UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

How the Brain Moves: Understanding Motion-Based Brain States Using Deep Learning Techniques

Author:
Joshua TAPPER

Supervisor:
Ignasi Cos AGUILERA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2025

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

How the Brain Moves: Understanding Motion-Based Brain States Using Deep Learning Techniques

by Joshua TAPPER

Advancements in deep learning techniques continue to provide more accurate and explainable results from predictive models. We tested some of these techniques on neurological data recorded using Local Field Potentials (LFPs) to classify stages of movement as well as explain which input features were most important for that classification. Before training, we applied various methods of data cleaning and processing to achieve better accuracies. During training, we used three model architectures: a Convolutional Neural Network (CNN), a CNN that feeds into a Long Short-Term Memory (LSTM) model, and a parallel CNN and LSTM model, all of which made use of an intramodel decoder. Our strategies yielded high test accuracy results, and the analysis of those models provided interesting insights into our input features, most notably the significance of the left dorsal premotor cortex, high gamma frequency band, and multiunit frequency band in determining model predictions. [1]

[1] Code available on GitHub

Acknowledgements

A huge thank you to Dr. Ignasi Cos for his guidance and wisdom throughout my work on this project. I also owe much gratitude to Michael DePass, who was an extremely helpful subject matter expert that gave great insight during both the struggles and successes of this project. I also thank Luca Di Croce, who was a wonderful partner to work with. Finally, I would like to thank my family and friends who have supported me during this project and beyond.

Contents

Chapter 1

Introduction

1.1 Introduction

Neurological data is of great use in medicinal, psychological, and other fields of research due to the immense amount of information it can provide about our thoughts, actions, and decisions. One source of neurological data comes from local field potentials (LFPs), which use electrodes to capture electrical signals emitted from groups of neurons within the brain. The data provided by LFPs is highly dimensional, nonlinear, and very noisy. As a result, it is quite challenging to recognize, learn, and explain its meaningful patterns.

The continued advancements of deep learning algorithms, such as attention mechanisms used by Large Language Models, have proven that machines can learn data as broad and complex as human language. Therefore, it stands to reason that deep learning techniques can also be applied to neurological data to create highly successful predictive tasks. Similarly, improvements in algorithms that explain deep learning models have the potential to provide accurate and interpretable explanations of the data features that are most important for a model's predictive success.

Using machine or deep learning algorithms on neurological data is not a new idea; many papers have been written on the subject. However, the vast range of topics within neuroscience and the aforementioned continued advancements mean that there is always room for new contributions to the field. The topic and contribution of this paper is within the area of neural decoding of movement activity, specifically how certain regions and emitted frequencies within the brain drive and communicate movement. We used data collected from an LFP-based experiment in an attempt to decode neural activity, building on the work of others and introducing new ideas of our own.

1.2 Paper Overview

The purpose of this paper is to provide examples of deep learning models that make the classification of movement using LFP data possible. In addition, we will use these models to provide insights into the parts and frequencies of the brain that are most important for making predictions. This paper will contribute to the study of neurological data in the following ways. First, the use of LFPs for neurological decoding is much less common than other methods, such as electroencephalography (EEG), so we will help establish its utility for this objective. Next, we will expand on the classification work already performed on this dataset by almost doubling the number of class sizes, testing the ability of our models to make more specific classifications on relatively less data per class. We will also introduce a model architecture that has barely, if ever, been used on neurological data before. Finally, we

will introduce our ideas on how to best use trained models to extract valuable and interpretable information on the brain's relation with movement.

In the next section, we provide some more technical background of our data and summarize other work related to decoding neural activity. Then, we introduce our dataset and explain our transformation and preprocessing techniques. Afterwards, we introduce our model architectures and review our training and feature extraction methodologies. Finally, we present our results and offer some final analysis and conclusions.

Chapter 2

Background

2.1 Theoretical Background

2.1.1 Brain Physiology

The brain is a highly complex organ that controls and regulates almost all of the body's processes. It can be divided into various regions that are more or less responsible for different functions (John Hopkins Medicine, 2025). The specific region we are interested in is the cerebral cortex, which contains the primary motor cortex and the premotor cortex. The primary motor cortex serves as a common output path for movement commands, sending that output to the spinal cord. The premotor cortex can be further divided into the dorsal premotor cortex, which facilitates movement triggered by environmental cues such as a shape or noise, and the ventral premotor cortex, which assists in movement related to within-reach objects (Rothwell, 2012). See Figure 2.1 for a visualization of where these areas are located.

The way the brain controls movement (and other processes) is through the sending and receiving of electrical signals by billions of neurons (John Hopkins Medicine, 2025). The output of these neurons for a given movement task can be thought of as its own sort of model prediction based on many parameters, including the type of movement (reflexive or volitional), stimulus input (such as a light indicating a movement to begin), and feedback received from other parts of the body and brain as the movement is being performed (the sensation of touch when trying to grasp an object). In other words, the signals for movement dynamically change as the movement progresses (Evarts, 1979 and Schwartz, 2016).

As neurons transmit electrical signals, they will synchronize with other neurons to create repeating electrical signals known as neural oscillations (Psych, 2017). There are a couple of ways to measure these oscillations, the first of which is the amplitude, or size of the oscillation, measured in microvolts. The other measurement is to evaluate the oscillations based on the rate at which they repeat, known as frequency. Frequency, which is defined in Hertz (Hz), can vary from about 0.05 to 500 Hz in the brain (Buzsáki and Draguhn, 2004). Higher frequencies are generally associated with a more alert state, and therefore different bands are associated with different brain states and processes. For example, during sleep, more lower frequencies are transmitted. Different parts of the brain are known to have preferences for different frequencies, and individuals also demonstrate different frequency patterns (Klimesch, 2018).

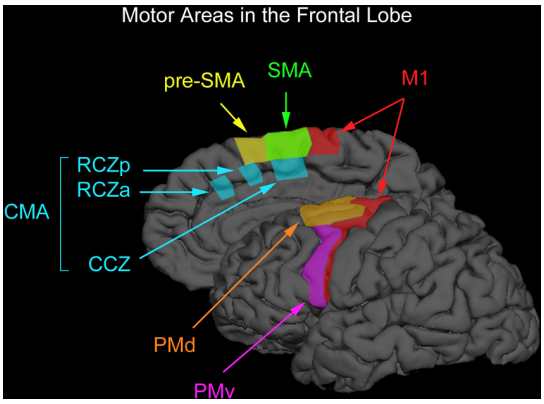


FIGURE 2.1: Image of the different parts of the brain, courtesy of Chouinard and Paus, 2010. The parts relevant for our data are the PMd, PMv, and M1.

2.1.2 Data Recording Methods

There are different methods of recording the electrical activity of a brain over time. We will discuss two methods: electroencephalograms (EEGs) and local field potentials (LFPs). EEGs, perhaps more commonly used as a result of being non-invasive, record data from the scalp. They capture a wide range of neural activity but have low spatial resolution, meaning it is difficult for them to decipher the activity in a small specific region (Cohen, 2017). LFPs on the other hand require an invasive surgical procedure and make use of electrodes to capture data of higher spatial resolution from several regions of the brain.

Although EEGs and LFPs fundamentally capture the same data, they differ in the ranges that define frequency bands. For LFPs, those bands are: 0-band (4-7 Hz), α-band (8-15 Hz), β-band (15-30 Hz), low γ-band (30-70 Hz), high γ-band (70-100 Hz).

Note that 500 Hz is the upper cutoff for recorded frequencies using LFPs (Destexhe and Goldberg, 2022). This is the case in our data, and as a result three additional bands are considered: low-ripple band (100-150 Hz), high-ripple band (150-200 Hz) and multiunit band (200-500 Hz).

2.2 Related Work

As mentioned in Chapter 1, using LFP data for neurological decoding is much less common than using EEG data. However, since the data are fundamentally the same, the work done on EEGs is very relevant to our experiments.

This paper directly builds on the work done in DePass et al., 2022, which used LFP data to classify and analyze stages of movement. They also introduced the novel idea of using higher LFP frequency bands (100+ Hz) for decoding purposes. They used Multinomial Logistic Regressors and 1-nearest neighbor classifiers on a total of 33 classification tasks. Their results identified the multiunit band, as well as the M1 and left PMd as very important for classification. Alonso, 2025 performed additional data processing and classification experiments on the same dataset with results that were limited but showed potential for future improvement. Both papers will be further discussed in Chapter 3.

Lu et al., 2021 compare and contrast neural decoding using the various activity recording methods. They explain how LFPs can encode movement-related variables;

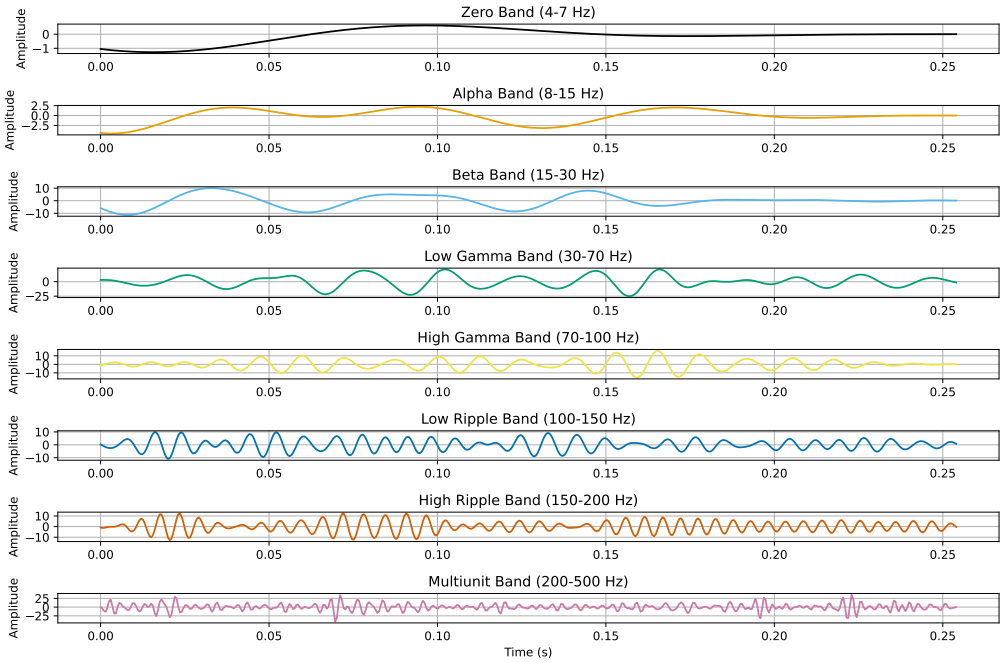


FIGURE 2.2: Example of every frequency band, taken from a sample of our data.

it has been found that higher frequency LFPs best encode speed while lower frequency LFPs encode movement initiation. More generally, they state that different recording methods are better or worse at capturing specific pieces of information.

Altaheri et al., 2023 review deep learning classification techniques for EEG motor imagery signals. They discuss how many reports have had success using minimally preprocessed raw EEG signals, and that Convolutional Neural Networks (CNNs) are the most frequently used and best performing models for classification tasks.

Fawaz et al., 2019 do a deep dive into the various deep learning methods that can be used for time series classification. Through discussion and their own testing, they demonstrate that CNNs can be effective when used for time series classification tasks.

