n_estimators=30`), a **maximum tree depth of 3** (`max_depth=3`), and a **learning rate of 0.1**. Class balancing was again addressed with SMOTE. The same 80-20 train-test split was maintained with `random_state=42`. This model leverages the sequential learning of weak classifiers to optimize classification performance over multiple iterations.

### 3.5.6 Ensemble Learning (Voting Classifier)

For further enhancement, we employed a soft voting ensemble that combines multiple classifiers, each with complementary strengths. The ensemble includes a **Gradient Boosting Classifier** (`n_estimators=50`, `max_depth=3`, `learning_rate=0.1`), a **Random Forest Classifier** (`n_estimators=50`, `max_depth=3`), a **Support Vector Classifier (SVC)** with an RBF kernel and probability estimates enabled, and a **Bagging Classifier** (`n_estimators=30`, `max_samples=1.0`, `max_features=1.0`). These classifiers were integrated using a **soft voting strategy** in the `VotingClassifier(voting='soft')` to average their probabilistic predictions.

Feature reduction was again performed using PCA with 95% variance retention. This ensemble approach leverages model diversity to achieve improved generalization and more stable predictions across sleep classes.

## 3.6 Results and Evaluation With Machine Learning

### 3.6.1 Random Forest

For the Random Forest model, we present the following evaluation metrics:

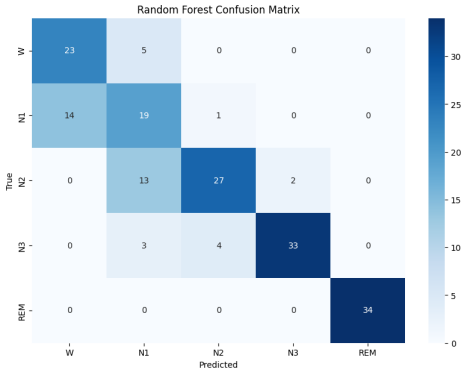


Figure 3.4. Random Forest Confusion Matrix


Figure 3.5. Random Forest Accuracy Curve

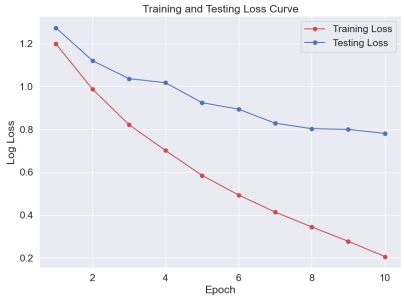


Figure 3.6. Random Forest Loss Curve

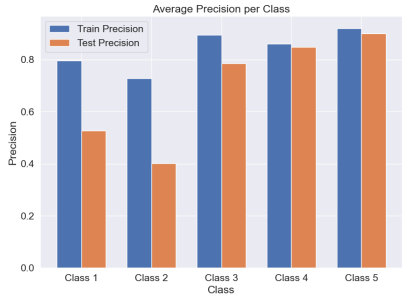


Figure 3.7. Random Forest Precision Per Class

### 3.6.2 Bagging Classifier

For the Bagging Classifier model, we present the following evaluation metrics:

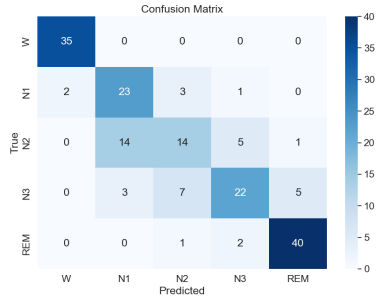


Figure 3.8. Bagging Classifier Confusion Matrix


Figure 3.9. Bagging Classifier Accuracy Curve

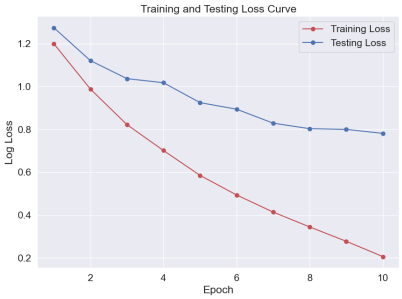


Figure 3.10. Bagging Classifier Loss Curve

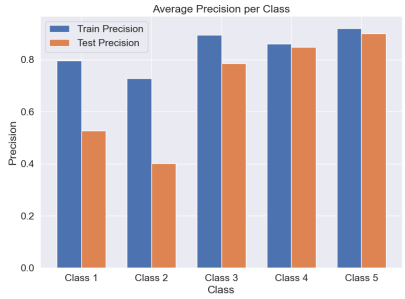


Figure 3.11. Bagging Classifier Precision Per Class

### 3.6.3 Ensemble Learning

For the Ensemble Learning model, we present the following evaluation metrics:

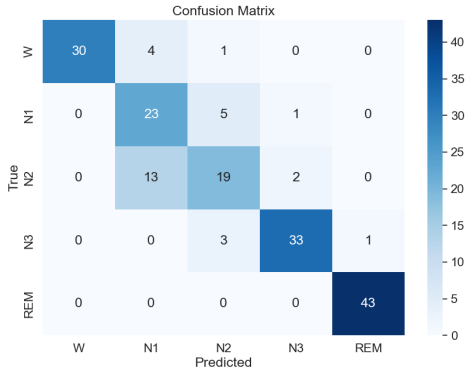


Figure 3.12. Ensemble Learning Confusion Matrix


Figure 3.13. Ensemble Learning Accuracy Curve

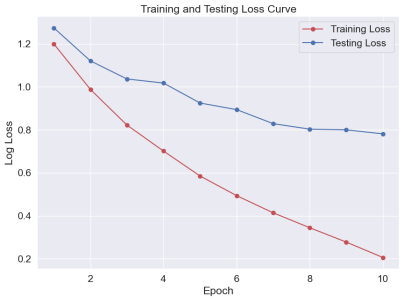


Figure 3.14. Ensemble Learning Loss Curve

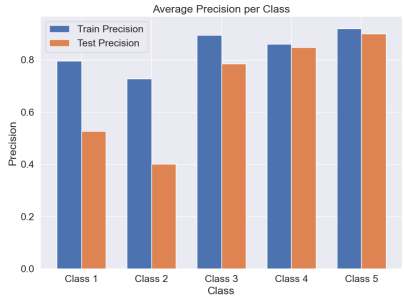


Figure 3.15. Ensemble Learning Precision Per Class

### 3.6.4 Gradient Boosting

For the Gradient Boosting model, we present the following evaluation metrics:

