Project Title

Predicting Sleep Stage Transitions from single-channel EEG data: A feature-based pipeline that aids customers in monitoring their sleep.

One-Sentence Summary

I built and evaluated an ML pipeline that attempts to classify 30-second EEG segments into 5 distinct sleep stages by using feature-based methods designed to support real-time consumer sleep technology.

Key Result

Accurately identifying the different sleep stages from various wearable EEG devices is a historical and critical challenge for all of the consumer sleep technologies that currently exist, where real-time insights have to be derived from low-cost and single-channel signals (single channel making up the majority of consumer-grade EEG wearables). This project will attempt to tackle the problem of classifying these 30-second EEG segments into their corresponding sleep stages using ML pipelines. The findings of the project were surprising insofar as a simple model such as Random Forest, which was trained on engineered spectral features, happened to outperform a deep CNN, achieving 88.6% accuracy compared to the CNN's marginally lower 86.6%. Interestingly (and I will talk about this more in depth later), despite the CNN's greater capacity, it was disadvantaged by the more tabular nature of the input, since the engineered spectral power features were derived from Fast Fourier Transforms, not the raw time-series data that CNN's are typically known to handle well. It seems like, without access to the temporal structure of the data, the CNN ended up just adding complexity without any clear benefit, while the Random Forest was able to leverage the engineered feature set more efficiently, which would highlight the value of a model baseline and the alignment of model-data.

Motivation and Reflection

This project is meant to directly address some of the issues that were highlighted in my earlier EEG classification work in our first project, where some of my unclear goals, unjustified model choices, and flawed validation techniques ended up leading to rather weak conclusions. I have tried to implement many of the suggestions that the professor made on my first project, primarily by using engineered spectral features and also making use of a baseline model with the intent of contextualizing the output of the CNN. I also attempted to handle the problem of short-term temporal context that plagued the previous project by implementing a prev_stage feature, and tried to avoid any potential data leakage by virtue of using subject-wise validation. In a bid for transparency, I also used permutation importance in order to better understand the contributions

that individual features had on the model's behavior. Overall, I hope that this project reflects a shift from my earlier exploratory modelling and embodies the spirit of learning from one's mistakes.

Hypothetical Audience

This project was designed with all of the emerging sleep tech companies in mind, with teams at places like Meta, and Fitbit (Google) who are now working to make sleep tracking increasingly more useful and accurate for their everyday users. I would say that my target audience would be comprised of UX designers who are primarily thinking about how to best present their sleep insights to the people using their product, as well as data scientists who actually build the models underlying these wearable products. Accurate sleep stage classification could enable these UX designers to better visualize recovery trends or even recommend behavior changes based on the actual sleep patterns of their clients. Data scientists would benefit from a more robust and interpretable modeling pipeline that seeks to avoid some of the common pitfalls, such as future-data leakage and overfitting.

Data Summary

The dataset I used is named the Sleep-EDF Expanded (PhysioNet, 2018), and the raw data, as well as more information, can be found at the link below:
https://physionet.org/content/sleep-edfx/1.0.0/sleep-cassette/#files-panel
The Sleep-EDF dataset is a publicly available collection of 197 full-night sleep recordings that were originally created in order to study just how sleep changes with age and with medication. It includes data from two distinct studies:

- The first study is labelled as Sleep Cassette, which documents 153 nights of at-home recordings from healthy adults who are aged 25–101, collected during the normal daily routines and by using portable EEG cassette recorders.

- The second study is called Sleep Telemetry, and consists of 44 nights recorded in a lab, studying the effects of a sedative drug called temazepam in people who were experiencing minor sleep issues.

Each night includes both:

- Polysomnography (PSG) data: EEG brainwaves (gathered from the Fpz-Cz and Pz-Oz scalp locations, two standard placements of electrodes during EEG scans), eye movement data (EOG), muscle tone (EMG), and sometimes even respiration and body temperature.

- The dataset also includes Hypnograms, which are essentially expert-labeled sleep stages that are scored every 30 seconds using the standardized “Rechtschaffen and

Kales system". Here are the different labels: W (Wake), N1 (Light Sleep), N2 (Deeper Sleep), N3 (Deep/Slow-Wave Sleep), and REM (Rapid Eye Movement sleep).

Our Focus:
This project is going to be exclusively focused on the first sleep Cassette subset. These 153 nights were recorded in natural, home environments using the single EEG channel (Fpz-Cz), which would more closely resemble the consumer-grade sleep tracking devices I seek to emulate. In this dataset, each EEG channel is sampled at 100 Hz, which provides 3000 data points per every 30-second sleep segment.

This dataset was intentionally chosen due to:

- It being large and diverse enough (100,000+ labeled segments) to be able to train and validate our ML models
- It offers real-world generalizability, unlike the other lab data that included drug influence.
- Aligning more with the use case of wearable sleep tech

Source Credibility:
The dataset is hosted by the organization PhysioNet, which appears to be a trusted, NIH-funded repository for physiological data.