Craik, He, and Contreras-Vidal, 2019 summarize the deep learning methods used for all types of EEG classification goals. They state that CNNs and Recurrent Neural Networks (RNNs) are higher performing than other networks. For RNNs, they suggest that Long Short-Term Memory (LSTM) models outperform both standard RNNs and Gated Recurrent Unit (GRU) models. They also mention that CNNs perform best when signal data is transformed into spectrograms for image input.

Mirchi et al., 2022 review the machine learning techniques used for neurological decoding. They reveal that lower frequency bands have been more useful for decoding motor tasks. They also reference other reports that have found better success using a combined CNN and LSTM model than when using just a CNN.

Chapter 3

Data Description and Processing

3.1 Data Overview

The data used comes from the experiment detailed in DePass et al., 2022. That paper provides the bulk of the foundation we used to perform our analysis, and to gain a full understanding of the data background we recommend reading that paper as well, although we will summarize what is most relevant to our work. The work completed in that paper includes some data processing and cleaning, which will be outlined in section 3.1.2.

3.1.1 Experiment Overview

The participants in the experiment were two adult male rhesus macaque monkeys (*Macaca mulatta*), who performed a series of precision grip tasks. Sitting in front of the structure shown in Figure 3.2, the monkeys were shown an LED signal that indicated which hand to use and when to start the movement. After the signal was given, the monkeys would move their hand from a resting position on "home plate" and reach for a target at one of four angles (0°, 45°, 90°, 135°). To acquire a juice reward, the monkeys pressed force transducers within the target using a precision grip. After the reward was delivered via a straw, the monkeys would return their hand back to home plate. These five stages of movement, denoted as Baseline, Pre-reach, Reach, Grasp, and Post-grasp (Figure 3.1), comprised a single trial of the experiment.

LFP data was collected using chronically implanted high-dimensional Utah electrode arrays. Data was recorded from five parts of the brain: the left and right ventral premotor cortex (PMv), the left and right dorsal premotor cortex (PMd) and the left primary motor (M1). The data was sampled from a total of 256 different channels at 2,034.5 Hz before being digitally recorded with a Tucker-Davis Technolgoies (TDT) system.

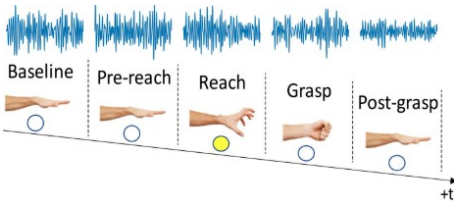


FIGURE 3.1: A visual of the five stages of movement performed in the experiment (from DePass et al., 2022).

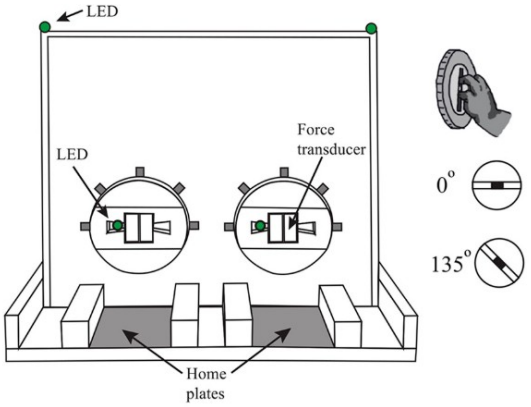


FIGURE 3.2: Experimental apparatus created for the outlined experiment (from DePass et al., 2022).

3.1.2 Data Summary

The data we worked with was an epoched version of the data recorded, meaning the data was split into five 0.25 second sections, one for each of the stages described in Section 3.1.1. This data was delivered in eight separate datasets, one for each combination of hand and angle.

The epoched data was composed of four dimensions: number of channels (always 256), number of time points (always 508), number of trials (varied depending on the dataset), and number of phases (always 5). In total, there were 315 trials, 155 using the left hand and 160 using the right hand. This, combined with our five different phases, gave us $315 * 5 = 1575$ total samples that we could use for classification training.

3.2 Data Processing

Before using the training methods described in Section 4, we put our data through various transformations and processing steps to maximize the learning potential of our models and improve the generalizability of our results. This included splitting our classes, transforming the epoched data into spectrograms, and considering and implementing various cleaning methods.

3.2.1 Class Splitting

We split our data into nine separate classes. These classes are the five stages of movement for both the left and right hands, with the exception of the baseline state, in which data from both hands were combined into a single baseline class. The reasoning behind this combination is twofold: there is no physical difference in the position of the baseline for trials done on either hand, and there is no indication of which hand will be used before the cue is given, meaning baseline data should be consistent across both hands. This created a class-size imbalance in which we had a little less than twice as many baseline samples than we did any other class.

The class labels were numerical and distributed as follows: 0 is the baseline, 1-4 represent the left-handed stages of movement in order of the experiment, while

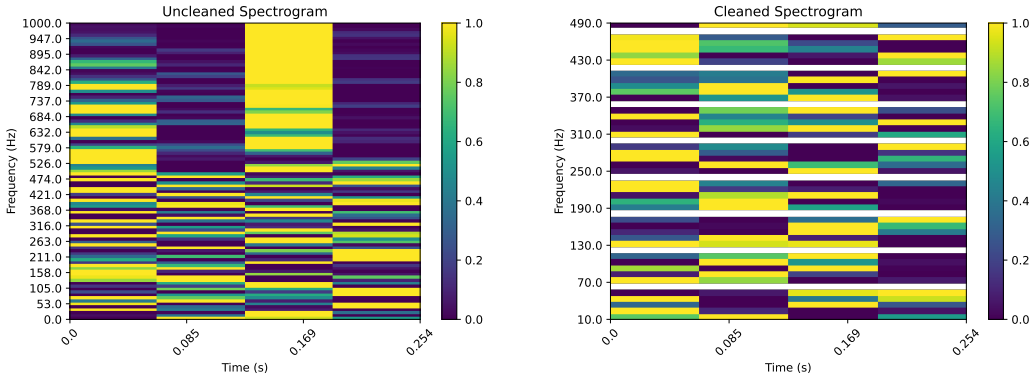


FIGURE 3.3: Example of an uncleaned and cleaned spectrogram. The white blocks in the cleaned spectrogram represent the frequencies removed by our powerline suppression.

5-8 represent the in order right-handed stages. One-hot encoding the classes was considered but ultimately deemed unnecessary.

3.2.2 Spectrogram Transformation

We opted to transform the raw data into spectrograms for training. In short, a spectrogram is a visual representation of how the frequency of a signal changes over time. This means that we are transforming our raw data into a two-dimensional image format for input into our models.

To create our spectrograms, we looped through every class and trial and used **scipy.signal.spectrogram** to create a spectrogram for each channel. This function required three important parameters. The first was our sampling frequency, which determines the number of samples per second to take to make the continuous signal data discrete. We followed the Nyquist principle, which states that the sampling frequency must be at least twice the maximum frequency of the sample to allow for perfect reconstruction (Federal Agencies Digital Guidelines Initiative, 2025), and set our sampling frequency to 2000 Hz. The other two parameters were segment length and the amount of overlap between segments. If we imagine a window being used to capture pieces of our time series, those parameters determine the size of the window and the amount it is moved in between captures. We chose a segment length of 200 and an overlap of 100, which gave us a total of four time steps in our final spectrograms.

The last thing to note for our spectrogram creation is that frequency values are grouped into bins, meaning that our analysis was performed on ranges of frequencies rather than individual values. The total number of bins in a spectrogram is calculated by $\frac{s}{2} + 1$, where s is segment length. In our case this gave us 101 frequency bins, which were divided into equal groups of our maximum frequency, $\frac{2000}{2} = 1000$, so every frequency bin within our spectrogram was made up of about 10 Hz.

3.2.3 Data Preprocessing

Before using our spectrograms for training, three additional data preprocessing methods were tested, two of which were used in the final training pipeline and one that was ultimately skipped. The overall idea was to remove any data that is potentially

or obviously faulty so that we had more confidence that our model had learned relevant and generalizable patterns.

Bad Channel Deletion

The first task was to identify any channels that were faulty or otherwise too noisy to be used in training. This strategy can be attributed to the work done in Komosar, Fiedler, and Haueisen, 2022 and was improved by Alonso, 2025. The process is as follows. First, the standard deviation of every channel is calculated on the raw data of a single sample. The median of all standard deviations is stored as a comparison point. After, an iterative process removes channels that are too flat (very low standard deviation), too noisy (very high standard deviation), or have a standard deviation too high above the overall median, until the overall median either falls below 5 or remains the same between iterations.

It is possible to apply this process on a per sample basis, however to keep our data uniform and avoid any large class imbalances, any channel identified as "bad" within a single sample had its data removed across all samples. This strategy resulted in the deletion of 11 channels, 8 from the left PMv and 3 from the left PMd.

Spectrogram Cleaning

The next preprocessing step was to clean our created spectrograms. This cleaning involved removing frequency bands with known or expected noisy and irrelevant information. To do this, we used a lower and upper frequency cutoff, as well as powerline suppression.

Our lower and upper frequency cutoffs were set at 10 and 500 Hz respectively. The upper cutoff is equal to the maximum recorded frequency of our data, meaning anything recorded at or above that frequency was unexpected and unexplainable. The minimum cutoff was designed to remove some artifacts, such as eye blinking or other small movements, from the data. These cutoffs, particularly the upper cutoff, were performed outside of spectrogram creation because it allowed for more control over the size of each frequency bin. These cutoffs also impacted our total number of frequency bins, reducing the number from 101 to 49.

Depending on the region in which the data is recorded, there is an expected narrow spike at every 50 or 60 Hz that originates from power lines and equipment. In our case, these spikes occur at 60 Hz and its harmonics, meaning we also see spikes at 120 Hz, 180 Hz, etc. To handle this noise, we implemented notch filtering, which applies a narrow band-stop filter centered on the interference frequency to eliminate the noise. This was applied on frequencies within 2 Hz of every harmonic.

Smoothing Amplitude Spikes

The final preprocessing strategy that was tested was to smooth out any large amplitude spikes seen in the data. Again, all instances of this data were likely due to artifacts outside of the experiment. Our strategy was to use a backward-looking rolling mean filter, which finds any amplitude that is over two times the average standard deviation across all channels and samples and sets it to the average of the previous five time points. If the spike occurs before there are five previous time points, the value is instead clipped to 0.

Initial testing revealed large drops in classification accuracy after smoothing was applied. While the exact reason is unknown, one theory is that the smoothing removed some amplitude spikes that were informative and not created by artifacts.

Similarly, too much smoothing could have perhaps made the data too clean and therefore more difficult for our models to generalize. Additionally, this method is more subjective in its design, and finding the right amount of time points to average or the threshold to be considered an artifact spike can be treated as its own parameter optimization problem. Thus, we opted to not implement this preprocessing on our data with the confidence that the other mentioned methods were sufficient to give us trust in the legitimacy of our results.

Chapter 4

Training and Analysis Strategy

4.1 Overview

Our training and analysis strategy was designed with two end goals: accurately classify different stages of movement and extract feature importance for interpretability. To achieve this, our training and analysis was divided into three different parts. First, we performed dimensionality reduction on the channel dimension of our spectrograms. Next, we fed the data into a classification model for classification training. Finally, we extracted and analyzed the feature importances of our models. The following sections detail the various techniques we used to accomplish all three phases.

