Non-Invasive Hormone Prediction Using EEG-Based CNNs

UNIVERSITY OF LINCOLN

SCHOOL OF COMPUTER SCIENCE

Maxwell Hill

26584263

maxwellhill2004@icloud.com

School of Engineering and Physical Sciences College of Health and Science University of Lincoln

Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science with Honours in Computer Science

Supervisor: Christos Frantzidis

May 8, 2025

Contents

List of Figures

Project Proposal

1.1 Abstract

This study aims to predict sleep-related hormone levels utilizing electroencephalogram (EEG) data. This could be an effective alternative, as it would provide a less invasive means of capturing vital biomarkers than blood sampling methods. Deep learning has proven to be very useful in means of EEG analysis, specifically by utilizing Convolutional Neural Networks (CNNs) we could correlate key hormones such as cortisol and testosterone. This approach eliminates the need for frequent blood draws as model prediction could be done in real-time, allowing for continuous hormonal sampling. Significant findings could have a meaningful effect for endocrine research.

1.2 Introduction

1.2.1 Significance

Hormone level measurement is important in many different sectors, for example endocrinologists use insulin (INS) to help treat diabetes, and thyroid conditions. Sports science can use prolactin (PRL) to indicate excessive overtraining, gastroenterologists can use glucose (GLU) to predict pancreatitis and liver disease.

1.2.2 Impact

Improvements on enhanced hormonal monitoring can have a groundbreaking impact in many fields. Some of these include; Sports and Physical performance could enable athletes to fine tune recovery and injury prevention, monitoring key hormones such as cortisol and testosterone. Chronic stress is a rising issue, often escalating to long term health problems such as cardiovascular disease. Polycystic Ovary Syndrome (PCOS) could be significantly improved with the use of specialized treatment and management.

1.2.3 Current Monitoring:

Current traditional approaches for hormone monitoring such as blood sampling have many limitations. These include the lack of real-time hormonal analysis, time delays after sample collection and side effects such as lightheadedness and soreness. Lab analysis of blood is also not cost effective, making it difficult for application in more fields.

1.2.4 Problem

Research has shown that there is significant correlation between several hormones and EEG patterns, particularly ones that impact cognitive function. Some of these hormones include; sex hormones (estrogen, progesterone, and testosterone) and metabolic hormones (cortisol, insulin and glucose). These hormones heavily impact factors such as alertness, focus and anxiety.

1.2.5 Solution

We could solve this by utilizing electroencephalogram (EEG) deep learning analysis, specifically using convolutional neural networks (CNN's). By dividing EEG data into epochs representing set time periods, we can quantify wave data into spectrograms and supply them to a neural network. This approach could have many advantages including richer temporal resolution and improved patient comfort. Analysis of multi-channel EEG is suited to convolutional neural networks (CNN's), as it makes it possible to extract complex features such as wave morphology

and patterns. Data collection and preprocessing are crucial in ensuring a high accuracy CNN. EEG data will need to be normalized and removed of any noise. Training data is likely to be relatively small so synthetic data generation and efficient pre-training is vital to improve model performance. Computational evaluation on the model to analyze for potential model pruning, whill allow the CNN to be deployed in real-world applications. Investigation and comparison of previous pre-trained CNN models will be very valuable to ensure a high prediction accuracy.

1.3 Objectives

Objectives

Data Collection:

- *Specific*: Obtain labeled EEG and hormonal level dataset.
- *Measurable*: Ensure dataset is high-quality and sufficiently large for diverse participants.
- *Achievable*: Use University of Lincoln resources and open datasets.
- *Relevant*: High-quality data is critical for accurate deep learning models.
- *Time-Bound*: Complete by end of 2024.

Model Development:

- *Specific*: Develop CNN model to predict hormone level trends from sleep EEG data.
- *Measurable*: Achieve hormonal level classification accuracy of 8 ± 2.
- *Achievable*: Build on prior EEG-based deep learning studies for preprocessing and feature extraction.
- *Relevant*: Non-invasive hormone prediction is the study's core goal.
- *Time-Bound*: Complete by early February 2025.

Validation:

- *Specific*: Compare EEG-based hormone prediction reliability to blood sampling.
- *Measurable*: Assess using R-squared, mean absolute error, and root mean squared error.
- *Achievable*: Cross-validate with independent EEG dataset.
- *Relevant*: Model reliability is essential for clinical use.
- *Time-Bound*: Complete within 2 months post-model development.

Project Time Planning:

Current project deadline approximations are indicated by a GANTT chart although further changes to the project and optimizations through research are to be realized in the literature review, where a PERT diagram and software development methodologies can be described.

Risk analysis

	Insignificant 1	Minor 2	Moderate 3	Major 4	Critical 5
Certain 5					Lack of Academia
Likely 4		Clinical Acceptance	Technical Application	Model Overfitting	
Possible 3		Model Bias	Data Quality	Time Management	Access to Data Sets
Unlikely 2	Model Training Failure		Computation	Data privacy	Ethical Concerns
Rare 1					

Figure 1: *Risk Table*

1.4 Risk Assessment

Low Risk (1–3)
- *Model Training Failure*: Power outages may lead to incorrect models. Mitigate by retraining and leveraging high-uptime cloud SLAs.

Moderate Risk (4–7)
- *Model Bias*: Small datasets may bias performance. Use diverse representation techniques.
- *Computational Cost*: High computational needs affect accuracy. Optimize with efficient architectures and pruning.

High Risk (8–13)
- *Clinical Acceptance*: Lack of clinical standard acceptance may block deployment. Engage regulatory groups and review studies.
- *Data Quality*: Noise or missing data impacts predictions. Apply filtering and quality checks.
- *Data Privacy*: Sensitive data raises GDPR/HIPAA concerns. Use secure storage and anonymization.
- *Time Management*: Delays affect timelines. Set clear objectives and timelines.
- *Technical Implementation*: Implementation issues cause delays. Use well-documented technologies.
- *Ethical Concerns*: Data misuse risks legal issues. Establish professional reviews and guidelines.

Extreme Risk (14+)
- *Model Overfitting*: Overfitting risks patient safety. Use cross-validation and large datasets.
- *Lack of Hormonal Prediction Studies*: Limited studies complicate planning. Leverage related field literature.
- *Lack of High-Quality Datasets*: Data scarcity reduces effectiveness. Source datasets or generate synthetic data.

Table 1: *Key Term Matrix for Literature Search*

	Group 1	**Group 2**	**Group 3**
Term 1	EEG	Machine Learning	Hormonal Biomarkers
Term 2	Electroencephalography	Deep Learning	Cortisol
Term 3	Sleep EEG	Convolutional Neural Networks	Cholesterol
Term 4	Neural Oscillations		Insulin
Term 5	NREM Sleep		

Literature Review

Abstract

This review seeks to critically evaluate methodologies for sleep EEG analysis. Prior work should contextualise and justify the projects implementation, identifying previous strengths and limitations. Areas that should be evaluated are as follows; signal processing, feature extraction in relation to EEG signals and machine learning analysis. Investigating and comparing emerging research and traditional methods will be crucial for a successful project. Additionally this review will also asses the current status and objectives of the project.

2.1 Introduction

Recent advancements in EEG signal analysis - particularly the use of deep learning have enhanced the ability to capture features from complex non-linear non-stationary data. Recently Convolutional neural networks (CNNs) have been used in EEG classification, demonstrating great success detecting abstract features directly from raw EEG signals. This method has proven to be superior to traditional methods being used widely by many scholars (Amin et al. 2019, Hou et al. 2020, Chriskos et al. 2020).

2.2 Search Methodology

To search for relevant literature I used EBSCOhost's search engine, as supplied by the University Of Lincoln. A key term matrix is demonstrates key words chosen; grouped via synonymous terms. The key words are described in Table 1 or in this boolean expression;

This search resulted in only 8 results, of which there was 5 academic journals. After language exclusions and duplicates only a single academic journal was left. The journal (Fu et al. 2022) focuses on mental stress levels induced from the Trier Social Stress Test (Allen et al. 2017) and is not particularly useful to justify my implementation. After evaluating my initial search I can deduce that the current research on EEG-hormonal classification is scarce, so I needed to redefine my search terms. As it would not be accurate to evaluate literature with largely different classification scenarios; I decided to only analyse papers on related to stress, disease diagnosis and BCI classification. Even though these influence EEG signals slightly differently we can still gain valuable insight on building and integrating a robust ML pipeline. My second search query was the following:

This search resulted in 106,981 results, and 83,589 academic journals. This is now too many! Even with language exclusion and type exclusions there was still 78,920 results. To narrow down the search and find valuable sources I only considered the academic peer review publisher "Frontier" which evaluates techniques used by many high quality studies. This narrowed my search to 617 results; significantly more manageable. The following evaluation is directed by these studies.

Table 2: *Second Key Term Matrix for Literature Search*

	Group 1	**Group 2**	**Group 3**
Term 1	EEG	Machine Learning	Stress
Term 2	Electroencephalography	Deep Learning	Disease Diagnosis
Term 3	Sleep EEG	Convolutional Neural Networks	BCI
Term 4	Neural Oscillations		
Term 5	NREM Sleep		

2.3 A review of EEG signal preprocessing

Typically the first step of preprocessing is segmentation as EEG data is typically very large and continuous, the data is split into chunks and downsampled which makes it easier to load the data into memory and significantly lowers the compute cost (Chriskos et al. 2020). Typically 70-90% of the raw EEG data can be removed with very small accuracy impact on classification (Abdullah et al. 2022). EEG data is very sensitive to noise, external sources can destroy meaningful patterns in the data so it is important to remove noise for an accurate model. The authors (Khosla et al. 2020) comparatively evaluate traditional filtering techniques used by a range of research disciplines, one of these being the band pass filter method. It is a commonly used to classify depression patients (Jebelli et al. 2018, Ding et al. 2019, Mumtaz et al. 2017) and has yielded impressive results with accuracies of (86.62%, 79.63% and 98%). The Butterworth is a similar method but (Oppenheim et al. 1997) and has proven to be very successful in sleep staging studies (Chriskos et al. 2017, Wang et al. 2019, Tian et al. 2017), with accuracies of (89.07%, 80% and 91.4%). ICA or Independent Component Analysis Pruning can help reduce specific enviromental artefacts such as eye blinks and recording equipment. (Chriskos et al. 2017, Jebelli et al. 2018). ICA also significantly reduces the dimensionality of the data which helps the computational and time cost of training a model.

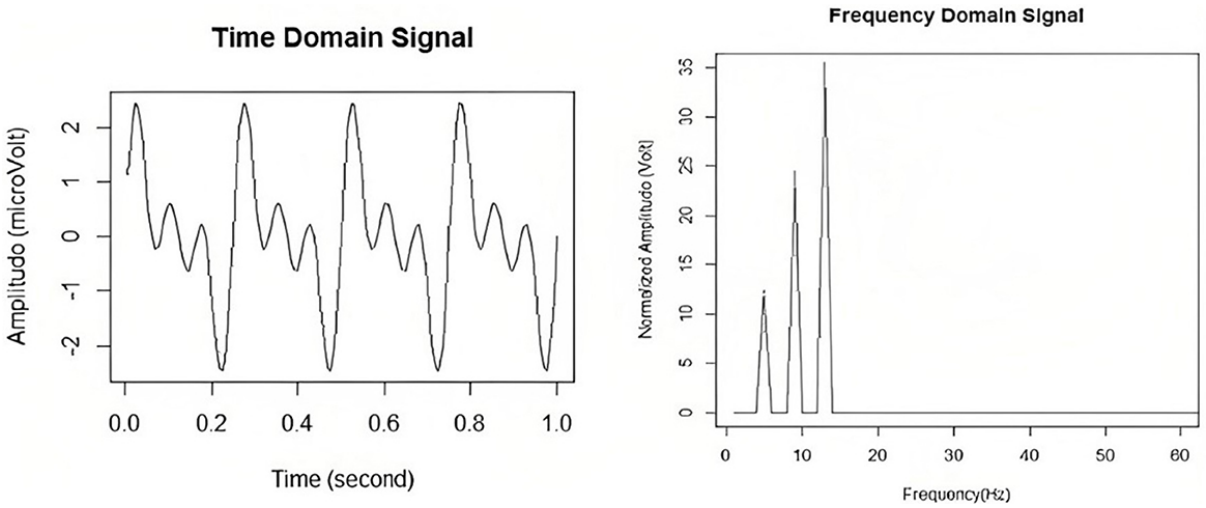

Figure 2: *EEG signals in the time and in the frequency domain. Taken from (Delimayanti et al. 2020)*

2.4 A review of Feature selection and extraction

Anupreet Kaur & Sridhar (2023) author a peer reviewed study that covers the performance of different feature extraction methods. The review analyses many features over different dimensions, and claims that the spatial domain is the most powerful when it comes to EEG analysis, citing the authors (Dornhege et al. 2006, Jin et al. 2019, Khan et al. 2019). We must understand

however that each dimension exponentially increases the size of our data and computation so forth. Feature selection is a difficult part of signal processing, and is individual to the dataset and research domain. With little prior literature on EEG hormonal classification, there is not a standardized processing pipeline to do this.