Data Preprocessing A

The first stage of preprocessing the data involved transforming the original Sleep-EDF .edf files (containing hours of continuous biomedical data) into more structured and labeled epochs suitable for our eventual machine learning. To do this, I used the load_and_qc.py script, which actually performs some data ingestion, along with also rescaling, quality checking, and various other formatting. As mentioned earlier, each of the recordings in the Sleep-EDF data subset consists of a polysomnography (PSG) file (which is just the raw EEG signal itself) and also a hypnogram file (which is the associated sleep labels). The script then automatically detects and pairs these files (prioritizing the Fpz-Cz EEG channel, known to be the standard in sleep analysis) and then will default to the first available EEG channel if the Fpz-Cz electrode recording happens to be missing. Once loaded, the script will then detect whether or not the EEG is in raw volts, which it will then rescale to microvolts, which is the standard unit in the practice of neurophysiology (it also makes it easier for me to interpret). Then, importantly, it divides the continuous signal that we have into 30-second epochs, each to be aligned with the corresponding sleep stage that we got from the hypnogram.

Each of these resulting epochs is then going to get passed through some basic quality controls, which aim to flag the recordings that feature excessive flatline periods above 90% that might be indicative of faulty sensors or some disconnected electrodes. From then on, any recording that fails these criteria is then going to be excluded from further processing. For all of these valid recordings, the script goes on to save two NumPy arrays per subject in the experiment: one array that contains the segmented EEG epochs and another that contains the label. Finally, the

cleaned output is then organized into a processed_data/ folder, where each of the 153 subjects will be given a unique identifier and also a set of accompanying metadata.

This step is important because it transforms the continuous and unparsable physiological recordings into inputs that are ML-ready. Without this processing, none of the models would be able to ingest the data. We want to ensure that early-stage hardware issues have been dealt with before they propagate into the training of our models.

Data Preprocessing B

The second major step in our data processing pipeline was more focused on actually refining the dataset through removing any extraneous wake-stage segments and then conducting exploratory analysis on the actual remaining (non-wake) sleep epochs. This was all handled by the new_EDA.py script, which effectively filtered out all of the stage 0 epochs (corresponding to Wake) from the preprocessed EEG data mentioned in the first section. Next, each subject's EEG recordings were then scanned for values of non-zero sleep stages (N1 - REM), and then only those epochs went on to be preserved. I then built a new directory structure under sleep_only_data/ that attempts to mirror the subject-by-subject layout of the original data, but now only contains the sleep-relevant segments. Each of these subject folders includes separate EEG channel files (such as EEG_Fpz-Cz.csv).

Once this sleep_only_data dataset has been created, the same script then went on to run several analyses in order to better understand both the data distribution and the spectral characteristics (Just the distribution of signal power across the different frequency bands). First, a distribution analysis quantified the relative frequency of each of the sleep stages. It found that approximately 25% of all recorded epochs represented sleep (the rest was the wake data that we since removed), with Stage N2 being the most dominant (at around 45%), followed by deep sleep (N3/N4) and REM sleep (~15% each), and then light sleep (N1) at around 10%. According to the research I did, these figures happen to closely align with the consensus around sleep architecture, confirming that the dataset does actually cover the full range of our physiological sleep states, albeit with an inherent class imbalance (the disproportionate waking stage that was removed).

Next, I performed a spectral profile analysis across all retained sleep stages (an example of a spectral feature would be increased delta wave power during deep sleep or an elevated alpha signal during your relaxed wakefulness). Through the aggregation of all of the bandpower values (delta through gamma) for each stage across our subjects, I confirmed expected neurophysiological patterns. Delta power was indeed highest in N3/N4, validating the presence of the deep and slow-wave sleep. Additionally, theta and alpha were most pronounced in the REM and N1 stages, which aligned with their known roles in memory processing and drowsy states. Unsurprisingly, the higher-frequency bands (beta and gamma) also decreased progressively with the onset of deep sleep. To account for some of the subject-to-subject variability that could have occurred, I then computed the coefficient of variation (CV) for each band and each stage. Deep sleep stages ended up exhibiting the lowest CV for delta (at around 25 to 30%), while REM and N1 showed higher variability, especially in the alpha and gamma

bands, which is also reflective of the natural heterogeneity inherent to some of the lighter or transitional modes of sleep.

Finally, in the same file, we applied ICA to all of the 5 frequency-band features across just 15 of the subjects in an attempt to investigate whether some nonlinear combinations of these features could end up doing better in the sleep separation task. I decided to pick ICA over PCA since EEG signals are known to arise from multiple overlapping sources in the brain, and ICA has been designed to disentangle those statistically independent components, not like PCA, which just tries to capture directions of maximum variance without the consideration of source separation. Since we are conducting EEG analysis, in which the goal is often just to isolate the physiologically meaningful patterns, I thought ICA would be a better fit. In our results, ICA ended up revealing two dominant components in the data: delta power separated deep sleep from lighter sleep, and theta and alpha activity distinguished REM sleep from non-REM stage sleep. With important implications for the rest of the project, this transformation confirmed that spectral features carry separable class information and also provided a sort of biologically interpretable space where data can be projected. At the end, this ICA mixing matrix was saved for the potential downstream use in feature extraction and eventually modeling.

