Seizure Forecasting for Epilepsy Management

Final Report

Team 11

Radi Elazazy

Marcus Emanuel

Morgan Ossolinski

Table of Contents

Table of Figures:

Table of Tables

Acknowledgements

The authors express their gratitude to Sabato Santaniello, PhD., for providing us guidance and support throughout the design of our model, as well as the University of Connecticut BME department and faculty for providing the tools necessary to complete this project. Additionally, we extend our appreciation to Physionet for providing us with the open-source Siena Scalp-EEG Database and Mathworks for the open-source software used for data preprocessing and visualization, including EEGlab and ZaplinePlus.

Abstract

Epilepsy is a neurological disorder characterized by recurrent seizures, or abnormal, sudden, and unpredictable electrical signals within the brain. Currently, EEG is used in combination with patient specific clinical data for seizure detection. However, forecasting seizures before they occur remains as a major obstacle for many patients. This project leverages non-invasive scalp-EEG data to develop a model which can alert patients and assist medical professionals in predicting epileptic seizures before they happen. Classification and forecasting modules were implemented in MATLAB, including time-frequency maps, statistical modeling, signal processing, and machine learning. The proposed design achieves a classification accuracy of approximately 70% and a forecasting precision of 74.3% thirty minutes before seizure onset. The application of this device would enhance patient quality of life without the need for invasive procedures, medication, medical assistance, or clinical information.

Section 1

1.1 Problem Statement

Epilepsy, one of the most prevalent neurological disorders worldwide, affects an estimated 70 million individuals [3]. It is associated with a range of significant consequences which include immediate risks such as falls, injuries, and, in extreme cases, death, as well as long-term challenges like psychological disturbances and cognitive impairments [3]. Recurrent seizures also add complexity to patients' lives as they strive to pursue academic, social, and occupational goals [3]. While pharmaceutical interventions like anticonvulsants and antiepileptic drugs are commonly used to manage seizures, they often entail undesirable side effects. Additionally, Drug-Resistant Epilepsy (DRE) affects 20-30% of patients, hindering consistent relief [42].

The ability to predict imminent seizures within a short time frame is crucial for the epilepsy community. A predictive analysis of EEG data could assist doctors in making decisions efficiently and with clarity. These forecasts are vital for enabling timely interventions, including alerts to caregivers and the administration of short-acting medications or stimulation. By leveraging non-invasive EEG data, a forecasting model could provide patients with a robust warning system, independent of surgical interventions or patient data, from the comfort of their homes. The EEG data used in our study is sourced from the Siena Scalp-EEG Database [43].

1.2 Overview of the Clinical Problem

In the human brain, neurons communicate through synchronized electrical activity. These brain waves vary in frequency depending on the current state of the brain and range from 0.5 - 35 Hz. Lower frequency ranges are seen during sleep/deeply relaxed states (lower range) [1]. In epileptic brain states, seizures manifest through abnormal, synchronous and excessive neuronal activity in the brain and are not confined exclusively to the aforementioned brain wave frequency range [2]. Concisely, seizures can be classified into two categories; generalized and focal. Focal seizures start in a localized region of the brain and can spread outward, whereas, generalized seizures can start as a focal seizure that spreads to both lateral hemispheres of the brain (can also occur as a “generalized onset” where the seizure impacts both sides of the brain at its origin) [4]. The unpredictability and complexity of the brain makes this problem particularly daunting for the patients who suffer as well as the doctors and scientists committed to their wellbeing. Epilepsy prevalence is also worth noting, with 70 million people worldwide, including nearly 3 million in the US, being directly impacted by it [3][7]. To further add complexity to the issue at hand, approximately 20-30% of epilepsy cases are drug resistant, meaning seizures persist despite antiepileptic medication. Oftentimes patients experience unwanted symptoms as a result of their medication (sedation, drowsiness, incoordination, and nausea) [5][6].

1.3 Impact of the problem on Patients

To reiterate, epileptic seizures are of two main types (generalized and focal). Consequently, symptom type and severity can vary widely as a result of the impacted region, intensity, etc.

Since the whole brain is involved and fires abnormally, generalized seizures have particularly devastating impacts on patients (grand and petit mal seizures). Petit mal seizures are characterized by a loss of consciousness and rapid blinking or staring, while grand mal seizures are characterized by falling to the ground, loss of consciousness, muscle jerks/spasms, and crying. Both types can lead to exhaustion, however, grand mal seizures are known to cause brain damage, amnesia, confusion, drowsiness, Todd paralysis, and/or death [12].

Alternatively, focal seizures are located within just one portion of the brain and are comprised into three different groups; simple (no loss of consciousness with motor/sensory/psychic irregularities), complex (unintentional repetitive movement preceding a simple focal seizure), and secondary generalized (a combination of simple and complex followed by a generalized seizure) [10][11].

In large scale studies it was found that 40-50% of untreated individuals can expect recurrence within 2 years following their initial seizure (with treatment reducing this risk by as much as half) [13][14]. It is worth noting that as mitigating the risk of recurrence becomes the focus of treatment, long term effects of epilepsy can go unchecked [7]. Long term effects, which include; psychiatric disorders (depression, anxiety, psychosis and personality change) and cognitive impairment (memory, attention, and executive function deficits), can be just as, if not more, debilitating than their short term counterparts[7][8].

The associated psychological/cognitive impedance, physical/internal stresses are two main contributing factors in a phenomenon known as Sudden Unexpected Death in Epilepsy (SUDEP). It is thought that over time, breathing difficulties (apnea), and heart arrhythmias cause SUDEP to occur 1.16 times for every 1000 people with epilepsy [15].

From the most extreme circumstances to the most seemingly minor, it is no surprise that any of these symptoms in combination can become a hindrance, or eventual end, to everyday life. Being able to successfully treat and accurately predict seizures would undoubtedly benefit each patient's quality of life.

1.4 Clinical Solutions Currently Available

Medication therapy is the most widely implemented means of seizure prevention. Antiepileptic drugs (AEDs) can be categorized as broad-spectrum AEDs and narrow-spectrum AEDs. Broad-spectrum AEDs, as their name suggests, can treat a wide range of various seizure types. Levetiracetam, lamotrigine, zonisamide, valproic acid, clobazam, clonazepam, topiramate, perampanel, and rufinamide are examples of broad-spectrum AEDs currently available. Alternatively, narrow-spectrum AEDs are used for the prevention of focal seizures, and include medications such as; lacosamide, gabapentin, carbamazepine, ezogabine, phenytoin, pregabalin, oxcarbazepine, and vigabatrin. Medications with the most promising therapeutic benefit are valproic acid for generalized seizures, and quick acting benzodiazepines/epinephrine used as preventative measures in the pre-seizure (pre-ictal) state.

However, 7-37% of therapeutic medication causes adverse side effects such as headaches, fatigue, dizziness, blurry vision, nausea, weight gain, and/or various psychological issues (mood disorders, cognitive issues). Chronic and continual use of medication can lead to osteoporosis, which can become dangerous during uncontrollable convulsions or fast/jerking motions. More severe side effects include Stevens-Johnson syndrome, agranulocytosis, aplastic anemia, hepatic failure, pancytopenia, multiorgan hypersensitivity, psychosis, and lupus syndrome [16].

Introduced in 1988, vagus nerve stimulation (VNS) serves to be a solution to drug-resistant epilepsy by providing a consistent shock to the vagus nerve (about 300 min/day) and preventing generalized seizures from occurring. VNS involves a small surgical incision and the implantation of electrodes onto the vagus nerve. 50% seizure reduction is observed after the first three months after implantation with an increase to about 80% after 8 years [17]. However, just like previously discussed therapies, side effects including exhaustion, throat irritation, change in voice do occur.

Video electroencephalography (EEG) and epilepsy monitoring units (EMUs) serve as primary methods for data collection in seizure detection, diagnosis, and evaluation of epilepsy. Here, EEG signals are captured using electrodes, and while specific electrode placements can vary, this study utilized a 10-20 electrode configuration (29 total electrodes)(refer to Figure 1) for consistency and comparability. EEGs can provide data of the main detectible features of epileptic seizures, such as spikes and slow waveforms, which can be processed through a machine learning (ML) algorithm and used for detection. Neuromodulation devices, often considered a part of responsive neurostimulation (RNS), use ML algorithms from either scalp / transcranial EEG data to predictively stop seizures before they occur. This is possible from the

detection of the seizure from the triggering event (pre-ictal) and subsequent electrical excitation propagated from the microcontroller within the device. After 7 years of follow-up responders reached 70-74% efficacy from RNS [18].

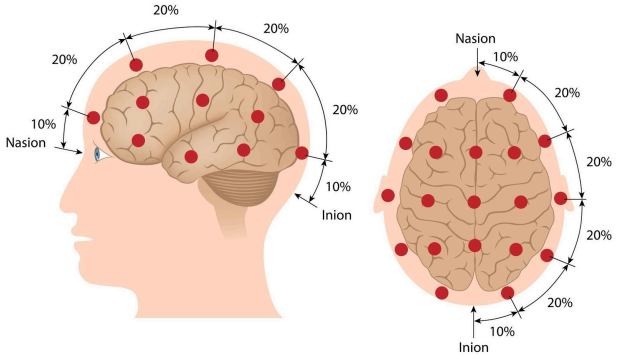

Figure 1: Electrode placement for EEG signal acquisition in 10-20 configuration.

There are also wrist watches which detect electrical impulses within the musculoskeletal system and/or heart rate variability to predictively warn a patient that a seizure is about to occur. The watch also uses a ML algorithm to selectively choose which impulses are indicative of a seizure. However, these devices can only detect tonic / clonic seizures and have no impact regarding focal seizures. This type of device has about 85.7% sensitivity [19].

Finally, there are less scientifically understood means of detection, such as trained dogs or a keen awareness of symptoms, such as olfactory disturbances, anxiety, hallucinations, etc.

1.5 Current State of Research

In the past, researchers have investigated EEG signals(scalp and intracranial) as a promising way to identify seizures as, or even before, they occur. Since 2016, given the vast amount of data that is associated with continuous acquisition of EEG data (typically 512 bits per second per node), deep learning algorithms, with their vast data analysis capabilities, have gained popularity in the search to predict seizure onset [23]. As of 2021, the majority of machine learning tools for seizure prediction were implemented on TensorFlow, Keras, and MatLab, and more than half of the proposed solutions were carried out using a 1D or 2D Convolutional Neural Network (CNN)[23]. These algorithm based solutions can loosely be simplified into three sequential pieces. First, how is the EEG data manipulated to suit the given algorithm? Second, what type of algorithm is used? And finally, given the algorithm's ability to classify the data, how is output data turned into a prediction/percent likelihood?

As an input, EEG data can be represented in a range of different ways, this is reflected in recent literature. Image-based (Continuous Wavelet Transform (CWT) Scalograms, Short Term Fourier Transform (STFT) Spectrograms), Power Spectral Density (PSD), variance, entropy, etc., are often used to convert EEG data into a more palatable form for algorithms [25][26][27][28]. Image-based classification is advantageous for its ease of implementation and tunability, particularly with pretrained networks. Pretrained networks, which are premade vary

wildly but have been proven successful in picking out distinct features within collected time-frequency maps. Furthermore, high contrast filters and distortions can be utilized to make different aspects of the image stand out, helping the algorithm classify. PSD is an example of a “raw” version of the same input data. It can serve to provide frequency resolution, temporal resolution using a matrix of numerical values. This lays a framework on which statistical features can easily be extracted by a deep learning algorithm.

Regarding the deep learning algorithm itself, there is a vast number of networks that are currently being explored. Pretrained image classifiers such as VGG16, ResNet50, GoogleNet, and InceptionV3 have emerged as a few promising CNN architectures. Due to these being premade and applicable to diverse projects, there seems to be a disproportionate amount of research in image classification [20][26][27]. However, for numerical-based inputs such as PSD, entropy, and variance, promising research has been conducted in deep learning techniques such as Long Short Term Memory (LSTM) Networks, Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) [22][24][30][31]. The utilization of these networks and deep learning techniques is versatile, allowing for various combinations. This field is inherently open-ended, with complexity seen in each study. Together, these research efforts strive to transform the management and comprehension of epileptic seizures, ultimately enhancing the quality of life for individuals impacted by epilepsy.

1.6 Intellectual Property

A range of patented methods and systems have been developed to address the complex challenge of seizure detection, with a primary focus on leveraging electroencephalography (EEG) signals alongside other physiological parameters. One introduces a sophisticated method that acquires EEG data from multiple channels, employing preprocessing techniques, feature extraction algorithms, and machine learning models for precise binary (seizure vs non-seizure) classification. Notably, a control policy is integrated to assess seizure burden, facilitating timely notifications when necessary [32]. Complementing this, another patent introduces the concept of ensemble classification models, which combine outputs from multiple models to enhance the accuracy of epilepsy seizure detection and prediction. Through weight adjustments, the system achieves optimized performance while minimizing power consumption, making the detection process more efficient [33]. Similar intellectual property presents a method that utilizes autoregression algorithms applied to sliding time windows of patient body signals to detect seizures. This approach relies on detecting variations in parameter vectors gotten from autoregression coefficients to determine seizure onset and termination [34]. Lastly, another proposes a medical system equipped with an adaptive seizure detection algorithm that dynamically adjusts based on patient parameters. By monitoring secondary parameters like patient activity, this system refines its detection ability, particularly in identifying specific target seizures [35]. Altogether, these patented innovations represent promising strides toward improving the accuracy, efficiency, and overall efficacy of seizure detection and management systems.

1.7 Project Stakeholders

Patients and Their Families

The main participants in this diagnostic process are epilepsy patients having EEG scans. To manage their disease, they look for precise diagnoses and efficient treatment alternatives. Additionally significant stakeholders are their relatives and caregivers, who aid patients every step of the way by arranging appointments, offering emotional support, and assisting with treatment adherence.

Medical Specialists (Neurologists and Epileptologists)

Neurologists and epileptologists play a crucial role in the EEG scan for the epilepsy process. To diagnose epilepsy, ascertain its kind and severity, and create individualized treatment strategies, they evaluate EEG readings. Their knowledge helps patients choose the best course of action, which may involve medication, a change in lifestyle, or even surgery in some situations.

Healthcare professionals and technologists

Stakeholders who carry out the scanning are crucial. By correctly positioning the electrodes, keeping an eye on the patient while they are being recorded, and maintaining the EEG equipment, they guarantee the accuracy and quality of the EEG recordings. Their knowledge guarantees that neurologists get trustworthy information for precise diagnosis and therapy planning.

Hospitals, clinics, and other healthcare facilities that offer EEG

Hospitals, clinics, and other healthcare facilities that offer EEG services are crucial stakeholders. To ensure that patients have access to this crucial diagnostic tool, they make investments in cutting-edge machinery, maintain a secure and welcoming atmosphere, and simplify scheduling. Their assistance makes sure that EEG scans are carried out effectively and swiftly.

Section 2

2.1 Objectives

Table 1: Objectives list.

#	Objective	Metric/Topic	Associated References
1	Utilize EEG Data	● Single channel ● Multi-channel avg. ● Time-Frequency Domain ○ STFT ○ CWT ● Time-Series Domain ○ PSD ○ Variance	[20][22][24][25][26] [27][30][31][36][37]
2	Classify Seizure/Non-Seizure Data	● VGG16 ● InceptionV3 ● Other networks	[20][26][36]
3	Predict Seizure Within Thirty Minutes	● Moving Mean ● Probability Ratios	N/A

		• Kurtosis	
4	Robust	• Relevant for wide range of patients within dataset	N/A

2.1.1 Explanation of Objectives

Utilizing EEG Data

EEG data, in its initial form, tends to be convoluted and indecipherable for an algorithm. The sensitive nature of EEG data means that any change in electrical potential at the point of each electrode can drown out what is actually occurring within the brain. Common disturbances include heartbeat, muscle spasms, blinking, ambient noise, faulty electrode contact with scalp, etc. Even shifts in physiological brain states (sleeping, relaxed, anxious, etc.) can elicit changes in the overall frequencies seen in EEG data. To achieve “clean” EEG data it is necessary to scrape any of these interfering signals (artifacts), this is done through preprocessing. Furthermore, data coming in from different electrodes can be stacked and averaged in order to cut down on the computing power required to classify and increase the efficiency of the overall system.