4.2 Dimensionality Reduction

Before data was fed into a classification model, one of two methods was used to reduce the size of the channel dimension. In doing so, we thought our models would better learn the most important channels, leading to an increase in accuracy and more distinct importance scores.

4.2.1 Encoding

Autoencoding is an unsupervised deep learning technique in which a model learns to represent data in a lower-dimensional space called the latent space. This process typically involves both an encoder and a decoder, the former using nonlinear combinations to transform the data into the latent space and the latter reconstructing the latent space into the original dimensions. The loss function for these models is computed by comparing the reconstructed data with the original.

Our models used the architecture of a Convolutional Autoencoder, which uses convolutional layers to reduce and reconstruct data. This architecture has proven useful in previous EEG analyses (Al-Marridi, Mohamed, and Erbad, 2018). In our case, we only made use of the encoder to transform our data into the latent space before feeding the reduced data into a classification model. Importantly, both of these "separate" models are part of one overall model. This means that our encoder received feedback and adjusted its parameters from our model's loss function, which will be described in more detail in the following section. Due to this, the act of reconstructing the latent space using a decoder was unnecessary.

For all models, the encoder was composed of three convolutional layers, with filter sizes of 128, 64, 32 and kernel sizes of (5, 1), (5, 1), and (3, 1) respectively. This resulted in an output shape of (49, 4, 32). Note that while 49 and 4 represent the same dimensions as our input spectrograms, the nonlinear nature of the encoder creates new spectrograms that cannot be found in the original data.

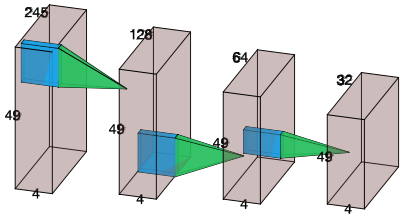


FIGURE 4.1: Visualization of our Encoder structure. The dimensions of our spectrograms remain the same, while the number of channels is reduced.

4.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a dataset into uncorrelated variables called principal components while maintaining as much of the original variance as possible. For example, one might reduce a dataset with a large amount of features to two or three principal components for the purpose of visualizing the data in a comprehensible dimensionality. PCA can also improve the performance of a deep model, exemplified in the work done by Choi and Kim, 2021.

PCA was performed on our cleaned spectrograms before model training, serving as a contrastive method to our Autoencoder. Our theory was that given the nature and complexity of our data, a model will perform better when given the opportunity to learn how to reduce the data rather than using a predetermined algorithm. We used sklearn's PCA class and opted for 32 principal components so that the input shape to our classification models would be equal to the decoder output of (49, 4, 32).

4.3 Classification Architecture

Three model architectures were used for spectrogram classification: a Convolutional Neural Network (CNN), a CNN that feeds into a Long Short-Term Memory (LSTM), and a parallel CNN and LSTM network.

All three models used the Adam optimizer with a starting learning rate of 0.001 and gradient clipping of 1.0 for increased stability. The loss used was Sparse Categorical Crossentropy Loss, which is commonly seen in multiclass classification problems when the labels are numerical and each sample belongs to a single class C. The loss is defined as follows. Let:

- $y \in \{0, 1, \ldots, C-1\}$ be the true class label
- $\hat{p}_c$ be the predicted probability for class C
- $\hat{\mathbf{p}} = [\hat{p}_0, \hat{p}_1, \ldots, \hat{p}_{C-1}]$ be the model's softmax output

Then the Sparse Categorical Crossentropy loss for a single sample is:

$$\mathcal{L}(y, \hat{\mathbf{p}}) = -\log(\hat{p}_y)$$

And for N samples:

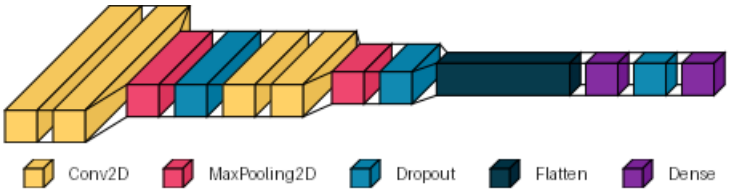


FIGURE 4.2: Architectural guidelines for our CNN models.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}_{y_i}^{(i)})$$

20% of our data was split into testing, while the rest was used for training. The same random state (42) was used when splitting to ensure consistency. During training, the learning rate was reduced by a factor of 0.95 if validation loss did not improve over 5 epochs, with a minimum learning rate of 0.00001. To decrease the risk of overfitting, early stopping was employed with a patience of 15 epochs and was set to restore the model's best weights. Training took place over a maximum of 100 epochs and used batch sizes of 128, 20% of which were split for validation.

Additionally, parameter selection for all models was optimized using a Grid Search. The parameters that produced the highest validation accuracies during training were then applied to new models and evaluated using the test data. The models that subsequently received the best test accuracies were used in the Results section.

4.3.1 CNN

Convolutional Neural Networks (CNNs) are neural networks commonly used for image classification. They use convolutional layers with learnable filters to extract features from the input data. Combined with max pooling layers that further aggregate data, CNNs are able to detect patterns within an image while preserving the spatial relationships.

Our CNN framework used two sets of stacked convolutional layers, each sharing the same layer and kernel sizes. This strategy allows us to increase the nonlinearity and thus discrimination of our models through the use of more activation functions while using less parameters (two stacked 3 x 3 layers have $2 * 3^2 * C^2 = 18C^2$ parameters while one 5 x 5 layer has $1 * 5^2 * C^2 = 25C^2$ parameters), reducing our potential overfitting. This method has been shown to improve model accuracy (Simonyan and Zisserman, 2015). Every convolutional layer used the ReLU activation function and performed zero-padding on the spectrograms. After each stack, we added a max pooling and dropout layer. Then, we flattened the data for input into a ReLU dense layer, added a final dropout layer, and finished with a Softmax dense layer for classification output.

We also now note our grid searches for all three model architectures considered asymmetric kernels of shapes (5, 3) and (7,3). This was due to the large difference in the dimension of our frequency axis (49) and time axis (4). We believed that larger sizes along our frequency axis would be beneficial in capturing trends, but did not want to capture the entirety of the time axis in a single kernel.

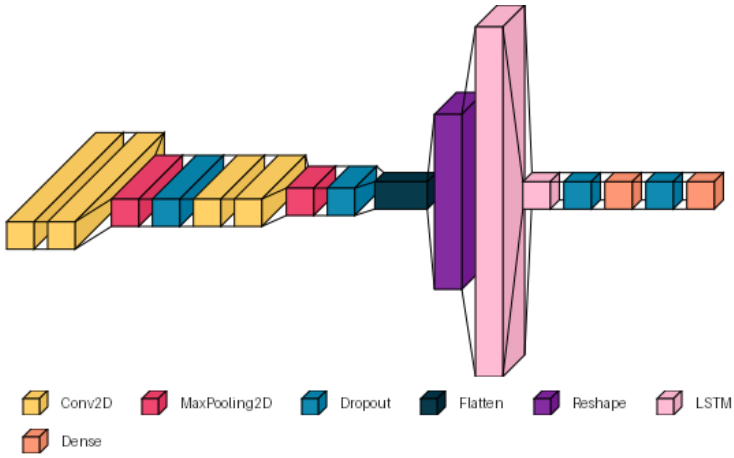


FIGURE 4.3: Architectural guidelines for our CNN into LSTM models.

4.3.2 CNN into LSTM

These models begin with a CNN architecture similar to what is described in 4.3.1, however after flattening, the data was reshaped and fed into a Long Short-Term Memory (LSTM) model. An LSTM is a type of Recurrent Neural Network (RNN) designed to operate over sequences of vectors and capture over-time features. LSTMs improve upon vanilla RNNs by adding an internal cell state and gating mechanisms, which control information flow and help avoid problems such as vanishing gradients. Our model includes two stacked LSTM layers to encourage hierarchical feature learning. That stack is followed by a dropout layer. The model then ends with a ReLU dense layer, a final dropout layer, and a softmax dense classification layer.

4.3.3 Parallel CNN and LSTM

This model architecture is based on the work of Xu et al., 2020 and Bae, Choi, and Kim, 2016, both of whom used it to achieve better accuracy results for classification on spectral data than they did using CNNs or LSTMs. The idea behind this architecture is that rather than force the LSTM or CNN to learn from the output of the other, we allow them to separately compute the temporal and spatial features and then combine the two for classification. To the best of our knowledge, this model has not yet been tested with neurological data, so our work can be considered a novel application of this architecture for neural decoding.

As with our other models, this model receives the input provided by our encoder, however this input is then split and directly fed into both CNN and LSTM layers. The CNN side of the model is similar to our other models except that we omitted the dropout in between our layer stacks and opted for a single dropout layer after the ReLU dense layer output. The LSTM side is the same as in our CNN-LSTM model, with two stacked layers and a dropout layer. The outputted features from both sides are then merged into a single concatenated output, which is fed into a fully connected ReLU and softmax layer, with a dropout layer in between.

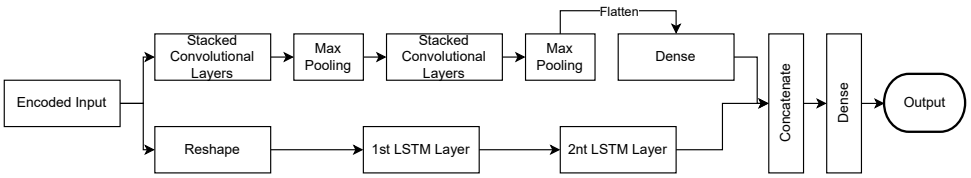


FIGURE 4.4: Overview of the parallel model structure. Dropout layers are not shown.

4.4 Feature Extraction

After training the models, we used various techniques to identify the importance of both channels and frequencies so that we could offer explainability of our results and either provide new insights or reinforce preexisting beliefs on the regions and frequencies of the brain that are most important for understanding movement.

Of note is that the methods described could only be applied to our models that used encoders, as they were fed the original data, whereas our PCA-based models were trained on the linear combinations of our original data. For PCA, we made use of its **.components_** attribute to extract importances for comparison.

4.4.1 SHAP

SHAP (SHapley Additive exPlanations) is an explainability framework introduced in Lundberg and Lee, 2017. SHAP unifies previous approaches in interpreting model prediction and introduces a new class of additive feature importance measures based on Shapley values, which originate from game theory. These SHAP values are thus a theoretically grounded method of determining the importance of input features by assessing the impact of those features on a model's predictions.

The SHAP library provides various classes that can be used on models to provide feature importance. Our preferred method is to use SHAP's DeepExplainer, which uses the DeepLIFT feature importance technique, introduced in Shrikumar, Greenside, and Kundaje, 2019. DeepLIFT is a recursive feature attribution method that explains model predictions by measuring changes in activation between an input and a "reference" input. The key differences between SHAP and the original DeepLIFT is that the SHAP method uses multiple background samples instead of a single reference value, and SHAP uses Shapley equations to linearize components. For our CNN-LSTM and Parallel models, we were forced to use SHAP's model agnostic KernelExplainer, as the DeepExplainer is unable to process the Reshape layer used in those models. Our code was written with the guidance of the SHAP GitHub repository (SHAP, 2025). Further limitations of SHAP will be discussed in Section 4.5.