2.5 A review of Machine Learning Analysis

Traditional classifications using time and/or frequency-domain features can yield accuracies in of up to 80-90%. It would be unfeasible and beyond my expertise for me to investigate manual feature extraction methods for my project. This is why I believe that the usage of deep learning could be successful. CNNs are suitable for complex EEG classification tasks, and have achieved good results being used by scholars (Amin et al. 2019, Hou et al. 2020, Chriskos et al. 2020). They are designed to use minimal preprocessing, although can be improved by redicing the signal-noise ratio to improve accuracy, (Hajinoroozi et al. 2016) established a CNN using only raw EEG signals to predict a drivers cognitive state. However (Chriskos et al. 2017) had remarkable improvements in sleep stage research using 3 dimensional inputs to yielding a higher accuracy compared to standard 1D implementations. No prior work was found on standard CNN architecture in the EEG hormonal analysis so the task will require different CNN architectures to be evaluated, of which can be inspired by alternative research fields.

Figure 3: *RGB image formation using 3 dimensions; graph metric, and energy ratio value. Taken from (Chriskos et al. 2020)*

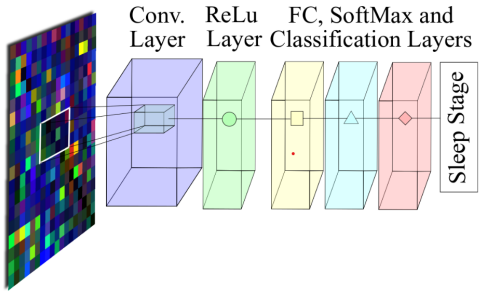

Figure 4: *Diagram of typical CNN architecture used for EEG analysis. Taken from (Chriskos et al. 2020)*

Progress Update

3.1 Introduction

This section of the report evaluates the status of my research and implementation against the goals defined in the project proposal . The focus so far has been learning neuroscience analysis implementation techniques from other successful studies. Progress defined in proposal represented in the GANTT chart has been met, where research of data preprocessing techniques and model design have been stated in this report.

3.2 Implementation Frameworks

While many scholars choose to use frameworks such as EEGLAB and BRAINSTORM (Tadel et al. 2011) as they are robust tools for analysis and provide a graphical user interface, I believe that the MNE-Python framework a better choice for my project. My familiarity with the Python ecosystem will allow me to integrate other powerful tools such as TensorFlow and SciPy. This will also allow me to provide more transparency to my methodology than traditional methods, enabling future research. Possible concerns could be that this method is not as validated as **EEGLAB/Brainstorm**. This can be justified by existing studies from authors such as (Gramfort et al. 2013, Banihashemi et al. 2021, Bowen et al. 2019) which include benchmarking MNE-Python against existing tools.

3.3 Data Handling

As described within my proposal, data privacy was designated as "High Risk" as data being analysed raises ethical and legal concerns (GDPR, HIPAA). Improving transparency by utilising the MNE-Python framework does not mean that ethical or methodological diligence can be sidelined. Raw EEG files should be ingested from a trusted cloud storage platform with appropriate GDPR/HIPAA certifications in a secure manner. Platforms such as Amazon Web Services (AWS) and Google Cloud Platform (GCP) should be utilised for their data residency controls and audit management.

3.4 Supervisory engagement

To date 3 in-person meetings and 2 online meetings have taken place focused on project objectives and planning. Supervisor feedback has massively shaped the direction of(technically it's the no left, right, forward, or back world record. i use a w and s for search crafting. all of my movement keys are unbound) the project, influenced by Mr Frantzidis expertise in neuroscience and sleep analysis. My supervisors prior works on characterizing sleep stages (Chriskos et al. 2017, 2020) has directly informed my decisions of data preprocessing and artefact handling. Mr Frantzidis also has access to private sleep datasets which allows me to work on this analysis. However communication has not been as frequent as it should have been, which may have limited guidance on the project caused by my poor time management. More proactive communication would have ensured that the best practices were followed.

Implementation

4.1 Introduction

This chapter will provide a comprehensive review and justification for my methodologies, tools, workflows to provide rationale for key design choices. The implementation is structured into two core components, backend consisting of data curation, preprocessing, model architecture, training pipelines and strategies. Frontend focuses on model deployment, web configuration, security and containerization. A review of the development environment - remote SSH, training hardware, and version control - is also included to contextualise my decisions.

4.2 Requirements

Functional Requirements Functional requirements specify baseline capabilities of the final project implementation, these are derived from aims in the project proposal and literature review, whilst making accurate changes during the creation process. The final artefact must process EEG data processing consisting of 19 channels and 15000 samples in the 1020 system, extract accurate and relevant features that are compatible with a CNN. The system must make an attempt to predict multiple hormones, and evaluate results in an effective manner. The final artefact must provide an interactive interface, that allows for a user to receive a predicted value with evidence or confidence behind said value. This must completed keeping computational efficiency, organisation and data security a top priority.

Non Functional Requirements Non functional requirements can address overall performance, usability, deployment and scalability of the final artefact, ensuring that choices are justified, and accurate based on current literature and technical standards.

4.3 Data Curation

The dataset is derived from an experiment that assesses the effect of simulated microgravity on the human body including but not limited to cardiac function, hormone levels, muscle activity, body mass, and body composition simulated over 59 days. The participants were 23 healthy male adults between the ages of 23 and 45 (mean: 29 $\pm$ 6 years). All of them were informed for the RSL study and then they signed a written informed consent form (Kramer et al. 2017). The acquired signals included EEG recordings from 19 Ag/AgCl electrodes positioned according to the International 10–20 System. In this paper, hormone prediction was investigated by extracting features from the N1 sleep stage EEG recordings in the BDC-14 phase of the experiments. This is in order to avoid EEG fluctuations due to the head-down tilt protocol.

Source and Accessibility The data used in this paper is a subset of the experimental data recorded during the experiment conducted in the ENVI- HAB premises of the German Aerospace Agency, as discussed in (Frantzidis et al. 2018), funded by Authorized licensed use limited to: University of Lincoln -UK. Downloaded on March 26,2025 at 17:18:38 UTC from IEEE Xplore as described in (Kramer et al. 2017).

Licensing and Ethical Considerations Restrictions apply. 118 IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 31, NO. 1, JANUARY 2020 the European Space Agency as described in (Kramer et al. 2017).The ethics committee which approved the study was that of the Northern Rhine Medical Association (Arztekammer Nordrhein) in Duesseldorf, Germany Strahlenschutz.

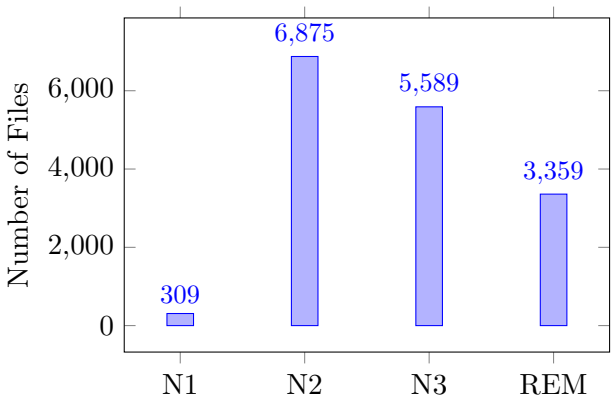

Figure 5: *Distribution of EEG files across sleep stages.*

4.3.1 Preprocessing

Raw Data Cleaning The initial EEG data consisted of significantly more recordings, recordings although due to the nature of EEG data, many epochs need to be removed due to unwanted noise. The signals were filtered with third-order Butterworth filters in order to remove unwanted spectral content such as power line interference, noise due to body movements, and noise from other unknown sources. After filtering independent component analysis (ICA) was performed as described in (Bell & Sejnowski 1995), and a subset of the resulting components were used to reconstruct the final EEG signal, in order to remove noise components related to body movements, interference from other biological signals such as EOG, EMG, and electrocardiographic (ECG) modulation. Finally, the EEG signal was split into 30-sec epochs according to the guidelines given by the AASM (Berry et al. 2017).

Custom DataLoader Due to the nature of the task, it was very important to enable computationally efficient processing, as well as having a large amount of configuration for later hyper parameter testing. This is implemented using a custom `RawDataset` class, built from PyTorch's `Dataset` interface. This allows for integration with Pytorch for further analysis. The custom `Dataset` interface allows for effective parallelisation using several independent subprocesses together, to manage many batches concurrently. Parameters such as `sleep_stages`, `feature_freqs` and `hormones` are custom parameters to allow for further testing and evaluation later in the project.

Data Interpreter The raw MATLAB `.mat` data is loaded into memory using SciPy's `loadmat` method, and stored in memory as a Numpy array. The original biomarker data that are represented as float values in a `.xlsx` spreadsheet file does not facilitate fast lookups in Python, so all the values are converted into a csv table which can easily be interpreted using the `pandas.DataFrame` architecture that facilitates dictionary-based lookups. This allows for more efficient dataset labelling when training. To also facilitate the testing of a classification model, a seperate biomarkers file is created, to allocate regressional values into uniformly distributed bins, (0, 1, 2). Uniform bins were selected to benefit training convergance, even though nessacarily populations may not fit into three discrete bins. Please note that later these are sometimes references as (low, normal and high). To remove participants that either have no epochs (participant Y) or no labels (participant S), `collate_fn` is used to remove these samples, running simultaneously during batch creation, PyTorch can easily filter this homogeneous data.

Train/Test Splitting Strategy It is important that the sampling implemented uses participant-wise splitting, otherwise our models predictions would be dishonest, if participant's epochs are

leaked between the training and test dataset, it would be expected to see unnatural convergence on the validation set. Further hash validation comparisons between the training and testing dataset quickly ensure that this sampling has been successful. The model could not accurately be evaluated, and would most likely overfit, so it would be unable to generalise to new data if participants epochs were included in the training and testing set. `participant_kfold_split` K-fold cross-validation involves dividing the dataset into `kfold`'s, training and testing the model on each respective fold. It is feasible to use this method rather than alternative cross validation such as LOOCV (Leave one out cross validation) because the dataset is large enough to support a sustainable amount of participants data in the datasets. As represented in the figures below, there is a non uniform amount of participants, this is particularly important for the N1 sleep stage. Using Python library `shutil` files can be copied into a new folder; fifteen random epochs from each participant in each sleep stage are selected and put into smaller sets. This is particularly useful when investigating various features and machine learning optimisations.