In preprocessing, there is no unanimously decided-on technique to clean out unwanted frequencies. In fact, there are a combination of viable techniques for this purpose, and, since they tend to be complicated and repetitive, literature tends to gloss over this aspect of seizure classification. However, it is commonly seen that signals are filtered using high/low pass filters to cut out artifacts at polar ends of the frequency spectrum, trimmed by simply cutting out pieces of data that are unwanted with respect to seizure onset, preprocessing filters implemented (spatial preprocessing, ICA, etc.), and frequency adjustments made for EEG data [37]. All the advantages for this lie within the artifact removal, for our purposes we implement a combination of all of these techniques in order to clean our data.

The organization of data is a critical decision preceding the application of classification models. Data can be structured and condensed to highlight or drown out specific information. An important characteristic of EEG electrodes is that they offer researchers valuable spatio-temporal data. Organizing data into spatial and temporal groupings can significantly enhance the overall performance and efficiency of the final product, depending on how it is done. For instance, in our analysis of the PhysioNet database (Table 1), we were provided segmented EEG signals from 11 patients into temporal intervals representing inter-ictal (non-seizure), pre-ictal (minutes preceding seizure), ictal (seizure), and post-ictal (minutes after seizure) periods, while noting the seizure onset location. One key design consideration involves deciding whether to alter temporal assignments and whether to average adjacent electrodes, electrodes within the same brain region, or every electrode together. This decision, termed as single-channel versus multi-channel averaging, holds significant implications. While multi-channel averaging reduces computational burden and enhances efficiency, potentially enabling earlier seizure prediction, it may obscure subtle features unique to specific brain regions. Conversely, by averaging all signals together, smaller characteristics within focal regions, where seizures manifest, might be overlooked. Thus, this decision plays a pivotal role in shaping our final design, influencing the algorithm's ability to accurately predict seizures.

Table 2: Simplified physionet dataset.

Patient Number (Gender - Age)	Seizure Type	Localization	Lateralization	# of Seizures
PN00 (M-55)	IAS	T	R	5
PN01 (M-46)	IAS	T	L	2
PN03 (M-54)	IAS	T	R	2
PN05 (F-51)	IAS	T	L	3
PN06 (M-36)	IAS	T	L	5
PN07 (F-20)	IAS	T	L	1
PN09 (F-27)	IAS	T	L	3
PN10 (M-25)	FBTC	F	Bilateral	10
PN11 (F-58)	IAS	T	R	1
PN12 (M-71)	IAS	T	L	4
PN13 (F-34)	IAS	T	L	3
PN14 (M-49)	WIAS	T	L	4
PN16 (F-41)	IAS	T	L	2
PN17 (M-42)	IAS	T	R	2

Classification Method

As EEG data undergoes cleaning, its underlying structure remains unchanged. Initially presented in graphical form, the data depicts the frequency activity of each electrode over time. However, raw graphs may not be optimal for algorithm-based analysis. To remedy this, EEG data can be transformed in two main ways: image-based and numerical-based conversions, both encompassing various representation methods.

Image-based conversion entails generating time-frequency maps, typically utilizing techniques like the Continuous Wavelet Transform (CWT) (Figure 2) or Short-Term Fourier Transform (STFT). These methods transform the original data into visual "maps" representing the frequency content over time for each electrode. Such maps consolidate vast amounts of information into a more digestible format. Moreover, this approach offers tunability, allowing adjustments in time shifts and window sizes. For instance, refining the time window during seizure periods can enhance the focus on seizure-related data, which constitutes a small fraction of the overall EEG data. It's noteworthy that CWT excels in pinpointing specific temporal

and frequency features, whereas STFT boasts advantages in computational efficiency, ease of implementation, and uniform frequency resolution. Thus, the choice between CWT and STFT is reliant on the specific requirements of the classification, and is dependent upon individual needs for precision versus practicality.

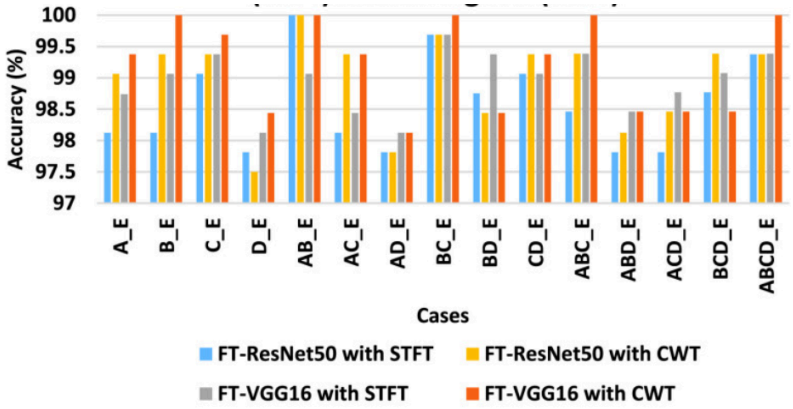

Figure 2: Comparison of CWT and STFT converted EEG data using pretrained networks [20].

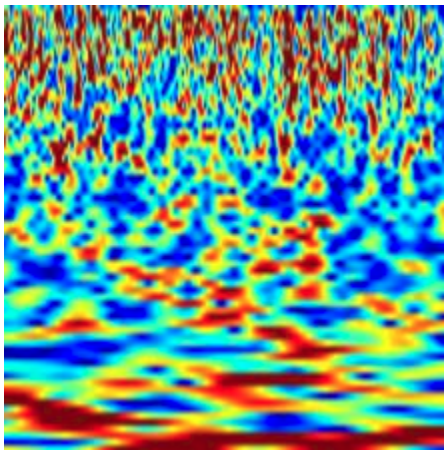

Figure 3: Patient 17, seizure 2, electrode T3 (area of focal seizure localization). Seizure scalogram.

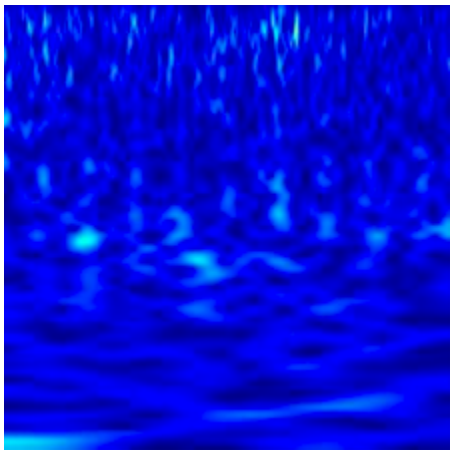

Figure 4: Patient 17, seizure 2, electrode T3 (area of focal seizure localization). Non-seizure scalogram.

Conversely, numerical-based conversions take the initial raw graph and represent large amounts of frequency and temporal bits in matrices. These matrices can then be broken down further by means of statistical, variance, PSD analysis, etc. These methods tend to be more computationally intensive as each bit of information can be recorded for data acquired at 512 Hz (512 bits per second per channel). However, they can give a more comprehensive view of the data and allow implementation of feature extraction which can lead to meaningful assumptions deduced from a given network. It is important to note that both image-based and numerical-based conversions are justifiable approaches to predicting seizures, the advantages that each contribute and whether or not to implement each can be variable depending on the design parameters and nature of the EEG data.

Forecasting Method

Our classifier outputs a 1x29 matrix which tells our forecaster for each electrode what is our VGG16 network's confidence that the given electrode is currently in a seizure state. We used a 70% confidence threshold for our forecaster. To conceptualize how our electrodes exhibited activity throughout non-seizure, seizure, and pre-ictal states we used a moving mean which showed the average number of electrodes over 70% confidence in a given time window. This mean was then shifted to show a gradient during the pre-ictal state from relatively low activation throughout all electrodes to a relatively high activation state during seizure. However, the aforementioned states were not always true, at times there were extremely active non-seizure states.

So, we utilized variance using kurtosis, a statistical feature which assigns a numerical value to the "fat-tailedness" of a gaussian curve. Throughout our exploration of a moving mean we found that non-seizure and seizure states were relatively uniform from seizure to seizure and patient to patient. Although there was still much variation, the variation seen in the pre-ictal state was much more in comparison, and so we could use it to distinguish pre-ictal from everything else.

To implement kurtosis we found which windows worked best and gave us optimal predictions. We found two windows worked best 120 and 30 frames of scalograms. These windows were then used in conjunction to create a multi flag system where both kurtosis windows had to be satisfied in order to trigger a hard warning. In this, we were able to cut down significantly on false positives while still notifying the patient reliably.

2.1.2 Objective Pairwise Comparison

Table 3: Objective pairwise comparison.

Obj #	Pairwise Matrix			CWT	STFT	PSD					Decision
1	Single Channel	-	1								Single Channel
	Multi Ch. Avg.	0	-								
2	CWT			-	1	1	1				CWT
	STFT			0	-	0	0				
	PSD			0	0	-	1				
	Variance			0	0	0	-				
3	VGG16							-	1		VGG16
	Inception V3							0	-		
4	Forecasting									-	

2.2 Constraints

The deep learning aspect of this project offers considerable flexibility, allowing us as creators to interpret data according to best judgment. Projects utilizing deep learning algorithms often unfold in an open-ended manner, with constraints typically determined by the given dataset and capabilities of the chosen network and software. Despite much room for creative freedom, certain constraints must be carefully considered.

A significant constraint arises from the provided PhysioNet dataset. Factors such as dataset size, duration, frequency, the number of channels, and electrode orientation present limitations as design options are contemplated. Understanding how the data was acquired is crucial, as it directly impacts workload, feature extraction, general pattern analysis, and data conversion. Dataset size and duration totaling 11 patients and 94 hours, while giving enough information to work with, limits our ability to broadly generalize the seizure data. This is

imperative when constructing a “one size fits all” model. Data frequency of 512 Hz gives us higher temporal and frequency resolution, meaning that high-frequency components can be scraped for cleaning or used for seizure classification/prediction. Additionally, the electrode number and orientation confines us to specific brain regions, possibly constraining the scope of analysis.

Computation workload poses another critical constraint, particularly in the context of the computing power available through MatLab. Convolutional Neural Networks (CNNs), while powerful, demand substantial computational resources, including processing power and memory. Balancing this computational demand with the need for low processing speed is essential, enabling real-time alerts for patients and ensuring the practical utility of the final product.

Early aspects of the project, such as data acquisition, directly influence the efficiency and usefulness of the final product for patients. The complexity of CNN training further shows the importance of thoughtful consideration of network architecture, dataset size, and neural network depth. Each of these factors plays a pivotal role in determining the success of the project in achieving its intended objectives.

2.2.1 Constraints List

Table 4: Seizure forecaster constraints list.

Constraint	Parameters
Dataset Size	11 patients
Dataset Duration	94 total hours 1-10 seizures per patient recording
Data Frequency	512 Hz
Number of EEG Channels	29 Channels
Electrode Orientation	10-20 orientation
Computational Load	*Hard to quantify* *Must be implementable in MATLAB*
Real Time Processing	Processing delay of 1-2 seconds
Feature Extraction	Limited to complexity and compatibility

2.3 Project Functions and Specifications

The first function of our software is to preprocess the data. Our preprocessing function should result in cleaned EEG data ready for input into the neural network. Secondly, the data must be converted from raw graphical data to a form that is readable by an algorithm. This could be inputting cleaned EEG signals numerically or utilizing time-frequency imaging. Next, our

network must extract meaningful features from the data to deduce accurately the percent likelihood of seizure onset. It is important that the decisions made in this function are compatible with the decisions made in the previous function, this way we can build upon already meaningful input data to create a more accurate representation of what is actually occurring inside the patient's brain. Our network must also output an accurate assessment of seizure likelihood in real time, meaning real time outputs with relatively high sensitivity and low false positive rates. Lastly, our algorithm must be able to adapt based on feedback, self evaluation, and calibration.

2.3.1 Functions and Specifications List

Table 5: Seizure forecaster functions and specifications list.

Functions	Specifications
Preprocess EEG Data	- ICA artifact reduction - High pass filtering for noise reduction - Low pass filtering for noise reduction

Process EEG Data	- Input data into NN comprehensively
	  - STFT
	  - CWT
	  - Matrix representation
	  - etc.

Feature Extraction for Classification	- Compatible with method of input
	  - PSD
	  - Entropy
	  - Statistical measures
	  - HCTSA

Feature Extraction for Forecasting	- Probability ratios - Kurtosis

Section 3

3.1Alternative Design Solutions

3.1.1 Morphological Chart

Table 6: Seizure forecaster morphological chart.

Functions	Means

	1	2	3	4
Preprocess EEG Data	ICA Artifact Reduction	Apply Filters Individually to Raw Data	Use EKG Signals to Attenuate Heartbeat Noise	Highpass/Notch /Bipolar Filters
Input EEG Data	STFT	CWT	Time-Series Data	
Extract Features for Classification	PSD	HCTSA	Statistical Measures	Entropy
Extract Features for Forecasting	Probability ratios	Kurtosis		

3.1.2 Alternative Designs

3.1.2.1 Design Option 1: CWT-VGG16

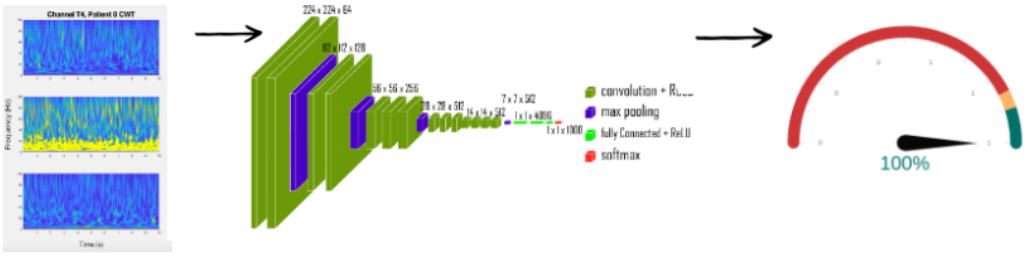

Figure 5: shows CWT input data inputted into a VGG16 based CNN for percent based seizure prediction.

How Design 1 Works

Cleaned EEG signals are inputted channel by channel. For non seizure data; ten second windows are shifted by ten seconds over the entirety of the non seizure data. For seizure data; ten second windows are shifted by one second. So, a seizure period will have 10x the amount of raw data in comparison to a non seizure of an equivalent time period. Raw EEG data is then converted via Continuous Wavelet Transform into scalograms and fed into a VGG16 pretrained network. The VGG16 image analysis CNN breaks inputted images into 3x3 pixel squares and as it shifts (strides) by one pixel at a time puts together an all encompassing, comprehensive map of the image. From this map distinct edges and features using convolutional blocks with filter numbers ranging from 64 to 512 filters. The first convolutional block, with 64 filters, will pick out the most basic properties of the data, while the fifth convolutional block will pick out abstract features from the data. Herein, the VGG16 CNN can distinguish between general blue, yellow and red areas, and as focus increases, can pick up shapes and patterns within the data. The VGG16 architecture accounts for overfitting, the networks propensity to become too

accustomed to the data and call false positives, by incorporating max pooling layers at the back end of each of the 5 convolutional blocks. The whole VGG16 architecture will not be kept for our project, the back-end of the network (fully connected layers), aimed at assigning percent likelihood to 1,000 classifications, will need to be changed in order to accommodate our classifications (non seizure and seizure). As we move forward with implementation and testing, our CNN will need to be trained so it can pick out identifiable features from our data and begin to apply it to training data. This is going to be a massive undertaking in the background of our designing process as this takes a lot of processing power. Concisely, design one incorporates CWT images with a VGG16 image classifier for seizure likelihood classification. Similar designs have been evaluated, one which compares STFT and CWT found CWT in tandem with a modified VGG16 image classifier achieves an average accuracy of 99.2% of varied datasets [20]. Eventually, forecasting methods will be implemented in order to predict seizures before they happen.