Data Preprocessing C

The final step in the data quality control was performed in order to ensure that all of the data used for training and evaluation was free from any physiological or hardware-related artifacts. This was accomplished using the file named artifact_screening.py script, which essentially systematically scans every one of the epochs for any signs that may be indicative of data corruption. The first thing that the script does is load all of the EEG data, going on to perform (both on the recording-level and on the epoch-level) screening for three major categories of artifact that are common in the computational neuroscience space: flatlines, amplitude outliers, and spectral noise.

At the aforementioned recording level, the script aggregates all of the epochs from a single night and then computes the percentage of a flatline signal (which is indicative of disconnected electrodes), the proportion of samples that happen to exceed the plausible amplitude bounds +-500 μV, and finally the presence of the excessive high-frequency power or powerline interference (Needed since in EEG recording is it common to have unwanted electrical noise caused by muscle artifacts or nearby electronics at 50 and 60Hz respectively). In alignment with the current EEG literature, recordings are then excluded entirely if more than 5% of the data is flatlined, if the amplitude outliers end up exceeding 10%, or if the noise ratios end up surpassing 30%.

Through the combination of frequency-domain, time-domain, and statistical artifact detection, this final step in the quality control forms the final check before the modeling begins, which is intended to protect the integrity of the entire ML pipeline.

Feature Engineering

Once we had a clean dataset without the wake stage, the next step was to then design features that would help our models be able to tell the difference between the different sleep stages in a way that was both biologically meaningful and also ML-friendly. This was done using generate_spectral_features.py, a script that reads each subject's EEG recordings and then moves to add several new types of features in addition to the existing ones. These new features are then going to be saved in a structured folder, organized by both subject and by EEG channel, with accompanying metadata so that everything ends up traceable and easy to load later for further training.

The original features in each 30-second segment included things like the mean and the standard deviation of the EEG signal, as well as the amount of brainwave activity (the aforementioned power) in all of the different frequency ranges: delta, theta, alpha, beta, and gamma. Interestingly enough, these raw power values can end up varying a lot between different people due to some natural differences in head shape, skin resistance, or electrode quality. In order to make things more consistent, the script also calculates the log-transformed versions of each power band. In doing so, we normalized each band by the subject's average and then applied the logarithmic transformation. The intent behind this was to keep the important changes between the stages, while also reducing the chance that our model would learn some irrelevant differences that occurred between the subjects.

Additionally, I also created three different ratio features that aim to capture the relationships between the bands instead of just their absolute size:

- The delta/theta wave ratio should help to pick out deep sleep stages like N3 and N4, where delta power becomes increasingly dominant.

- The theta/alpha ratio is more useful for identifying the REM stage of sleep, which tends to show an elevated theta.

- Slow/fast wave ratio compares the slower waves (such as delta and theta) compared to the faster ones (like alpha and beta), which should hopefully give a general sense of how "deep" the sleep is.

In addition to these spectral features, I also added a "previous stage" feature that records what sleep stage the subject was going through during the previous 30-second epoch in their file. This should hopefully give the model a sense of "context" in their respective patients' sleep cycle, without giving away anything in the future. Since there's no "previous stage" for the very first 30-second segment, I just left it blank for those cases.

All together, this process ended up creating a rich set of inputs that happen to be normalized across all of the subjects (grounded in some real sleep science), designed to help the model focus on more meaningful differences between all of the stages.

Feature Selection

I chose to apply feature selection not just to reduce dimensionality, but to improve the model's generalization and focus on learning what the most informative aspects of the EEG signal are. I believe that, given the small sample size relative to the number of engineered features I have, including all inputs would have likely risked overfitting and, in turn, diluted some of the important patterns. In order to do this, I chose to use two widely accepted filter-based feature selection methods. The first is mutual Information (MI), and the second is the ANOVA F-test (Both chosen since multiple other studies cite their simplicity and their interpretability in EEG modelling). This all happens in feature_reduction.py. These methods were chosen for both their simplicity and their interpretability. MI essentially measures how much knowing the value of a feature can reduce the uncertainty about the corresponding sleep stage label (useful for power ratios that spike in specific sleep states). On the other hand, ANOVA F-tests are meant to evaluate whether or not the mean feature values differ significantly across their classes, a strong indicator of class separability.

We applied each of these methods to the full engineered dataset using a 70/30 train-test split and then imputed any of the missing values using some column means. The selected features were then standardized in order to ensure fair treatment by both classical and by neural models. We then tested feature subsets of different sizes (k = 10, 20, 30, 40, and 50) and then ultimately focused on MI-50 and ANOVA-20, respectively, since these had the best model performance. In runtime, the top features selected by both of the methods included the mean, the standard deviation, and the multiple band powers and log-transforms, confirming that the core spectral properties of the EEG happen to be strongly correlated with all of the sleep stages. Notably, many of the most important features that the tests selected for were ratios and log-normalized band powers, which goes to support our earlier EDA findings that these transformations would end up enhancing the stage separation.

This process allowed us to eliminate the redundant or noisy features while retaining those most relevant to the sleep-stage classification. The reduced feature sets were then saved as NumPy arrays and used as input to both the models in the next section.