4.4.2 Integrated Gradients

Another way of assessing input importance is by using Integrated Gradients, a concept introduced in Sundararajan, Taly, and Yan, 2017. Integrated Gradients are calculated by multiplying the difference of an input and "baseline" feature with the

integral of the gradients generated in the space between the baseline and input. In the authors' own words, this technique is easy to implement and satisfies two important axioms: sensitivity (a value is assigned to any feature that can change the prediction of different inputs) and implementation invariance (values are the same for two models with different architectures but equal outputs).

Our implementation of integrated gradients is based on the code found in Nain, 2020, adopted to handle our multichannel spectrogram input. We used the zero-baseline, which is an all black image, as it has been chosen and used successfully in previous EEG analyses (Warrick and Nabhan Homsi, 2018). Since Integrated Gradients are designed to explain the features of a single input, such as the pixels used for the classification of a single image, we obtained our overall feature importance by averaging the individual Integrated Gradients of 100 randomly selected samples from our testing data. For each individual calculation, we used 250 steps in the Riemann integral approximation (within the recommended 20-300 range), and took the average of two calculations for each test sample.

4.4.3 Ablation

Ablation refers to the process of removing or changing certain components of a model pipeline to evaluate their contribution to the overall performance of a model. For our research, ablation was applied to the data itself so that we could evaluate how setting channel or frequency data to 0 impacted our trained models' accuracies. The final ranking of "importance" is solely based on the change in accuracy caused by the ablated feature.

For all selected models, channel ablation was performed on individual channels and regions of the brain. Frequency ablation was band-based, performed on both individual bands and pairs of bands. Since our spectrograms are made of bins of 10 Hz, we were unable to be exact in our ablation for all bands. The 0 band could not be considered at all, as our final spectrograms begin at 10 Hz. The alpha band was rounded up to 20 Hz, thus limiting the beta band to 20-30 Hz. All other bands are exactly as described in Section 2.1.2.

4.5 Limitations

Several limitations were observed or expected before running our experiments. The first and perhaps most important limitation was our relative lack of data; 1575 total samples divided across 9 classes is far from a huge data size. We believe that access to more data would in and of itself increase the accuracies of all our models presented in the next section.

Inconsistencies in model training were observed throughout our experiments. Specifically, a single set of parameters could result in final test accuracies as different as 15 to 20 percentage points. This can be attributed to the complexity of the data, the amount of overfitting, and perhaps the inclusion of our encoder (more on that in Section 6.3). To account for this, we trained each set of top grid search parameters five times and selected the model with the highest test accuracy for analysis.

We expected our results to also be limited by the fact that we used epoched data that did not make up the full duration of each phase. The primary limitation was that our model could not learn from the full cycle of a phase; we expect different characteristics of each phase at its beginning, middle, and end, which could make

learning easier or more difficult. It also prevented us from performing a more thorough analysis on class differences, such as asking the question: "Is the similarity between end of reach and beginning of grasp closer than the middle parts of those phases?"

Other than being unable to use SHAP's DeepExplainer for two of our architectures, we also had much difficulty using the KernelExplainer method. This is because this explainer requires data to be flattened, which in our case results in a massive 48,020 features for analysis. SHAP then defaults to reevaluating the model a total of $2 * nfeatures + 2048$ times, which in our case yields 98,088 "samples" for overall evaluation. This number is well above the number needed to crash any hardware we have access to. We were able to limit the number of samples of the KernelExplainer through a parameter; however, the maximum number of samples we could use before crashing our hardware was about 700, meaning we expected a high variance in the SHAP values we received. In short, while using SHAP on all our models is theoretically possible, hardware limitations have put us into a situation where we cannot be confident with the consistency of the final values for our CNN-LSTM and Parallel models.

Chapter 5

Results and Analysis

5.1 Accuracy Results

We begin by presenting our models that achieved the highest test accuracy scores for each combination of data processing and classification model architecture. Afterwards, we will focus on our models that were trained using fully processed data and an intramodel decoder, which we will refer to as Method A.

5.1.1 Methodology Comparison

Table 5.1 compares the test accuracies of various methods. The first three rows contain the accuracies of our Method A models. The second trio are models that are trained on data that was preprocessed and reduced through PCA. The next three models were trained on data that was not preprocessed, but did use the encoding method. The final three models are trained using the same parameters as our Method A models but without any dimensionality reduction.

This table supports our hypothesis that Method A is the best training strategy. Across all three models, the incorporation of the encoder as a part of training resulted in a higher test accuracy than using PCA to reduce our channel dimension. This suggests that encoder-based models were able to learn a better representation of the latent space than the traditional PCA method. A deeper analysis of channel importance is provided in Section 5.2.

Additionally, we see that our preprocessing in general led to higher test accuracies, with an average test accuracy of 80.2% for models trained on preprocessed data versus an average test accuracy of 78.2% on data not preprocessed. It is noteworthy

Model	Preprocessing?	Dimensionality Reduction	Test Acc
CNN	Yes	Encoder	75.2%
CNN-LSTM	Yes	Encoder	83.2%
Parallel	Yes	Encoder	82.2%
CNN	Yes	PCA	74.6%
CNN-LSTM	Yes	PCA	71.4%
Parallel	Yes	PCA	71.7%
CNN	No	Encoder	81.2%
CNN-LSTM	No	Encoder	78.0%
Parallel	No	Encoder	75.5%
CNN	Yes	None	78.4%
CNN-LSTM	Yes	None	77.1%
Parallel	Yes	None	80.3%

TABLE 5.1: Comparison of different methodologies.

Model	Train Acc	Test Acc	Precision	Recall	Total Parameters
CNN	88.0%	75.2%	76.7%	76.9%	484,137
CNN-LSTM 1	89.9%	83.2%	85.6%	83.3%	256,201
CNN-LSTM 2	90.0%	81.6%	83.9%	81.7%	465,865
Parallel 1	93.9%	82.2%	83.1%	82.7%	719,049
Parallel 2	92.8%	81.6%	82.3%	82.3%	708,361

TABLE 5.2: A high-level comparison between different models trained using our Method A.

that our CNN trained on non-preprocessed data outperformed our CNN trained on preprocessed data, however this could be due to the model learning from unimportant and non-generalizable patterns, such as the 60 Hz powerline activity. We offer some evidence to support this theory in Section 5.2.

At 78.6%, the average accuracy of our models trained without dimensionality reduction is closest to our mean of Method A. We also again see a better CNN accuracy score using an alternative method than using our Method A. Therefore, it is possible that the addition of dimensionality reduction is ultimately unnecessary, even if it might help slightly increase accuracy. At the very least, we are confident that the addition of the decoder did not make learning more difficult for our models.

5.1.2 Model Analysis

We now dive deeper into analysis of our Method A models. For further comparison, we introduce additional CNN-LSTM and Parallel models that achieved high test accuracies using Method A.

Table 5.2 demonstrates a pattern of overfitting consistent across all models, although given the complexity of our data a gap between train and test accuracy is expected. The table also highlights that CNN-LSTM 1, which has the highest accuracy of any model and the fewest parameters, also has the smallest gap between training and testing accuracies. This suggests that future optimization attempts are better off trying to reduce parameter size, or at the very least avoiding increases.

Tables 5.3 and 5.4 present the parameters selected using our grid search. Note that our parallel models also had CNN-specific dense layers of 128 and 64 respectively. Despite the observed overfitting, most models did use a larger final dense layer, including CNN-LSTM 1. Similarly, maximum dropout sizes were not always used. This suggests that too much aggression in preventing overfitting could be damaging to a model's final test accuracy by making training too difficult and/or preventing models from learning even while overfitting is occurring (an observation noted in Dos Santos, Sabourin, and Maupin, 2009). There is not much agreement about the optimal LSTM or CNN kernel sizes, although the CNN-LSTM and Parallel models do make use of smaller filters and kernels than the pure CNN, which demonstrates that including an LSTM allows the use of less complex CNNs without sacrificing accuracy.

5.2 Feature Extraction

We present the results of our various feature extraction methodologies, keeping in mind the big picture idea that regions and frequencies identified as more important

Model	Layer 1	K 1	Layer 2	K 2	Final Dense	CNN D.O.	End D.O.
CNN	32	(7, 3)	64	(7, 3)	128	0.5	0.5
CNN-LSTM 1	16	(3, 3)	32	(5, 3)	128	0.4	0.4
CNN-LSTM 2	16	(3, 3)	32	(5, 3)	128	0.3	0.5
Parallel 1	16	(5, 3)	32	(7, 3)	128	0.4	0.5
Parallel 2	16	(5, 3)	32	(5, 3)	64	0.3	0.5

TABLE 5.3: Parameter comparison of selected Method A models for both the CNN and fully connected layers.

Model	LSTM Size	LSTM Dropout
CNN-LSTM 1	32	0.4
CNN-LSTM 2	128	0.4
Parallel 1	64	0.3
Parallel 2	64	0.4

TABLE 5.4: Parameter comparison for the LSTM-specific layers.

for classification could also be useful in understanding movement in non-machine learning settings.

5.2.1 SHAP Results

After computing the SHAP values for our models, we isolated the values for channels and frequency bins by taking the mean of the absolute SHAP values across the relevant dimensions. Figures 5.1 and 5.2 present visual representations of these values for our CNN model. The clear regional favorite is the left PMd, mostly because of three very important channels quite close together but also because of a general high importance across the region. The left PMv also has a few top channels, while the left motor channel scores are more consistent throughout the region. For the right-sided regions, the PMd appears to carry more weight than the PMv, although both have less importance than the left-sided regions.

For frequencies, the high gamma frequency bins are seen as more important than the bins in any other band, although the adjacent low gamma and low ripple bins also have high importance. Also of note is that the multiunit bins generally decline in importance as frequency increases.

Figure 5.3 shows our SHAP values using the KernelExplainer. As forewarned in Section 4.5, the low score values (the highest scores are about 40 times lower than the

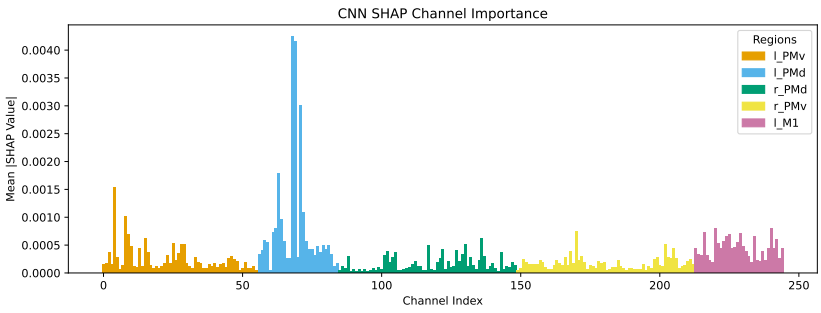


FIGURE 5.1: Channel importances for our CNN model using SHAP values.