Strengths and Weaknesses

Continuous Wavelet Transforms (CWT) offers distinct advantages over other forms of time-frequency domain inputs, particularly in the context of seizure detection. This is supported by the aforementioned literature where CWT consistently outperformed STFT given varying patient data [20]. Although InceptionV3 and ResNet classifiers performed better than VGG16 in a study by Gao et al, these results were achieved using STFT data and the small number of convolutional layers and ease of implementation of VGG16 serves the purpose of simplicity for this design [38]. These advantages are pivotal for our design's success in rapidly identifying seizures for the well-being of patients. Firstly, CWT excels in representing data in real-time, a critical aspect for promptly forecasting and identifying seizures. Minimizing processing time is important, and leveraging input data that accurately captures temporal dynamics is crucial for our algorithm. Secondly, CWT images provide precise representations of high-frequency bands, which is valuable for distinguishing seizure vs. non-seizure activity. If identifiable features for seizure detection occur in high-frequency bands, specialized CWT inputs enhance the network’s ability to discern subtle patterns, thereby improving overall classification accuracy. By employing the VGG16 architecture, we ensure both effectiveness and simplicity in our design. VGG16 offers a straightforward yet powerful framework for image analysis, which aligns seamlessly with our objectives. Its proven success as an image classifier for EEG-based data reinforces its suitability for our task. Our design strikes a balance between simplicity and efficiency by utilizing single-channel inputs and processing data in 10-second CWT segments. This approach preserves the nuances present in individual channels without sacrificing computational efficiency. Furthermore, the compatibility of VGG16 with CWT input data ensures the robustness of our network. In summary, our design leverages the strengths of CWT and VGG16 to achieve rapid and accurate seizure detection. By prioritizing simplicity and efficiency we can make significant strides in improving patient outcomes through timely seizure identification.

3.1.2.2 Design Option 2 (Adapted from Article [44])

How the Design Works

This design also implements single channel EEG data, meaning data from each electrode is left unaveraged with any other electrode. This design implements a numerical-based data matrix instead of CWT or other image-based methods. This alternative solution identifies epileptiform signals of a patient by analyzing their entropy relative to normal brain activity signals (inter-ictal). Many equations in biophysics, signal processing, and information theory quantify entropy, but based on literature review, Sample, Fuzzy, and Renyi entropy, when combined into a tensor can provide a comprehensive analysis of a given signal's entropy with a sensitivity > 95% and FP/hr < 0.5 [44].

Sample Entropy quantifies how deterministic a signal is by searching for patterns within two matrices, based on embedding dimensions which incrementally decrease into one representative of the signal. The function is given below, with A and B defined as separate matrices:

$$SampEn = -\ln \frac{A}{B}$$

Fuzzy Entropy quantifies the degree of fuzziness between a given length (mu) through membership functions in fuzzy set theory. It provides the uncertainty of the signal. It is often described as an improvement of Sample Entropy, but in literature has been shown to be effective in conjunction with it. The function is given below:

$$H_d(A) = \frac{2}{n}\sum_{i=1}^{n} | a_i - \mu_{A^0}(u_i) |$$

Renyi entropy is primarily used within quantum information theory, providing a general measurement of entropy of a probabilistic function, encapsulating Shannon (a = 1), Hartley, Collision and min entropies within a single function. The formula is given below, with a > 1:

$$\mathrm{H}_\alpha(X) = \frac{1}{1-\alpha}\log\left(\sum_{i=1}^{n} p_i^\alpha\right).$$

1-D wavelet decomposition was performed on a 20 second sample of all channels to produce sub-bands using a Daubechies mother wavelet, Theta [4 - 8] Hz, Alpha [8 - 12] Hz, Beta [12 – 30] Hz, Gamma Low [30 - 45] Hz, Gamma High [45 100] Hz used to describe entropic variation via sub-bands. The decomposition utilized db2 to db8 levels in order to capture the specific frequency ranges for the sub-bands. Using PN00 as an example, the sub-bands of importance in entropic analysis were more clearly defined, which is shown in 9.0.

The output of these transformations yields three vectors based on 6 sub-bands, which are each treated into a matrix, and combined to create a (3 x 6 x 6) matrix to extract features through a CNN, shown below (Note: the matrix below takes into account delta band but was removed within this design as HP filter removes most of the data). The result yields a probabilistic likelihood of seizure onset, using 20 seconds of cleaned data as an input.

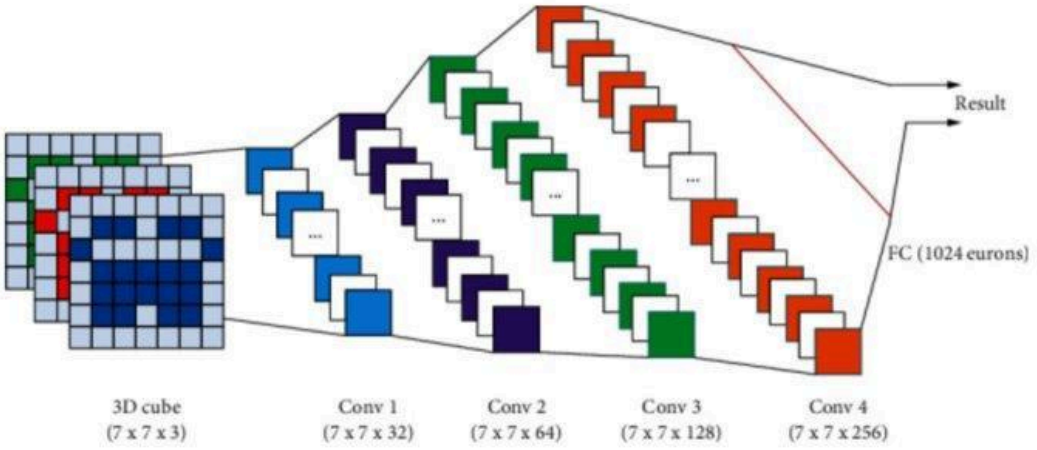

Figure 6: CNN design specifications of entropy feature extraction

Design Strengths and Weaknesses

While the design captures many features related to uncertainty, it does not take into account where variations occur specifically within the sample. Also, it must be noted that when transforming all channels, the focal channels will have less weight in describing a proper likelihood. Twenty second samples might also be two computationally dense to perform in real-time.

Section 4

4.1 Project Management Plan

Table 7: Seizure forecaster project management plan.

Phase	Task
1. Literature Review	-Image-Based vs Numerical-based
	-CWT vs STFT
	VGG16 vs InceptionV3
	-Report 1
	-Report 2

	-Report 3
	-Final Report
3. Data Preprocessing	- Clean data
	- Convert Data (CWT)
	- Organize Data (inter, pre, ictal, post)
4. Model Architecture Design/Picking	- Data Integration
	- Pretrained Model Modification
4. Model Training	- Split Data (training, testing)
	- Training Iterations
	-Hyperparameter Tuning (to optimize learning/testing)
5. Model Evaluation	
6. Forecasting	-TBD
	-TBD
	-TBD

Section 5

5.1 Applicable Iterations

Data cleaning is a crucial yet time-consuming process in the development of machine learning models, particularly when dealing with patient datasets. To streamline the whole process, iterations and testing often start well before the entire dataset is cleaned and converted. In fact, only half of the total patient dataset is required until the final production phase, where the model's performance against unseen patient data is evaluated. At the start, when we are tuning hyperparameters to prevent overfitting and extreme loss, the focus is narrow, requiring just a few electrodes from a single patient. As data cleaning progresses and electrodes from multiple patients are acquired, the scope widens. This expansion, from just a couple electrodes to a couple patients, allows for cross-patient validation (cross validation) during the model training and validation stages, where metrics such as accuracy, sensitivity, and specificity are assessed. The final stage of this iterative process lies in model testing, necessitating an even larger dataset. Here, the aim is to challenge the model with a diverse range of data, eliminating the possibility of its success being reliant on easily classifiable data. Each phase serves a distinct purpose, collectively ensuring that the model is robust enough to withstand the variability inherent in real-world scenarios.

5.1.1 Hyperparameter tuning

Beginning with a limited dataset, hyperparameter tuning plays a pivotal role in optimizing our model training process. As a result of varying model architectures, individual models respond uniquely to different hyperparameters. The two models we were interested in at this stage of production were VGG16 and InceptionV3 image classifiers. Individual responses to hyperparameters were seen in our iterations. inceptionV3, having a total of 48 convolutional layers in comparison to VGG16's 16 convolutional layers, was much more apt to overfit training data. Overfitting occurs when a network becomes proficient solely at classifying already-seen training data, leading to arbitrary classifications and diminished accuracy when faced with variability within the dataset. To mitigate this, we opted for pre-made optimizers that offer preset parameters, easing the decision-making process.

As previously stated, VGG16 has a simpler architecture and is less likely to overfit in comparison to the InceptionV3 architecture. For this reason we utilized an Adam optimizer for InceptionV3 that gave us an adaptable learning rate, in this the model would correct itself if overfitting was occurring. VGG16, conversely, would overfit on occasion, but, for the most part, this was not an issue. Consequently, we utilized an SGDM (Stochastic Gradient Descent with Momentum) optimizer.

The Adam optimization algorithm is a widely used extension of stochastic gradient descent (SGD) which is specifically tailored for deep learning applications such as image classification. Unlike SGD, which maintains a single learning rate (alpha) for all weight updates throughout training, Adam employs a separate learning rate for each network weight (parameter). This learning rate is adapted individually as learning progresses.

Adam implements the mean as well as the variance, this makes it optimal. To do this, Adam calculates exponential moving averages of both the gradient and the squared gradient.

The decay rates of these moving averages are controlled by parameters beta1 and beta2. Initially, these parameters, along with the initial values of the moving averages, are set close to 1.0, which introduces a bias towards zero in the moment estimates. So, in the beginning of any training the adjustment will be low. As more data is learned the bias shifts and Adam then calculates bias-corrected estimates.

Stochastic Gradient Descent with Momentum (SGDM) is a variant of SGD designed to mitigate oscillations along the path of steepest descent towards the optimum. It achieves this by incorporating a momentum term into the parameter update equation. By adding momentum, SGDM helps accelerate convergence and reduce oscillations, resulting in faster training and improved stability. It is important to note that it is crucial to have a low initial learning rate with SGDM as the oscillatory nature of this optimization technique can be further smoothed with a lower initial learning rate.

Default hyperparameters such as epochs (1), batchsize (2), and validation frequency (3), drop-out rate (4), class weights (5), initial learning rate (6) were provided in each optimizer function, although they could still be tuned manually in order to optimize frequency, which we did.

So, after our final iteration shown in Figure 8, we moved forward with SGDM and Adam optimizers for their respective networks. The SGDM optimizer, used for the VGG16 model (which we eventually moved forward with as it performed better) used the following parameters and achieved a validation accuracy of 91.3%:

Table 8: Hyperparameter settings for VGG16-SGDM training optimization.

Hyperparameter	Setting
Epochs	20
Batchsize	40
Validation Frequency	25
Drop-Out Rate	0.7
Class Weights	0.4 (ictal)
	0.05 (inter-ictal(before seizure))
	0.05 (inter-ictal(after seizure))
	0.2 (post-ictal)
	0.3 (pre-ictal)
Initial Learning Rate	5e^-5

(1) Epochs: Number of times the applicable dataset is fed through the model.
(2) Batchsize: Number of CWT images fed through the model at a time.
(3) Validation Frequency: Number of iterations in between each validation.

(4) Drop-out Rate: rate at which randomly selected data are temporarily excluded during training to prevent overfitting and enhance model generalization.
(5) Class weights: gives more importance to minority classes (ictal) during training.
(6) Initial Learning Rate: influences size of updates made to the model's weights during training, impacting the rate at which the model learns from the data.

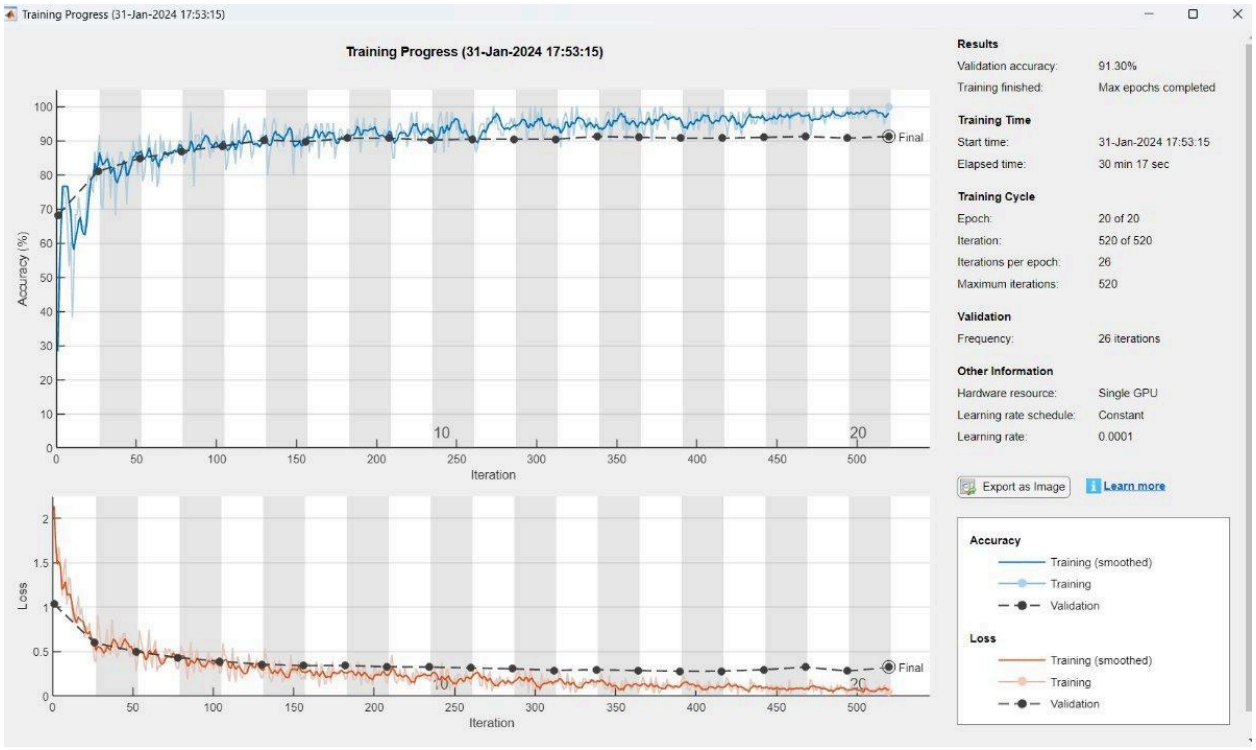

Figure 8: VGG16-SGDM results for PN00 (electrodes T4 and T6) split 80/20 for testing/training. Top graph(blue) shows validation accuracy over iterations. Bottom graph (red) shows the loss as a function of iterations.

5.1.2 Training and Validation: VGG16 vs. InceptionV3

Establishing hyperparameter optimization for each network was crucial; however, determining the superiority of one network over the other was imperative. Given the time-consuming nature of duplicating iterations and testing for two networks, a more streamlined approach was necessary.

Initially, our hyperparameter tuning involved five classifications for our data. Subsequently, we refined our methodology by excluding post-ictal and inter-ictal data while merging pre-ictal and inter-ictal data to enhance predictive accuracy for seizure onset. This consolidation enabled us to categorize our data into two distinct classes (seizure and non-seizure), facilitating binary classification within our models. Binary classification not only reduced computational workload but also simplified our analysis. To prove the superiority of one network over the other, various factors were evaluated, including overall efficiency, computational load, accuracy, sensitivity, and specificity. Consistency was ensured by maintaining constant training and testing data splits across iterations for both networks. In this,

we ensured that the network's performance was directly correlated to the respective result, and the input data was not impacting performance. Furthermore, datasets were progressively manipulated and expanded across iterations to analyze our models.

Conceptually, it was understood that because VGG16 had less convolutional layers (16) than InceptionV3 (48) it would be less computationally extensive when implementing. This was corroborated by shorter run times during the hyperparameter tuning phase. Simply put, this was a general observation that was confirmed by a simpler VGG16 architecture. InceptionV3 is also prone to overfitting and it was found that although the Adam optimizer did partially solve this problem, achieving equivalent accuracy with InceptionV3 on testing data was laborious. This showed us that despite the Adam optimizer, overfitting was still an issue for InceptionV3. It is worth noting that this could be due to our specific dataset and may not be related to the actual quality of the network. In fact, literature supports InceptionV3 as a more successful model, despite our findings [38][39][40][41].