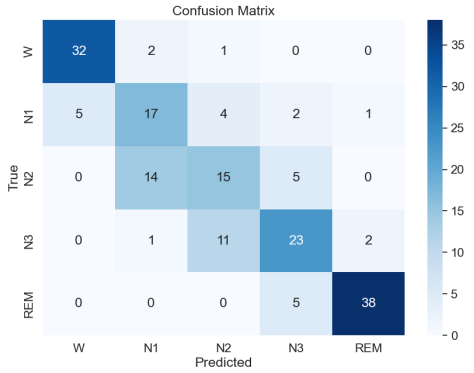


Figure 3.16. Gradient Boosting Confusion Matrix


Figure 3.17. Gradient Boosting Accuracy Curve

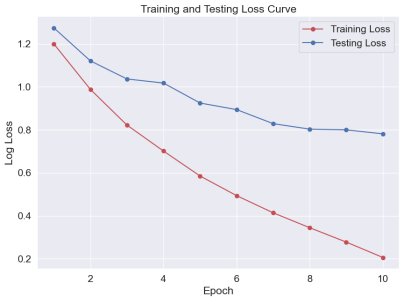


Figure 3.18. Gradient Boosting Loss Curve

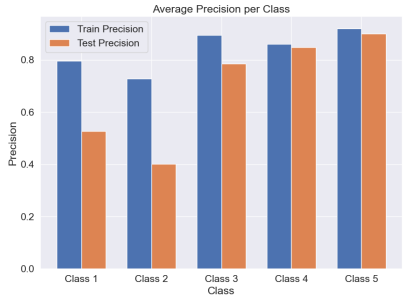


Figure 3.19. Gradient Boosting Precision Per Class

## 3.7 Results and Evaluation with K-fold Cross Validation

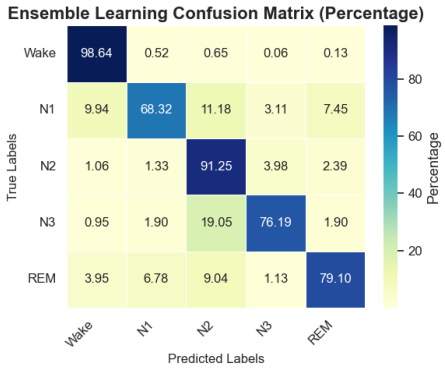


Figure 3.20. Ensemble Learning Confusion Matrix

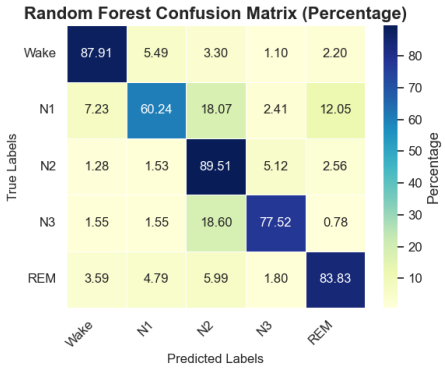


Figure 3.21. Random Forest Confusion Matrix

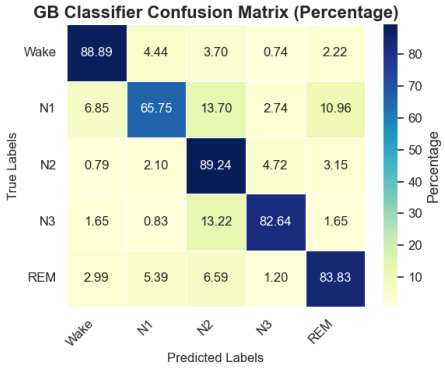


Figure 3.22. GB Classifier Confusion Matrix

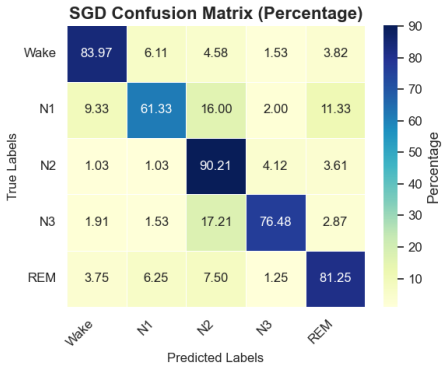


Figure 3.23. SGD Confusion Matrix

## 3.8 Comparison Sections

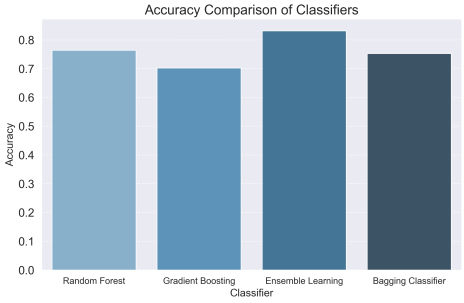


Figure 3.24. Accuracy Comparison

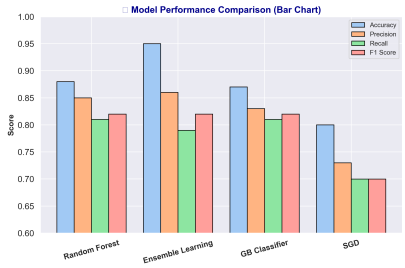


Figure 3.25. Improved Bar Chart

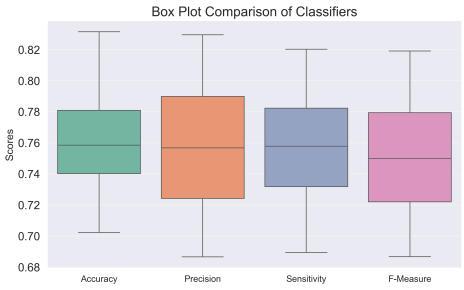


Figure 3.26. Box Plot Comparison

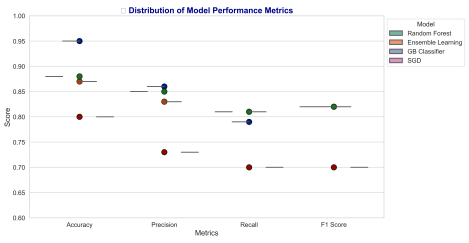


Figure 3.27. Improved Violin Plot

# Chapter 4

# Deep Neural Model for Automated Sleep Staging System using Single-Channel EEG Signal

## 4.1 Methodology

We employed the publicly available **Sleep-EDF** dataset for automated sleep stage classification. This dataset consists of two primary file types: Polysomnographic (PSG) recordings in European Data Format (EDF) and corresponding Hypnogram annotation files. The EDF files contain multi-channel biosignals, including EEG, while the Hypnogram files provide ground-truth sleep stage labels aligned with specific time intervals.