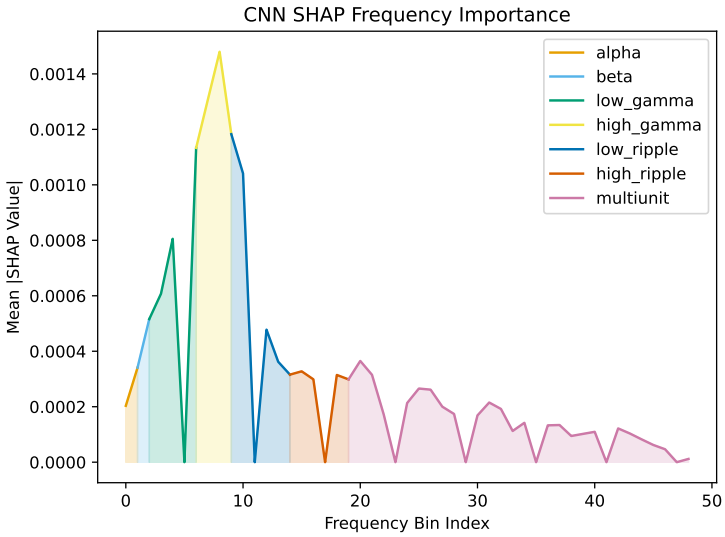


FIGURE 5.2: SHAP scores for all frequency bins. Note that the empty bins are due to the deletion of powerline noise.

highest CNN SHAP score) and many channels with scores close or equal to 0 imply that we are unable to effectively use SHAP for these architectures due to limitations in either our hardware or the functionality of the KernelExplainer. As a result, we will not reference this output in other analysis.

We also acquired the SHAP channel importances on our CNN model trained on unclean data. While the top three channels were the same as the CNN trained on clean data, the fourth most important channel was channel 9, one of the channels that was identified for removal in our bad channel deletion algorithm. This offers some evidence that the model has learned from bad features and is not the best representation of generalizable trends and findings.

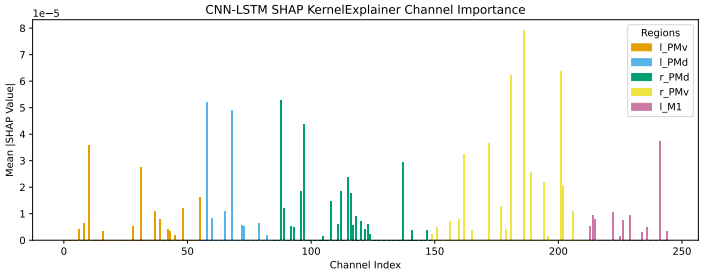
5.2.2 Integrated Gradients Results

As with our SHAP values, our Integrated Gradient values were calculated using the mean absolute values. The channel results can be viewed in Figure 5.4 All three models have similar top channels; they highly regard a "cluster" of channels in the left PMd and give high importance to some channels in the left PMv. The evaluations of the right PMd and left motor are not as consistent, although the general order of importance appears to be the right PMd, then the left motor, and finally the right PMv with very little importance.

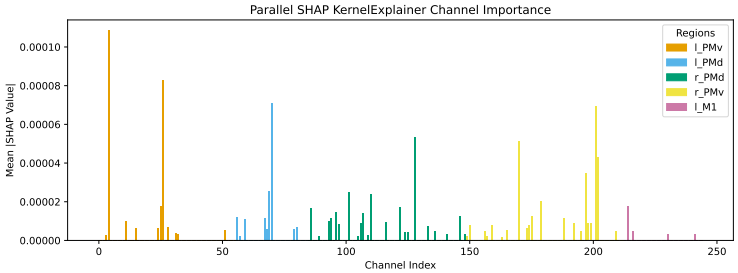
Figure 5.5 plots the frequency bin importances. Compared to the CNN SHAP values, the Integrated Gradients see the higher frequency bands: low ripple, high ripple, and multiunit, as relatively more important and the low and high gamma bands as relatively less important. Both methods assign low importance to the alpha and beta bands, although the difference is more exaggerated with Integrated Gradients. We also want to highlight that the Integrated Gradients assign values to the removed powerline frequencies, while SHAP does not.

5.2.3 Ablation Results

Ablation was performed on our top model of each Method A architecture type.



(A) CNN-LSTM



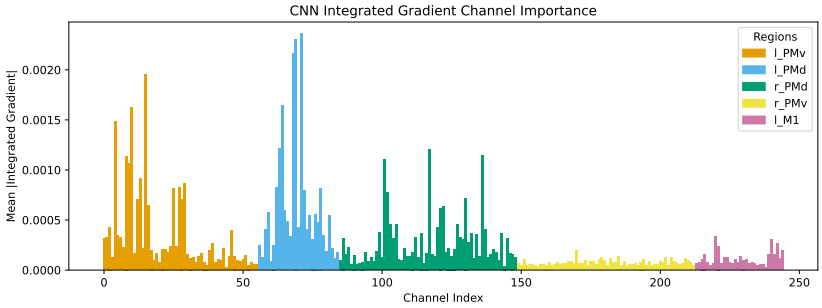
(B) Parallel

FIGURE 5.3: SHAP scores for our CNN-LSTM and Parallel models using the KernelExplainer. The low y-axis and missing values indicate that these results are not as trustworthy as others we provide.

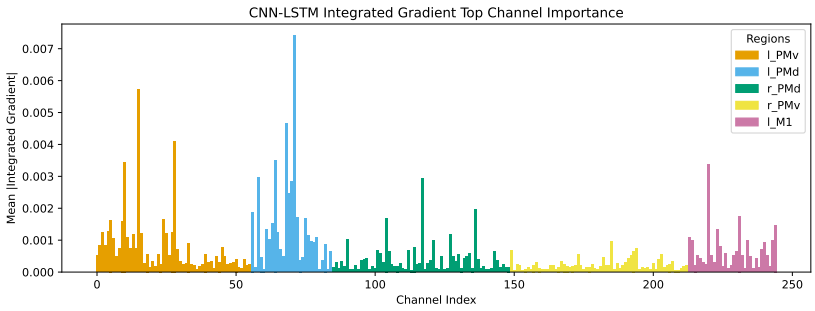
Channels

Individual accuracy drops are displayed in figure 5.7. There are a few observations that caught our attention. First, the cluster of channels in the left PMd that were given high importance values also caused very large accuracy drops. This provides further evidence of a region within the left PMd that is essential for determining and understanding the movements performed in the experiment. Next, the left PMv did not cause the accuracy drops that might have been expected when viewing the importance scores, especially from Integrated Gradients. This does not have to mean that the importance values are incorrect; our models could be more robust to the removal of important left PMv channels by accounting for its removal in the aggregate. Finally, we noticed that the removal of some channels caused an increase in the accuracy of our models. This surprising result indicates that our models could benefit from a more aggressive bad channel deletion strategy.

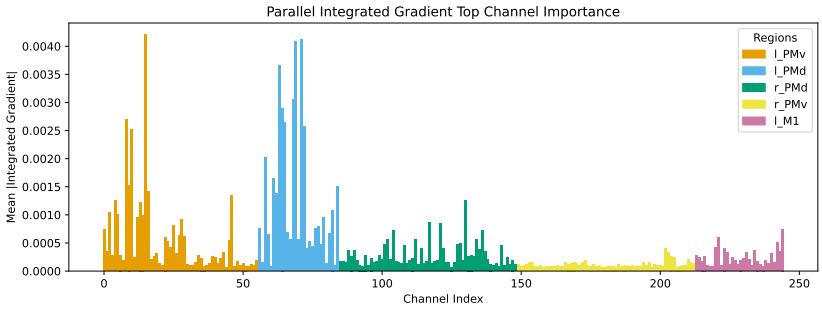
Turning to regional ablation, we again find some intriguing results. The high drops seen when the left PMd was ablated is unsurprising given our previous results, but what is surprising is the impact of ablating the right PMd, especially for our Parallel model, where ablating the right PMd caused a larger accuracy drop than ablating the left PMd. The reason this result is particularly curious is because of the low accuracy drops caused by individual channels when ablated. This could be due to channels being more equal in their individual impact when compared to other regions, even though as a whole they are vital for classification. This theory is related to the function of the contralateral hemisphere, which is discussed in further detail in Section 5.3.1.



(A) CNN



(B) CNN-LSTM



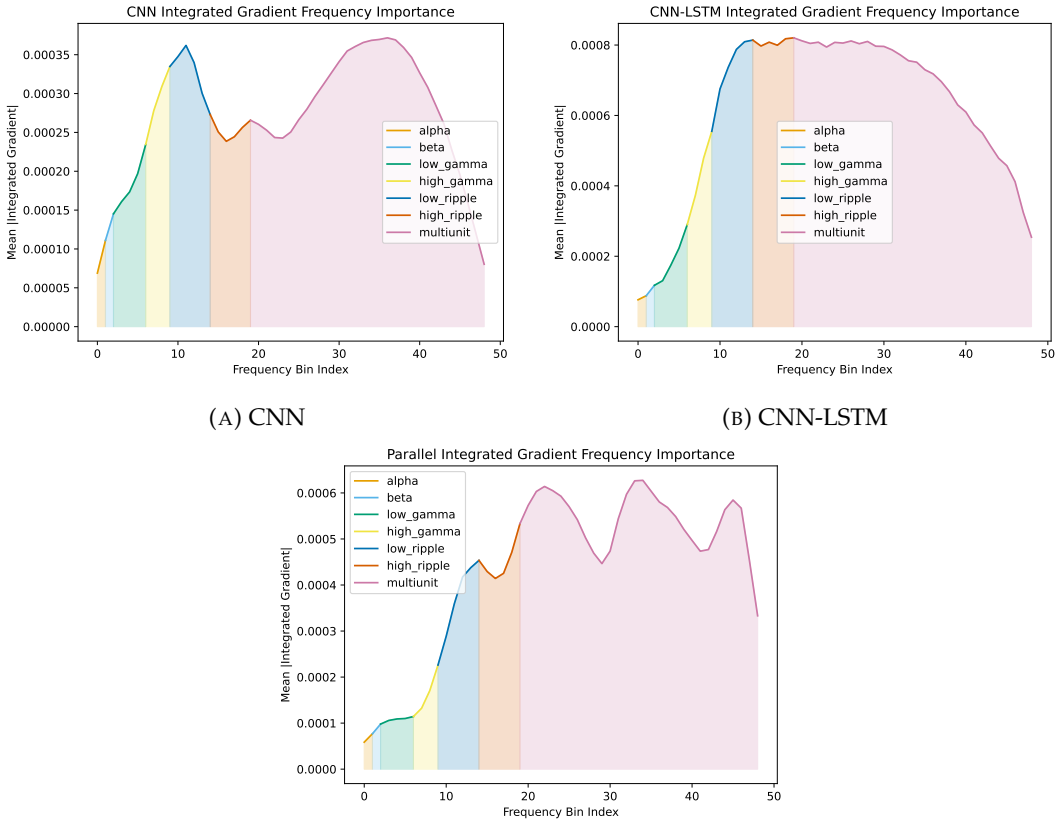
(C) Parallel

FIGURE 5.4: Channel importances for each model using average absolute Integrated Gradients.

Frequencies

Figure 5.8 shows the accuracy loss for all three models given every combination of frequencies, with single frequency tests along the left diagonal.