Sequentially, we started with small (two patients: 5 electrodes each with respect to individual seizure focalization) datasets with an 80/20 training/testing split to enable cross validation and incrementally built up to larger (5 patients: 5 electrodes each with respect to individual seizure focalization) with a 4 to 1 patient training/testing split. As mentioned before, ictal data was collected with a time shift of 1 second on a 10 second window leading to more ictal data. This increased ictal data was used to create a 50/50 split of ictal and interictal data.

Furthermore, we implemented two training techniques which are referred to as ensemble and consecutive. Ensembled models reflect an averaged or aggregated prediction based on every model within the ensemble. In this, we were able to train a VGG16 or InceptionV3 network 5 different times, each of these trained models contributed to their ensembles respectively. Consecutive models refer to a sequence of models where the output of the first becomes the input of the second, this pattern continues for each network in the system wherein the final network outputs a prediction that is correlated to each network in the system. Both of these techniques serve the advantage of bringing together many models (trained differently and separately) into one large prediction, it gives us as researchers a better indication of how each network performs on a larger scale.

With our largest and latest dataset of 5 total patients with a data split of 4 patients (training) to 1 patient (testing) we ensembled and consecutively ran a total of 10 individual networks for both VGG16 and InceptionV3. Not only were the individual networks for VGG16 consistently more accurate than InceptionV3, but this was supported by data found in confusion matrices. Confusion matrices outline True Positive/Negative and False Positive/Negative values and therefore give us accuracy (equation 5), sensitivity (equation 6) and specificity (equation 7). Accuracy gives the overall efficacy of the model (including all positives and negatives). Sensitivity refers to the proportion of correctly classified positive data vs. incorrectly classified negative data. Conversely, specificity refers to the proportion of correctly classified negative data vs. incorrectly classified positive data. It was shown, in Figures 10-13, that although ensembled networks were superior in classifying seizure data, they were far inferior in classifying non-seizure data. Since consecutive networks were proficient in both seizure and non seizure data, we established that consecutive networks were a more viable path forward. Furthermore, we concluded that the VGG16 model performed significantly better than

InceptionV3 for our dataset, and, therefore, we moved forward solely implementing a VGG16 based consecutive model.

	1	2	3	4	5	6	7	8	9	10
1	0.9265	0.9313	0.9265	0.9349	0.9313	0.9349	0.9211	0.9307	0.9229	0.9253

Figure 10: VGG16 accuracies for each individual network in the ensemble and consecutive.

	1	2	3	4	5	6	7	8	9	10
1	0.8789	0.8554	0.8741	0.9175	0.8620	0.8886	0.8855	0.8934	0.8928	0.8572

Figure 11: InceptionV3 accuracies for each individual network in the ensemble and consecutive.

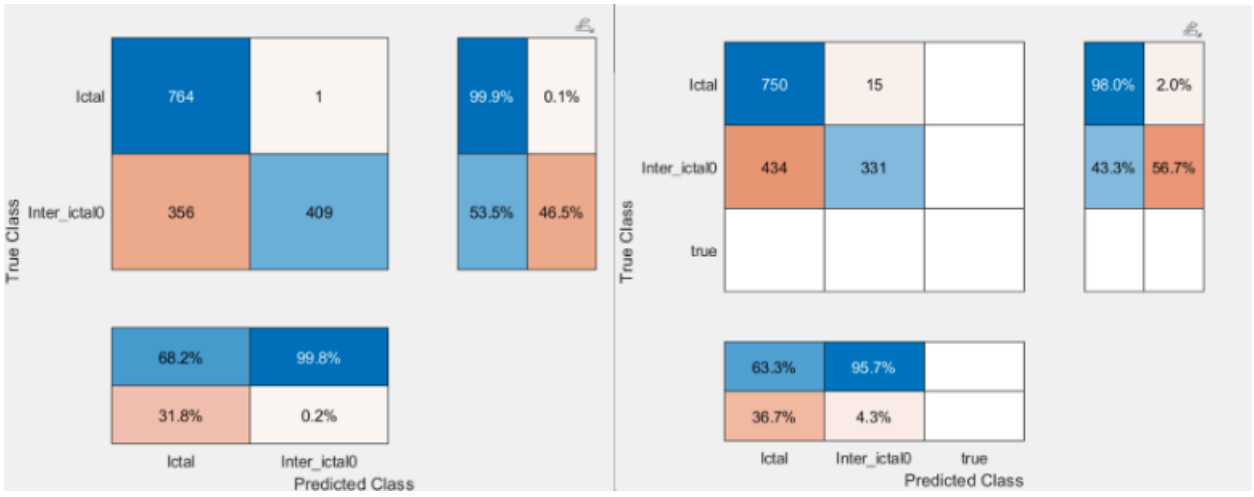

Figure 12: Ensembled confusion matrix for VGG16 (left) and InceptionV3 (right) with respect to network classification accuracies shown in Figures 9 and 10.

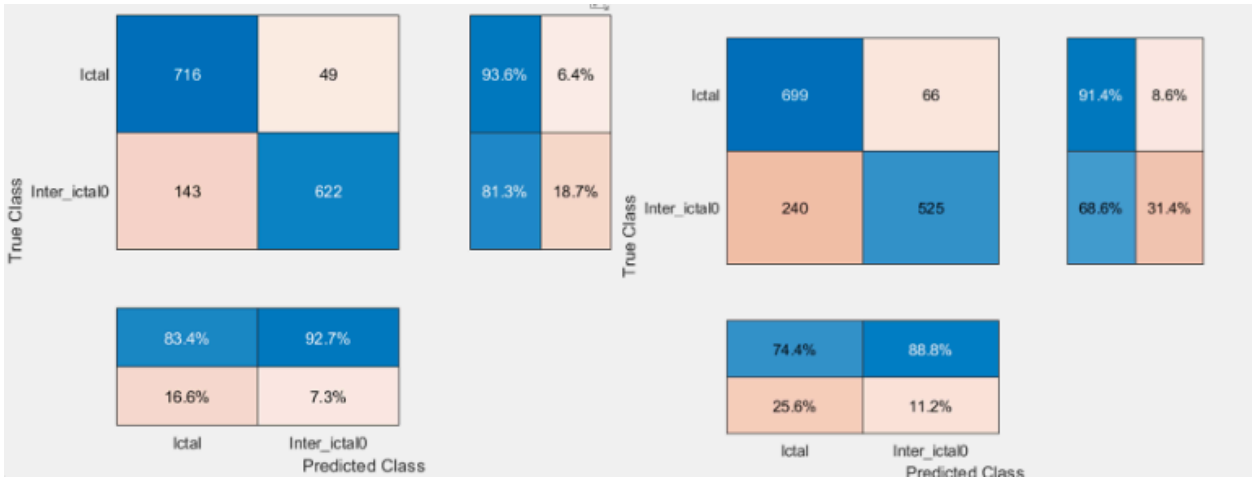

Figure 13: Consecutive confusion matrix for VGG16 (left) and InceptionV3 (right) with respect to network classification accuracies shown in Figures 9 and 10.

5.1.3 VGG16 Testing

Before we opted into using the HPC, we had already established key strategies for training our network. Namely, we were inclined to use the transfer learning method using VGG-16. With the capabilities of the HPC at our disposal, we opted to determine the validity of our dataset using k-fold cross validation.

Training and testing CNNs demands substantial computational power, which can be efficiently handled through use of a GPU. To address this issue, we utilized UCONN's High Performance Computing Cluster (HPC) which provides us with the processing power needed to complete each task in a timely manner. Through this virtual connection, coding scripts (training/testing iterations) are submitted to the "cluster." The HPC also allows for many CNNs to be run simultaneously, while also offering a much higher pace. This significantly expedites the efficiency of our project goals.

For the first set of runs on the HPC cluster, we had the scalograms of five patients cleaned. Similar to the preceding training (largest consecutive-VGG16 training run in Figure 13), inter-ictal and ictal data of Patients 00, 05, 13, 16, and 17 were used. For these, we used electrodes T3, T5, F9, CP5, and O1. For this first set of runs, we ran five different scripts, swapping the testing patient out each time. In other words, we used 4 training patients and 1 testing patient, each iteration swapped one of the previous training patients for the testing patient until every patient had been individually tested on. The resulting confusion matrix is as follows:

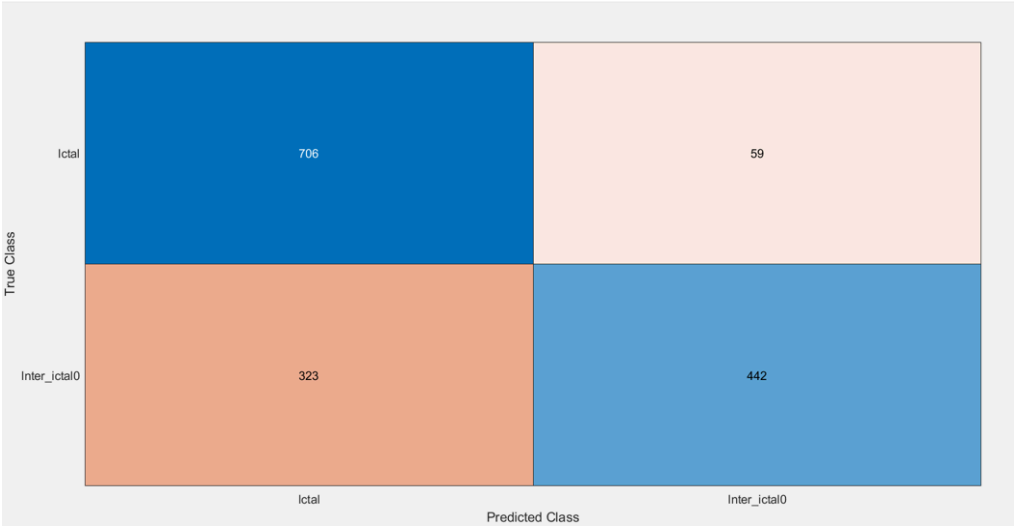

Figure 14: 8 VGG16 networks consecutively ran on a 5 patient dataset. Split into 8 data subsets for a total of 8 networks

Although the overall accuracy has room for improvement, it the false positives could be a sign that our model is picking up on seizure-predicting data. In this, if we could graphically lay

out how our model classifies non seizure data as seizure data it could shed light on predicting seizures before they happen.

Next, we expanded our dataset by preprocessing and converting patients 06, 09, and 12 data to their respective CWT format. All of these patients exhibited left-temporal lateralized seizures at onset and therefore electrodes T3, T5, F9, CP5, and O1 were used. The iterations were conducted in the same manner as the iterations explained before with testing patients swapped after each iteration. The results for this run is as follows:

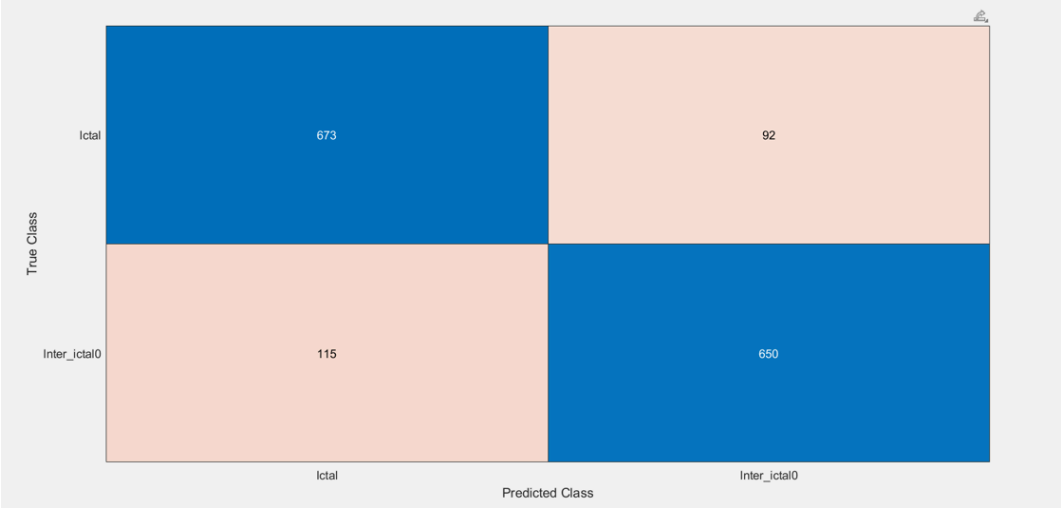

Figure 15: 8 VGG16 networks consecutively ran on a 8 patient dataset. Split into 8 data subsets for a total of 8 networks

This also shows promising results as with a larger dataset we get increased accuracy. For the third set of testing, our network's ability to successfully classify pre-ictal data was observed. For this, we separated our non-seizure data back into inter-ictal and pre-ictal data. Pre-ictal data is set as 30 minutes prior to seizure onset, whereas inter-ictal data was composed of all the data preceding pre-ictal. For runs including pre-ictal data adjustments were needed to our network architecture. Namely, we had to adjust the input layer, the last fully connected layer, and the output layer for 3 classifications instead of two. This first model did not yield promising results and so hyperparameters were adjusted to optimize pre-ictal classification. Mini-batch size was changed to 16, epoch count was increased to 100, and we fine-tuned the initial learning rate to 1e-4. L2 regularization was incorporated to prevent overfitting, alongside a new learning rate drop factor of 0.5 and a slower drop period of 20 epochs. These modifications were anticipated to enhance model training efficiency. However, the performance, as illustrated in the confusion matrix below, did not meet our expectations. As a result, binary classification remained our primary method as we moved forward into seizure prediction.

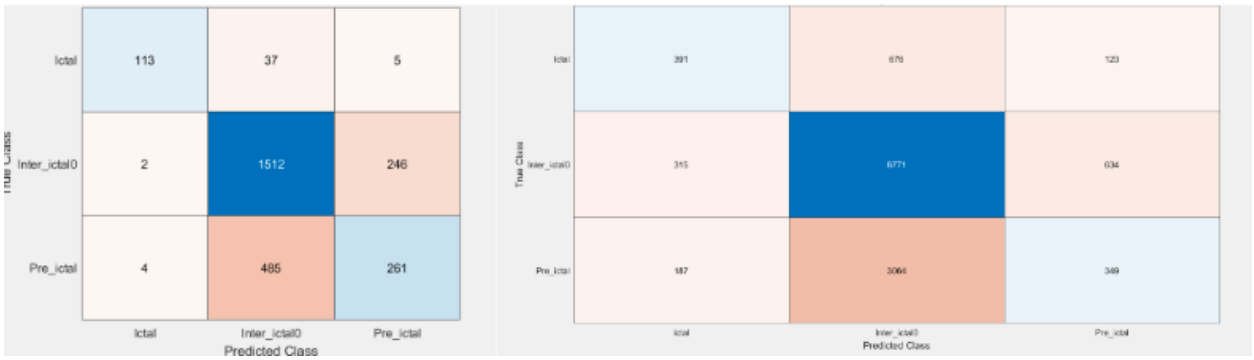

Figure 16: Left: 3-class classification on consecutive VGG16 model (8 patients) with same hyperparameters as previous model. Right: 3-class classification on consecutive VGG16 model (8 patients) with final adjusted hyperparameters.

5.1.4 Predicting Capabilities (Likelihood)

5.1.4.1 Electrodes Classified as Seizure

While the trained network gave a decent accuracy (~90% accuracy) regarding the testing data (patients 6 and 17), the activations were only tested on data 30 minutes before seizure onset (termed interictal) and marked ictal data. It must be noted that while the clinical onset and end have been determined by medical officials, the epileptiform waves are not consistent throughout the entire labeling. Conversely, interictal discharges are seen within people suffering from epilepsy that may mimic the waveforms seen within a seizure, but not to the same time and/or intensity. Therefore, we chose a conservative approach in marking a scalogram as resembling epileptic activity as determined by the trained network, where a prediction of 70% “Ictal” was flagged as epileptiform.

5.1.4.2 Moving Mean

To curb leftover artifacts in the data, a moving mean was used to attenuate the activations from the trained network; these mostly resemble heart and muscle artifacts, noted by their low frequency and rhythmic pattern.