Model Comparison: CNN vs. Random Forest for EEG Sleep Stage Classification

To explore the best sorts of modeling strategies for the EEG-based sleep stage classification, I chose to implement both a Convolutional Neural Network (CNN) and a Random Forest classifier in order to compare two fundamentally different but interestingly complementary approaches to the EEG-based sleep stage classification problem. First, the Random Forest was chosen as a strong, interpretable baseline that would be well-suited to the tabular, pre-engineered features I created, such as the spectral power bands and statistical summaries extracted from the EEG signal. RF models are robust to noise and handle nonlinearity well, with strong out-of-the-box performance that would make them ideal for validating whether or not our

engineered features were informative. In contrast, the more complex CNN represents a deep learning approach. While CNNs are typically applied to raw time-series data (such as the raw EEG signals), we adapted them here in order to operate on structured input to test whether spatial filters over our frequency-domain features could actually uncover complex relationships missed by the tree-based model. I paired these two models in the hopes that I would be able to evaluate the value added by deep learning in this context and then ground our performance claims in some semblance of interpretability and ML technique. I would like to think that this dual-model approach also ends up mirroring the real-world workflows, where simpler models are able to serve as robust baselines before more complex systems end up being deployed.

Model Architectures and Design

The CNN used in our project follows the Sleep-1D-CNN architecture that was proposed by Mohammed & Sagheer in 2024, consistently applying a deep 9-layer design across all of our conditions. Specifically, it consists of 3 convolutional blocks, each having its own 1D convolution. Additionally, the CNN also features a pooling layer to reduce the dimensionality and a LeakyReLU activation. These are then followed by the global average pooling and two fully connected layers before the final softmax output. The model processes all of the input features, such as the spectral powers and the ratios, as a one-dimensional sequence, which allows the CNN to be able to detect the local patterns across different feature types. The model also features a stopping and learning rate scheduling to better stabilize training and prevent overfitting. This structure was used consistently for both the ANOVA-20 feature and the MI-50 feature inputs.

The Random Forest, somewhat contrastingly, was a straightforward ensemble model that used 100 decision trees. Its design is naturally conducive to tabular and structured data, and it benefits from the strong regularization that is achieved through bagging and feature subsampling.

Dataset and Feature Space

Both of the models were trained and were evaluated using the same preprocessed and artifact-screened EEG dataset that was derived from the Sleep-EDF database in the beginning. As previously mentioned, two feature selection techniques were used: Mutual Information (MI-50) and ANOVA F-test (ANOVA-20) in order to select the most informative features from a larger set of spectral, statistical (mean and std), and engineered features (e.g., spectral ratios and previous sleep stage context). This yielded 16-dimensional input vectors for the CNN and direct inputs for the Random Forest. The CNN was evaluated on a held-out test set of 78,595 epochs.

CNN Performance Summary

Across both of the feature sets, the CNN demonstrated very consistent performance:

- **MI-50**: Accuracy of **86.31%**, test loss of **0.4304**
- **ANOVA-20**: Accuracy of **86.63%**, test loss of **0.4228**

These scores indicate that the CNN generalized well to the unseen data. However, the performance gains from using 50 features over 20 were minimal, which would suggest that the model sees diminishing returns past all of the top-ranked features.

The classification report for ANOVA-20 showed some particularly strong results for the different sleep stages:

- **Stage 1 (N1)**: Precision 0.89, Recall 0.92, F1-score 0.91
- **Stage 4 (N3/N4)**: Precision 0.93, Recall 0.94, F1-score 0.93
- **REM (Stage 5)**: Recall was 0.00 due to lack of predicted samples, which could be an indicator of the model's difficulty handling extremely rare classes