All three models showed the greatest accuracy losses when the multiunit band was removed. Our SHAP results would suggest this is due to the sheer amount of bands being removed summing up to make a large lack of data, while our Integrated Gradients would say it is because the multiunit is the most important band. Our CNN is also reliant on the high gamma band for classification, as it was the band that caused the largest accuracy drop when ablated individually, something better predicted by our SHAP values. The CNN-LSTM and Parallel models were more robust to non-multiunit ablation but more impacted by the multiunit ablation, suggesting that LSTMs might find useful sequential information in that band that is less obvious to a CNN. Besides that, they both suffered the greatest accuracy losses when the low and high gamma bands were ablated.



(A) CNN

(B) CNN-LSTM

(C) Parallel

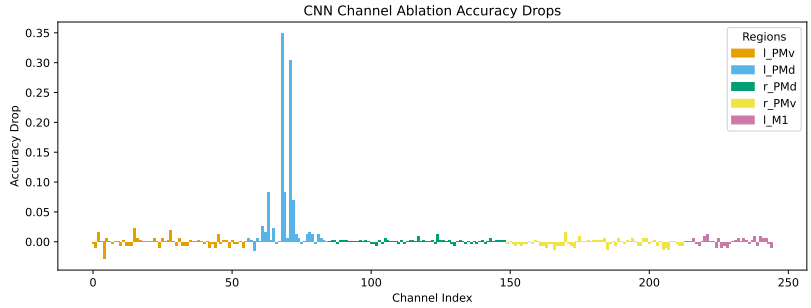
FIGURE 5.5: Frequency bin importance for each model using Integrated Gradients.

5.2.4 PCA on Spectrograms

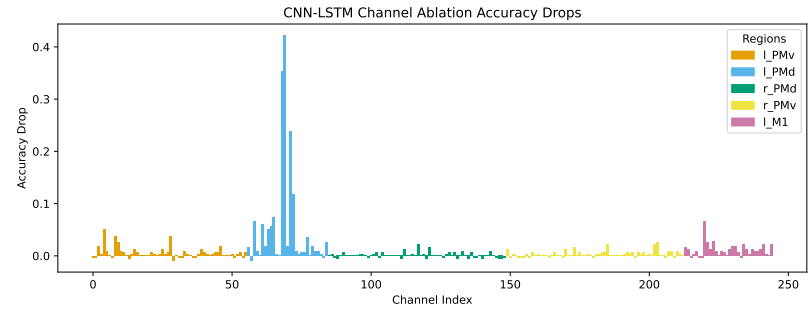
As PCA was performed on the data before training, the model type is irrelevant to the importance score. Starting with the channels in Figure 5.9, we see that the importance values differ from what we have seen in our other results. For example, while the left PMd is still shown as most important, PCA did not highlight a specific region within the PMd as clearly more important. Moreover, its top two important channels in that subregion, channels 69 and 70, are numerically close but not the same as the most important channels obtained via other methods (76, 77, and 79). PCA also places much more importance on the right PMv, which is in contradiction with our other observations. Similar conclusions are reached when looking at the frequency bins in Figure 5.10. The low ripple band is not an unimportant band, but its representation as the clear most important band is not in line with our other results. It is also interesting that PCA gives more importance to higher multiunit bins, more or less the opposite of our SHAP and Integrated Gradient values.

5.3 Class Analysis

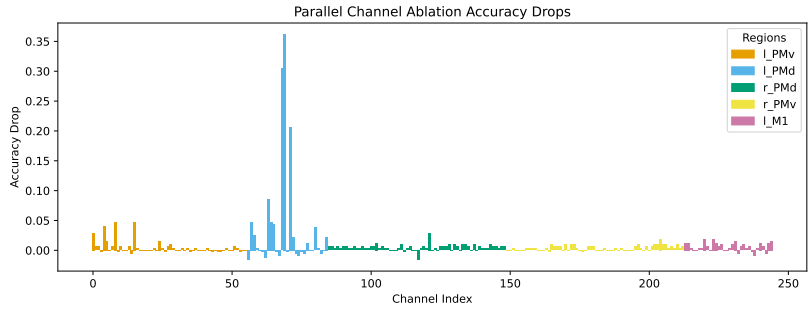
We conclude our results section by analyzing some key differences between our nine classes.



(A) CNN



(B) CNN-LSTM



(C) Parallel

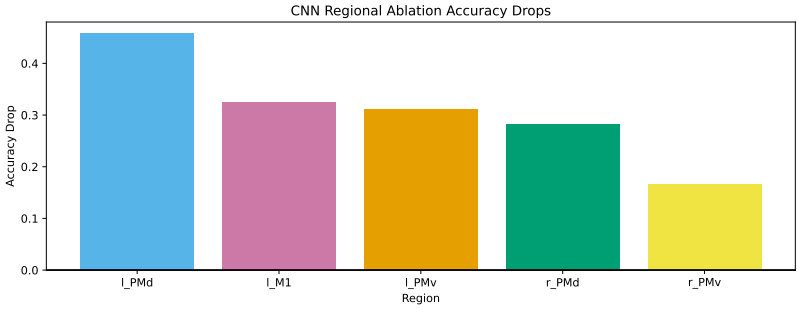
FIGURE 5.6: Top accuracy drops on individual channel ablations.

5.3.1 Confusion Matrices

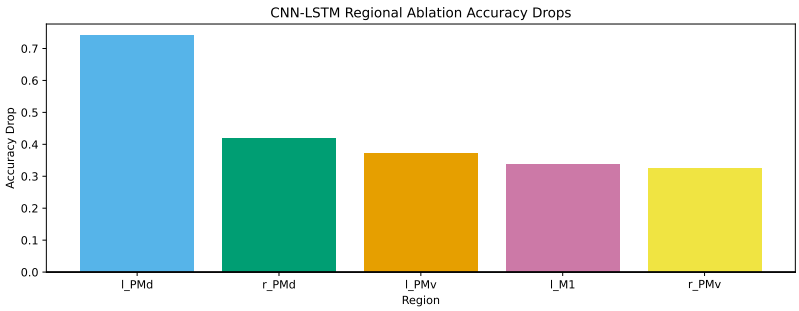
We plotted confusion matrices for all three of our models for a more detailed breakdown of their predictions. All three models struggled with predicting the baseline class in terms of both precision and recall, often incorrectly predicting the pre-reach or post-grasp stages or vice versa. This is most apparent when analyzing the class 1 (left-handed pre-reach) predictions from the CNN model, although similar but less pronounced patterns can be seen across all three models for classes 4, 5, and 8.

All three models have higher success with predictions involving the reach and grasp stages. In fact, the CNN-LSTM model achieves 100% precision on classes 6 and 7, while the CNN model does so on just class 6. This difference is less obvious for the Parallel model, which is more consistent in its predictions across all classes than the other two models, but it is still observable.

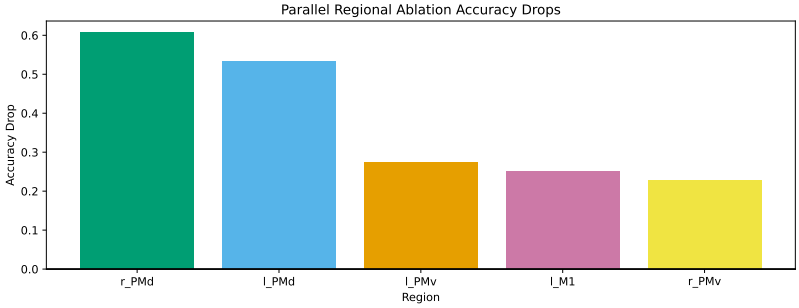
Also interesting is that our models perform better on the right-handed classes than the left-handed ones. This difference is best observed by comparing the CNN's predictions of classes 1 and 5. We offer three possible theories as to why this difference in performance exists. The first is the placement of the electrodes; motor



(A) CNN



(B) CNN-LSTM



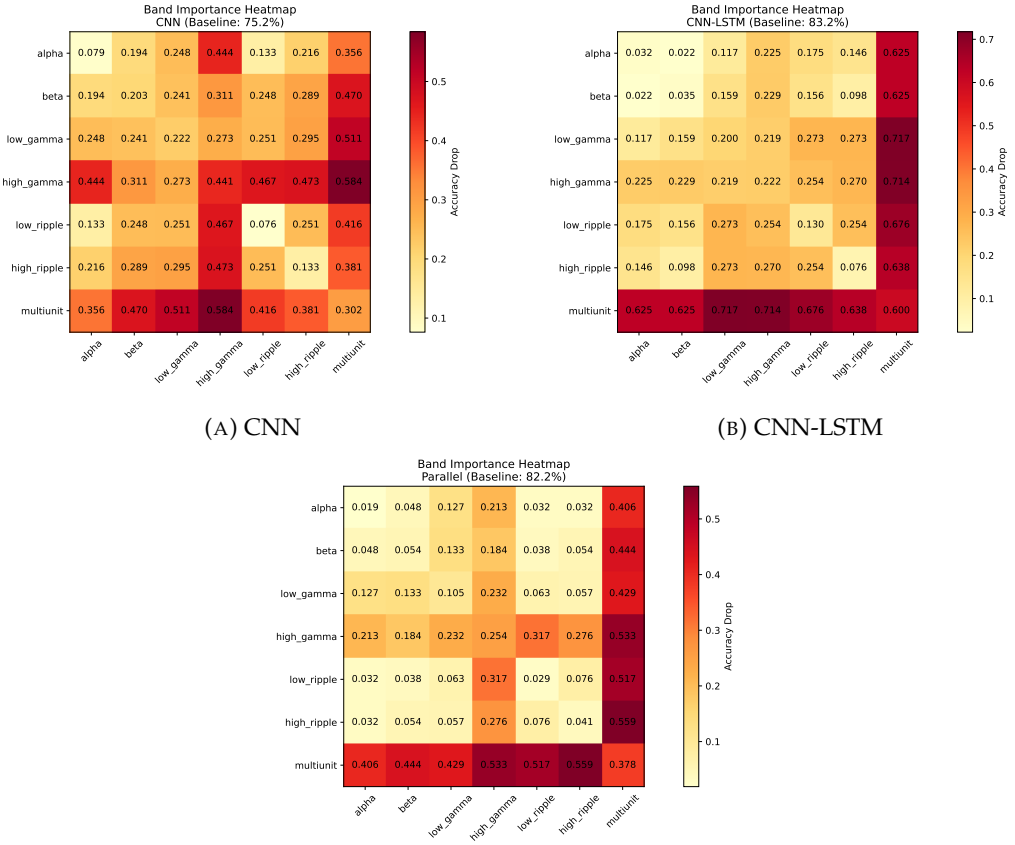
(C) Parallel

FIGURE 5.7: Accuracy drops when ablating entire regions of the brain.

control is dominated by the contralateral hemisphere, meaning one side of the brain controls movement for the opposite side of the body. We have more data from electrodes on the left side of the brain, so perhaps our data lacks useful information for left-sided movement. More abstract is the possibility that the macaques in the experiment were right dominant and therefore emitted stronger signals from that side, although we were unable to confirm what if any dominance the macaques in the experiment showed. Finally, our data has five more trials from the right-handed side than the left-handed one, so the explanation could be as simple as more data led to better results, although the difference in trials does not appear significant enough to be the sole cause.