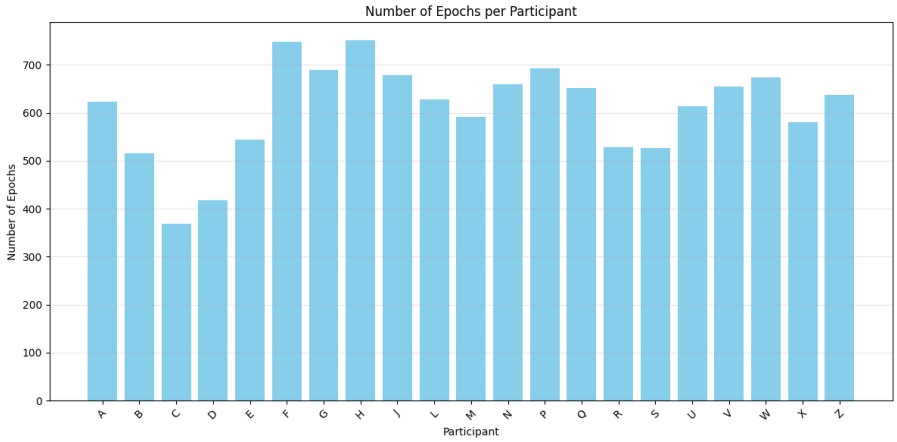

Figure 6: *Distribution of amount of participant epochs on N1, N2 and N3 sleep stages.*

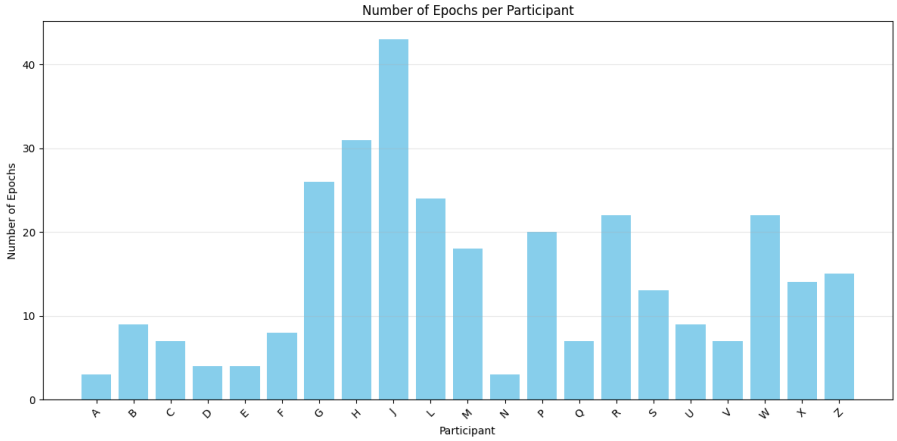

Figure 7: *Distribution of amount of participant epochs on only the N1 sleep stage.*

4.4 Model Development

`Feature Extractor` is a modular component designed to extract various EEG features to allow for extensive testing and evaluation. Features implemented are listed as `_coh`, `_pdc`, `_lc`, `_psd`, `_se` for any specified frequency bands of the delta, theta, alpha, beta and gamma waves,

Figure 8: *Normalised amount of epochs per participant.*

specified as `1-4 Hz, 4-8 Hz, 8-13 Hz, 13-30 Hz, 30-45 Hz`. These features are appended to a matrices of size 19 x 19, and combined into a multi channel depth matrix; such as single greyscale(1 channel) or full RGB(3 channels).

4.4.1 Feature Engineering

Coherance_coh Coherence is a measure of the linear correlation between two signals; in my case between two brain electrodes in the specified frequency domain, its a normal metric, ranging from 0 (no correlation) to 1(perfect correlation). The result of this method is a lower triangular matrix, and two of these can be combined to form a full 19 x 19 square matrix. The original 30 second epochs are created into fixed length events of 5 seconds, to obtain averages of Power/Cross spectral densities to reduce variance and noise. The implementation relied on the Python MNE's `mne_connectivity.spectral_connectivity_epochs` interface using multi-taper averaging. The spectral coherence $C_{xy}(f)$ between two EEG signals x and y at frequency f is computed as:

$$C_{xy}(f) = \frac{|S_{xy}(f)|^2}{S_{xx}(f)S_{yy}(f)}, \tag{1}$$

where $S_{xy}(f)$ is the cross-spectral density between x and y, $S_{xx}(f)$ and $S_{yy}(f)$ are the auto-spectral densities of x and y, respectively, and $|\cdot|^2$ denotes the squared magnitude.

Shannon Entropy_se Shannon entropy is computed using the Orthogonal Discrete Wavelet Transform (ODWT). Each 30-second EEG epoch is segmented into non-overlapping 5-second windows using MNE's `make_fixed_length_events`. Downsampling to around 100 Hz is required to reduce computation , yielding 500 time points per window. Shannon entropy is computed and normalized to range between 0 and 1 using Python's Numpy package, and is defined as:

Partial Directed Coherance_pdc Partial Directed Coherence (PDC) is captures directed interactions between channels. The input data is also downsampled to 200 time points using `scipy.signal.resample`, yielding a sampling rate of 133.33 Hz. To further optimise computation, the midpoint frequency is used, as defined by (Baccalá & Sameshima 2001) $f = (f_{\min} + f_{\max})/2$ The Fourier transform of the coefficients is computed as:

$$H = -\frac{1}{2}\left(p_{j1}\log_2 p_{j1} + p_{j2}\log_2 p_{j2}\right). \tag{2}$$

where p_{j1} and p_{j2} are the relative energies at level j for the first and second electrodes, respectively.

$$A(f) = I - \sum_{k=1}^{10} A_k e^{-2\pi i f k}, \tag{3}$$

where I is the identity matrix. The PDC is then calculated in a vectorized manner across output channels, and the diagonal is set to zero. The resulting 19×19 matrix, averaged over the epoch and shaped as $(1, 19, 19)$.

Power Spectral Density (_psd) The `_psd` function computes Power Spectral Density (PSD) features for hormonal prediction by processing 30-second EEG epochs with 19 channels into a diagonal matrix capturing channel-specific power in a specified frequency band (e.g., α: 8–13 Hz). The input epoch, converted to an MNE (Gramfort 2013)`RawArray` via `_epochtoRawArray`, is segmented into 5-second epochs using `mne.make_fixed_length_events`. PSD is calculated via `epochs.compute_psd`, which is then expanded to shape $(1, 19, 19)$ and transposed to align with PyTorch's channel-first format. This PSD matrix complements other features such as Partial Directed Coherence and Shannon entropy in the EEG pipeline. Frantzidis et al. (2018).

$$\text{PSD}_i = \frac{1}{N_f} \sum_{f=f_{\min}}^{f_{\max}} S_{xx,i}(f) \tag{4}$$

Figure 9: *where PSD_i is the power spectral density for the i-th signal, N_f is the number of frequency bins, f is the frequency variable, f_{min} is the minimum frequency, f_{max} is the maximum frequency, and $S_{xx,i}(f)$ is the power spectrum of the i-th signal at frequency f.*

Regression Layer Configuration This section expands on neural network architectures for regression hormone prediction

- **Model Design**: The bases of my model is designed to process spatial maps derived from electrode placements, consisting of convolutional layers and a fully connected classifier (Lawhern et al. 2016). The architecture has specific emphasis on balancing computational efficiency and predictive accuracy. The CNN is needs to be able to support different filter sizes and input channels for iterative testing.
- **Layer Configuration**: The basis of the `EEG-CNN` is the input layer has two convolutional layers paired with Relu activation, and the output layer has three Linear Relu activation pairs layers.
- **Evaluation and Implementation**: This was implemented and evaluated using PyTorch `nn.Module`. Integrations with the data loader, visualisation and loss methods meant that it was the best choice for my project. The evaluation investigates achieved metrics over

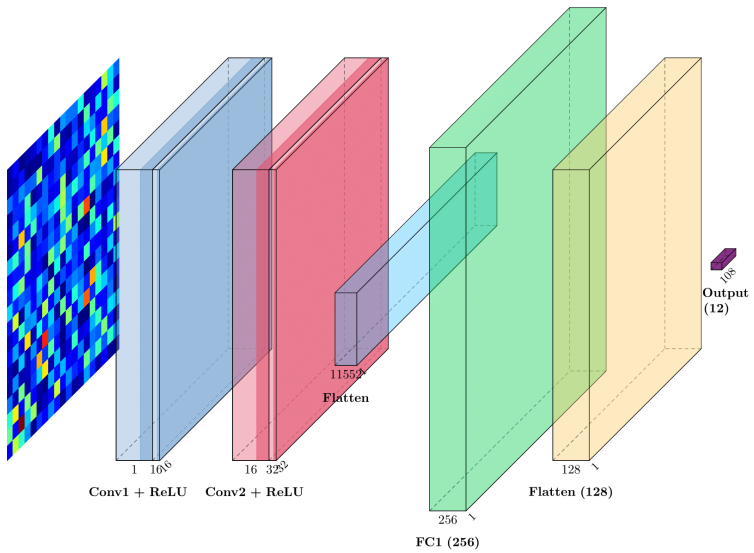

Figure 10: *Diagram of implemented CNN architecture used for EEG analysis. Image creation tool used : (Iqbal 2018)*

many different features extraction methods, hormones and sleep stages. The standardised metrics that will be primary used will be Coefficient of determination (n^2) and MSE, discussed in further detail in the next section.

Classification Layer Configuration Due to challenges later described in this chapter, an classification model was also implemented for comparison, this paragraph details the CNN architecture for hormonal multi-classification into bins: low, normal, and high.

- **Model Design**: This is different to the regression model, where a instead of the output layer representing the size of the input, in the classification model only a single hormone is trained and predicted at a time; although similar to the regression model, the `EEG-CNN` feature extraction follows the same methodology, a 19x19 square matrix consisting of spatial maps that are computed from electrode pairings to extract features that could be relevant to the hormonal fluctuations. This architecture aims to balances computational efficiency with classification accuracy.
- **Layer Configuration**: The input layer consists of similar convolutional layers, the difference being that an extra dropout layer is used to reduce over fitting. The same ReLU activation layer is used in the multi-classification, designed with the option to specify filter sizes and input channels. The output layer is a fully connected layer that produces three estimations that are finally converted using a SoftMax activation layer during inference, within the Cross-Entropy Loss method.
- **Evaluation and Implementation**: This architecture aims to balances computational efficiency with classification accuracy, leveraging PyTorch `nn.Module` for implementation.

Hyper Parameters Both models follow very similar hyper parameter tuning. Hyper parameter tuning was extremely challenging due to the high-dimensional parameters space such as feature extraction methods (e.g. Spectral coherence, wavelet transformation), sleep stages, group sampling(Leave one out vs k-fold), participants, loss methods, optimiser type, learning rate scheduling, batch sizing, epoch iterations and CNN architecture. To manage this complexity, the first objective was to create much smaller subsets of data with accurate proportions of

sleep stages and participants to analyise which parameters had the largest weights in prediction accuracy. This is a good idea as with good validation convergance it is likely that the model could be scaled in an effective manner to achieve linear performance gains. For an effective evaluation of individual sleep stages, separate datasets organised as `N1_small, N2_small, N3_small and RE_small` were defined(Cheng et al. 2023) (Gamel et al. 2025) . This was implemented a using Python module `shutil`and visualised using a custom sampler method defined in `hpset.py`. To measure effectiveness of possible parameters, a set was defined using a Python dictionary located in `parameters.py` and iterated training using Python module `itertools.product`. Standardized performance metrics (e.g. Coefficient of determination and Mean squared average) were used to asses performance of parameters. Demonstration of the `parameters.py` file as follows:

```
params = {
       "b_size" : 16,
       "filter_size" : 5,
       "iterations" : 50,
       "k_folds" : 3,
       "in_channels" : 1
}

param_options = {
    "feature_freq" :
        [[{'coh' : 'delta'}],
        [{'coh' : 'alpha'}]
        [{'coh' : 'delta'}, {'coh' : 'beta'}],
        [{'coh' : 'alpha'}, {'coh' : 'theta'}],],
    "hormones" : [
        ['BDC1'],
        ['BDC1.1'], ['BDC1.2'], ['BDC1.3'], ['BDC1.4'],
        ['BDC1.5'], ['BDC1.6'], ['BDC1.7'], ['BDC1.8'],
        ['BDC1.9'],['BDC1.10'], ['BDC1.11']],


    "sleep_stages" : [['N1', 'N2', 'N3', 'REM']],
}
```

4.5 Testing Strategy

4.5.1 Optimizers

- **Adam (Adaptive Moment Estimation)**: The Adam optimizer is well established in CNN models Kingma & Ba (2014), and is particularly useful in the context of EEG data, where the feature matrix is likely to have significant noise. This is typically more beneficial then stochastic gradient optimisers. The optimiser was implemented using Pytorch `torch.optim.Adam`, for both the classification and regression model. Adam is also inherently calculates its own momentum, which is particularly useful in our case where we have such a large amount of hyper parameters to evaluate (Kingma & J.Ba 2014).

  where $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected moving averages of the gradient and squared gradient, respectively.