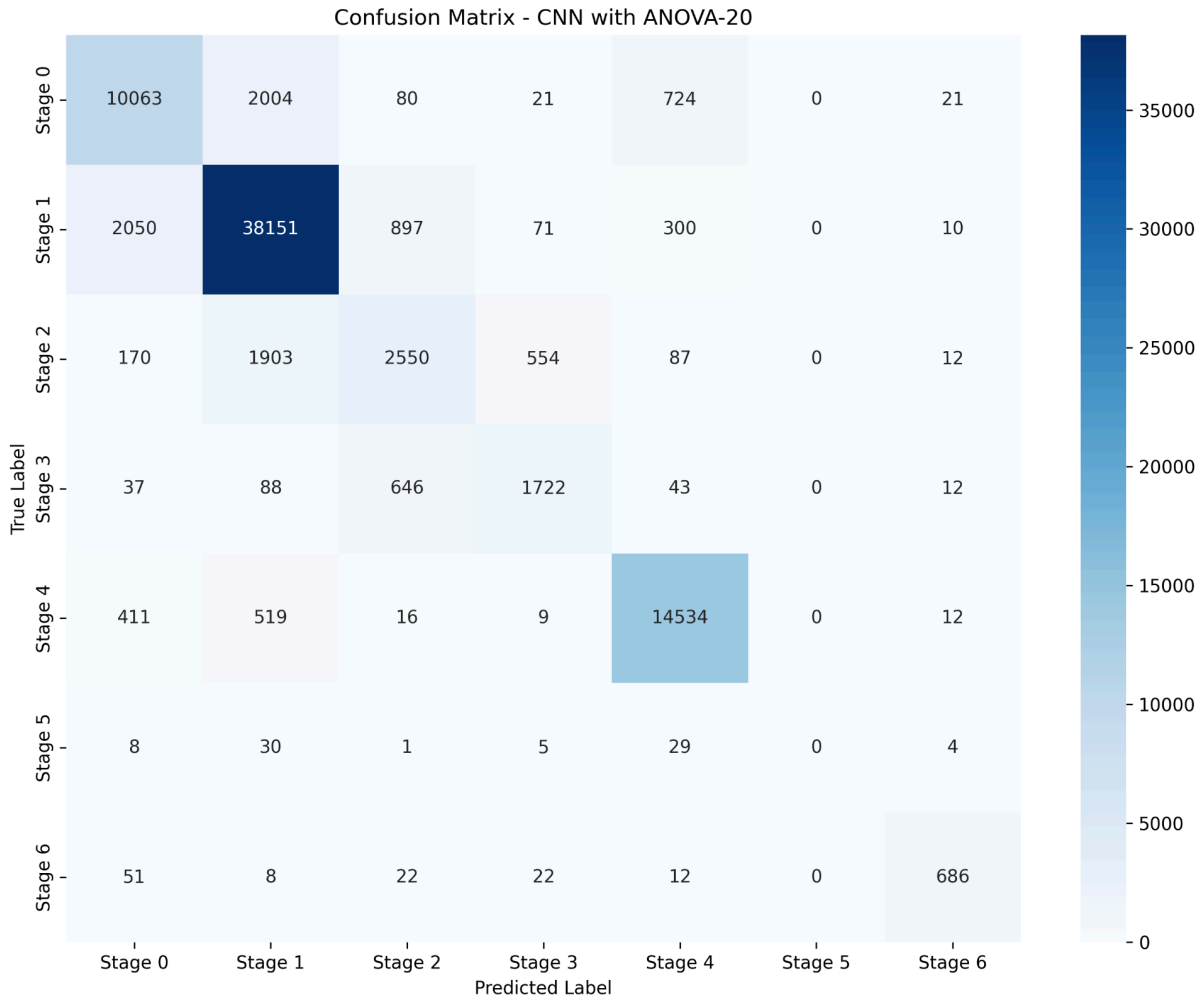

Figure 1: Confusion Matrix – CNN with ANOVA-20 Feature Selection

Figure 1 shows the confusion matrix for the CNN that was trained on the ANOVA-20 features. The model performs very well on common stages like Stage 1 (light sleep) and Stage 4 (deep sleep). However, like most EEG models, it struggles with the less frequent stages, like REM (Stage 5)