The dataset was partitioned into **training**, **validation**, and **testing** subsets using a **90:5:5** split. Given the inherent class imbalance in sleep stage distribution, we employed the **SMOTE** (Synthetic Minority Over-sampling Technique)

to synthetically balance the dataset, improving model robustness across minority classes.

### 4.1.1 Modeling Approach

We implemented and evaluated several machine learning and deep learning architectures for the classification task. One of the core models used is the **Deep Neural Network (DNN)**, designed as a fully connected feedforward architecture suitable for non-sequential input.

#### Deep Neural Network (DNN)

The DNN consists of an input layer followed by three fully connected (`Dense`) layers comprising **128**, **64**, and **32** neurons respectively. Each layer employs the **ReLU** activation function to introduce non-linearity. To mitigate overfitting, **Dropout layers** with a dropout rate of **0.2** were interleaved between each dense layer. The final output layer uses a **Softmax** activation to produce class probabilities for the five sleep stages.

## 4.2 Dataset Information

ensure efficient implementation and reproducibility, this research adheres to a well-structured computational framework, including consistent system configuration, version control, and the integration of essential libraries for data processing and deep learning tasks. The experiments were conducted on both Windows and Linux operating systems using an **Intel Core i5 11th Gen** processor (with *Iris*

*Xe Graphics* and a clock speed of **3.9 GHz**). The programming environment was built with **Python version 3.13.1**, managed via the `pip` package installer.

The study utilized the **Sleep-EDF** dataset, which contains EEG recordings and corresponding hypnograms for sleep stage classification. Prior to model training, the dataset was preprocessed into uniform, fixed-length segments. The dataset is publicly available at: `https://www.physionet.org/content/sleep-edf/1.0.0/`.

For project collaboration and version control, **GitHub** was employed, with the full implementation accessible at: `https://github.com/tanmay007thor/ContraWR`.

Several key libraries were used throughout the development pipeline:

- **NumPy (2.2.2)** – Numerical computations
- **scikit-learn (1.6.1)** – Machine learning utilities
- **imbalanced-learn (0.13.0)** – Handling class imbalance (SMOTE)
- **MNE (1.9.0)** – EEG signal processing
- **TensorFlow/Keras (2.16.1)** – Deep learning model implementation
- **Matplotlib (3.10.0)** and **Seaborn (0.13.2)** – Data visualization

This setup ensures smooth execution of machine learning and deep learning models, including BiLSTM and other architectures, for automating sleep stage classification.

## 4.3 Preprocessing Techniques

To ensure consistency in the input data, all recordings were resampled to a **100 Hz** sampling frequency. Each signal was segmented into **30-second epochs**, resulting in **3000 time steps per sample**. These samples were stored in `PKL` (Pickle) format, each approximately **48 KB** in size, enabling efficient storage and fast access, particularly beneficial for edge device deployment.

Each sample consisted of input-output pairs: the inputs ($X$) included two EEG channels—**FPZ-CZ** and **PZ-OZ**—while the labels ($Y$) represented the corresponding sleep stages. The stages were categorized into five classes: **Wake, N1, N2, N3,** and **REM**. To streamline processing, raw labels were mapped to a standardized class format.

## 4.4 Model Architecture and Learning Framework

### Deep Neural Network (DNN)

A Deep Neural Network (DNN) is a type of multilayer feedforward architecture composed of successive fully connected layers. In this model, the input layer is followed by three dense layers containing 128, 64, and 32 neurons respectively. Each dense layer uses the Rectified Linear Unit (ReLU) activation function to introduce non-linearity.

To improve generalization and prevent overfitting, dropout layers with a dropout rate of 0.2 are inserted between the dense layers. The final output layer employs a Softmax activation function to map the learned features to the five target sleep

stages: Wake, N1, N2, N3, and REM.

TABLE 4.1
Architecture of the Deep Neural Network

| Layer Type | Units | Activation Function |
|---|---|---|
| Input Layer | `X_train` | - |
| Dense Layer | 128 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 64 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 32 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Output Layer | Number of classes (e.g., 5) | Softmax |

While DNNs are effective in extracting complex feature representations from EEG signals, they are inherently limited in modeling temporal dependencies due to their feedforward nature. This limits their standalone utility for time-series classification tasks like sleep staging.

### Forward Propagation

Forward propagation computes the activations at each layer using the following equations:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \tag{4.1}$$

$$a^{(l)} = \sigma(z^{(l)}) \tag{4.2}$$

where:

- $z^{(l)}$ is the pre-activation (linear combination of inputs) at layer $l$,

- $a^{(l)}$ is the activation output at layer $l$,
- $W^{(l)}$ and $b^{(l)}$ denote the weights and biases of layer $l$,
- $\sigma$ is the activation function (ReLU or Softmax depending on the layer).

**Backward Propagation**

During training, gradients of the loss function with respect to weights and biases are calculated using backpropagation. The process involves the following steps:

$$\delta^{(L)} = \frac{\partial L}{\partial a^{(L)}} \odot \sigma'(z^{(L)}) \tag{4.3}$$

$$\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \odot \sigma'(z^{(l)}) \tag{4.4}$$

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)} \left(a^{(l-1)}\right)^T \tag{4.5}$$

$$\frac{\partial L}{\partial b^{(l)}} = \sum \delta^{(l)} \tag{4.6}$$

where:

- $\delta^{(l)}$ is the error signal at layer $l$,
- $\sigma'(z^{(l)})$ is the derivative of the activation function,
- $\frac{\partial L}{\partial W^{(l)}}$ and $\frac{\partial L}{\partial b^{(l)}}$ are gradients used for weight and bias updates.