- **SGD (Stochastic Gradient Descent)** Unlike Adam, SGD is a static optimiser and so there forth parameters such as loss and momentum. This means that the optimiser uses less memory overhead, but the parameters have to be fine tuned. This is particularly

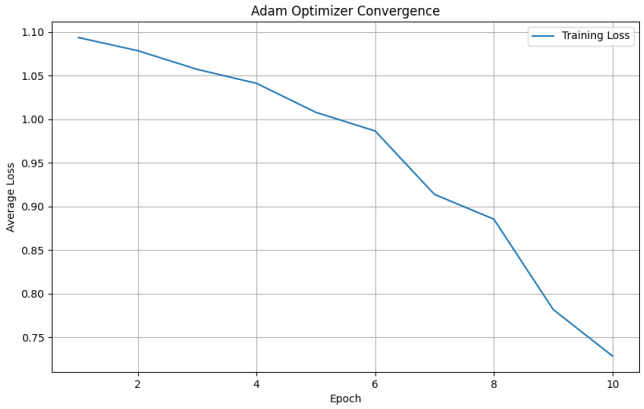

Figure 11: *Diagram of Cross Entropy convergence with the Adam optimiser*

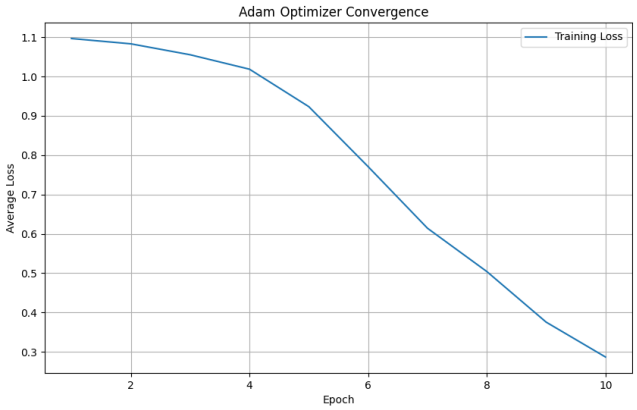

Figure 12: *Diagram of MSE convergence with the Adam optimiser*

difficult when using a large amount of hyper parameters such as features and sleep stages. As demonstrated in the figure below, it demonstrates the sharp osculations in convergence, reflecting noisy batches, where the magnitude is heavily impacting the gradient updates.

- **Conclusion** Adams ability to provide fast convergence over noisy gradients make it significantly more favoured than SGD. It significantly eases the tuning process.

4.5.2 Batch Sizing Strategies

Batch sizing impacts the convergence rate and computational requirements. In context of resource-limited cloud deployment, it is important to reduce memory usage whilst balancing accuracy.

- **Full Batch Gradient Descent** FBCD computes the loss function gradient from all data points within the training batch, making it extremely robust and very accurate convergence; it is a good performance baseline for experimental testing. However, it requires

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{5}$$

Figure 13: *Equation of Adam Optimiser*

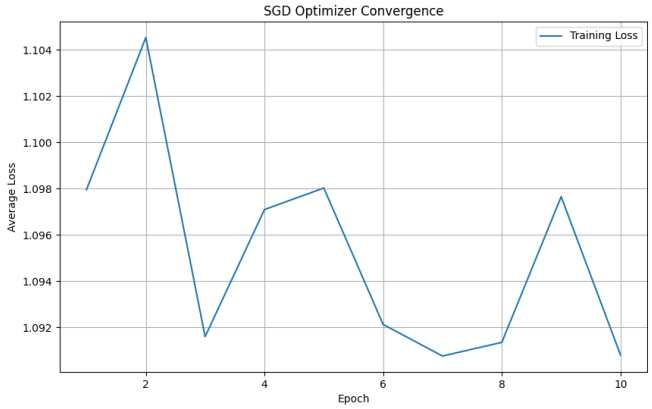

Figure 14: *Diagram of convergence with the SGD optimiser*

a large amount of memory to store the entire dataset. In the interest of cloud deployment inference; the EEG dataset size is carefully managed, including the procurance of smaller datasets and spatial reduction. Other reduction techniques include Independent Component Analysis (ICA) (retaining specific electrodes or channels), although necessitating baseline testing.

4.5.3 Epochs and Training Duration

- Early Stopping Early stopping allows for better hyper parameter tuning, as training times can be significantly optimised. Early stopping incudes evaluating a separate validation set during the training. My implementation does this after each epoch. A patience value allows configuration on how many epoch iterations can complete before the training is abandoned. Because of the nature of the dataset, lots of noise and variation is expected in batches; so a high patience value of 10 epochs is implemented. This was implemented using the respective metrics for classification (accuracy) and regression(coefficient of determination). To maintain robustness, the validation set has an equal amount of epochs in per participant, as to maintain class balance and representation.

- Learning Rate Scheduling The initial learning rate used for both optimisers was 0.0001, providing a good baseline for the models, to have little initial over fitting. To further optimise this, Pytorch step decay `torch.optim.lr_scheduler.StepLR` was originally used; paired with Adam. The results of this were not significant in my final prediction metrics; although effected my computational cost(due to a more amount of epochs), and worsed the generalisation of the model.

- Convergence Criteria Convergence of the model is primarily determined by early stopping, although the maximum epoch limit of 100 is a safeguard to prevent excessive training; this is defined within `args`. Furthermore the usage of the Adam optimiser lightly tackles the risk of over fitting. The usage of kfold cross validation validates convergence, addressing inter-participant variability in the datasets.

- Over fitting and Under fitting Detection Identifying over fitting during training is critical to ensure high performance on unseen data. By optimising the amount of actual training data used and investigating performance by proactive monitoring formatted as `Traning Acc: 0.69 Val Acc: 0.63`. Additional regulation techniques such as additional dropout layers investigated in the CNN to try to enhance over fitting and robustness.

As seen in the figure below, it is a apparent that my training data is converging particularly well, but not generalising well to unseen data; this will be further investigated later.

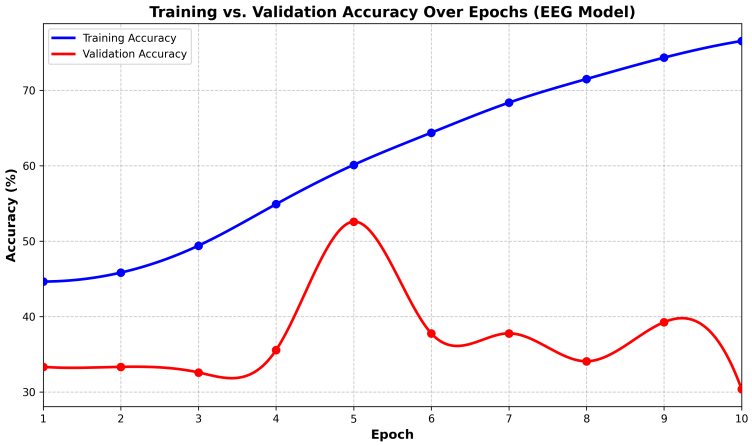

Figure 15: *Diagram of training and validation accuracy.*

4.5.4 Loss Functions

- **Regression:** The loss method investigation in context with EEG, is extremely important. Since EEG data is prone to artifacts, the choice of loss functions must account for robustness and probabilist correctness. Below will define implementations for both regression and classification models. These include MSE, MAE, and Huber loss. Definitions include the derived mathematical formulations.

- Mean Squared Error (MSE) The mean squared deviation is an estimator that measures the squared difference between the predicted value and the truth value, implemented in PyTorch as `torch.nn.MSELoss`. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

- **Mean Absolute Error (MAE):** The mean absolute error measures the absolute difference between predicted values and truth values, it is not accurate to use when classes are weighted or are non-uniformly distributed. Class labels are normalised using PyTorch `sklearn.preprocessing.StandardScaler`. MSE is implemented in PyTorch as `torch.nn.L1Loss`. It is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

  where y_i is the true value, $\hat{y}_i$ is the predicted value, and n is the number of observations.

- **Huber Loss:** The Huber loss combines the benefits of MSE and MAE, providing robustness to outliers while maintaining differentiability. It is implemented in PyTorch as `torch.nn.HuberLoss`. It is defined as:

$$\text{Huber Loss} = \frac{1}{n}\sum_{i=1}^{n} L_\delta(y_i, \hat{y}_i).$$

- **Classification:**

- **Cross Entropy** Cross Entropy Loss is widely used in EEG studies for classification. In our case predicting probabilities for a low, normal or high hormone value. This is different then quantifying the difference directly from the output layer, as the `torch.nn.CrossEntropyLoss` internally applies the Softmax operation, effectively converting the raw values into probabilities. Mathematically, for a single sample it is defined as :

$$\text{Cross Entropy} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

.

4.5.5 Monitoring and Debugging

- Training/Validation Curves
- Gradient Flow Analysis

4.5.6 Performance Metrics

This section justifies performance metrics used to evaluate implemented regression and classification models. The classification's output is three classes, based upon if the hormonal value is low, normal or high The regression models output layer produces the specified amount of hormones; although for testing single classes are mainly used for simplicity and abstraction. Further evaluation such as saliency maps to discuss electrode pairings that are commonly the highest weighting.

- **Regression:**
  - **R2 (Coefficient of Determination)**: Measures the proportion of variance in the dependent variable explained by the model. It is defined as:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2},$$

    where y_i is the true value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of true values. Higher values indicate better model fit.
  - **Spearman Correlation**: Assesses the monotonic relationship between predicted and true values by computing the Pearson correlation of their ranks. It is robust to non-linear relationships and outliers.
- **Classification:**
  - **Per-Class Accuracy**: Measures the proportion of correct predictions for a specific class i, defined as:

$$\text{Accuracy}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

    where TP_i is the number of true positives and FN_i is the number of false negatives for class i. This metric highlights model performance on individual classes, especially in imbalanced datasets.
  - **Per-Class F1 Score**: The harmonic mean of precision and recall for a specific class i, defined as:

$$\text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}, \quad \text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \quad \text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

    where FP_i is the number of false positives. The per-class F1 score is critical for evaluating performance on minority classes and can be aggregated into macro or weighted F1 scores for overall assessment.

4.5.7 Saliency Maps for Model Interpretation

Saliency maps are employed to interpret neural network's predictions, and investigate feature extraction. The EEG dataset follow the 10–20 system which is an internationally standardized method to describe the location and organisation of scalp electrodes. This is particularly important, as previous literature indicates that specific channels correlate to specific biological processes. The classification task categorizes hormonal values into three distinct classes (low, normal, and high). Computing saliency maps identify which electrode pairings have the largest weights that contribute towards the final class prediction. These saliency values can be visualised to make it more intuitive to interpret the findings. Additionally the saliency values can be applied to topological heat maps to visualise the spatial information of features, this is implemented in `data_io.saliencymap`. This section aims to further describes the saliency map implementation, and its application to electrode-based data.

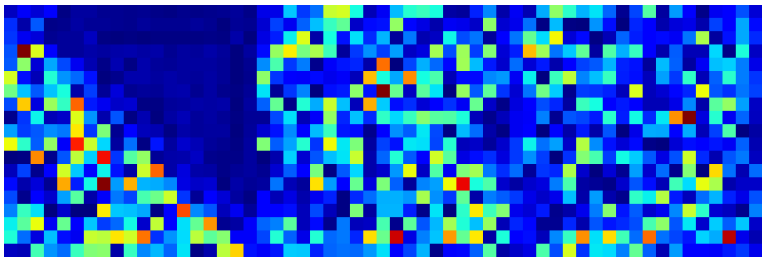

Figure 16: *Diagram of three saliency maps for coherence of α, coherence of α and β, coherence of δ and γ (Adenosine Deaminase 2 Enzyme).*

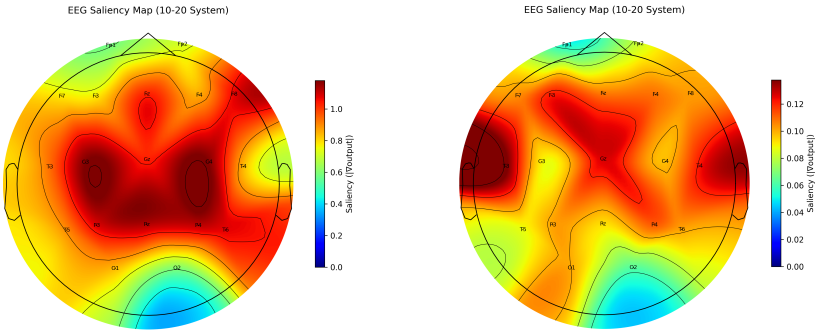

Figure 17: *Channel Topology for GLU mg/dL (left) HDL mg/dL (right)*

- **Saliency Map Methodology**: Saliency maps compute the gradient of the model's output with respect to the input tensor, indicating which features most affect the prediction. My implementation is demonstrated in `compute_saliency_map`, by retrieving the gradients using `torch.Tensor.grad` after a forward and backwards pass. Saliency is defined as:

$$\text{Saliency}_i = \left| \frac{\partial \hat{y}_j}{\partial x_i} \right|,$$

  where $\hat{y}_j$ is the logit for class j, x_i is the i-th input feature (e.g., an electrode pair signal), and $|\cdot|$ denotes the absolute value to capture magnitude. For regression, the output is the an exact hormone value. High gradient values indicate that electrode pairings have a strong predictive influence, aiding in biological interpretation.