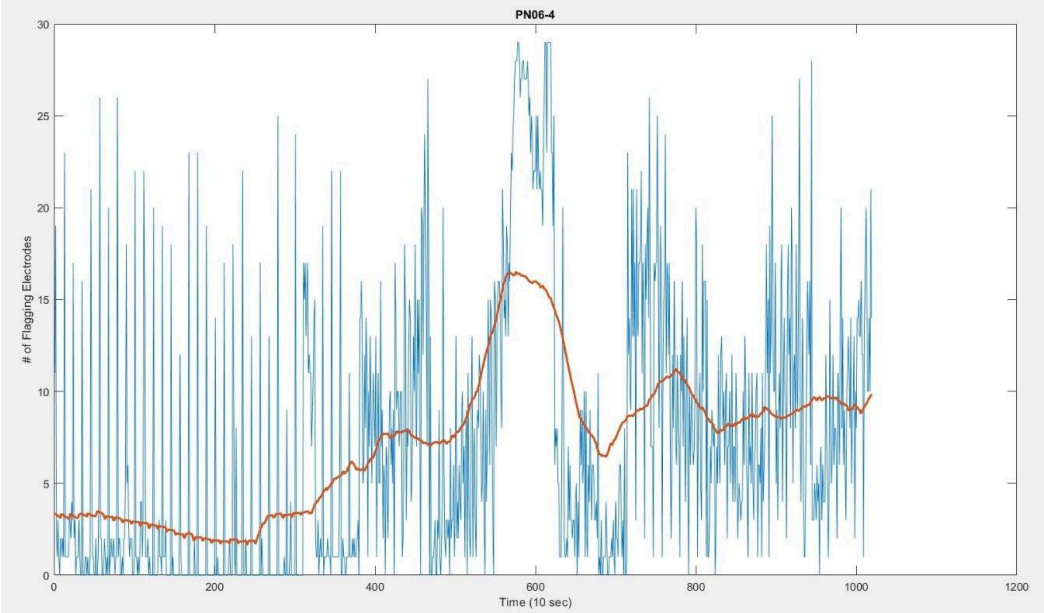

Figure 17: Move mean graph: applied to data to taper remaining artifacts

The 10-second frames used by the movmean were optimized according to the likelihood ranges produced by the respective distributions. This image shows an example of using 120 data points (20 minutes of EEG data).

5.1.4.3 Probability Distributions

In capturing the statistical components of flagged electrodes, three groups (Interictal, Pre-ictal, Ictal) were assessed in terms of their fit to a Gaussian distribution, using the function fitdist with a Gaussian kernel. From these distributions, the PDFs of the distributions were calculated and formed likelihood ranges based on the ratio of their likelihoods. A cursory assessment of the goodness-of-fit was shown in the lower figures, where a gaussian distribution did not seem to retain the true distribution of the periods leading up to seizure onset (as shown by kernel fit)

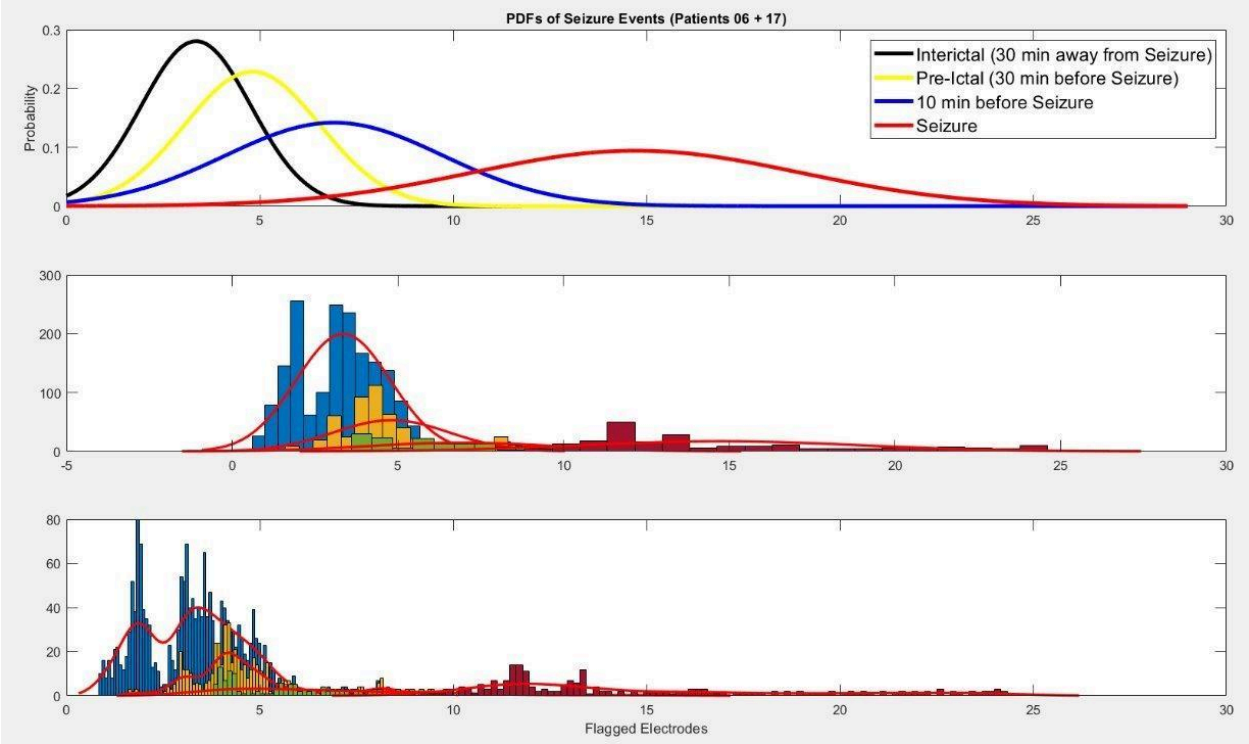

Figure 18: Gaussian distribution fitting of flagged electrodes (>70% confidence) for ictal (red), pre-ictal (blue/yellow), and inter-ictal (black).

From here we calculated the ratios of likelihoods based on the number of flagged electrodes, generating the plot described below. We also saw this as an opportunity to tune the moving mean function as the boundaries determined by the likelihood function trended to stable values.

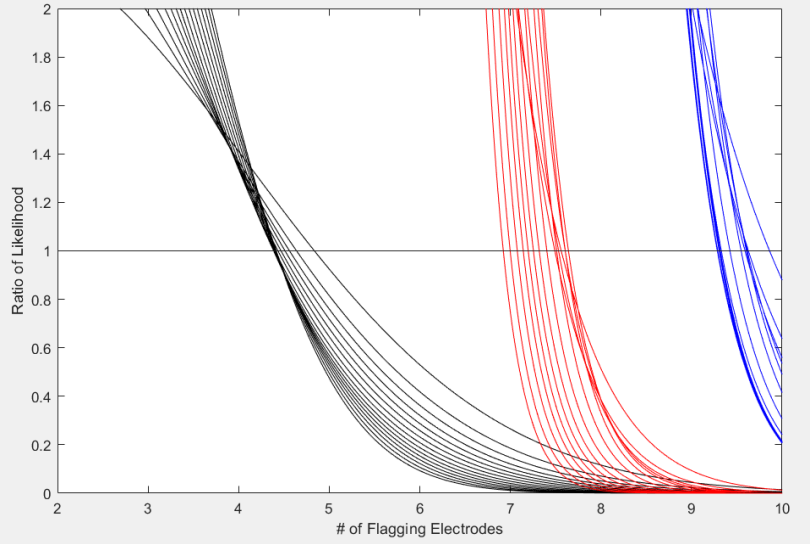

Figure 19: Ratios of likelihoods based on number of flagged electrodes. Black: P(inter-ictal)/P(pre-ictal). Red: P(inter-ictal)/P(ictal). Blue: P(ictal)/P(pre-ictal).

The idea is to mark the amount of flag electrodes based on the ranges described by these boundaries, [Low, Medium, High, and Seizure] risks. However, these boundaries did not capture the patterns of the pre-ictal region as the amount of flagged electrodes did not always monotonically increase leading to seizure onset.

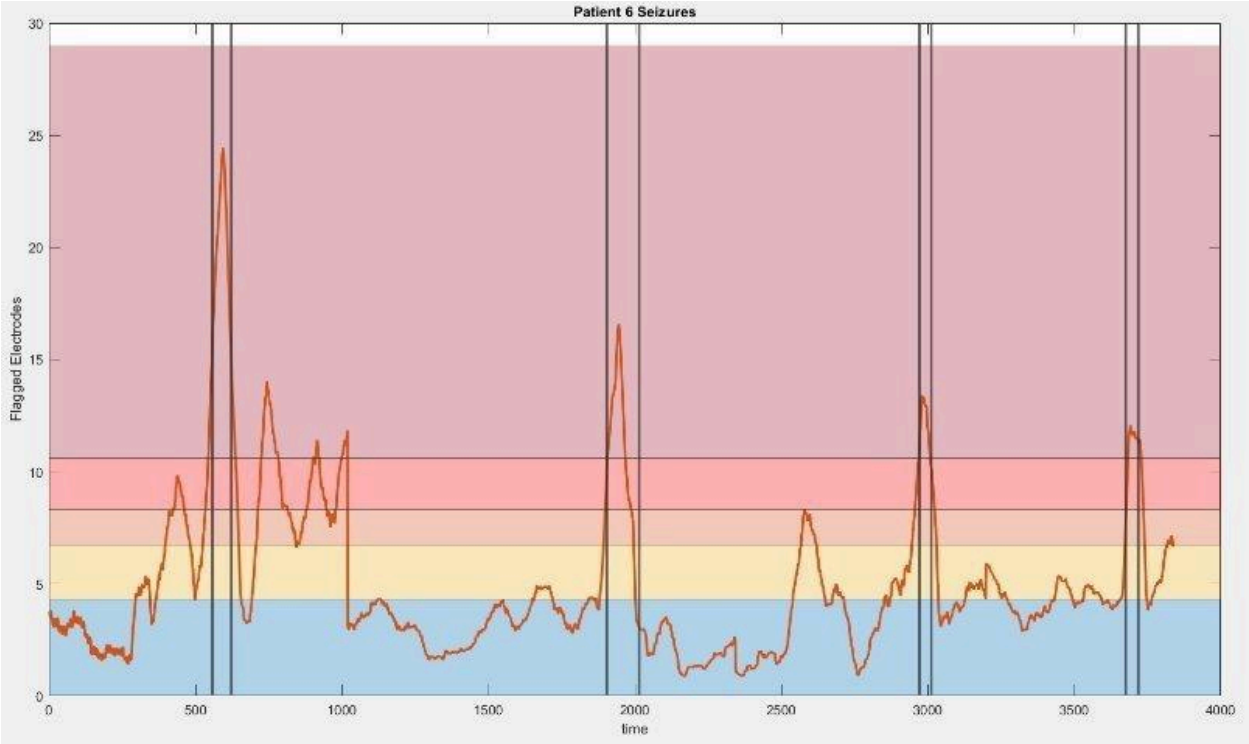

Figure 20: Risk Zones plotted against Patient 6 seizure files

With an additional boundary set for the 10 minutes before seizure, the seizure itself was only reliably captured by the dark red boundary [P(10 minutes before onset)/P(Seizure)]. As such, a regressor was not able to derive the ratio likelihood as a function of time before onset. However, a new approach was developed as a realization of this shortcoming, namely the amount of variation within the distribution leading to the seizure onset.

5.1.4.4 Kurtosis

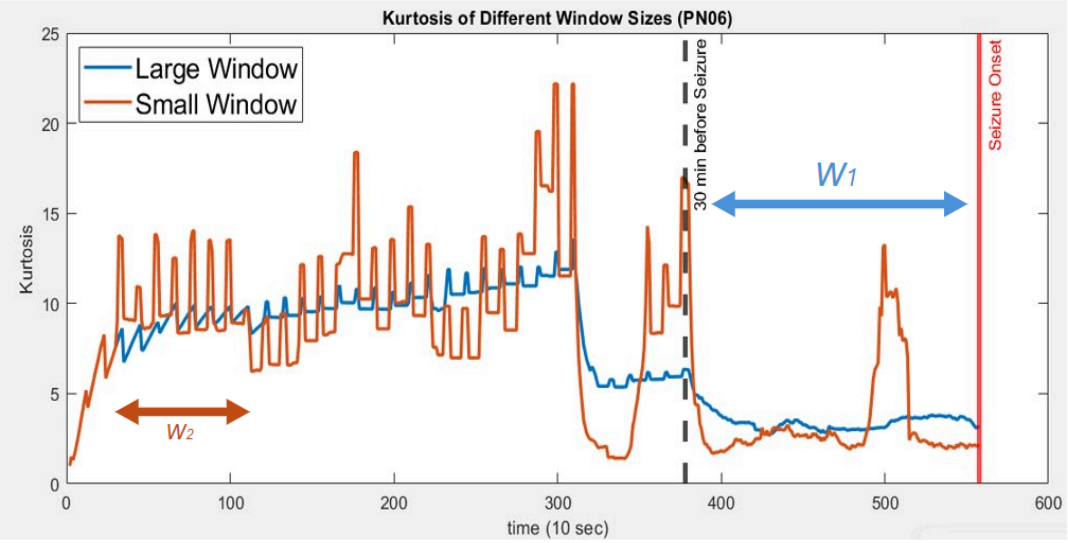

Figure 21. Small and Large kurtosis windows plotted against the periods leading up to seizure onset (file 06-1) Small window: 30 frames, Large window (120 frames)

The distribution of flagged electrodes in the pre-ictal region were found to be distinguishable by use of kurtosis. As the patient progresses towards a seizure, the flagged electrodes tend to be more erratic, hence the decrease in kurtosis value. MATLAB presents a kurtosis value with 3 being the benchmark, i.e. the value for a normal distribution, which suggests that the pre-ictal state is typically platykurtic (fat-tailed) before onset. As correlated previously from the distribution fits (Figure 18), both the ictal and pre-ictal distributions followed a poor fit, with large standard deviations like that of a beta distribution, which verifies the use of kurtosis as a feature to use as a contrast between the inter-ictal and pre-ictal.

5.1.4.5 Semaphores and Optimization

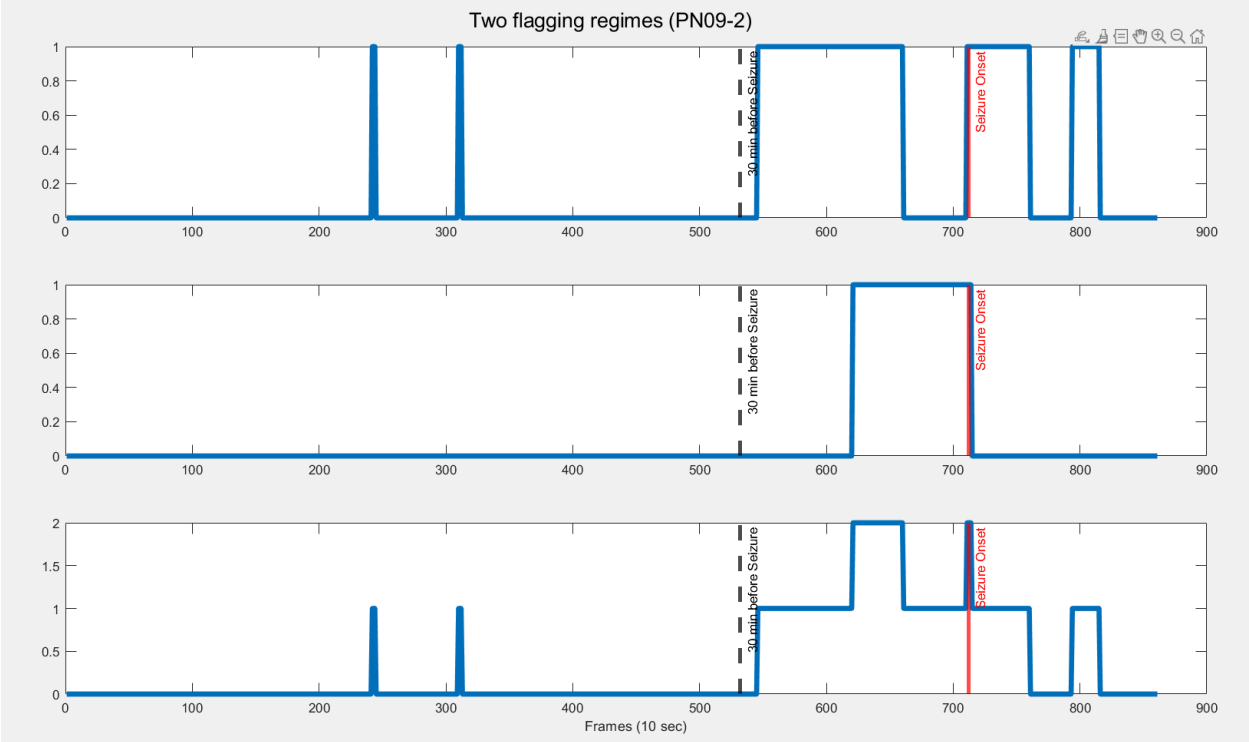

Figure 22. Optimized flagging regimes: top plot (Large kurtosis window, small moving mean window), middle plot (Small kurtosis window, large moving mean window), bottom plot (combination of both flags). Patient is warned when both flags are active.