**Activation Functions**

The Softmax function is used in the output layer to produce class probabilities:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{4.7}$$

where:

- $z_i$ is the input to the $i^{th}$ output neuron,
- $N$ is the number of output classes,
- The denominator ensures the outputs form a probability distribution summing to 1.

For hidden layers, the ReLU activation function is applied:

$$f(x) = \max(0, x) \tag{4.8}$$

where:

- $x$ is the input to the neuron,
- The function outputs $x$ if $x > 0$, and 0 otherwise.

**Recurrent Neural Network (RNN)**

Recurrent Neural Networks (RNNs) are specifically designed for modeling sequential data by incorporating feedback connections, allowing them to retain information across time steps. This architecture is well-suited for EEG signal analysis due to its ability to capture temporal dependencies.

In the proposed model, two RNN layers are employed: the first with 128 units and the second with 64 units, both using the Tanh activation function. These

layers process sequential inputs while maintaining internal memory of past computations.

TABLE 4.2
Recurrent Neural Network Architecture for Sleep Stage Classification

| Layer Type | Units | Activation Function |
|---|---|---|
| Input Layer | `X_train` | - |
| RNN Layer 1 | 128 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| RNN Layer 2 | 64 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 64 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 32 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Output Layer | Number of classes (e.g., 5) | Softmax |

Dropout layers with a 0.2 rate are introduced after each RNN and dense layer to mitigate overfitting. The dense layers, activated using ReLU, help to refine temporal features extracted by the RNN layers. The final classification is performed by a Softmax-activated output layer corresponding to the target sleep stages.

Although RNNs are effective for learning short-term dependencies in sequential EEG data, they are prone to issues such as vanishing gradients, which can hinder the learning of long-range patterns.

The operation of an RNN cell at each time step $t$ is mathematically represented as:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h) \tag{4.9}$$

where:

- $h_t$ is the hidden state at time step $t$,

- $x_t$ is the input at time step $t$,
- $W_h$ and $W_x$ are the recurrent and input weight matrices, respectively,
- $b_h$ is the bias vector,
- tanh is the hyperbolic tangent activation function.

### Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks extend traditional RNNs by incorporating memory cells and gating mechanisms, enabling them to retain long-term dependencies in sequential data. This makes them particularly effective for time-series applications like EEG-based sleep stage classification.

The proposed LSTM architecture includes two stacked LSTM layers with 128 and 64 units, respectively, both using the Tanh activation function to handle non-linear temporal patterns in the data. To prevent overfitting, dropout layers with a rate of 0.2 are placed after each LSTM and dense layer.

TABLE 4.3
LSTM Neural Network Architecture for Sleep Stage Classification

| **Layer Type** | **Units** | **Activation Function** |
|---|---|---|
| Input Layer | `X_train` | - |
| LSTM Layer 1 | 128 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| LSTM Layer 2 | 64 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 64 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 32 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Output Layer | Number of classes (e.g., 5) | Softmax |

The dense layers following the LSTM blocks (with 64 and 32 units) refine the extracted temporal features using ReLU activations. The final output layer uses the Softmax function to perform multi-class classification of sleep stages. LSTM models are particularly advantageous in capturing long-range dependencies and complex temporal dynamics, thus often outperforming simple RNNs for EEG sequence modeling.

LSTM cells operate through a series of gating mechanisms that regulate the flow of information. The key equations governing an LSTM cell at time step $t$ are:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad \text{(Forget Gate)} \tag{4.10}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad \text{(Input Gate)} \tag{4.11}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad \text{(Candidate Cell State)} \tag{4.12}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(Cell State Update)} \tag{4.13}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad \text{(Output Gate)} \tag{4.14}$$

$$h_t = o_t \odot \tanh(c_t) \quad \text{(Hidden State Update)} \tag{4.15}$$

where:

- $f_t, i_t, o_t$ represent the forget, input, and output gates, respectively.
- $\tilde{c}_t$ is the candidate cell state.
- $c_t$ is the updated memory cell state.
- $h_t$ is the hidden state at time $t$.
- $\sigma$ denotes the sigmoid activation function.

- $\odot$ represents element-wise multiplication.
- $W, U, b$ are the learned weight matrices and biases.

### Bidirectional Long Short-Term Memory (Bi-LSTM)

The Bidirectional Long Short-Term Memory (Bi-LSTM) network enhances the LSTM model by processing input sequences in both forward and backward directions, enabling the model to learn context from both past and future data. This bidirectional approach improves the model's ability to capture dependencies in temporal data, making it highly effective for tasks such as EEG-based sleep stage classification.

The architecture consists of two bidirectional LSTM layers with 128 and 64 units, respectively, both utilizing the Tanh activation function. Dropout layers (with a rate of 0.2) are applied after each LSTM layer to prevent overfitting. Following the bidirectional LSTM layers, dense layers with 64 and 32 units, using ReLU activation, are added to refine the learned features before the final classification layer, which uses the Softmax activation to classify the sleep stages.

Bi-LSTM networks are particularly well-suited for EEG-based sleep stage classification, as they allow the model to capture both past and future dependencies in the EEG signal, leading to improved accuracy in detecting sleep patterns.

The Bi-LSTM model processes input sequences in both forward and backward directions, maintaining two hidden states for each timestep. The key equations describing the forward and backward passes are:

TABLE 4.4
Bidirectional LSTM Neural Network Architecture for Sleep Stage Classification

| Layer Type | Units | Activation Function |
|---|---|---|
| Input Layer | `X_train` | - |
| Bi-LSTM Layer 1 | 128 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Bi-LSTM Layer 2 | 64 | Tanh |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 64 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Dense Layer | 32 | ReLU |
| Dropout Layer | - | 0.2 (Dropout Rate) |
| Output Layer | Number of classes (e.g., 5) | Softmax |

$$\overrightarrow{h}_t = \text{LSTM}(x_t, \overrightarrow{h}_{t-1}) \tag{4.16}$$

$$\overleftarrow{h}_t = \text{LSTM}(x_t, \overleftarrow{h}_{t+1}) \tag{4.17}$$

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t] \tag{4.18}$$

where:

- $\overrightarrow{h}_t$ is the forward LSTM hidden state at time step $t$.
- $\overleftarrow{h}_t$ is the backward LSTM hidden state at time step $t$.
- $h_t$ is the concatenated hidden state, combining both forward and backward states, used for final predictions.