- **Application to Classification**: Per-class saliency maps are generated by computing the gradients for the three classes, low, normal and high. This lets us investigate which electrode pairings are the most valuable . For example, a practical insight from the saliency

map would be finding a common high saliency values (specific channels that are visualised in red) for an electrode pair This is particularly valuable in the medical field to identify physiological patterns linked to hormonal states. This can also be used for later research to improve computational efficiency, as instead of an entire 19x19 square matrices, only a single value could be used. This is slightly more valuable than a regression model, as it could be possible that different electrode pairs indicate whether high and low values.

- **Application to Regression**: For regression, similar use of saliency maps can be used to highlight electrode pairings that influence predicted hormones. Single-class testing (individual hormone values) allows class-specific saliency analysis. This supports the identification of electrode pairs driving accurate predictions in specific contexts.
- **Advantages of Saliency Maps**: Saliency maps are model-agnostic, requiring only a differentiable model, making them suitable for various neural network architectures used in bio signal processing. They provide feature-level importance for electrode pairings, and are computationally lightweight, requiring a only a single back propagation pass.
- Saliency maps thus provide a robust and interpretable method for understanding the contribution of electrode pairings to hormonal level predictions, offering advantages in flexibility, interpretability, and compatibility over Grad-CAM for this application. Per-class metrics are particularly valuable in multi-class classification to identify model weaknesses, such as poor performance on minority classes, which may be obscured by global metrics like overall accuracy.

4.6 Final Model Performance Evaluation

This section is evaluates the final performance of the convolution neural network (CNN) model for predicting hormones; and is separated into two parts; regression and classification. Results and visualisations are organised by sleep stages - N1, N2, N3, and REM. For standardisation all results were generated using the same k-fold cross-validation participant groups (defined below) and CNN parameters (batch size 4, filter size 3). Feature extraction focused on 19x19 coherence matrices at different frequencies, with topographic visualisations of electrodes, to further investigate specific channels. The aim of this evaluation is to find which sleep stages, features and channel pairings of the 10-20 system produce the best results. Notable findings in the classification model (0.8816 for ADA2 U/L using alpha coherence) demonstrate the models efficacy and potential for non-invasive biomarker monitoring.

Hormone Regression in Sleep The first iteration of the model included all four sleep stages in its traning set, and a single output layer to predict regression values for all 12 hormones (TAC mmol/L, ADA U/L, ADA2 U/L, %ADA2, GLU mg/Dl, PHOS mg/Dl, CA mg/Dl, CHOL mg/Dl, TRI mg/Dl, HDL mg/dL, LDL-C mg/Dl, CPK U/L) at once. This was very challenging, as the model struggled to converge on all 12 labels at the same time. The model's input being a 19x19 square matrix, originally it seemed that the model could not successfully optimise all of the hormones hormones due to the diverse scale of the hormone levels. This was resolved by normalising the values to 0-1. This change improved the models convergence during training, but still could not generalise well to the unseen data, with the best performance a mere: 0.175 r^2. Due to the challenges in the regression task, the modelling approach was adapted to suit a classification approach, where hormone levels were categorised into three discrete groups to enhance generalisation. Sleep stages were also separated during training to remove possibility of frequency sensitive hormonal fluctuations. These changes are further demonstrated using the regression performance as outlined below.

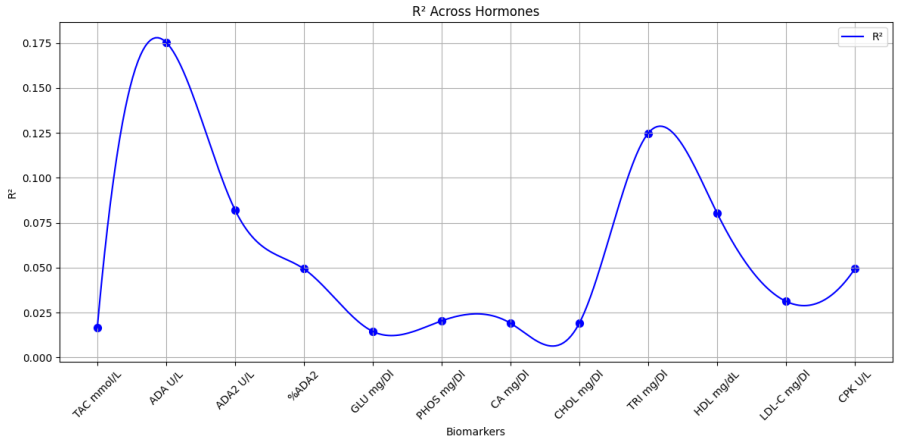

Figure 18: *Best r^2 values for individual hormones from all sleep stages.*

Hormone Classification in N1 Sleep The N1 sleep stage is characterised by light sleep, only lasting for around 5 minutes; there is not much data available, so training is completed on the entire dataset. This is particularly noticeable in comparison to the other sleep stages, demonstrating a much larger amount of variance between feature extraction methodology. This does not seem to prevent results in N1 however, as notable findings include 0.8816 for ADA2 U/L (alpha) and 0.7829 for TAC mmol/L (alpha), it is to be expected that the delta wavelength appears to perform the worse, as the N1 sleep stage has very little delta wave activity. Low performing hormones such as CHOL, TRI and LDL-C may indicate limitations for EEG-based correlations in lipid biomarkers. Topographic visualisations reveal critical electrode channels Fp3 and Fp4 driving N1's high performance. These frontal channels reveal coherence patterns relevant to early sleep. Future work that have access to larger computation and datasets could expand on this key pairing with further frequency analysis, to improve performance.

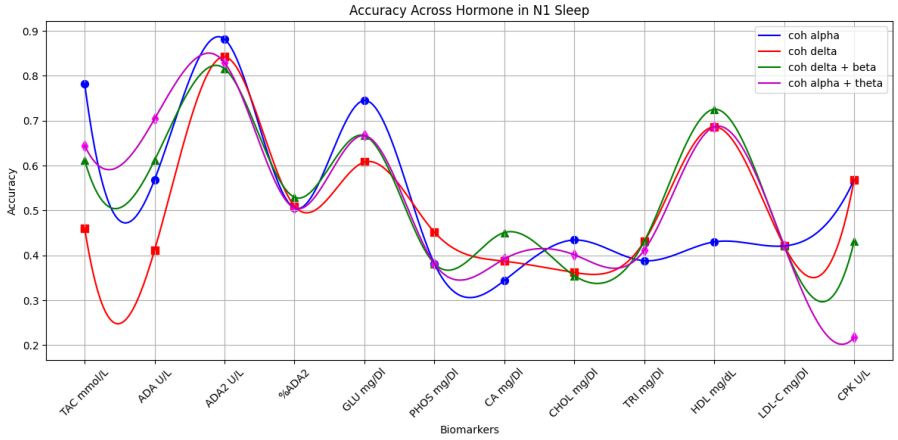

Figure 19: *Accuracy for individual hormones for N1 sleep stage.*

Hormone Classification in N2 Sleep The N2 sleep stage is defined as a light sleep that follows after N1, typically lasting for around 15 minutes, characterised by its sleep spindles and theta waveband. which means we have a much larger amount of data available than N1; this is also reflected in the variance increase from 0.12 to 0.17 between feature extraction methods. N2 achieves a slightly lower overall accuracy than N1, but this seems to be because of sharper gradient changes in N1, as it is a smaller dataset. Coherence of alpha and theta seem to outperform all other features; this supports N2's higher theta activity. N2 exceeds at predicting

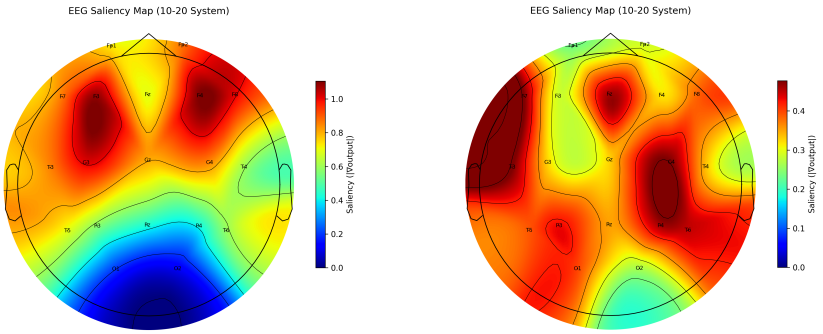

Figure 20: *N1 Channel Topology for alpha ADA2 U/L (Left) HDL mg/dL delta (Right)*

GLU mg/dL (0.7345) and HDL mg/dL (0.5855). These findings could be relevant in supporting diabetes management with further performance optimisations. The topology map indicates a different compared to the hormones evaluated in N1. Electrode channels for GLU mg/dL seem to demonstrate use of different channels, particularly G3 and G4.

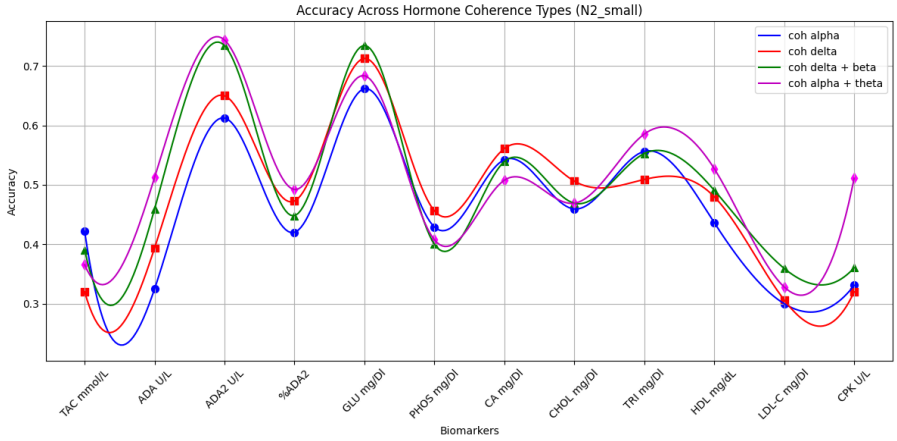

Figure 21: *Accuracy for individual hormones for N2 sleep stage.*

Hormone Classification in N3 Sleep The N3 sleep stage is charterized with slow-wave activity, and is important in evaluating EEG-based predictions. Classification average accuracy across hormones evaluates to a conservative 0.5181, and demonstrates the best predictions over all sleep stages for CA mg/dL and %ADA2, driven by delta and beta coherence, likely reflecting the dominance of slow-wave activity in deep sleep.The variability (standard deviation 0.15) suggests moderate consistency, similar to N2_small (0.12) but less than N1 (0.17). Unexpectedly, in comparison with other sleep stages N3 overall performs the lowest, its unique efficacy for %ADA2 highlights its complementary role in EEG-based hormone prediction, with its accuracy likely skewed by low hormone accuracy values for TAC mmol/L, LDL-C mg/Dl and CPK U/L. These results could be potentially linked to deep sleep's metabolic regulation activity. Electrode channels for CA mg/dL and %ADA2, seem to demonstrate use of different channels, particularly FP2 for %ADA2 and X for . Further research could see large performance improvements for delta specific features to improve overall model performance and generalisation.