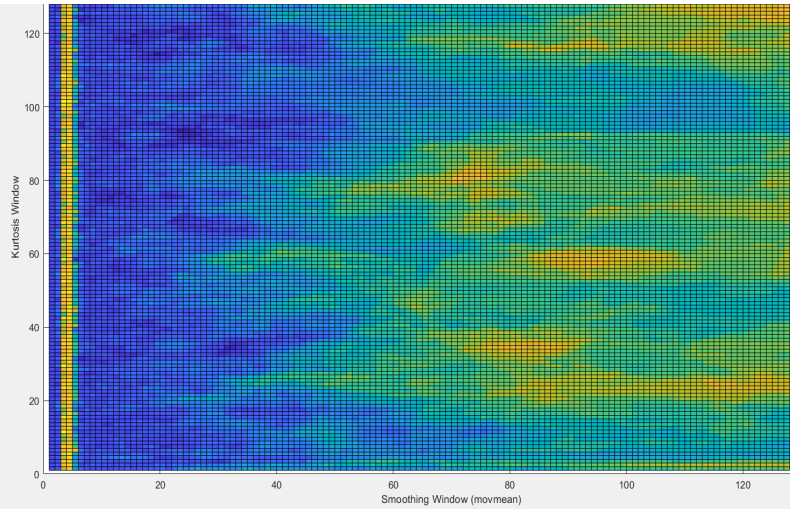

Figure 21. Grid search of smoothing window (moving mean) and kurtosis window. Color depicts the difference between flags beyond 30 minutes and flags within 30 minutes to onset. A threshold of 3 was consistently applied to this strategy.

With kurtosis as a valid feature in identifying the pre-ictal region, the parameters of the window and smoothing factor (moving mean) had yet to be implemented. As such, a brute force optimization strategy was employed (grid search), where parameters of the flagging strategy were chosen by the least amount of yielded false positives. The optimization was run on multiple patients at once, where the difference between flags in the inter-ictal and pre-ictal region determined the goodness of fit. A minimal amount of false positives tended toward two different regimes, a smaller smoothing factor and larger kurtosis window and the inverse. With both regimes considered, the false positives were minimized. Additionally, a length of consecutive satisfaction of these conditions were implemented to further reduce false positives and were run through the grid search likewise.

Table 9. Optimized Kurtosis Parameters via Grid Search

Moving Mean Window	Kurtosis Window	Length of Consecutive Runs
[2]	[85, 110]	[47, 75]
[20, 50]	[15, 27]	[50, 73]

Section 6

6.1 Final Prototype Specifications

The final prototype is intended to forecast seizures generally for people suffering with epilepsy, but may be more precise through training and testing within a single patient, as the semaphore flagging regime requires parameter tuning for a better goodness-of-fit. Regardless, the prototype requires an information input of scalograms and data cleaning, which may be optimized through use of a commercially available GPU. Additionally, the best performing CNN model coupled with the general forecasting parameters will analyze incoming data with a lag time much less than the input of scalograms (10 seconds).

The model is intended to perform over long time periods, both for calibration and the large kurtosis window, therefore, it would serve best in assisting medical health professionals in stabilizing/monitoring a patient with severe epilepsy throughout the night and during resting periods.

6.2 Final Data Collection

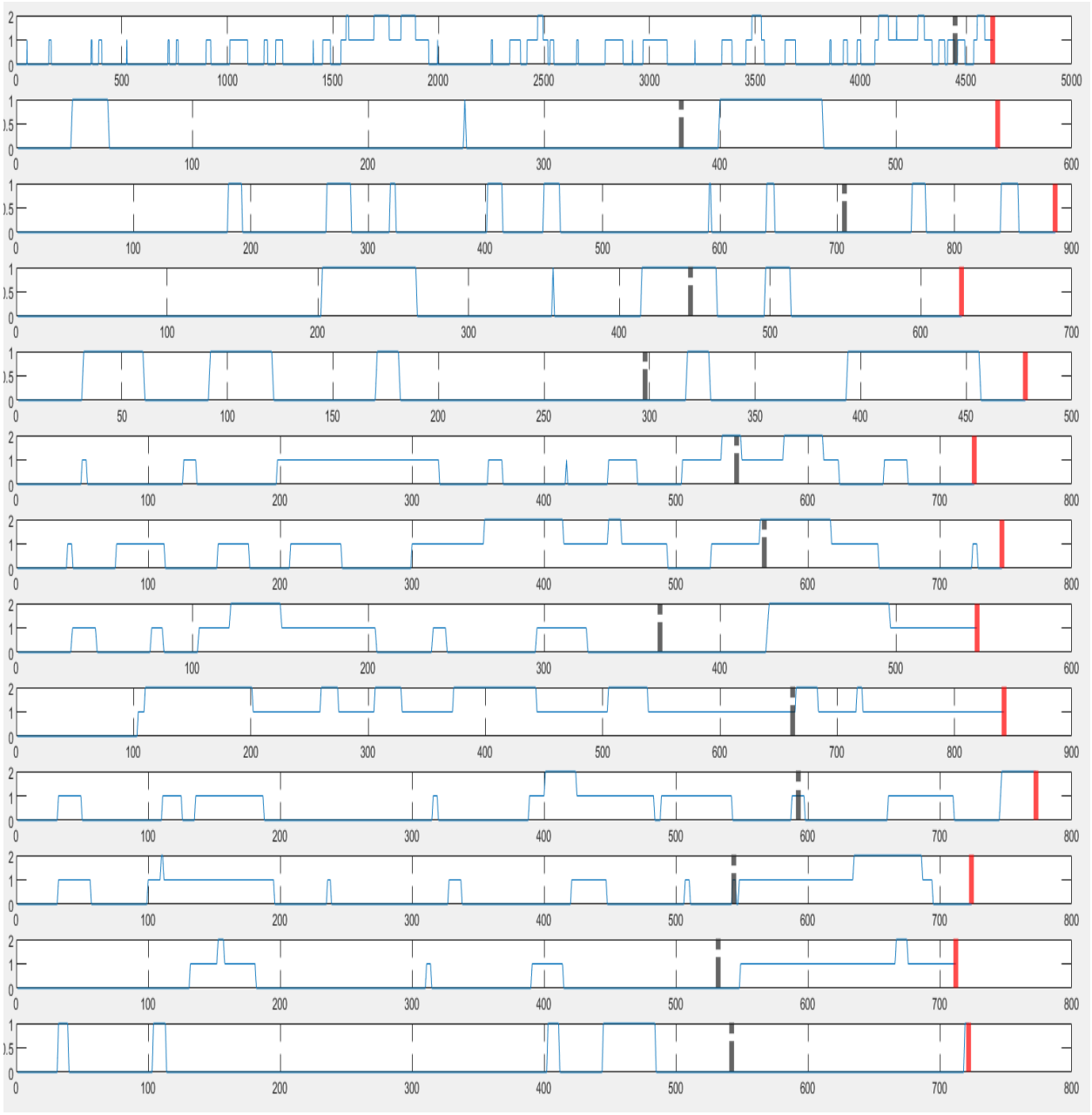

Figure 22. Sample of 13 seizures files taken from all patients in the database (Red: Seizure onset, Black: 30 minutes before onset)

The flagging regime was unable to capture seizures from patients 3 and 13, which was likely due to poor data preprocessing. The CWT scaling did not demonstrate a significant distinction in capturing the pre-ictal period likely due to the noisiness in the recordings.

6.3 Results

Table 10. Testing results (Macro-average) of 11 patients

	Precision (%)	Sensitivity (%)	AUC	Avg time of Flag (before seizure)	Accuracy (Seizures flagged ≥1 times)
Flag 1	37.11	**87.86**	0.63	15.02 min	100%
Flag 2	60.73	48.98	0.55	13.59 min	83.87%
Smart Flag*	**74.33**	18.14	0.46	13.46 min	83.87%

After fitting the forecasting parameters to each fold of the training/testing regime (6 trained CNNs), the sensitivity of the forecaster in flagging each and every frame within the 30 minute period was 18% which was vastly underwhelming; however, the purpose of the alarm is to only notify the patient once as, where enough time is given to administer some short-acting prophylaxis. Using this metric as a means of accuracy, flagging schemes were able to identify a pre-ictal state of 26/31 seizure files with a precision of 74.33%. On average, the flag warns the patient about 13.46 minutes before seizure, which leaves enough time to take precautionary measures.

Section 7

7.1 Materials

The total cost of materials for this project was 0. We used a no cost online database (PhysioNet) for our dataset, signal processing software (EEGLab), and programming platform (MatLab).

7.2 Regulatory Issues

The most critical regulatory consideration concerning our project is medical liability. Given that our predictor is not entirely accurate and false positives remain a possibility, obtaining informed consent from patients is critical. This consent would acknowledge the potential medical risks associated with false or premature alarms, ensuring that patients understand and accept these concerns before using this product. Furthermore, it is important to mention that our product has not undergone real-world testing yet, which is a mandatory step before bringing it to market. Real-world testing is crucial to validate the efficacy and safety of the predictor in diverse clinical settings, providing essential data for regulatory approval and ensuring the product's reliability and suitability for widespread use.

Section 8

8.1 Discussion

We have demonstrated that scalp-EEG data from 11 patients can effectively support a robust solution for predicting seizure onset. Our binary classification model reliably distinguishes between non-seizure and seizure states, enabling accurate forecasts within a 30-minute pre-ictal region of interest.

Our study focused on a dataset comprising focal seizures (localized onsets that subsequently spread) which heavily influenced our design considerations. Given the dataset's relatively small size, we opted for a simplified CNN architecture such as VGG16. A relatively small dataset such as this one led us to a simpler CNN architecture such as the VGG16. These focal seizures lead us to focus training around 5 focal and focal adjacent electrodes that exhibited more activity in the pre-ictal window. In this, we turned inherent limitations into our own advantage.

The VGG16 classifier excelled in testing on focal electrodes, demonstrating enhanced precision in discerning seizure-related data that ultimately minimized false positives. This resulted in a more robust forecasting model. Notably, our findings suggest that training a model on focal electrodes enhances its performance specifically within this context, offering promising implications for patients with focal seizure disorders. Additionally, our results reveal distinct brain states between non-seizure and seizure conditions as a consequence of our model's good precision. These distinguished states were used to model a state within the pre-ictal region leading up to seizure which were paramount in our forecaster.

With our forecaster we found that variance, what initially seemed like an obstacle, could be used to our advantage using higher order statistical methods. Kurtosis was found to be a viable way to model the variation seen in electrode activity in the thirty minutes leading up to seizure. This kurtosis was seen to drastically simplify the CNN output and utilized the CNN’s good precision to predict seizure onset. Kurtosis was particularly successful with our dataset because in the preceding minutes leading up to seizure much more variation was seen throughout our data and flagging our patients based off of this allowed us an accurate method to notify patients.

Many design options were decided based on our dataset, others were made based off of a literature review. Although logical and justified steps were made constantly throughout the process, future steps include much room for improvement despite successful results. To start, data preprocessing is a crucial step, it serves as the foundation for the whole model. In this, a more precise removal of noise and artifacts could have led to a superior classification and forecasting. We could have expanded on time stamps provided in the dataset and by identifying greatest activation could have cut out inactive seizure data. In this we could have trained on strictly active seizure data which would have established a better picture of what a seizure looks like for our classifier.

Furthermore, we could have utilized the time spectrum to a greater extent. Although CWT scalograms offer us time-specific resolution, we lose specificity on the progression of non seizure data by shifting ten second blocks by ten seconds at a time. This is a very generic way of parsing non-seizure data. Although this more detailed approach of shifting the scalograms by

less time is computationally extensive it could offer future researchers and their image classifiers a better understanding of the data at hand.

Within our VGG16 network, techniques could have been used to boost accuracy and efficiency. For example, Principal Component Analysis (PCA) could have been used to reduce the dimensionality of our scalograms to reduce noise. This technique also would have provided an increased computational efficiency, allowing us to run more tests in less time.

The choice in choosing VGG16 pretrained architecture was made due to superior functionality in comparison to other networks, which was supported by literature. However, this may have simply been a reflection of our small dataset which would have required a less intensive spatial feature analysis in order for the network to not overfit. In this, VGG16 worked for us, however, in future endeavors with larger datasets, it will be imperative to test on many different networks. For example, an autoencoder/decoder scheme could enable the model in tracking the progression of waveforms (as they appear in the scalograms) over time.

Overall, our model provides a strong assessment of seizure likelihood. Utilizing a multi-flag kurtosis and conditional system, we issued a definitive warning within a thirty-minute window preceding 83.7% of seizures across all 11 patients. This tool holds significant potential in professional scientific settings for studying the intricate dynamics of seizure states in the brain, in medical environments to corroborate doctors' decisions, and even for use by patients in their homes. The capacity to predict seizure onset without the necessity for invasive procedures, medication, or constant medical oversight remains a significant challenge for millions of individuals worldwide grappling with epilepsy. As part of our pursuit to identify seizures preemptively, our project represents a substantial stride towards achieving this goal.

References

[1] Pal, P.R., Khobragade, P., & Panda, R. (2011). Expert system design for classification of brain waves and epileptic-seizure detection. IEEE Technology Students' Symposium. DOI: 10.1109/TECHSYM.2011.5783822.

[2] Huff JS, Murr N. Seizure. 2023 Feb 7. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2024 Jan–. PMID: 28613516.

[3] Kobau R, Luncheon C, Greenlund K. Active epilepsy prevalence among U.S. adults is 1.1% and differs by educational level-National Health Interview Survey, United States, 2021. Epilepsy Behav. 2023 May;142:109180. doi: 10.1016/j.yebeh.2023.109180. Epub 2023 Apr 7. PMID: 37031584.

[4] **ttps://www.hopkinsmedicine.org/health/conditions-and-diseases/epilepsy/types-of-seizures#:~:text=Focal%20onset%20seizures%20start%20in,both%20sides%20of%20the%20brain.**

[5] Slimen, I. B., Boubchir, L., and Seddik, H., 2020, "Epileptic Seizure Prediction Based on EEG Spikes Detection of Ictal-Preictal States," Journal of Biomedical Research, **34**(3) pp. 162-169.

[6] Kennedy GM, Lhatoo SD. CNS adverse events associated with antiepileptic drugs. CNS Drugs. 2008;22(9):739-60. doi: 10.2165/00023210-200822090-00003. PMID: 18698874.

[7] Kedare JS, Baliga SP. Management of Psychiatric Disorders in Patients of Epilepsy. Indian J Psychiatry. 2022 Mar;64(Suppl 2):S319-S329. doi: 10.4103/indianjpsychiatry.indianjpsychiatry_17_22. Epub 2022 Mar 23. PMID: 35602355; PMCID: PMC9122168

[8] Novak A, Vizjak K, Rakusa M. Cognitive Impairment in People with Epilepsy. J Clin Med. 2022 Jan 5;11(1):267. doi: 10.3390/jcm11010267. PMID: 35012007; PMCID: PMC8746065.

[9] Berg AT. Risk of recurrence after a first unprovoked seizure. Epilepsia. 2008;49 Suppl 1:13-8. doi: 10.1111/j.1528-1167.2008.01444.x. PMID: 18184149.