## 4.5 Results and Evaluation

This section presents the performance evaluation of different deep learning models, using accuracy plots and confusion matrices for visual comparison.

### 4.5.1 LSTM Model

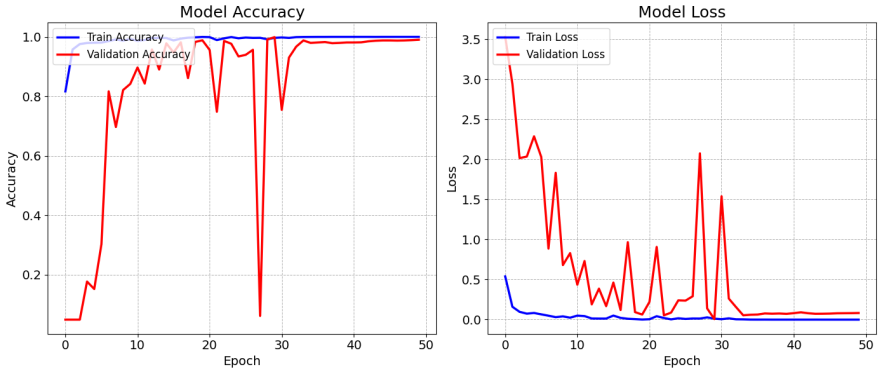


Figure 4.1. LSTM Accuracy and Loss Plot

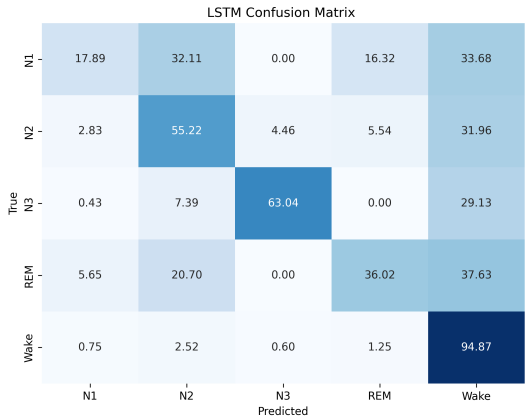


Figure 4.2. LSTM Confusion Matrix

### 4.5.2 RNN Model

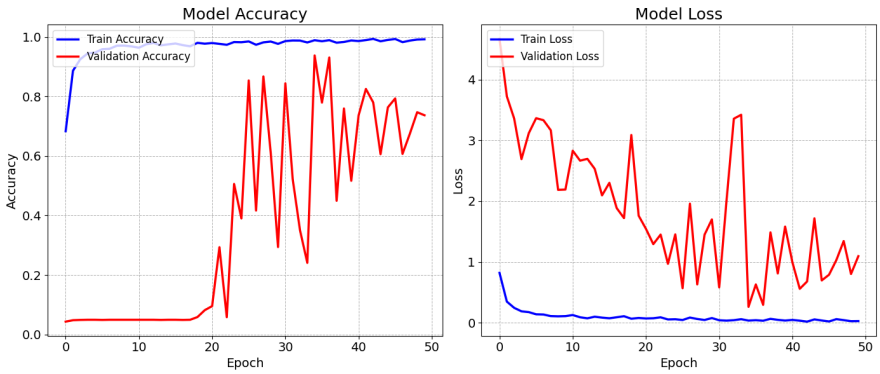


Figure 4.3. RNN Accuracy Plot

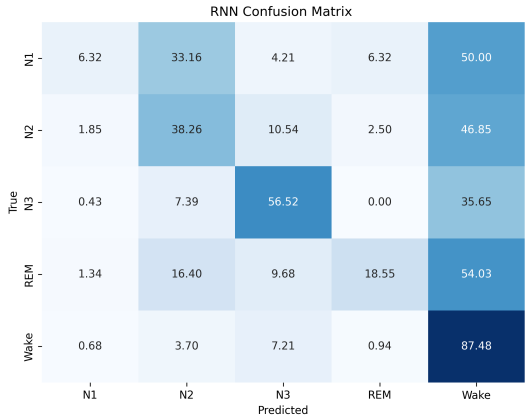


Figure 4.4. RNN Confusion Matrix

### 4.5.3 BiLSTM Model

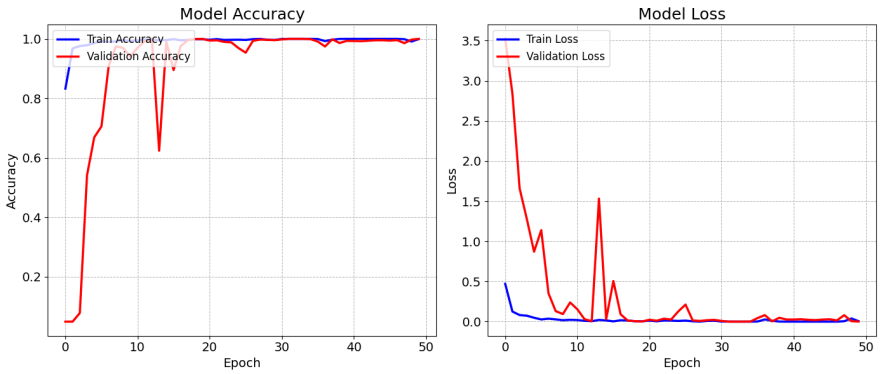


Figure 4.5. BiLSTM Accuracy Plot

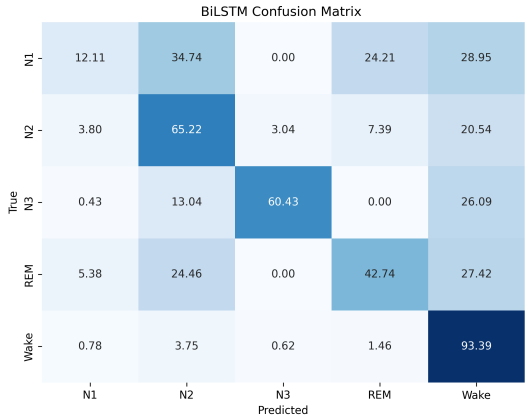


Figure 4.6. BiLSTM Confusion Matrix