5.3.2 PCA on Model Parameters

Finally, we performed three-dimensional PCA on our models' parameters to visualize the separation between different classes. In general, the relative positions of left-handed classes 1, 2, 3, and 4 and right-handed classes 5, 6, 7, and 8 follow similar



(A) CNN

(B) CNN-LSTM

(C) Parallel

FIGURE 5.8: Heat maps representing loss in accuracy after frequency ablation.

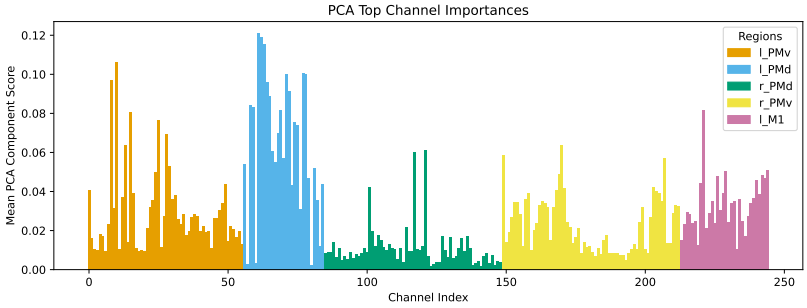


FIGURE 5.9: Channel importance derived from PCA scores.

patterns, but the same movement for both arms is not typically very close. In other words, we have similar intra-side patterns, even though there is a distinguishable difference between the inter-side patterns. The models disagree on the placement of classes relative to the other classes within the space. For example, the CNN model places class 3 closest to class 2, while the Parallel model places it very close to class 7, while the CNN-LSTM model places it about equidistant between classes 2, 4, and 7. Similarly, the CNN model has the baseline class 0 closest to class 1, whereas it is closest to class 5 for the CNN-LSTM model and interestingly class 2 for the Parallel model. These differences in placement make it difficult to conclude anything concrete about how individual classes relate to each other neurologically.

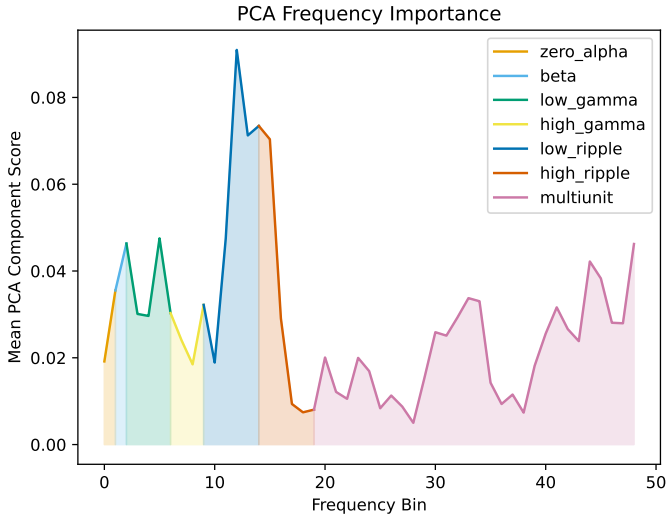
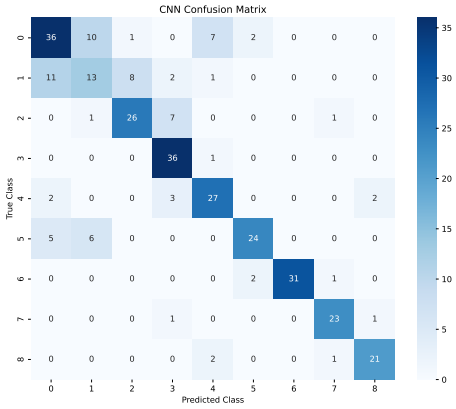
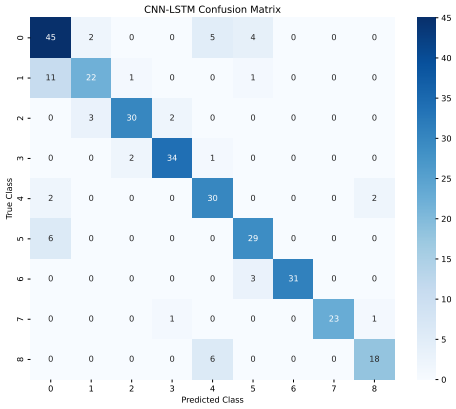


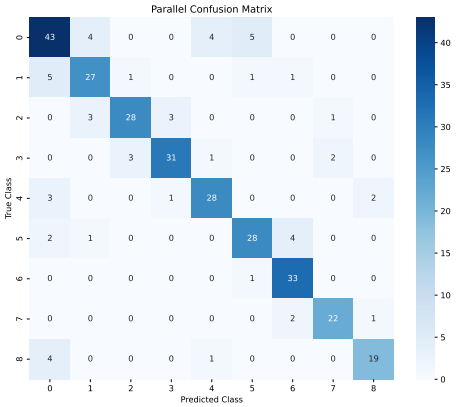
FIGURE 5.10: Frequency importance derived from PCA scores.



(A) CNN

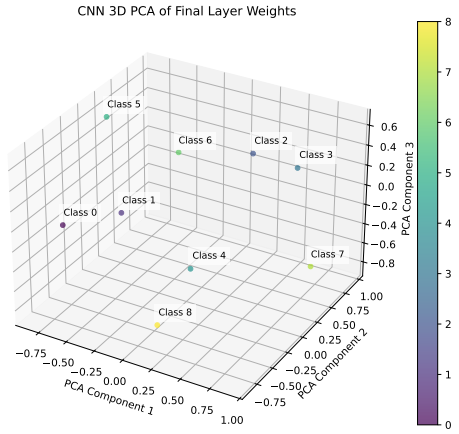


(B) CNN-LSTM

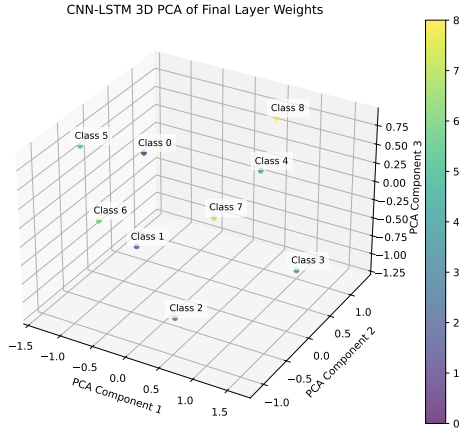


(C) Parallel

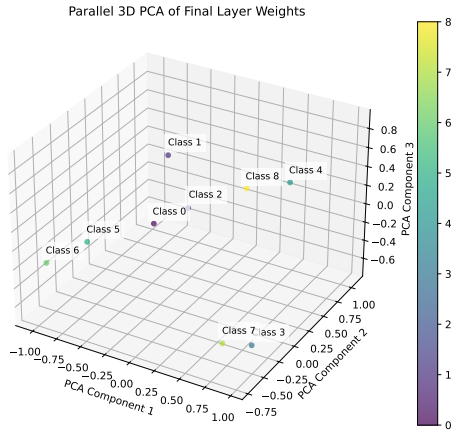
FIGURE 5.11: Confusion matrices of our three top Method A models. Class 0 is the baseline class, classes 1-4 are left-handed movements, and classes 5-8 are right-handed movements.



(A) CNN



(B) CNN-LSTM



(C) Parallel

FIGURE 5.12: A three-dimensional projection of the classes using the output layer weights from our models.

Chapter 6

Discussion

6.1 Comparison With Di Croce Results

The work described in this paper was done in parallel with the work described in Di Croce, 2025. We both used the same data with the same overarching goals, however rather than using deep learning models Di Croce used an Echo State Network. An Echo State Network (ESN) is a reservoir computing algorithm that leverages a fixed sized non-trainable network (reservoir) as a dynamic feature generator, from which a linear regressor is trained to classify and differentiate the different neural signatures.

6.1.1 Classification Results

Di Croce was able to achieve a top accuracy of 89%, surpassing our results by a notable margin. That result alone is impressive, however there are two nuances to that score that further elevate it. First, Di Croce used amplitude smoothing to achieve this accuracy, suggesting that the technique can be effective, even if seemed to confuse our models. Second, while we made the decision to treat all baseline samples as one class, Di Croce kept them separate due to the fact that his model was able to spatially separate and predict left and right-handed baselines. This result goes against our logical understanding of the data, making it an extremely curious result which at the moment does not have an explanation. We note that while not shown in Section 5, training our models on data with separate baseline classes produced the expected outcome, which was an inability to accurately distinguish the two.

One result shared between both methods was the better predictive accuracy for right-handed classes. This is further evidence that there are stronger right-hand signals within the data, although Di Croce's results do not add any theories beyond what was discussed in Section 5.3.1.

6.1.2 Feature Observations

Both methods also reached different conclusions when evaluating the importance of input features. As a forewarning, Di Croce evaluated importance by training ESNs on specific regions or bands, in other words, a pretraining strategy as opposed to our post-training strategies. To start, while our models demonstrated distinct differences in regional importance, Di Croce found minimal differences in accuracy changes when training with specific regions. Despite the differences in our evaluation methods, the consistency displayed by the ESN when trained on each region is both surprising and impressive. On the other hand, it can be argued that our methods provided more applicable and interpretable real-world results, assuming, of course, that they were accurate. Our evaluations on frequency bands were more similar; higher frequency bands are more useful for accurate predictions than lower

bands. There are some differences though, as while both methods establish the high importance of the multiunit band, Di Croce's results show the gamma and ripple bands as about equal whereas our results showed a preference (one for the high gamma band from SHAP and ablation, one for the ripple band from Integrated Gradients). Again, this could be attributed to a higher robustness of the ESN compared to our models.

6.2 Areas for Improvement and Future Ideas

We propose various ideas to expand or improve upon the work presented in this paper.

First, much can be done to try and fully optimize model performance. Although our grid searches tested a few hundred combinations per architecture type, there are still thousands of parameter combinations to try. This includes new options for parameters we did include in our search, such as CNN kernel size, and adding parameters we did not include, such as the size of our encoder or the number of CNN layers to stack. There is also the question of dimensionality reduction; reducing our channels dimension from 245 to 32 felt like a solid and safe choice for our data, but it was arbitrary and more and less aggressive approaches could affect the results. Ideally, any effort focused on optimization also uses a larger dataset than was available to us.

There is also plenty of room for more testing of our data loading and cleaning processes. Our first thought is to try different variations of spectrogram creation: changing bin sizes, adjusting the time dimension, etc., but different approaches to the data preprocessing, particularly the amplitude smoothing, could yield new and fascinating results.

"Ablation" performed before training, that is, only training models on specific parts of the brain and frequency bands, will also likely provide interesting new insights into feature importance, particularly on how the predictions of specific classes are impacted by including or not including a region or band.

Finally, and most challenging, is attempting to make our models more generalizable. Currently, the models are entirely dependent on the number of channels as well as the order of those channels. More broadly, the data itself is dependent on the exact placement of the electrodes. Any way to make our models more dynamic in their reading of data would allow for much more extensive generability testing and interpretation of the brain regions and frequencies most important in determining and understanding movement.