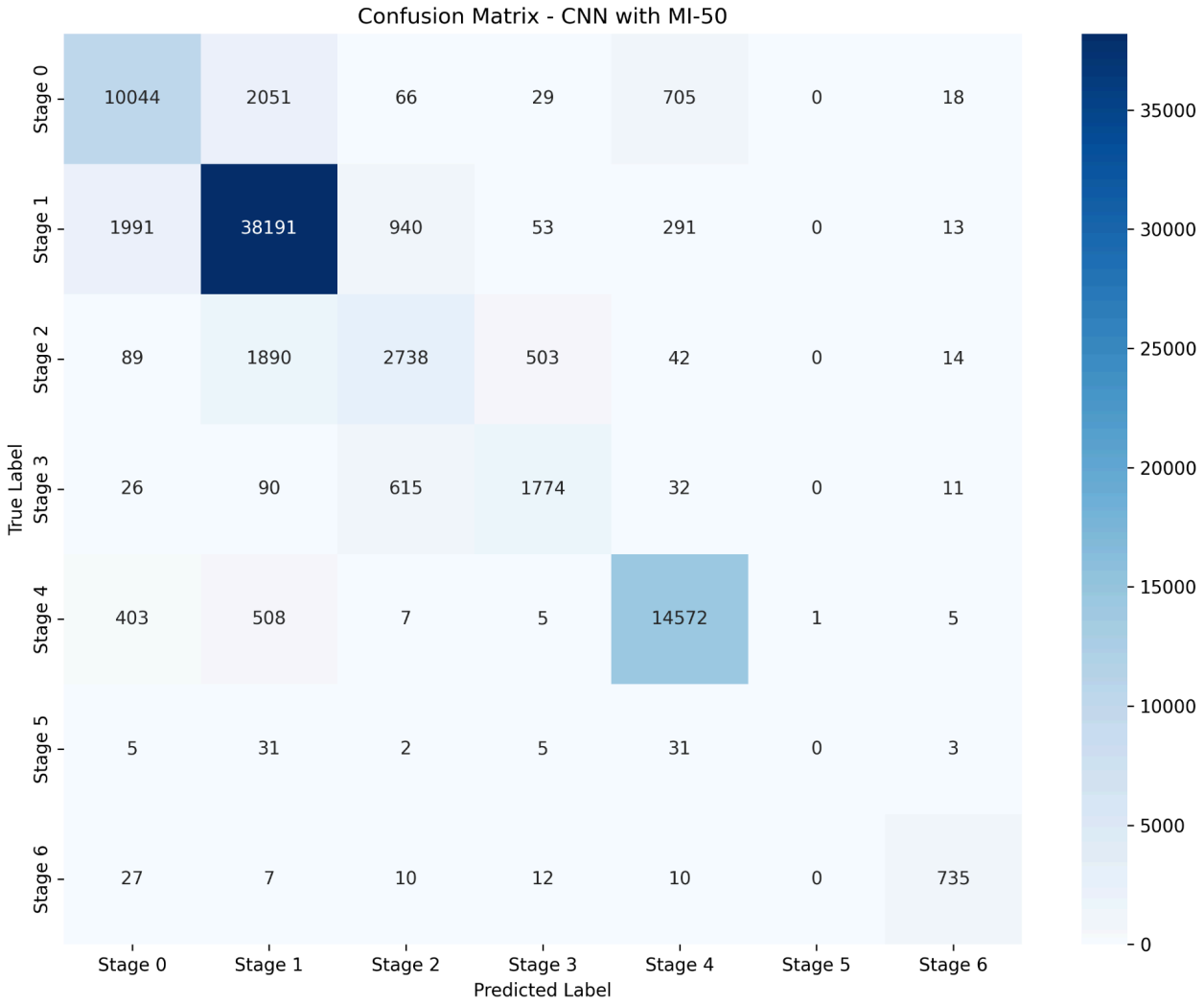

Figure 2: Confusion Matrix – CNN with MI-50 Feature Selection

Figure 2 displays the confusion matrix for the CNN trained on the MI-50 feature set this time. As with the ANOVA-20, the model excels at predicting the 1st and 4th stages. However, confusion remains between the neighboring stages. REM remains poorly predicted due to the class imbalance in the dataset.

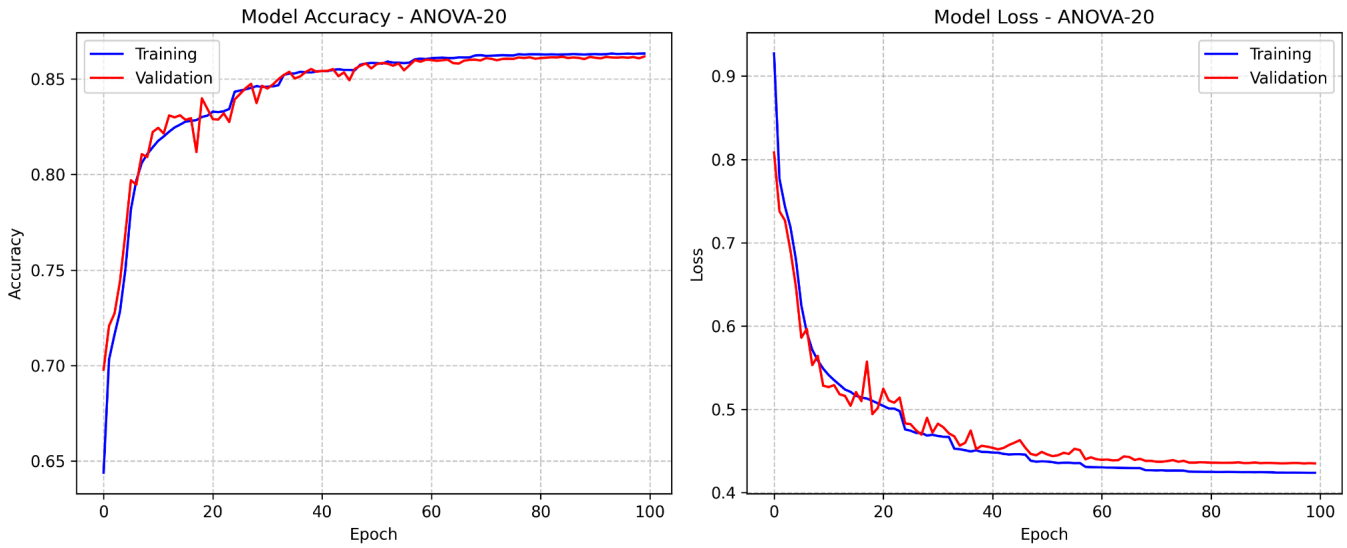

Figure 3: Training and Validation Curves – CNN with ANOVA-20

Figure 3 is a training history plot for accuracy and for loss using the ANOVA-20 features over 100 epochs. Both the validation and training curves converge smoothly, which could suggest minimal overfitting and stable learning. The accuracy plateaus around that 86% mark, and the loss reaches a floor near 0.42, confirming what I would consider to be effective model convergence.