### 4.5.4 Neural Network (NN) Model

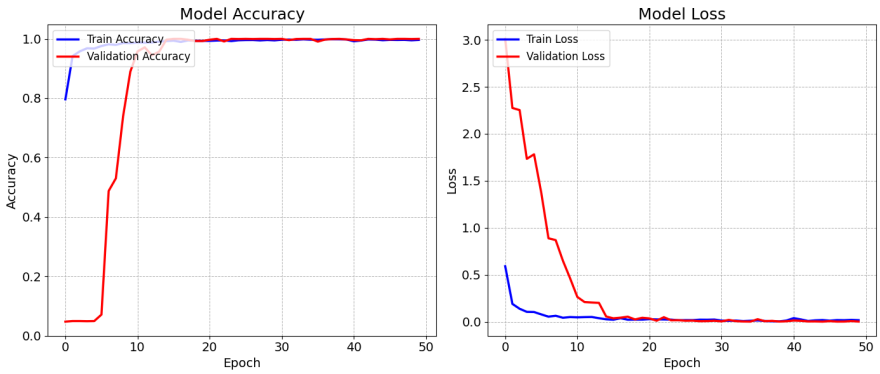


Figure 4.7. Neural Network Accuracy Comparison

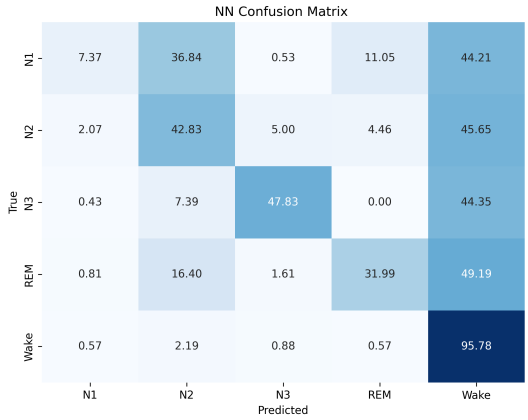


Figure 4.8. Neural Network Confusion Matrix

## 4.6 Comparison of Results

In this section, we compare the performance of various models, including Neural Networks, RNN, LSTM, and Bi-LSTM, based on several evaluation metrics such as accuracy, precision, recall, and F1 score.

### 4.6.1 Comparison Plot

The following plot presents a comparison of the accuracy scores for different models.

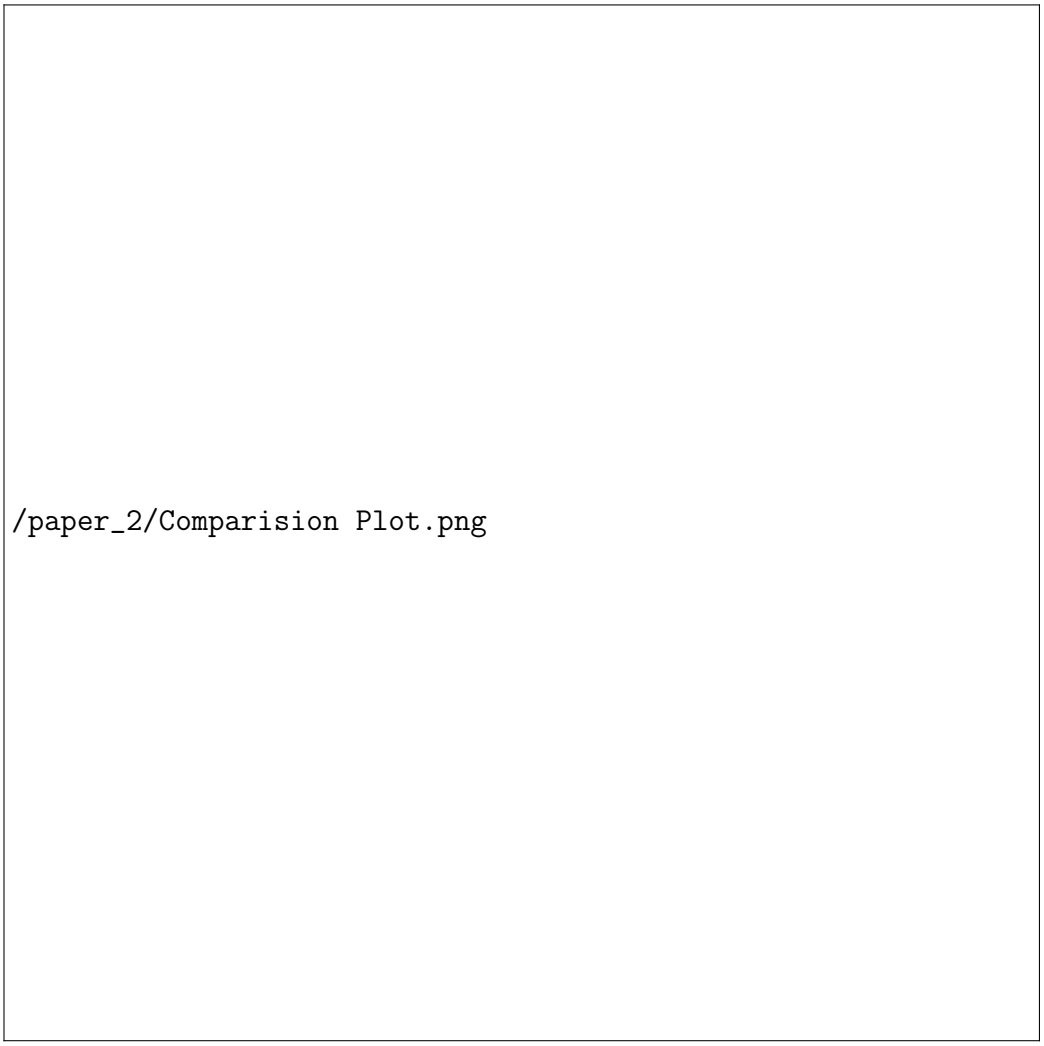


Figure 4.9. Accuracy Comparison Plot

### 4.6.2 Neural Network Comparison

The performance of the Neural Network model is compared in the plot below, showing its accuracy.

