DeepMedico™ Sleep Breathing Irregularity Detection

Final Submission Report

Prepared by: Akhilesh Y Ragate

1. Introduction

This report outlines the design and implementation of a comprehensive machine learning pipeline for detecting breathing irregularities such as apnea and hypopnea from full-night sleep recordings. The project also explores classifying sleep stages using time-aligned physiological signals.

2. Data Overview

- Participants: 5 subjects (8 hours each)
- Signals:
    - Nasal Airflow (32 Hz)
    - Thoracic Movement (32 Hz)
    - SpO_2 (Oxygen Saturation, 4 Hz)
- Annotations:
    - Breathing events: Apnea, Hypopnea
    - Sleep Profile: Wake, N1, N2, N3, REM

3. Visualization & Exploration

- Signals were read, time-synchronized, and visualized for each participant.
- Color-coded overlays mark apnea/hypopnea events on time-series plots for clear interpretation.
- Visualizations were exported to PDF for detailed review.

Key Points:

- Resampling and alignment handled differences in sampling rate.
- Visual reports enabled insight into sleep and event structure over the night.

4. Data Cleaning

- Filtering: A third-order Butterworth bandpass filter (0.17–0.4 Hz) was applied to airflow and thoracic channels.
- Purpose: Attenuated motion artifacts and high-frequency noise, isolating the natural breathing frequency range (10–24 BrPM).

5. Dataset Creation

- Windowing: 30-second segments, 50% overlap (standard in sleep analysis).
- Labeling:
    - If >50% of a window overlapped with an annotated event (apnea/hypopnea), that event's label was assigned.
    - Otherwise, the window was marked as "Normal."
- Format: Data stored as Parquet for efficiency, type safety, and ML compatibility.

6. Modeling

1D CNN Architecture

- Extracted time-local features from signal windows.
- Input: Flattened vectors (nasal airflow, thoracic movement, SpO_2).

Conv-LSTM Architecture

- Combined convolutional feature extraction with LSTM sequence modeling for richer temporal context.

Training & Evaluation

- Cross-validation: Leave-one-participant-out (LOPO) method:
    - For each fold, the model trained on 4 participants and tested on the remaining one.
- Rationale: Prevented data leakage; ensured results reflect new subject generalization.

Metrics Reported (for each class and fold):

- Accuracy
- Precision
- Recall / Sensitivity
- Specificity
- Confusion Matrix

7. Results Summary

Model	**Accuracy (mean ± std)**	**Sensitivity (Apnea)**	**Sensitivity (Hypopnea)**	**Sensitivity (Normal)**
1D CNN	XX.X% ± XX.X%	XX.X%	XX.X%	XX.X%
Conv-LSTM	XX.X% ± XX.X%	XX.X%	XX.X%	XX.X%

- All metrics computed for each participant/fold, then averaged.
- Confusion matrices and additional statistics produced per model.

8. Bonus: Sleep Stage Classification

- Implemented the same windowing and modeling, substituting sleep stage labels.
- Models evaluated for Wake, N1, N2, N3, and REM classification using LOPO cross-validation.

9. Technical Implementation

- Scripts:
  - `vis.py`: PDF signal visualization with event overlays
  - `create_dataset.py`: Filtering, window segmentation, labeling, Parquet export
  - `modeling.py`: Deep learning models and evaluation
  - `sleep_stage_classification.py`: Bonus sleep stage prediction
- Key Libraries: NumPy, Pandas, SciPy, Matplotlib, TensorFlow/Keras, Scikit-learn, PyArrow

10. Discussion

- Careful signal filtering and event overlaying were essential for reliable datasets.
- LOPO cross-validation offered a fair assessment of generalization to unseen patients, avoiding pitfalls of random splits.
- Parquet format supported fast, scalable machine-learning workflows.

Recommendations:

- Consider boosting dataset diversity in larger studies.
- Integrate additional clinical variables (BMI, age, comorbidities) for improved personalization.

11. Sample Code Excerpt

```
python

from scipy.signal import butter, sosfilt

def bandpass_filter(data, fs, lowcut=0.17, highcut=0.4,
order=3):
    nyq = 0.5 * fs
    sos = butter(order, [lowcut / nyq, highcut / nyq],
btype='band', output='sos')
    return sosfilt(sos, data)
```

12. References

- Medical signal processing best practices.
- DeepMedico™ Challenge documentation.
- Official documentation: Pandas, SciPy, Scikit-learn, TensorFlow.

Prepared by:
Akhilesh Y Ragate
July 2025