Hormone Classification in REM Sleep The REM (rapid eye movements) sleep stage is characterised by desynchronized EEG activity, and by theta and beta waves. Classification accuracies demonstrate great performance, including 0.7333 for ADA2 U/L (delta and beta),

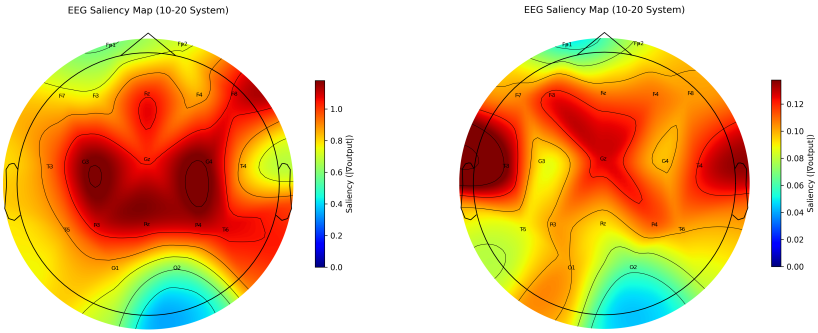

Figure 22: *N2 Channel Topology for delta GLU mg/dL delta and theta (left) HDL mg/dL (right)*

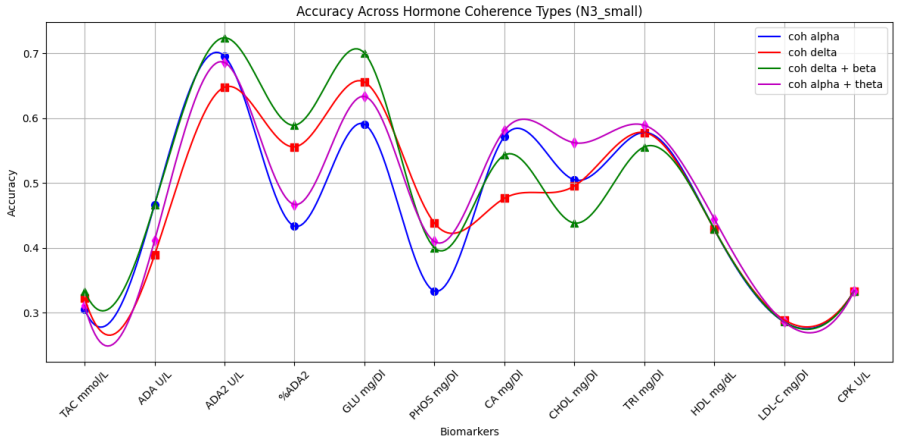

Figure 23: *Accuracy for individual hormones for N3 sleep stage.*

0.6667 for HDL mg/dL (alpha and theta), 0.6333 for TRI mg/Dl (delta and beta). These strengths likely reflect REM's mixed-frequency patterns and metabolic activity. Weaker predictions such as LDL-C mg/Dl (0.3523), TAC mmol/L(0.3905), and CPK U/L (0.3333), indicate limited EEG-hormone correlations, although overall it surpasses accuracies of N3_small (0.5181) and N2_small (0.5243) but trails N1 (0.6023). This demonstrates N1's strength in generalising over all of the hormones, but does not nessesarily demonstrate suitability for hormones such as CA mg/Dl and CHOL mg/Dl. Future work could expand larger training sets and theta specific features.

Overall Conclusion This CNN investigation evaluated the feasibility of hormone prediction of 12 hormones TAC (mmol/L), ADA (/L), ADA2 (/L), %ADA2, GLU (mg/dL), PHOS (mg/dL), CA (mg/dL), CHOL (mg/dL), TRI (mg/dL), HDL (mg/dL), LDL-C (mg/dL), and CPK (/L)—using EEG coherence data across four sleep stages: N1, N2_small, N3_small, and RE_small. The evaluation revealed significant changes in hormone values in different sleep stages, highlighting stage-specific strengths. The change from the original aim of a regression model significantly improved generalisation and interpretability of the data. The models performance supports potential for future studies, with applications for diabetes management, metabolic activity and clinical diagnostics. By leveraging features inside of EEG data, particular alpha in the N1 sleep stage and beta in REM sleep, the model could monitor longitudinal hormone fluctuations. Limitations of computational resources, sample demographics and past literature contribute to non perfect accuracy values, but can still indicate a possible argument for further studies to be completed. These could particularity involve additional coherence features such as theta for REM and sigma for N2. Investigation into other use cases such as

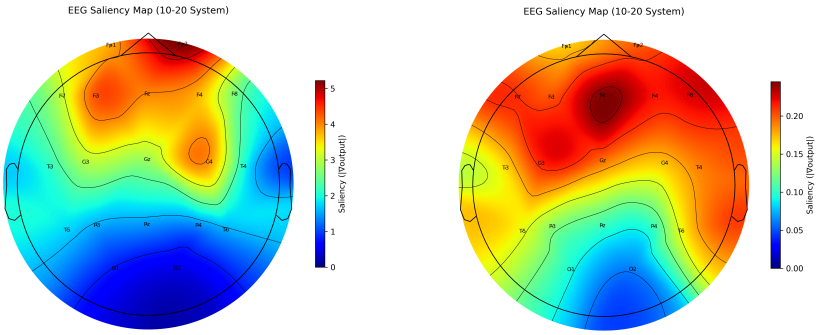

Figure 24: *N3 Channel Topology for delta %ADA (left) delta and beta CA mg/dL (right)*

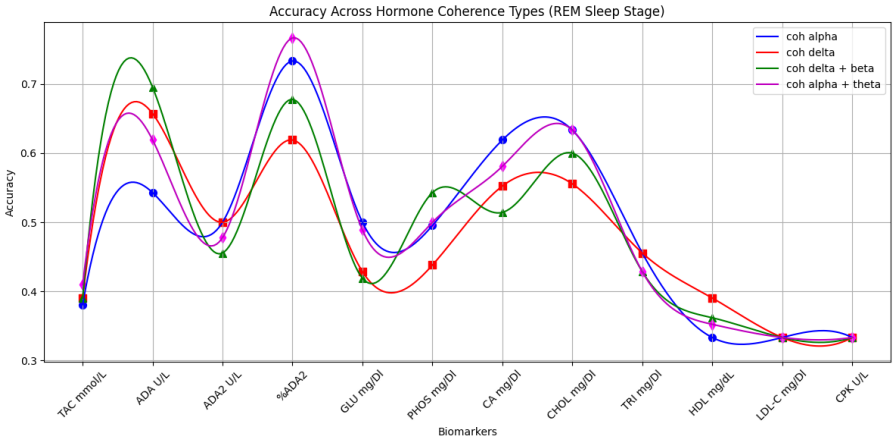

Figure 25: *Accuracy for individual hormones for REM sleep stage.*

hormone prediction based upon baseline hormonal markers could also prove to be valuable.

Authors Note I want to express gratitude to Christos Frantzidis for his idea and providing guidance on the dataset for EEG analysis which were critical for the success of this study. Special thanks to Youssef Zairi, Muhammad Vadia and Pruthvi Mohanlal for their valuable insight and motivation during this task, for unwavering support. Thanks to Sam Carson for extra guidance and expertise on frontend frameworks and practices (418 I'm a teapot!). Te also acknowledge the two supervisors in INB1101 for their contri- butions to shaping the final ablation thoughts, enhancing the study's predictive approach. Acknowledgments to Oracle Cloud Free Tier that allowed me to train and host my models, as well as host my frontend website made my entire application possible. I am also grateful to the University of Lincoln School of Computer Science for the extended open hours over the dissertation period, allowing for uninterrupted access to essential resources. This work was supported by the University of Lincoln.

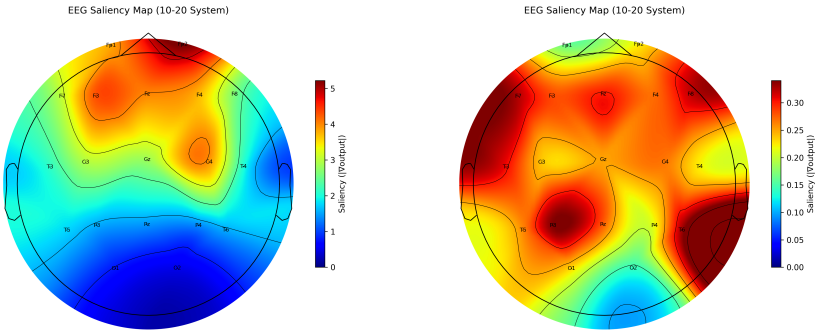

Figure 26: *N3 Channel Topology for delta PHOS mg/dL (left) delta CHOL mg/dl(right)*

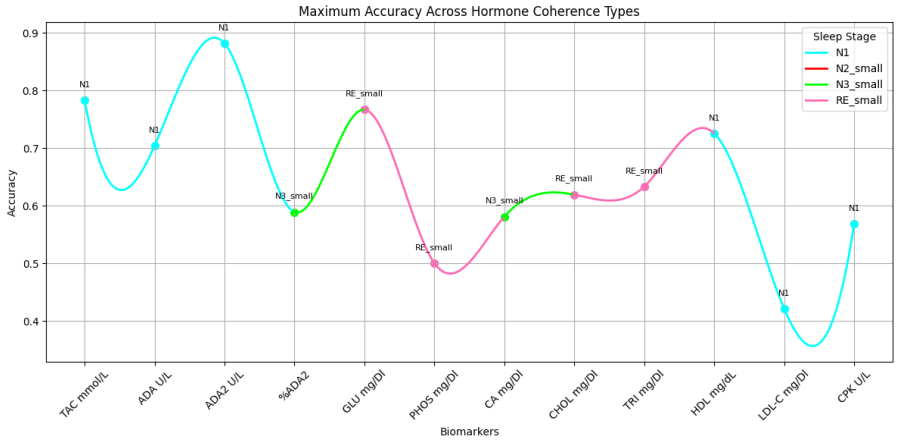

Figure 27: *Distribution of amount of participant epochs on all sleep stages.*

4.7 Frontend Server Stack

The web stack is designed to maximize performance, redundancy and cost-effectiveness. The frontend website uses lightweight frameworks whilst maintaining successful execution. Below is an overview of the stack's foundation:

- **Performance**: Utilizes static assets with a efficient CDN to allow for caching, minimizing latency and server load.
- **High Redundancy**: High containerization and usage of availability zones to maintain availability, even during failures.
- **GDPR**: Implements AES-256 encryption for long term storage.
- **Cost-Effectiveness**: Leverages Oracle Cloud's Always Free tier and lightweight technologies like Tailwind CSS, and Gunicorn to reduce operational costs.

4.7.1 Static Frontend

The frontend is implemented to allow users to upload a given 30 second epoch, select from a range of hormones; and view regression or classification predictions from a self hosted model. A

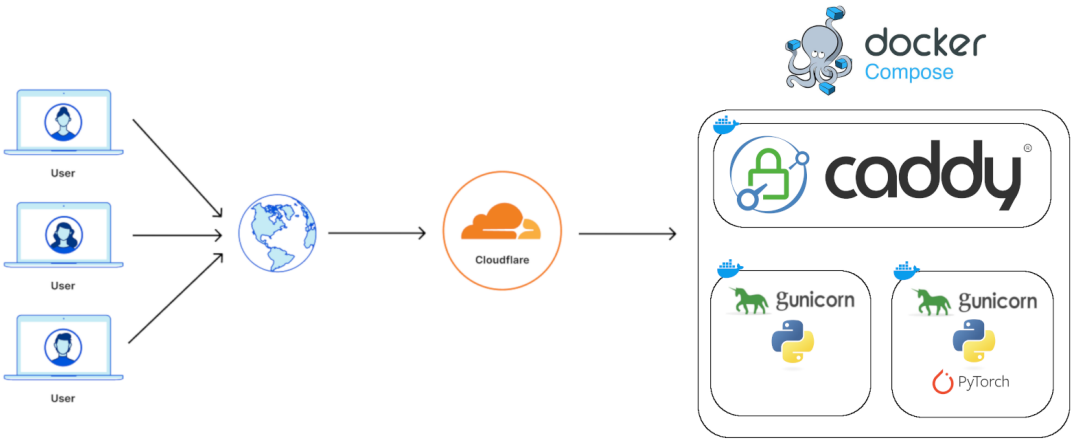

Figure 28: *Diagram of implemented tech stack for web interface*

Saliency map is also displayed to allow for further confidence in the prediction. The application has to ensure strict security protocols such as HTTPS and encrypted data transfer because of the nature of the EEG data, which is likely to be sensitive user data.