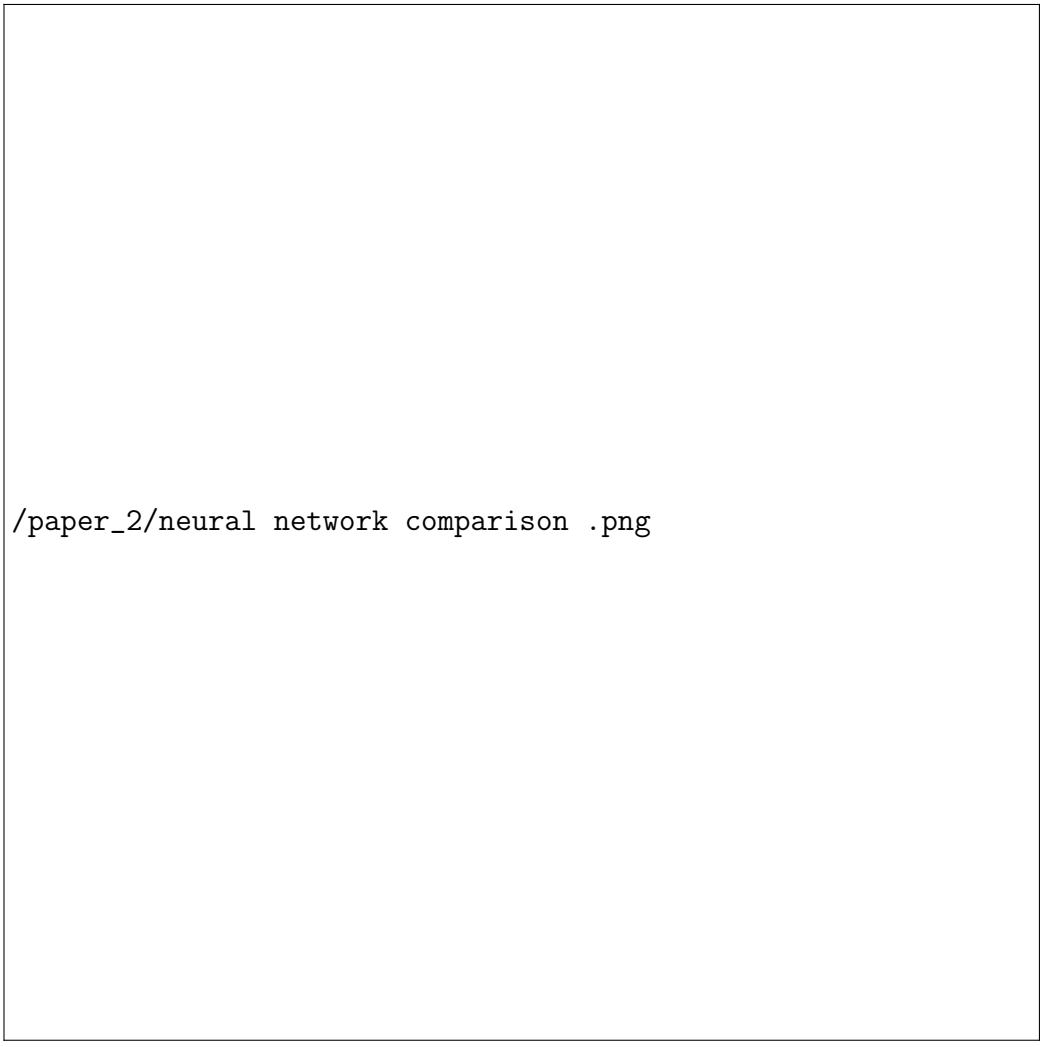


Figure 4.10. Neural Network Comparison

### 4.6.3 Bi-LSTM Performance

The Bi-LSTM model's accuracy is evaluated and compared with other models.