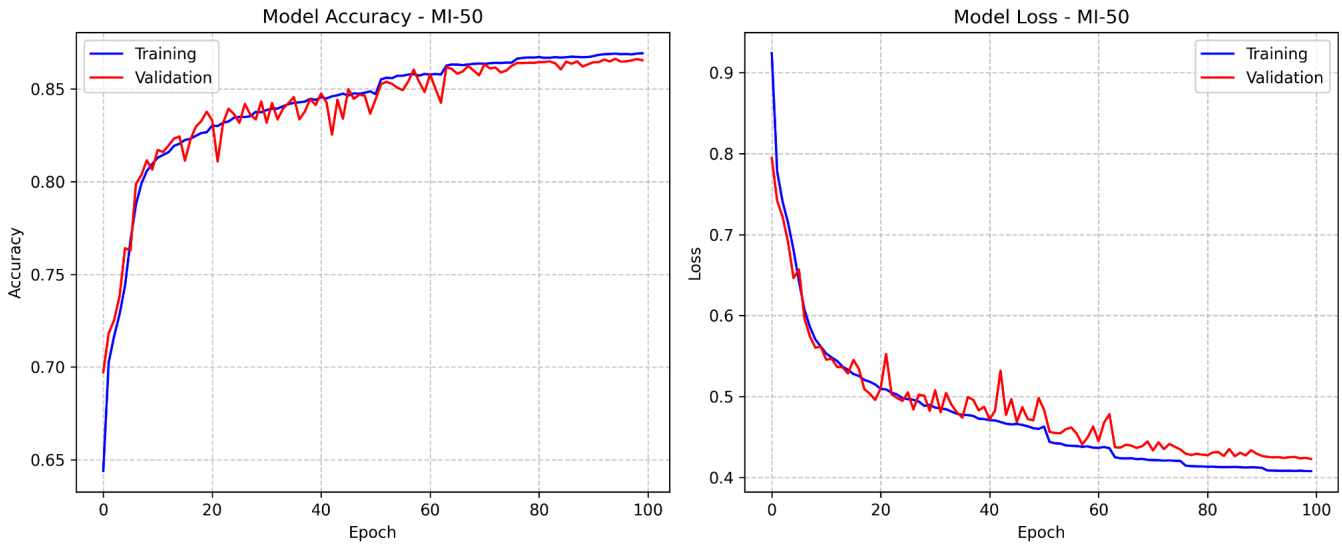

Figure 3: Training and Validation Curves – CNN with MI-50
The findings are very similar to the previous ANOVA-20 graph, although the slight fluctuation in validation accuracy around epoch 20–60 could be reflective of the model's sensitivity to more redundant features when compared to the ANOVA subset.

Random Forest Performance Summary

Surprisingly, the Random Forest model outperformed the CNN across both of the feature sets:

- **MI-50**: Accuracy of **88.61%**
- **ANOVA-20**: Accuracy of **88.61%**

This consistent result indicates strong robustness to the feature set choice and could highlight the Random Forest's ability to be able to handle these sorts of tabular EEG-derived features effectively. Moreover, the classification reports for both feature sets showed impressive metrics for each of the classes.

- **Stage 1 (N1):** Precision 0.91, Recall 0.93, F1-score 0.92
- **Stage 4 (N3/N4):** Precision 0.94, Recall 0.94, F1-score 0.94
- **Stage 2 (N2):** F1-score 0.66, slightly higher than CNN achieved
- **REM (Stage 5):** As with the CNN, performance was rather negligible (F1-score 0.00) due to the extreme class imbalance in the dataset.

These results confirm that the Random Forest, although thought to be somewhat “dumb” by deep learning standards, is actually well-suited for structured EEG feature data and happens to offer a compelling combination of interpretability and performance.

Overall Results summary.

Model	Feature Set	Test Accuracy	F1
CNN	MI-50	86.31%	0.68
CNN	ANOVA-20	86.63%	0.69
Random Forest	MI-50	**88.61%**	0.73
Random Forest	ANOVA-20	**88.61%**	0.73

Feature Importance

The code for the CNN also included a permutation analysis, which revealed that out of all of the features, the prev_stage dominated the model performance, contributing nearly 46% of predictive value in both MI-50 and ANOVA-20 runs. The top features after prev_stage were slow_fast_ratio, delta_theta_ratio, and log-transformed spectral power following up the rear. Additionally, class-specific analysis showed that:

- N3 (deep sleep) was the most influenced by delta_power and slow_fast_ratio
- Stage transitions were predicted with help from prev_stage

Interestingly, the Random Forest model, using its built-in feature importance, ended up highlighting spectral ratios and power bands as the most valuable of the predictors, but performed much more reliably without depending that heavily on a single dominant feature.

Why Did the Random Forest Model Outperform the CNN?

- **Data Format Compatibility**: Like previously mentioned, the Random Forest model is just better suited for this kind of tabular, pre-engineered sort of features, whereas CNNs typically shine with raw time series data. Since the data had already been summarized into these spectral and statistical features, the CNN did not have this temporal structure to exploit.

- **Class Imbalance Handling**: The Random Forest model inherently handles some of these class imbalances (such as REM) better through its stratified splits and its weighted voting. The CNN training required more deliberate balancing or class weighting in order to match this robustness.