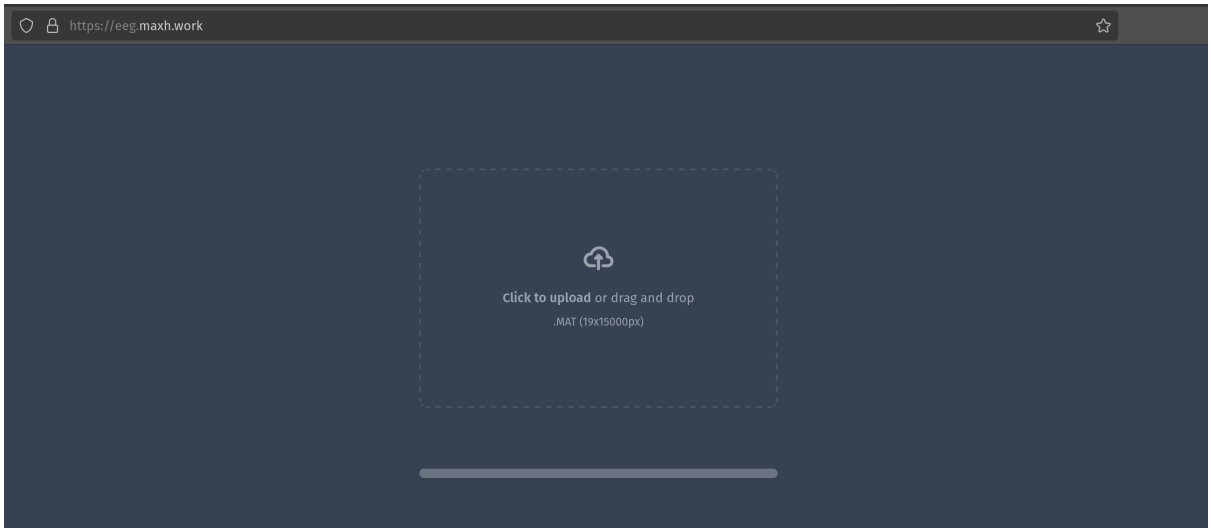

Figure 29: *Implemented index.html*

4.7.2 Frameworks

The frontend is implemented with a web interface, using HTML and Tailwind CSS. Tailwind is a well-maintained project, and reduces the need to write new CSS, it provides a large range of features that allow for cross-platform compatibility, ideal for responsible web design. Pre-compiled assets eliminates the need for server-side rendering overhead, which is especially vital in this projects, as the frontend resources are shared with the inference model (due to cost).

4.7.3 Deployment

Flask was chosen as the web framework, as it had easy integration already with my PyTorch models. Gunicorn is used as the WSGI server to handle and serve dynamic content for the Flask

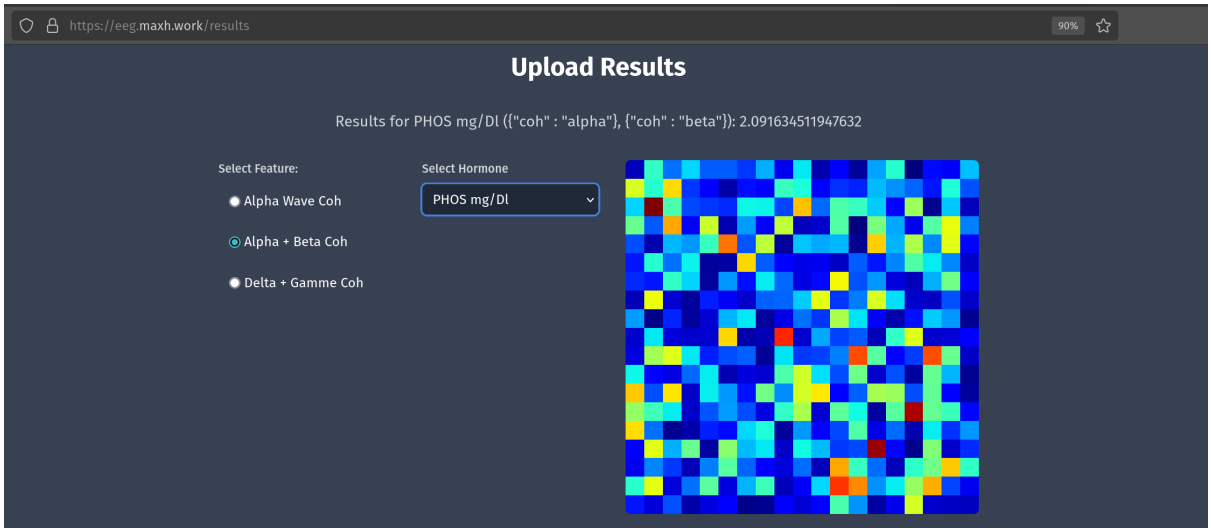

Figure 30: *Implemented results.html*

application. It is chosen over Flask's built-in WSGI server, as Flask's WSGI is not intended for production deployment. Gunicorn is a robust, production-grade WSGI server optimized for handling high traffic, gracefully handling worker restarts and timeouts. Gunicorn supports asynchronous worker processes and threads, enabling it to handle concurrent requests efficiently. The configuration uses 4 workers, as follows:

```
gunicorn --workers=4 --threads=2 --bind=0.0.0.0:8000 app:app
```

4.8 Server Infrastructure

The application is hosted on Oracle Cloud Infrastructure (OCI) using Canonical Ubuntu 24.04 LTS, chosen for its efficiency and compatibility:

- **Operating System and Hardware**: Canonical Ubuntu 24.04 LTS offers 12-year support via Ubuntu Pro, has relatively low overhead. Hosted on `VM.Standard.A1.Flex`, Leveraging cost effective Ampere Altra processors. 4 OCPUs and 24 GB of RAM.

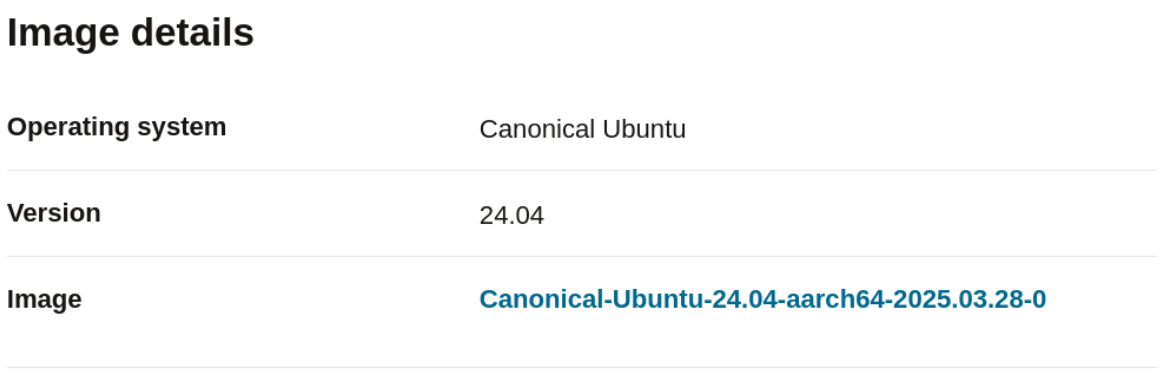

Figure 31: *Operating system Image*

- **Storage**: Utilizes 100 GB of Oracle Cloud's block storage, encrypted by AES-256. Data is stored over multiple isolated fault domains(FD's) to improve redundancy. My training set is particularly sensitive, so the block storage is restricted to specific non admin user

'eeg'. The block storage is hosted in Oracle's South London region, following EU Standard Contractual Clauses (SCCs). It is also protected under Oracle's Data Processing Agreement (DPA) to ensure security.

- **Network Configuration**: Virtual Cloud Network (VCN) with security lists restricts inbound traffic to essential ports (e.g., 80, 443). The frontend server is isolated in a public subnet, while a backed services are included on a separate subnet, enhancing GDPR-compliant network segmentation. Para virtualized networking ensures efficient, low-latency data transit, encrypted with TLS 1.3. The virtual cloud network (VCN) is configured with security lists to restrict inbound traffic to port 443 (HTTPS). This is also enforced on the Ubuntu instance with `UFW` in cases of external firewall failures. Isolation frontend and back end services segmented into two Virtual Cloud Network's (VCN), enhancing security through network separation. '

0.0.0.0/0	TCP	All	80	TCP traffic for ports: 80
0.0.0.0/0	TCP	All	443	TCP traffic for ports: 443 HTTPS

Figure 32: *Security group rules for Oracle instance*

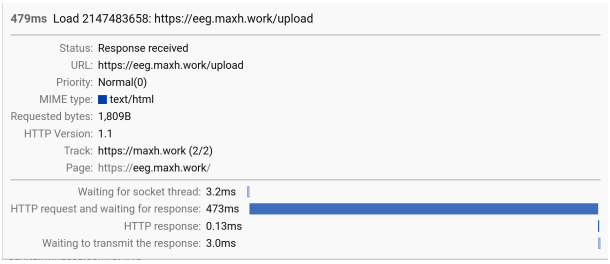

Figure 33: *Diagram of call stack times for /upload*

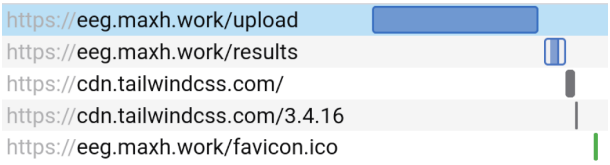

Figure 34: *Diagram of call stack compared to Tailwind loadtimes*

4.9 Cloudflare DNS & SSL

- **DNS Handling**: Cloudflare's DNS service ensures fast domain resolution, serving an 'A' record for eeg.maxh.work to forward the data to 84.8.153.220.
- **WHOIS Privacy**: NameCheap provides free WHOIS privacy to protects domain registration and ownership details.
- **Cloudflare Strict SSL**: Cloudflare enforces strict SSL/TLS policies, verifying the origin server's SSL certificate and requiring connections to use TLS 1.3. This ensures end-to-end encryption of the traffic, ensuring GDPR laws compliance and preventing man in the middle attacks.
- **SSL Key Management**: Cloudflare can create and manage SSL certificate's effectively, integrating seamlessly with the Caddy web server. Certificates are renewed automatically, ensuring uninterrupted connections.

- **Security Features**: Cloudflare's provides a free security against DDoS protection, exhaustion floods, and negotiation attacks; preventing malicious or suspicious traffic. Rate limiting is can be configured to prevent abuse of API endpoints, and bot detection mechanisms filter out automated attacks.
- **CDN** : A CDN is integrated to cache static assets closer to users, reducing latency and improving load times.

4.10 Containerization

The application is containerized using Docker Compose, which orchestrates services to ensure redundancy, scalability, and GDPR-compliant data processing:

- **Orchestration**: Manages Caddy (reverse proxy), PyTorch (model inference), and the web server, ensuring seamless integration.
- **Redundancy**: Isolates services to maintain uptime, even if individual containers (e.g., model inference) fail.
- **Resource Efficiency**: Separates resource-intensive services like model inference, optimizing performance on Arm-based Ampere Altra processors.
- **Deployment Automation**: Integrates with GitHub Actions for automated, consistent deployments, reducing human error.
- **Volume Binds**: Persistent storage is achieved through Docker volumes (`caddy_data`, `caddy_config`, `static_data`). These volumes store Caddy's certificates, configuration and the machine learning implementation. Volumes have persistent data; so image dependencies don't have to be reinstalled on startup.
- **Docker Network**: A docker bridge network (`app_network`) enables secure communication between containers, using a subnet defined as 172.17.0.0/16. Services are accessible via their service names 84.8.153.220(caddy, web and backend) within the network, simplifying configuration.
- **Forwarded Ports**: The Caddy service exposes ports 80 for HTTP and 443 for HTTPS. Other services, such as the webserver (Gunicorn), are not exposed directly, and can only be accessed via Caddy's reverse proxy.
- **Scaling**: Docker Compose supports scaling services to handle increased traffic, in cases of high traffic. Caddy's load balancer distributes requests across the scaled instances.

```
version: '3.8'

services:
  caddy:
    image: caddy:latest
    container_name: caddy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./caddy:/etc/caddy # Full config dir bind mount
      - /home/eeg/ssl:/etc/ssl/caddy
```

```
      - ./requirements.txt/:/app
      - caddy_data:/data # Caddy's internal TLS data
      - caddy_config:/config # Caddy's dynamic config
    restart: unless-stopped
  web:
    image: python:3.11-slim
    container_name: frontend
    working_dir: /app
    volumes:
      - ./:/app
    command: >
      sh -c "pip install --no-cache-dir -r requirements.txt &&
             gunicorn app:app --bind 0.0.0.0:8000"
    restart: unless-stopped
  backend:
    image: python:3.11-slim
    container_name: backend
    working_dir: /predict
    volumes:
      - ../backend:/predict
    command: >
      sh -c "pip install --no-cache-dir -r requirements.txt &&
             gunicorn server:server --bind 0.0.0.0:6000"
    ports:
      - "6000:6000"
    restart: unless-stopped
volumes:
  caddy_data:
  caddy_config:
```

4.11 Caddy

Caddy serves as the reverse proxy, handling incoming HTTP/HTTPS requests and managing TLS/SSL certificates. Its simplicity, performance, and Docker compatibility make it an ideal choice for the stack.