[10] Kwan, P., Schachter, S., and Brodie, M., 2011, "Drug-Resistant Epilepsy," The New England Journal of Medicine, 365pp. 919-26.

[11] Pathak, S.J., Yousaf, M.I.K., and Shah, V.B., 2023, "StatPearls,"StatPearls Publishing, Treasure Island (FL), .

[12] Schindler, K., Leung, H., Lehnertz, K., 2007, "How Generalised are Secondarily “generalised” Tonic–clonic Seizures?" Journal of Neurology, Neurosurgery, and Psychiatry, 78(9) pp. 993-996.

[13] Das CP, Sawhney IM, Lal V, Prabhakar S. Risk of recurrence of seizures following single unprovoked idiopathic seizure. Neurol India. 2000 Dec;48(4):357-60. PMID: 11146601.

[14] Shinnar S, Berg AT, Moshe SL, O'Dell C, Alemany M, Newstein D, Kang H, Goldensohn ES, Hauser WA. The risk of seizure recurrence after a first unprovoked afebrile seizure in childhood: an extended follow-up. Pediatrics. 1996 Aug;98(2 Pt 1):216-25. PMID: 8692621.

[15] Tang, J., El Atrache, R., Yu, S., 2021, "Seizure Detection using Wearable Sensors and Machine Learning: Setting a Benchmark," Epilepsia, 62(8) pp. 1807-1819.

[16] Subbarao BS, Silverman A, Eapen BC. Seizure Medications. 2023 Jul 10. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2024 Jan–. PMID: 29489165.

[17] Binnie CD. Vagus nerve stimulation for epilepsy: a review. Seizure. 2000 Apr;9(3):161-9. doi: 10.1053/seiz.1999.0354. PMID: 10775511.

[18] Ryvlin P, Jehi LE. Neuromodulation for Refractory Epilepsy. Epilepsy Currents. 2022;22(1):11-17. doi:10.1177/15357597211065587

[19] Yamakawa T, Miyajima M, Fujiwara K, Kano M, Suzuki Y, Watanabe Y, Watanabe S, Hoshida T, Inaji M, Maehara T. Wearable Epileptic Seizure Prediction System with Machine-Learning-Based Anomaly Detection of Heart Rate Variability. Sensors (Basel). 2020 Jul 17;20(14):3987. doi: 10.3390/s20143987. PMID: 32709064; PMCID: PMC7411877.

[20] Rashed-Al-Mahfuz M, Moni MA, Uddin S, Alyami SA, Summers MA, Eapen V. A Deep Convolutional Neural Network Method to Detect Seizures and Characteristic Frequencies Using Epileptic Electroencephalogram (EEG) Data. IEEE J Transl Eng Health Med. 2021 Jan 11;9:2000112. doi: 10.1109/JTEHM.2021.3050925. PMID: 33542859; PMCID: PMC7851059.

[21] Dissanayake T, Fernando T, Denman S, Sridharan S, Fookes C. Geometric Deep Learning for Subject Independent Epileptic Seizure Prediction Using Scalp EEG Signals. IEEE J Biomed Health Inform. 2022 Feb;26(2):527-538. doi: 10.1109/JBHI.2021.3100297. Epub 2022 Feb 4. PMID: 34314363.

[22] Tsiouris KM, Pezoulas VC, Zervakis M, Konitsiotis S, Koutsouris DD, Fotiadis DI. A Long Short-Term Memory deep learning network for the prediction of epileptic seizures using EEG signals. Comput Biol Med. 2018 Aug 1;99:24-37. doi: 10.1016/j.compbiomed.2018.05.019. Epub 2018 May 17. PMID: 29807250.

[23] Shoeibi A, Khodatars M, Ghassemi N, Jafari M, Moridian P, Alizadehsani R, Panahiazar M, Khozeimeh F, Zare A, Hosseini-Nejad H, Khosravi A, Atiya AF, Aminshahidi D, Hussain S, Rouhani M, Nahavandi S, Acharya UR. Epileptic Seizures Detection Using Deep Learning Techniques: A Review. Int J Environ Res Public Health. 2021 May 27;18(11):5780. doi: 10.3390/ijerph18115780. PMID: 34072232; PMCID: PMC8199071.

[24] Geng M, Zhou W, Liu G, Li C, Zhang Y. Epileptic Seizure Detection Based on Stockwell Transform and Bidirectional Long Short-Term Memory. IEEE Trans Neural Syst Rehabil Eng. 2020 Mar;28(3):573-580. doi: 10.1109/TNSRE.2020.2966290. Epub 2020 Jan 13. PMID: 31940545.

[25] Peng P, Xie L, Wei H. A Deep Fourier Neural Network for Seizure Prediction Using Convolutional Neural Network and Ratios of Spectral Power. Int J Neural Syst. 2021 Aug;31(8):2150022. doi: 10.1142/S0129065721500222. Epub 2021 May 7. PMID: 33970057.

[26] A. Narin (2022). "Detection of Focal and Non-focal Epileptic Seizure Using Continuous Wavelet Transform-Based Scalogram Images and Pre-trained Deep Neural Networks," IRBM, Volume 43, Issue 1,

Pages 22-31. ISSN 1959-0318. https://doi.org/10.1016/j.irbm.2020.11.002.

[27] O. Alshaltone et al., "Epileptic Seizure Detection using short time Fourier transform and Convolutional Neural Network," The 3rd International Conference on Distributed Sensing and Intelligent Systems (ICDSIS 2022), Hybrid Conference, Sharjah, United Arab Emirates, 2022, pp. 103-112, doi: 10.1049/icp.2022.2424.

[28] Usman SM, Usman M, Fong S. Epileptic Seizures Prediction Using Machine Learning Methods. Comput Math Methods Med. 2017;2017:9074759. doi: 10.1155/2017/9074759. Epub 2017 Dec 19. PMID: 29410700; PMCID: PMC5749318.

[29] Rashed-Al-Mahfuz M, Moni MA, Uddin S, Alyami SA, Summers MA, Eapen V. A Deep Convolutional Neural Network Method to Detect Seizures and Characteristic Frequencies Using Epileptic Electroencephalogram (EEG) Data. IEEE J Transl Eng Health Med. 2021 Jan 11;9:2000112. doi: 10.1109/JTEHM.2021.3050925. PMID: 33542859; PMCID: PMC7851059.

[30] Janjarasjitt S. Epileptic seizure classifications of single-channel scalp EEG data using wavelet-based features and SVM. Med Biol Eng Comput. 2017 Oct;55(10):1743-1761. doi: 10.1007/s11517-017-1613-2. Epub 2017 Feb 13. PMID: 28194648.

[31] Gopal Chandra Jana, Ratna Sharma, Anupam Agrawal (2020). "A 1D-CNN-Spectrogram Based Approach for Seizure Detection from EEG Signal," Procedia Computer Science, Volume 167, Pages 403-412. ISSN 1877-0509. https://doi.org/10.1016/j.procs.2020.03.248.

[32] https://patents.google.com/patent/US10743809B1/en?q=(seizure+detection)&oq=seizure+detection

[33] https://patents.google.com/patent/US11219405B2/en?q=(seizure+detection)&oq=seizure+detection

[34] https://patents.google.com/patent/US11672431B2/en?q=(seizure+detection)&oq=seizure+detection

[35]https://patents.google.com/patent/US20200155829A1/en?q=(seizure+detection)&oq=seizure+detection

[36] Song K, Fang J, Zhang L, Chen F, Wan J, Xiong N. An Intelligent Epileptic Prediction System Based on Synchrosqueezed Wavelet Transform and Multi-Level Feature CNN for Smart Healthcare IoT. Sensors (Basel). 2022 Aug 27;22(17):6458. doi: 10.3390/s22176458. PMID: 36080916; PMCID: PMC9460721.

[37] L. Lechner et al., "Fine-tuning of pre-processing filters enables scalp-EEG based training of subcutaneous EEG models," 2023 IEEE 19th International Conference on Body Sensor Networks (BSN), Boston, MA, USA, 2023, pp. 1-4, doi: 10.1109/BSN58485.2023.10331106.

[38] Gao Y, Gao B, Chen Q, Liu J, Zhang Y. Deep Convolutional Neural Network-Based Epileptic Electroencephalogram (EEG) Signal Classification. Front Neurol. 2020 May 22;11:375. doi: 10.3389/fneur.2020.00375. PMID: 32528398; PMCID: PMC7257380.

[39] Fang, Zhou & Leung, Howan & Choy, Chiu. (2018). Spatial temporal GRU convnets for vision-based real time epileptic seizure detection. 1026-1029. 10.1109/ISBI.2018.8363746.

[40]]Khan, M. H., Khan, Y. U., Sarfraz, M., and Al Maathidi, M. M., “A fast and novel deep learning approach for automatic classification of epileptic seizures using spectrograms”, in American Institute of Physics Conference Series, 2023, vol. 3015, no. 1. doi:10.1063/5.0188341.

[41] Rasheed K, Qadir J, O'Brien TJ, Kuhlmann L, Razi A. A Generative Model to Synthesize EEG Data for Epileptic Seizure Prediction. IEEE Trans Neural Syst Rehabil Eng. 2021;29:2322-2332. doi: 10.1109/TNSRE.2021.3125023. Epub 2021 Nov 10. PMID: 34727036; PMCID: PMC8592500.

[42] Dalic L, Cook MJ. Managing drug-resistant epilepsy: challenges and solutions. Neuropsychiatr Dis Treat. 2016 Oct 12;12:2605-2616. doi: 10.2147/NDT.S84852. PMID: 27789949; PMCID: PMC5068473.

[43] Detti, P. (2020). Siena Scalp EEG Database (version 1.0.0). PhysioNet. https://doi.org/10.13026/5d4a-j060.

[44] Dastgoshadeh M, Rabiei Z. Detection of epileptic seizures through EEG signals using entropy features and ensemble learning. Front Hum Neurosci. 2023 Feb 1;16:1084061. doi: 10.3389/fnhum.2022.1084061. PMID: 36875740; PMCID: PMC9976189.

Appendix

PSDs + Subbands:

```
addpath("C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Patient Data\Clean_Data")
%edf to array
tt = edfread('PN00-1.ICA4.edf');
info = edfinfo('PN00-1.ICA4.edf');
X = cell2mat(table2array(tt));
fs = 512;
t0 = (1:length(X))./fs;
X2 = X(1144*fs+1:1214*fs, 1:29);
%%
% time parameters
[X1, X2, X3, X4] = get_Times(X, 50, 60, 1150, 1160, 1134, 1144, 1215, 1225, fs);
%%
%Parameters%
noverlap = fs/16;
window0 = 2*noverlap;
window = hamming(window0);
nfft2 = noverlap*2;
figure;
[pxx, f] = pwelch(X2, window,noverlap,nfft2,fs);
plot(f, 10*log10(pxx), 'LineWidth', 2);
xlim([2 90]);
legend(info.SignalLabels,'Location', 'bestoutside','Fontsize', 14);
ylabel('Power Spectral Density (\muvolt^2 / Hz )'); xlabel('Frequency (Hz)');
title("Power Spectral Density of Ictal Event")
set(gca, 'fontsize', 20)
%%
figure;
hold on;
for i = 1:length(X2)
   [pxx, f] = pwelch(i, window,noverlap,nfft2,fs);
   plot(f, 10*log10(pxx), 'LineWidth', 2);
end
hold off;
%%
[pxx1, f1] = pwelch(X1, window,noverlap,nfft2,fs);
[pxx2, f2] = pwelch(X2, window,noverlap,nfft2,fs);
[pxx3, f3] = pwelch(X3, window,noverlap,nfft2,fs);
[pxx4, f4] = pwelch(X4, window,noverlap,nfft2,fs);
figure;
plot(f1, 10*log10(pxx1), 'LineWidth', 5, 'DisplayName', 'Inter-ictal'); hold on;
plot(f3, 10*log10(pxx3), 'LineWidth', 5, 'DisplayName','Pre-ictal'); hold on;
plot(f2, 10*log10(pxx2), 'LineWidth', 5, 'DisplayName','Ictal'); hold on;
plot(f4, 10*log10(pxx4), 'LineWidth', 5, 'DisplayName','Post-ictal'); hold on;
ylabel('Power Spectral Density (\muvolt^2 / Hz )'); xlabel('Frequency (Hz)');
title("Power Spectral Density of Stages of Focal Epilepsy")
set(gca, 'fontsize', 20)
legend('Location', 'Northwest','Fontsize', 20);
xlim([5 90])
%% Differences in Power based on subbands Inter vs. Ictal
tt = edfread('PN00-1.ICA4.edf','SelectedSignals', 'T4', 'DataRecordOutputType','vector');
X = cell2mat(table2array(tt));
fs=512;
[X1,X2,X3,X4] = get_Times(X, 90, 100, 1180, 1190, 844, 854, 1514, 1524, fs);
delta = [1 4];
theta = [4 8];
alpha = [8 12];
beta = [12 30];
gamma = [30 90];
d1 = bandpower(X2, fs, delta) - bandpower(X1, fs, delta);
d2 = bandpower(X3, fs, delta) - bandpower(X1, fs, delta);
d3 = bandpower(X4, fs, delta) - bandpower(X1, fs, delta);
t1 = 10*log10(bandpower(X2, fs, theta) - bandpower(X1, fs, theta));
t2 = 10*log10(bandpower(X3, fs, theta) - bandpower(X1, fs, theta));
t3 = 10*log10(bandpower(X4, fs, theta) - bandpower(X1, fs, theta));
a1 = 10*log10(bandpower(X2, fs, alpha) - bandpower(X1, fs, alpha));
```

```
a2 = 10*log10(bandpower(X3, fs, alpha) - bandpower(X1, fs, alpha));
a3 = 10*log10(bandpower(X4, fs, alpha) - bandpower(X1, fs, alpha));
b1 = 10*log10(bandpower(X2, fs, beta) - bandpower(X1, fs, beta));
b2 = 10*log10(bandpower(X3, fs, beta) - bandpower(X1, fs, beta));
b3 = 10*log10(bandpower(X4, fs, beta) - bandpower(X1, fs, beta));
g1 = 10*log10(bandpower(X2, fs, gamma) - bandpower(X1, fs, gamma));
g2 = 10*log10(bandpower(X3, fs, gamma) - bandpower(X1, fs, gamma));
g3 = 10*log10(bandpower(X4, fs, gamma) - bandpower(X1, fs, gamma));
figure
x = ["Before" "Seizure" "After"];
y = [t2, a2, b2, g2; t1, a1, b1, g1; t3, a3, b3, g3];
b = bar(x,y);
legend({'Theta', 'Alpha', 'Beta', 'Gamma'},'Fontsize', 20);
title('Subband Power Differences in Seizure Events vs. Inter-ictal')
set(gca, 'fontsize', 15);
ylabel('Power Difference (dB)');
%%
b2 = bar(nexttile,x,y,'stacked');
b2(1).FaceColor = [0.9290 0.6940 0.1250];
b2(2).FaceColor = [0.8500 0.3250 0.0980];
b2(3).FaceColor = [0.4940 0.1840 0.5560];
ylabel('Power Spectral Density (\muvolt^2 / Hz )');
set(gca, 'fontsize', 15);
%% Alternate (simplified)
x1 = ["Delta (1 - 4)Hz" "Theta (4 - 8)Hz" "Alpha (8 - 12)Hz" "Beta (12 - 30)Hz" "Gamma (30 - 90)Hz"];
y1 = [d1 t1 a1 b1 g1];
figure;
bar(x1,y1);
title('Power Differences by Brain Wave Subbands (Seizure vs. Non-seizure)')
set(gca, 'fontsize', 15);
ylabel('Power Spectral Density (\muvolt^2 / Hz )');
%%
function [X1,X2,X3,X4] = get_Times(X, non_0, non_n, inter_0, inter_n, pre_0, pre_n, post_0, post_n, fs)
   X1 = X(non_0*fs+1:non_n*fs); % (no seizure)
   X2 = X(inter_0*fs+1:inter_n*fs); % inter-ictal
   X3 = X(pre_0*fs+1:pre_n*fs); % pre-ictal
   X4 = X(post_0*fs+1:post_n*fs); % post-ictal
end
```