6.3 Conclusions

This paper had two goals: to prove that the decoding of complex, high-class LFP data is possible and to derive explanations for what is most useful for that decoding. To accomplish this, we used LFP data recorded in an experiment from DePass et al., 2022 in which macaques went through five different stages of movement. We further split that data into left and right-sided classes, as well as transformed the data into spectrograms and performed some initial preprocessing to clean it. For training, we performed grid searches to optimize the parameters of three different model architectures: a standard CNN, a CNN that feeds into an LSTM, and a parallel CNN and LSTM model, all of which also made use of a dimensionality reducing

decoder. After training, we evaluated feature importance using SHAP, Integrated Gradients, and ablation.

The results of our models, especially given our relatively low amount of data, show that LFP data can be used for accurate neural decoding. We also established that a CNN and LSTM working in tandem generate more success than using just a CNN model. And, while CNN-LSTM models have well-documented use on neurological data, the use of the Parallel strategy is rather novel, and can be considered for other neurological classification tasks in the future. In addition, we show how including a decoder within a model's architecture leads to improved results over using standard reduction methods such as PCA.

Our extracted feature importances varied both among our models and among the extraction methods we used. Still, there are some general conclusions we can reach. First, the left PMd appears to be the most important part of the brain for classifying movement. The fact that many of the most important individual channels are numerically close to each other means that there might even be a more specific part within the PMd that holds a major key for movement decoding, although we lack the details on electrode placement to confirm this. From there, the left PMv and left motor both offer some utility, and the right PMd does not have much individual channel importance but is important as a whole, perhaps in our specific case to assist with left-handed classification. Our results have less agreement on frequency band importance, but what is clear is the high importance of the multiunit and high gamma bands.

Finally, we offer an assessment of our methods. Although our encoder might have helped improve our accuracy by a couple of percentage points, we also observed that testing accuracies are more consistent across training trials when the encoder is not included. So, we are not confident in saying that the encoder must be included in models (at least in its current form), even if it is a better dimensionality reducer than standard methods. For architectures, we recommend both the CNN-LSTM and Parallel models, as they both performed similarly well. For feature extraction, despite our more limited successful use of it, we do think SHAP showed more potential than Integrated Gradients, as it was able to account for our powerline suppression and, although not used in our analysis, does make it easy to perform a class-by-class importance breakdown. Still, SHAP is more architecturally dependent, and we do have confidence in our Integrated Gradient values. We believe that further advancements in SHAP or other explainability methods will allow for an even deeper understanding of neurological data and how our brains drive our thoughts and actions.

Appendix A

Description of Group Work

The entirety of Chapters 1, 2, 4, and 5 are written by the author alone. Within Chapter 3, the Data Preprocessing subsection was written with the assistance of Luca Eric Di Croce, who also wrote the original code to perform the described steps. That subsection has been changed from what is found in his paper to fit better with the work represented in this paper. In Chapter 6, the Method Comparison section was the joint effort of both authors. Di Croce also supplied the wording of Appendix B.1.

Appendix B

Statements and Disclosures

B.1 Author Contributions and Rights

The data and some algorithms, as mentioned explicitly in the paper and attributed to in the code, are taken from DePass et al., 2022. All appropriate rights are reserved to their respective authors.

The algorithm `automatic_bad_channel_detection` was taken from Alonso, 2025, who improved on the existing code published in Komosar, Fiedler, and Haueisen, 2022.

B.2 Transparency on the Use of Generative AI

This thesis used ChatGPT for assistance in code development and debugging. In addition, ChatGPT reviewed some of the explanations provided in Sections 3 and 4 to verify language clarity. Finally, ChatGPT assisted with some of the LaTeX coding in this report, such as citation and figure formatting. All code, ideas, and statements provided by ChatGPT were reviewed and cross-checked to verify correctness, and all ideas, analysis, and interpretation provided within this report are directly from the author.

Bibliography

Al-Marridi, Abeer Z., Amr Mohamed, and Aiman Erbad (2018). "Convolutional Autoencoder Approach for EEG Compression and Reconstruction in m-Health Systems". In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 370–375. DOI: 10.1109/IWCMC.2018.8450511.

Alonso, Manuel A. Hernández (2025). "Cross-Task Multiclassification Approach to Neural Processing". MA thesis. Universitat de Barcelona.

Altaheri, Hamdi et al. (2023). "Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: a review". In: *Neural Computing and Applications* 35.20, pp. 14681–14722. ISSN: 1433-3058. DOI: 10.1007/s00521-021-06352-5. URL: https://doi.org/10.1007/s00521-021-06352-5.

Bae, Soo Hyun, In Kyu Choi, and Nam Soo Kim (2016). "Acoustic Scene Classification Using Parallel Combination of LSTM and CNN." In: *DCASE* 585, pp. 11–15.

Buzsáki, Gyorgy and Andreas Draguhn (July 2004). "Neuronal Oscillations in Cortical Networks". In: *Science (New York, N.Y.)* 304, pp. 1926–9. DOI: 10.1126/science.1099745.

Choi, Sang Won and Brian H. S. Kim (2021). "Applying PCA to Deep Learning Forecasting Models for Predicting PM2.5". In: *Sustainability* 13.7. ISSN: 2071-1050. DOI: 10.3390/su13073726. URL: https://www.mdpi.com/2071-1050/13/7/3726.

Chouinard, Philippe and Tomas Paus (Oct. 2010). "What have We Learned from "Perturbing" the Human Cortical Motor System with Transcranial Magnetic Stimulation?" In: *Frontiers in Human Neuroscience* 4, p. 173. DOI: 10.3389/fnhum.2010.00173.

Cohen, Michael X. (2017). "Where Does EEG Come From and What Does It Mean?" In: *Trends in Neurosciences* 40.4, pp. 208–218. ISSN: 0166-2236. DOI: 10.1016/j.tins.2017.02.004. URL: https://doi.org/10.1016/j.tins.2017.02.004.

Craik, Alexander, Yongtian He, and Jose L Contreras-Vidal (2019). "Deep learning for electroencephalogram (EEG) classification tasks: a review". In: *Journal of Neural Engineering* 16.3, p. 031001. DOI: 10.1088/1741-2552/ab0ab5. URL: https://dx.doi.org/10.1088/1741-2552/ab0ab5.

DePass, Michael et al. (2022). "A machine learning approach to characterize sequential movement-related states in premotor and motor cortices". In: *Journal of Neurophysiology* 127.5. PMID: 35171745, pp. 1348–1362. DOI: 10.1152/jn.00368.2021. eprint: https://doi.org/10.1152/jn.00368.2021. URL: https://doi.org/10.1152/jn.00368.2021.

Destexhe, Alain and Joshua A. Goldberg (2022). "LFP Analysis: Overview". In: *Encyclopedia of Computational Neuroscience*. Ed. by Dieter Jaeger and Ranu Jung. New York, NY: Springer New York, pp. 66–70. ISBN: 978-1-0716-1006-0. DOI: 10.1007/978-1-0716-1006-0_782. URL: https://doi.org/10.1007/978-1-0716-1006-0_782.

Di Croce, Luca Eric (2025). "What's in a grasp? Describing motion-based brain states using reservoir computing". MA thesis. Universitat de Barcelona.

Dos Santos, Eulanda M., Robert Sabourin, and Patrick Maupin (2009). "Overfitting cautious selection of classifier ensembles with genetic algorithms". In: *Information Fusion* 10.2, pp. 150–162. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2008.11.003. URL: https://www.sciencedirect.com/science/article/pii/S1566253508000730.

Evarts, Edward V. (1979). "Brain Mechanisms of Movement". In: *Scientific American* 241.3, pp. 164–179. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24965294 (visited on 06/12/2025).

Fawaz, Hassan Ismail et al. (2019). "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4, pp. 917–963. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1. URL: https://doi.org/10.1007/s10618-019-00619-1.

Federal Agencies Digital Guidelines Initiative (2025). *Term: Sampling Rate (Audio)*. Accessed: 2025-06-13. URL: https://www.digitizationguidelines.gov/term.php?term=samplingrateaudio (visited on 06/12/2025).

John Hopkins Medicine (2025). *Brain Anatomy and How the Brain Works*. Accessed: 2025-06-12. URL: hhttps://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain (visited on 06/12/2025).

Klimesch, Wolfgang (2018). "The frequency architecture of brain and brain body oscillations: an analysis". In: *European Journal of Neuroscience* 48.7, pp. 2431–2453. DOI: https://doi.org/10.1111/ejn.14192. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/ejn.14192. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/ejn.14192.

Komosar, Milana, Patrique Fiedler, and Jens Haueisen (Sept. 2022). "Bad channel detection in EEG recordings". In: *Current Directions in Biomedical Engineering* 8, pp. 257–260. DOI: 10.1515/cdbme-2022-1066.

Lu, Hung-Yun et al. (2021). "Multi-scale neural decoding and analysis". In: *Journal of Neural Engineering* 18.4, p. 045013. DOI: 10.1088/1741-2552/ac160f. URL: https://dx.doi.org/10.1088/1741-2552/ac160f.

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Mirchi, Nykan et al. (2022). "Decoding Intracranial EEG With Machine Learning: A Systematic Review". In: *Frontiers in Human Neuroscience* Volume 16 - 2022. ISSN: 1662-5161. DOI: 10.3389/fnhum.2022.913777. URL: https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2022.913777.

Nain, Aakash (2020). *Model interpretability with Integrated Gradients*. https://github.com/keras-team/keras-io/tree/master. GitHub repository. URL: https://github.com/keras-team/keras-io/blob/master/examples/vision/integrated_gradients.py.

Psych, SciShow (2017). *What Do Different Brainwaves Mean?* YouTube video produced by educational platform. URL: https://www.youtube.com/watch?v=gvpuOBezW0w (visited on 06/12/2025).

Rothwell, J.C. (2012). "Overview of neurophysiology of movement control". In: *Clinical Neurology and Neurosurgery* 114.5. Restorative Neurology and Motor Control, pp. 432–435. ISSN: 0303-8467. DOI: https://doi.org/10.1016/j.clineuro.2011.12.053.

Schwartz, Andrew B (Mar. 2016). "Movement: How the Brain Communicates with the World". In: *Cell* 164, pp. 1122–1135. DOI: 10.1016/j.cell.2016.02.038.

SHAP (2025). *shap*. https://github.com/shap/shap. GitHub repository. URL: https://github.com/shap/shap.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje (2019). *Learning Important Features Through Propagating Activation Differences*. arXiv: 1704.02685 [cs.CV]. URL: https://arxiv.org/abs/1704.02685.

Simonyan, Karen and Andrew Zisserman (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (2017). *Axiomatic Attribution for Deep Networks*. arXiv: 1703.01365 [cs.LG]. URL: https://arxiv.org/abs/1703.01365.

Warrick, Philip A and Masun Nabhan Homsi (2018). "Ensembling convolutional and long short-term memory networks for electrocardiogram arrhythmia detection". In: *Physiological Measurement* 39.11, p. 114002. DOI: 10.1088/1361-6579/aad386. URL: https://dx.doi.org/10.1088/1361-6579/aad386.

Xu, Mingdong et al. (2020). "Spectrum Sensing Based on Parallel CNN-LSTM Network". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5. DOI: 10.1109/VTC2020-Spring48590.2020.9129229.