- **Reverse Proxy**: Caddy routes incoming requests to the appropriate services, as to ensure that inbound traffic cannot directly interact with local services.
- **TLS/SSL Management**: Caddy automatically provisions and renews SSL certificates from Cloudflare, removing the need for manual certificate management. It supports zero-downtime certificate renewals and enforces secure TLS configurations.
- **Load Balancing**: Caddy's built-in load balancer distributes traffic evenly across multiple services. Automatic Health checks on services are supported to divert traffic away from down instances.
- **Docker Integration**: Caddy runs inside of a Docker container, defined in the Docker Compose file. It integrates with the bridge Docker network, allowing seamless communication with other services. Volume binds are used to persist Caddy's configuration, SSL certificate and key.

- **Configuration**: Caddy's configuration is defined in a `Caddyfile`, which specifies routing rules and TLS settings. Below is my `Caddyfile`:

```
eeg.maxh.work {
    tls /etc/ssl/caddy/cert.pem /etc/ssl/caddy/key.pem
    reverse_proxy web:8000
}
```

4.12 Remote Development

This section evaluates the organisation of the project, efficient methods for development, data safety and version control. A crucial part of the development is ensuring data safety, to ensure that data privacy is prioritised. All development to the project was completed within a remote SSH, utilising privileged users and private authentication keys.

- **Block Storage Access**: Oracle Cloud's block storage is used to store the encrypted training data for the machine learning model development. The data is encrypted using AES-256 and mounted as a volume to a Ubuntu instance. Access is restricted to specific authorized users that have been authenticated by the leadership of the project. SSH keys stored in authorised_keys are stored on the host instance, to allow access to key holders, encrypted using RSA.
- **Development Workflow**: GitHub is used for version control and a CI/CD pipeline to deploy updates to production to enable automated testing utilising GitHub Actions. Using GitHub also allows for mutiple instances of the repository in branches, and this was particularly useful when changing the projects direction to a classification model. Specific tmux environments are utilised during testing to enable persistency whilst model training. This ensures that if the remote SSH connection is terminated, models remain training.
- **Project Structure** By organising the project structure, it enables development efficiency and modulation. All implantations of classes follow PEP-8 naming structure and doc strings. Modular functions such as the data loader or feature extraction has accurate type handling and object instantiation. Inheritance is used for the EEGCNN to ensure that it follows PyTorch standards, and the individual training and testing sets inherit from the same data loader with corresponding indices to ensure that the initial EEG data is only loaded into memory once.

```
LICENSE
README.md
backend/
   analysis/
      __init__.py
      models.py        # Model definitions
      train.py         # Training logic
   data_io/
      __init__.py
      dataloader.py    # Data loading/pipeline
      saliencymap.py   # Saliency map generation
      writeresults.py  # Output handlers
   features_extract/
      __init__.py
      _coh.py          # Coherence features
      _utils.py        # Shared utilities
```

```
      core.py          # Feature extraction core
   main.py              # Entry point
   params.py            # Configuration/hyperparameters
   requirements.txt     # Backend dependencies
   server.py            # API/backend server
   trained_models/      # Persistent model storage
       ([...]).pt       # (Keep only latest models)
frontend/
   app.py               # Frontend server (Flask/etc.)
   static/
      dist/output.css   # Compiled CSS
   templates/
      index.html        # Main UI
      results.html      # Results page
   requirements.txt     # Frontend dependencies
literature/              # (Optional) Research docs
    ...pdf
```

References

Abdullah, Faye, I. & Islam, M. R. (2022), 'Eeg channel selection techniques in motor imagery applications: A review and new perspectives', *Bioengineering* **9**(12).
URL: *https://www.mdpi.com/2306-5354/9/12/726*

Allen, A. P., Kennedy, P. J., Dockray, S., Cryan, J. F., Dinan, T. G. & Clarke, G. (2017), 'The trier social stress test: Principles and practice', *Neurobiology of Stress* **6**, 113–126. SI:Stressors in animals.
URL: *https://www.sciencedirect.com/science/article/pii/S2352289516300224*

Amin, S. U., Alsulaiman, M., Muhammad, G., Mekhtiche, M. A. & Shamim Hossain, M. (2019), 'Deep learning for eeg motor imagery classification based on multi-layer cnns feature fusion', *Future Generation Computer Systems* **101**, 542–554.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X19306077*

Anupreet Kaur, S. & Sridhar, K. (2023), 'Trends in eeg signal feature extraction applications.', *Frontiers in Artificial Intelligence* **5**.

Baccalá, L. A. & Sameshima, K. (2001), 'Partial directed coherence: a new concept in neural structure determination', *Biological Cybernetics* **84**(5), 463–474.

Banihashemi, L., Peng, C. W., Verstynen, T., Wallace, M. L., Lamont, D. N., Alkhars, H. M., Yeh, F.-C., Beeney, J. E., Aizenstein, H. J. & Germain, A. (2021), 'Opposing relationships of childhood threat and deprivation with stria terminalis white matter', *Human Brain Mapping* **42**(8), 2445–2460.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.25378*

Bell, J. & Sejnowski, T. J. (1995), 'An information-maximization approach to blind separation and blind deconvolution', *Neural Computation* **7**(6), 1129–1159.

Berry, R. B. et al. (2017), 'Aasm scoring manual updates for 2017 (version 2.4)', *Journal of Clinical Sleep Medicine* **13**(5), 665.

Bowen, Z., Winkowski, D. E., Seshadri, S., Plenz, D. & Kanold, P. O. (2019), 'Neuronal avalanches in input and associative layers of auditory cortex', *Frontiers in Systems Neuroscience* **13**.
URL: *https://api.semanticscholar.org/CorpusID:155955008*

Cheng, X., Huang, K., Zou, Y. & Ma, S. (2023), 'Sleepegan: A gan-enhanced ensemble deep learning model for imbalanced classification of sleep stages', *arXiv preprint arXiv:2307.05362* .
URL: *https://arxiv.org/abs/2307.05362*

Chriskos, P., Frantzidis, C. A., Gkivogkli, P. T., Bamidis, P. D. & Kourtidou-Papadeli, C. (2020), 'Automatic sleep staging employing convolutional neural networks and cortical connectivity images', *IEEE Transactions on Neural Networks and Learning Systems* **31**(1), 113–123.

Chriskos, P., Kaitalidou, D. S., Karakasis, G., Frantzidis, C., Gkivogkli, P. T., Bamidis, P. & Kourtidou-Papadeli, C. (2017), Automatic sleep stage classification applying machine learning algorithms on eeg recordings, *in* '2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)', pp. 435–439.

Delimayanti, M. K., Purnama, B., Nguyen, N. G., Faisal, M. R., Mahmudah, K. R., Indriani, F., Kubo, M. & Satou, K. (2020), 'Classification of brainwaves for sleep stages by high-dimensional fft features from eeg signals', *Applied Sciences* **10**(5).
URL: *https://www.mdpi.com/2076-3417/10/5/1797*

Ding, X., Yue, X., Zheng, R., Bi, C., Li, D. & Yao, G. (2019), 'Classifying major depression patients and healthy controls using eeg, eye tracking and galvanic skin response data', *Journal of Affective Disorders* **251**, 156–161.
URL: *https://www.sciencedirect.com/science/article/pii/S0165032718330064*

Dornhege, G., Blankertz, B., Krauledat, M., Losch, F., Curio, G. & Muller, K.-R. (2006), 'Combined optimization of spatial and temporal filters for improving brain-computer interfacing', *IEEE Transactions on Biomedical Engineering* **53**(11), 2274–2281.

Frantzidis, C. A. et al. (2018), 'Cortical connectivity analysis for assessing the impact of microgravity and the efficacy of reactive sledge jumps countermeasure to nrem 2 sleep', *Acta Astronautica* . to be published.
URL: *https://www.sciencedirect.com/science/article/pii/S0094576518311111*

Fu, R., Chen, Y.-F., Huang, Y., Chen, S., Duan, F., Li, J., Wu, J., Jiang, D., Gao, J., Gu, J., Zhang, M. & Chang, C. (2022), 'Symmetric convolutional and adversarial neural network enables improved mental stress classification from eeg', *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **30**, 1384–1400.

Gamel, F. et al. (2025), 'Eeg electrode setup optimization using feature extraction techniques for neonatal sleep state classification', *Frontiers in Computational Neuroscience* **19**.
URL: *https://www.frontiersin.org/articles/10.3389/fncom.2025.1506869/full*

Gramfort, A. (2013), 'Meg and eeg data analysis with mne-python', *Frontiers in Neuroscience* **7**, 267.

Gramfort, A., Luessi, M., Larson, E., Engemann, D., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L. & Hämäläinen, M. (2013), 'Meg and eeg data analysis with mne-python', *Frontiers in Neuroscience* **7**, 267.

Hajinoroozi, M., Mao, Z., Jung, T.-P., Lin, C.-T. & Huang, Y. (2016), 'Eeg-based prediction of driver's cognitive performance by deep convolutional neural network', *Signal Processing: Image Communication* **47**, 549–555.
URL: *https://www.sciencedirect.com/science/article/pii/S0923596516300832*

Hou, Y., Zhou, L., Jia, S. & Lun, X. (2020), 'A novel approach of decoding eeg four-class motor imagery tasks via scout esi and cnn', *Journal of neural engineering* **17**(1), 016048.

Iqbal, H. (2018), 'Plotneuralnet', https://github.com/HarisIqbal88/PlotNeuralNet. Version 1.0.0, archived on Zenodo: https://doi.org/10.5281/zenodo.2526396.

Jebelli, H., Hwang, S. & Lee, S. (2018), 'Eeg-based workers' stress recognition at construction sites', *Automation in Construction* **93**, 315–324.
URL: *https://www.sciencedirect.com/science/article/pii/S092658051830013X*

Jin, J., Miao, Y., Daly, I., Zuo, C., Hu, D. & Cichocki, A. (2019), 'Correlation-based channel selection and regularized feature optimization for mi-based bci', *Neural Networks* **118**, 262–270.
URL: *https://www.sciencedirect.com/science/article/pii/S0893608019301960*

Khan, J., Bhatti, M. H., Khan, M. U. & Iqbal, R. (2019), 'Multiclass eeg motor-imagery classification with sub-band common spatial patterns', *EURASIP Journal on Wireless Communications and Networking* **2019**.

Khosla, A., Khandnor, P. & Chand, T. (2020), 'A comparative analysis of signal processing and classification methods for different applications based on eeg signals', *Biocybernetics and Biomedical Engineering* **40**(2), 649–690.
URL: *https://www.sciencedirect.com/science/article/pii/S0208521620300231*

Kingma, D. & J.Ba (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .
URL: *https://arxiv.org/abs/1412.6980*

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kramer, A., Kümmel, J., Mulder, E., Gollhofer, A., Frings-Meuthen, P. & Gruber, M. (2017), 'High-intensity jump training is tolerated during 60 days of bed rest and is very effective in preserving leg power and lean body mass: An overview of the cologne rsl study', *PLoS ONE* **12**(1).

Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P. & Lance, B. J. (2016), 'Eegnet: A compact convolutional network for eeg-based brain-computer interfaces', *arXiv preprint arXiv:1611.08024* .
URL: *https://arxiv.org/abs/1611.08024*

Mumtaz, W., Ali, S. S. A., mohd yasin, m. a. & Malik, A. (2017), 'A machine learning framework involving eeg-based functional connectivity to diagnose major depressive disorder (mdd)', *Medical biological engineering computing* **56**.

Oppenheim, A. V., Willsky, A. S. & Nawab, S. H. (1997), *Signals and systems.*, Prentice-Hall signal processing series, Prentice Hall.

Tadel, F., Baillet, S., Mosher, J. C., Pantazis, D. & Leahy, R. M. (2011), 'Brainstorm: A user-friendly application for meg/eeg analysis', *Computational Intelligence and Neuroscience* **2011**(1), 879716.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1155/2011/879716*

Tian, P., Hu, J., Qi, J., Ye, X., Che, D., Ding, Y. & Peng, Y. (2017), 'A hierarchical classification method for automatic sleep scoring using multiscale entropy features and proportion information of sleep architecture', *Biocybernetics and Biomedical Engineering* **37**(2), 263–271.
URL: *https://www.sciencedirect.com/science/article/pii/S0208521616302455*

Wang, Q., Zhao, D., Wang, Y. & Hou, X. (2019), 'Ensemble learning algorithm based on multi-parameters for sleep staging', *Medical Biological Engineering Computing* **57**.