- **Model Simplicity and Stability**: It could be that Random Forests are known for their low variance and their interpretability, while the CNNs are in need of tuning of their architecture, learning rates, and more. In our case, the simpler model happened to be more effective and more efficient.

Final Takeaway

The unexpected success of the Random Forest model over the CNN in this project is illustrative of a fundamental principle in applied machine learning: the idea that perhaps choosing the most complex model isn't always the right move, and that what matters is actually choosing the right model that fits the structure of the data.

Our dataset happened to consist of these sorts of hand-engineered and domain-informed features that we extracted from EEG segments. These features were deliberately designed in order to summarize the most relevant physiological signals that could aid in differentiating the sleep stages. By the time that these features actually reached our model, much of the important temporal and spectral structure had already been sort of distilled into these compact, fixed-length vectors. As it stands, this flattened, tabular format is exactly the kind of data that these classical machine learning models like our baseline Random Forests excel at.

CNNs, by contrast, are designed to pick up bits and pieces of the local structure, such as spatial edges in images or temporal sorts of patterns in raw waveforms. CNNs are special since power lies in extracting new representations from the raw or minimally processed inputs. But when they are applied to some already-summarized features, as we had in our case, their representational capacity is definitely underused. For what it is worth, the CNN still performed quite well, but I think it is reasonable to conclude that the added complexity offered only some marginal gains.

All in all, this project reaffirms what we have heard all year in class, that model choice should be data-aware and purpose-driven (Let's not model for modelling's sake!). Rather than chasing

some complexity, it may be a better idea to build on strong feature engineering and align the model assumptions with the given data structure.

Limitations and Future Work

While this project demonstrates promising results in classifying sleep stages using EEG-derived features, several limitations temper the conclusions and offer avenues for further development.

Imbalanced Class Distribution:
As shown in our data summary early on in the report, classes like REM (stage 5) were severely underrepresented. In the results section, this imbalance impacted the respective recall and F1-score for these stages, with the CNN often failing to predict any of the samples from class 5. While our engineered features and model designs attempted to compensate for this shortcoming, further improvements could have possibly involved some sort of oversampling or perhaps synthetic data generation in order to better address these minority stages

Feature Leakage in for prev_stage:
One of the most predictive features, prev_stage, contributed over 45% to the CNN's performance according to the permutation test. While I suppose it was helpful for capturing temporal transitions, this feature could risk introducing some form of hidden dependencies. In the future, it might prove advantageous that systems test performance both with and without prev_stage in order to assess its true value in generalization scenarios.

Dataset Demographic:
From what I have seen online, the Sleep-EDF dataset is a widely used benchmark, but its demographic actually skews older (from 25–101 years) and also includes the sleep captured only from the Fpz-Cz scalp electrode. As in a lot of research, this limits generalization to a younger population or any multi-channel EEG systems. Additionally, since all of the recordings were conducted with professional-grade equipment in sleep-lab-like conditions, results may not translate that cleanly to a more noisier, consumer-grade wearable sensor.

Validation and Robustness of Results

In order to evaluate the robustness of our models and, in turn, verify that their performance is not due to chance, we implemented a structured 5-fold cross-validation procedure using both the MI-50 and the ANOVA-20 feature subsets. This was also supported by a label-shuffling control test so that I could explicitly check whether our models were learning true structure or were simply overfitting to some noise.

True vs. Shuffled Labels Performance

With the MI-50 feature selection set, the CNN achieved a mean accuracy of **85.14% (+-0.13%)** and a mean F1 score of **84.76% (+-0.17%)** across all of the five folds. In comparison, training the model on the same input data but this time with shuffled sleep stage labels resulted in a significantly lower accuracy of **52.78%** and F1 score of **36.46%**. The validity also included the mean difference in accuracy, 32.36 percentage points, yielded a t-statistic of **497.65**, which confirms that the CNN's performance is statistically significant and is unlikely to have resulted from random chance. ANOVA-20 had nearly identical results.

Generalization and Outlier Control

I believe that the small standard deviation in the performance across all of the folds would suggest that the CNN results are likely not driven by any specific outliers or particular subsets in our data. Each fold was performed within a narrow margin of the average, which likely means that the model generalized well across the different partitions of the dataset.

As previously mentioned in the limitations section, the subject pool remains fixed. To fully ensure generalizability to a wide variety of different individuals, perhaps future work should implement leave-one-subject-out or leave-one-night-out validation techniques in order to properly simulate exposure to some new and completely unseen data distributions.

Comparison to a "Dumb but Reasonable" Baseline

As previously mentioned, in order to benchmark the CNN's performance, we compared it to a baseline Random Forest classifier that was trained on the same MI-50 and ANOVA-20 features. Thus, this part of the validation procedure confirms the CNN's predictions to be grounded more in real EEG patterns rather than in chance, but also reveals that some simpler models may be better suited for this feature representation.

Disclaimer:

I used ChatGPT in order to help generate some of the plots and also to help with understanding and debugging (including many print statements) some of the foreign concepts, such as the CNN code and the loading of EEG data.

Here is the link to the Repo:

https://github.com/SpeerioCheerio/AML-Final