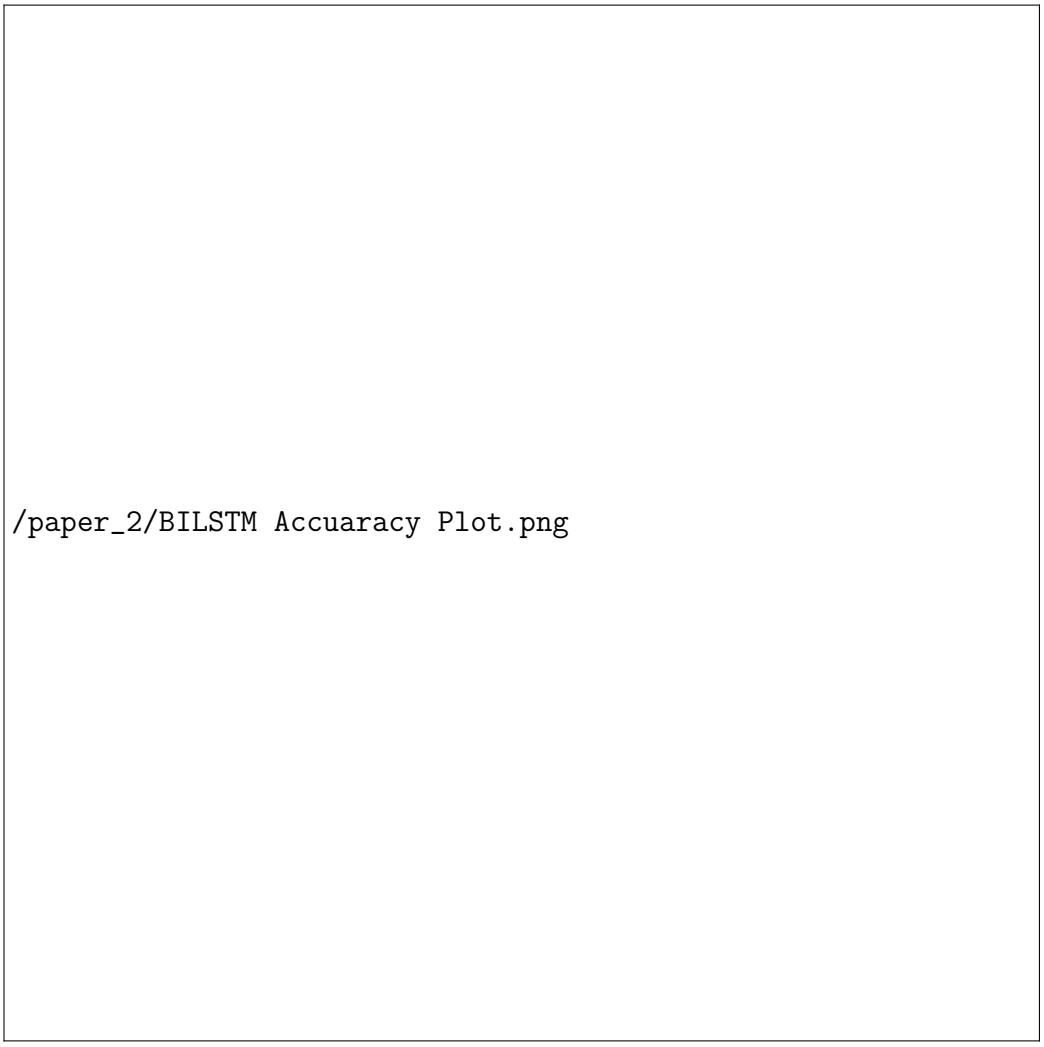


Figure 4.11. Bi-LSTM Accuracy Plot

### 4.6.4 LSTM Performance

The following figure shows the accuracy of the LSTM model, highlighting its effectiveness in comparison to other models.

/paper_2/lstm accuracy and loss plot.png

Figure 4.12. LSTM Accuracy and Loss Plot

### 4.6.5 RNN Performance

Lastly, the performance of the RNN model is shown below.

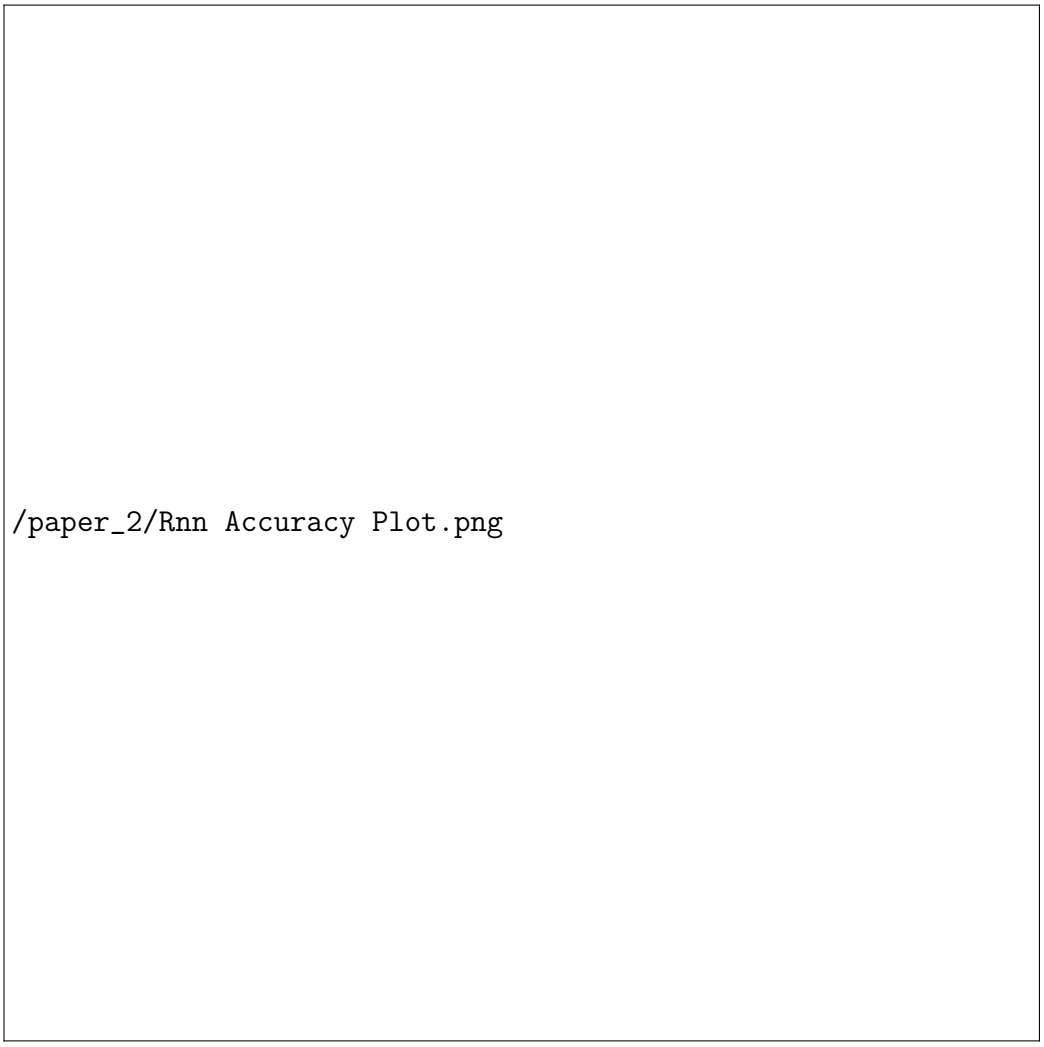


Figure 4.13. RNN Accuracy Plot

# Chapter 5

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

## 5.1 Methodology

## 5.2 Dataset Information

## 5.3 Preprocessing Techniques

## 5.4 Model Architecture and Learning Framework

## 5.5 Results and Evaluation

# Chapter 6

# 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 7

# Reference