Organizing and creating CWTs:

```
clc
clear all
%%
addpath('\Patient Data\Clean_Data\');
tt1 = edfread('PN00-1.ICA4.edf', 'DataRecordOutputType','vector');
tt2 = edfread('PN00-2.IAE.edf', 'DataRecordOutputType','vector');
tt3 = edfread('PN00-3.IAE.edf', 'DataRecordOutputType','vector');
tt4 = edfread('PN00-4.IAE.edf', 'DataRecordOutputType','vector');
tt5 = edfread('PN00-5.IAE.edf', 'DataRecordOutputType','vector');
tt6 = edfread('PN13-1.common.edf', 'DataRecordOutputType','vector');
tt7 = edfread('PN13-3.common.edf', 'DataRecordOutputType','vector');
tt8 = edfread('PN05-2.edf', 'DataRecordOutputType','vector');
tt9 = edfread('PN05-4.common.edf', 'DataRecordOutputType','vector');
%%
% Get Seizure Period in Dataset (please leave quotes)
r1 = '21.11.29'; % Registration Start Time
%skip Resistration End Time
s1 = '23.39.09'; % Seizure Start Time
s2 = '23.40.18'; % Seizure End Time
times(r1, s1, s2)
%%
% Get [Pre, Post]-ictal Bounds (You choose how long pre, post is) in seconds
seizure_start = 8860;
seizure_end = 8929;
length_pre_ictal = 60*30;
length_post_ictal = 60*1;
bounds(length_pre_ictal, length_post_ictal, seizure_start, seizure_end)
%%
% Ordering of Electrodes
%PN00
t_labels_PN = {'T5', 'T3', 'T4', 'T6'};
f_labels_PN = {'FP1', 'FP2', 'F9', 'F7', 'F3', 'FZ', 'F4', 'F8', 'F10', 'FC5', 'FC1', 'FC2', 'FC6'};
c_labels_PN = {'C3', 'CZ', 'C4', 'CP5', 'CP1', 'CP2', 'CP6'};
p_labels_PN00 = {'P3', 'PZ', 'P4'};
o_labels_PN = {'O1', 'O2'};
electrode_array_PN00 = {t_labels_PN, f_labels_PN, c_labels_PN, p_labels_PN00, o_labels_PN};
%PN 5 & 13
t_labels_PN = {'T5', 'T3', 'T4', 'T6'};
f_labels_PN = {'FP1', 'FP2', 'F9' 'F7', 'F3', 'FZ', 'F4', 'F8', 'F10', 'FC5', 'FC1', 'FC2', 'FC6'};
c_labels_PN = {'C3', 'CZ', 'C4', 'CP5', 'CP1', 'CP2', 'CP6'};
p_labels_PN5_13 = {'P9', 'P3', 'PZ', 'P4' 'P10'};
o_labels_PN = {'O1', 'O2'};
electrode_array_PN5_13 = {t_labels_PN, f_labels_PN, c_labels_PN, p_labels_PN5_13, o_labels_PN};
% Modifying data into structure w/ labels and splitting of seizure events`
fs = 512;
%%
PN00_1 = genStructure(tt1, electrode_array_PN00, 843, 1143, 1213, 1813, fs);
PN00_2 = genStructure(tt2, electrode_array_PN00, 920, 1220, 1274, 1874, fs);
PN00_3 = genStructure(tt3, electrode_array_PN00, 465, 765, 825, 1425, fs);
PN00_4 = genStructure(tt4, electrode_array_PN00, 706, 1006, 1080, 1680, fs);
PN00_5 = genStructure(tt5, electrode_array_PN00, 604, 904, 971, 1571, fs);
PN13_1 = genStructure(tt6, electrode_array_PN5_13, 6762, 7062, 7110, 7710, fs);
PN13_3 = genStructure(tt7, electrode_array_PN5_13, 7253, 7553, 7704, 8304, fs);
PN05_2 = genStructure(tt8, electrode_array_PN5_13, 6863, 7163, 7198, 7798, fs);
PN05_4 = genStructure(tt9, electrode_array_PN5_13, 3308, 3608, 3647, 4247, fs);
PN00_Data = {PN00_1,PN00_2,PN00_3,PN00_4,PN00_5};
dataset_labels_PN00 = {'PN00_1','PN00_2','PN00_3','PN00_4','PN00_5'};
Patient0 = overallstructure(PN00_Data,dataset_labels_PN00);
PN13_Data = {PN13_1,PN13_3};
PN05_Data = {PN05_2, PN05_4};
dataset_labels_PN13 = {'PN13_1','PN13_3'};
dataset_labels_PN05 = {'PN05_2', 'PN05_4'};
Patient13 = overallstructure(PN13_Data,dataset_labels_PN13);
Patient5 = overallstructure(PN05_Data,dataset_labels_PN05);
%
Patients = {Patient0,Patient5,Patient13};
Patient_labels = {'Patient 0','Patient 5','Patient 13'};
TrainingData = patient_structure(Patients, Patient_labels);
%
```

```
%%
clc
proj = matlab.project.rootProject;
[root, patientFiles, electrodes, seizure_events] = obtainFolders(TrainingData(3));
disp([root, patientFiles, electrodes, seizure_events])
%%
disp(patientFiles)
%% Creating Directory
clc
channel_directory(root, patientFiles, electrodes, seizure_events)
%% Generating Scalograms
clc
%CWT_spawner(TrainingData, root, patient, electrodes)
disp(TrainingData(1).patients(1).files(1).electrode_data)
%%
CWT_spawner_cheap(TrainingData(3).patients, root, patientFiles, electrodes, 1)
function CWT_spawner_cheap(EEGdata, masterFolder,patientFiles,electrodeFolder, percentage_removed)
 for n = 1:length(patientFiles)
   y = patientFiles{n};
   for m = 1:length(electrodeFolder)
       z = electrodeFolder{m};
       M = getMaxima(EEGdata(n).files(m).electrode_data);
       threshold = removeOutliers(M, percentage_removed);
       electrodeScalogram(EEGdata(n).files(m).electrode_data, masterFolder,y,z,threshold)
   end
 end
end
function electrodeScalogram(EEGdata,masterFolder,patientFiles,electrodeFolder,threshold)
imageRoot = fullfile(masterFolder,patientFiles,electrodeFolder);
Fs = 512;
for i = 1:length(EEGdata)
   events = EEGdata(i).events';
   t1 = 1;
   t2 = Fs*10;
   k=0;
   while t2 <= Fs*length(events)
       segment = events(t1:min(t2,length(events)));
       if i == 3
           t1 = t1 + Fs;
           t2 = t2 + Fs;
           k = k + 1;
           if length(segment) > .9*Fs

               [~,signalLength] = size(segment);
               fb = cwtfilterbank(SignalLength=signalLength,SamplingFrequency=512, ...
               VoicesPerOctave=12,FrequencyLimits=[2 100], Wavelet='bump');
               cfs = abs(fb.wt(segment));
               norm_cfs = (cfs - min(cfs))./max(threshold);
               powerLimit = (max(max(norm_cfs)));
               im = ind2rgb(round(rescale(cfs,0,255*powerLimit)),jet(128));
               imgLoc = fullfile(imageRoot,char(EEGdata(i).labels));
               imFileName = char(EEGdata(i).labels)+"_"+num2str(k)+".jpg";
               imwrite(imresize(im,[224 224]),fullfile(imgLoc,imFileName));
           end
       else
           t1 = t1 + Fs*10;
           t2 = t2 + Fs*10;
           k = k + 1;
           if length(segment) > 8*Fs
               [~,signalLength] = size(segment);
               fb = cwtfilterbank(SignalLength=signalLength,SamplingFrequency=512, ...
               VoicesPerOctave=12,FrequencyLimits=[2 100], Wavelet='bump');
               cfs = abs(fb.wt(segment));

               norm_cfs = (cfs - min(cfs))./max(threshold);
               powerLimit = (max(max(norm_cfs)));
               im = ind2rgb(round(rescale(cfs,0,255*powerLimit)),jet(128));
               imgLoc = fullfile(imageRoot,char(EEGdata(i).labels));
               imFileName = char(EEGdata(i).labels)+"_"+num2str(k)+".jpg";
               imwrite(imresize(im,[224 224]),fullfile(imgLoc,imFileName));
           else
           end
```

```
        end
    end
end
end
function CWT_spawner(EEGdata, masterFolder, patientFolder,patientFiles,electrodeFolder)
for i = 1:length(patientFolder)
 x = patientFolder{i};
 for n = 1:length(patientFiles)
    y = patientFiles{n};
    for m = 1:length(electrodeFolder)
        z = electrodeFolder{m};
        electrodeScalogram(EEGdata(i).patients(n).files(m).electrode_data, masterFolder, x,y,z)
    end
 end
end
end
function [root, patientFiles, electrodeFolder, seizure_events] = obtainFolders(EEGdata)
    root = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT2';
    %patientNum = getlabels(EEGdata);
    patientFiles = getlabels(EEGdata(1).patients);
    electrodeFolder = getlabels(EEGdata(1).patients(1).files);
    seizure_events = getlabels(EEGdata(1).patients(1).files(1).electrode_data);

end
function B = getlabels(s)
B = [];
    for i = 1:length(s)
        B{i} = s(i).labels;
    end
end
%%
%%
function channel_directory(masterFolder, patientFiles, ElectrodeFolder, eventsFolder)
 for n = 1:length(patientFiles)
    y = patientFiles{n};
    for p=1:length(ElectrodeFolder)
      z = ElectrodeFolder{p};
      for m = 1:length(eventsFolder)
          a = eventsFolder{m};
          mkdir(fullfile(masterFolder, y, z, a));
      end
    end
 end
end
function M = getMaxima(EEGdata)
n = 0;
Fs = 512;
M = [];
for i = 1:length(EEGdata)
    events = EEGdata(i).events';
    t1 = 1;
    t2 = Fs*10;
    k=1;

    while t2 <= Fs*length(events)
        segment = events(t1:min(t2,length(events)));
        t1 = t1 + Fs*10;
        t2 = t2 + Fs*10;
        k = k + 1;

        if length(segment) > 8*Fs
            n = n + 1;
            [~,signalLength] = size(segment);
            fb = cwtfilterbank(SignalLength=signalLength,VoicesPerOctave=12);
            cfs = abs(fb.wt(segment));
            M{n} = max(cfs);
        else
        end
    end
end
end
%%
```

```
function threshold = removeOutliers(max, percentage)
   A = cell2mat(max);
   x = 100 - percentage;
   y0 = prctile(A,[0,x]);
   threshold = A(A<y0(2));
end
%%
% FUNCTIONS
function times(r1, s1, s2)
   reg0 = datetime(r1, 'InputFormat', 'HH.mm.ss');
   sz0 = datetime(s1, 'InputFormat', 'HH.mm.ss');
   sz1 = datetime(s2, 'InputFormat', 'HH.mm.ss');
   t_onset = seconds(time(between(reg0,sz0)));
   length = seconds(time(between(sz0,sz1)));
   t_end = length + t_onset;
   fprintf('Seizure Range: \t %g - %g seconds \n\nSeizure Length: \t %g seconds \n\n ', t_onset,t_end,length)
end
function bounds(length_pre_ictal, length_post_ictal, seizure_start, seizure_end)
   pre_ictal_start = seizure_start - length_pre_ictal;
   post_ictal_end = seizure_end + length_post_ictal;
   fprintf('Pre-ictal Start: \t %g seconds \n\n Post-ictal End: \t %g seconds \n\n ',
pre_ictal_start,post_ictal_end)
end
% 5 groups of classification
function [inter0,pre,ictal,post, inter1] = get_Times(X, pre_ictal_start, ictal_start, ictal_end,
post_ictal_end, fs)
   inter0 = X(1 : (pre_ictal_start-1)*fs);                % (before seizure)
   pre = X(pre_ictal_start*fs : (ictal_start)*fs);        % pre-ictal
   ictal = X(ictal_start*fs+1 : ictal_end*fs);            % ictal
   post = X(ictal_end*fs+1 : post_ictal_end*fs);          % post-ictal
   inter1 = X(post_ictal_end*fs+1: length(X));            % (after seizure)
   % where X = dataset
end
% Obtaining electrode data from Timetable
function A = EEGsubset(timetable, label_array)
   for i = 1:length(label_array)
       A{i} = cell2mat([timetable.(label_array{i}); i + 1]);
   end
end
% Converting electrode data into structure w/ labels
function structure = subset2structure(subset, label_array)
   for i = 1:length(subset)
       structure(i).electrode_data = subset{i};
       structure(i).labels = label_array{i};
   end
end
% proper naming
function structure = subset2structure2(subset, label_array)
   for i = 1:length(subset)
       structure(i).events = subset{i};
       structure(i).labels = label_array{i};
   end
end
% Master structure for data manipulation where seizure events are indexed
% and labeled
function patient_data = genStructure(tt, electrode_array, pre_ictal_start, ictal_start, ictal_end,
post_ictal_end, fs)
   t = EEGsubset(tt, electrode_array{1});
   temporal = subset2structure(t, electrode_array{1});

   f = EEGsubset(tt, electrode_array{2});
   frontal = subset2structure(f, electrode_array{2});
   c = EEGsubset(tt, electrode_array{3});
   central = subset2structure(c, electrode_array{3});
   p = EEGsubset(tt, electrode_array{4});
   parietal = subset2structure(p, electrode_array{4});

   o = EEGsubset(tt, electrode_array{5});
   occipital = subset2structure(o, electrode_array{5});
   patient_data = [temporal, frontal, central, parietal, occipital];
   for i = 1:length(patient_data)
       X = patient_data(i).electrode_data;
```

```
       [inter0, pre, ictal, post, inter1] = get_Times(X, pre_ictal_start, ictal_start, ictal_end,
post_ictal_end, fs);
       A = {inter0, pre, ictal, post, inter1};
       event_labels = {'Inter_ictal0','Pre_ictal','Ictal','Post_ictal','Inter_Ictal1'};
       patient_data(i).electrode_data = subset2structure2(A, event_labels);
   end
end
% Converting electrode data into structure w/ labels
function structure = patient_structure(subset, label_array)
   for i = 1:length(subset)
       structure(i).patients = subset{i};
       structure(i).labels = label_array{i};
   end
end
% Converting electrode data into structure w/ labels
function structure = overallstructure(subset, label_array)
   for i = 1:length(subset)
       structure(i).files = subset{i};
       structure(i).labels = label_array{i};
   end
End
```

CNN Training:

```
%% YOU CHOOSE PARAMETERS
clc
clear all
%%
events = {'Ictal', 'Inter_ictal0'};
electrodes = {'\T4', '\T6', '\F10', '\CP6', '\O2'};
root = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT3\Patient 0';
files = {'\PN00_1', '\PN00_2', '\PN00_3','\PN00_4','\PN00_5'};
fileLocations1 = getLocs(root, files, electrodes, events);
electrodes2 = {'\T3', '\T5', '\F9', '\CP5'};
root2 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT4\';
files2 = {'\PN13_1', '\PN13_3'};
fileLocations2 = getLocs(root2, files2, electrodes2, events);
electrodes3 = {'\T3', '\T5', '\F9', '\O2','\CP5'};
root3 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT4\';
files3 = {'\PN05_2', '\PN05_4'};
fileLocations3 = getLocs(root3, files3, electrodes3, events);
root4 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT4\';
files4 = {'\PN16_1', '\PN16_2'};
fileLocations4 = getLocs(root4, files4, electrodes3, events);
fileLocations = [fileLocations1, fileLocations2, fileLocations3, fileLocations4];
%% GROUPING IMAGE DATA
clc
allImages = imageDatastore(fileLocations, ...
   "IncludeSubfolders",true, ...
   "LabelSource",'foldernames');
%% RATIO BETWEEN SEIZURE / NO SEIZURE DATA
T = countEachLabel(allImages);
ns_to_s = T{2,2} / T{1,2}
% SPLITTING EVENLY
numIctal = T{1,2};
imdsS = splitEachLabel(allImages, numIctal, 'randomized', "Include",'Ictal');
imdsNS = splitEachLabel(allImages,(ns_to_s^-1)*round(ns_to_s),'randomized','Exclude','Ictal');
%%
subds1 = partition(imdsNS,5,1);
subds2 = partition(imdsNS,5,2);
subds3 = partition(imdsNS,5,3);
subds4 = partition(imdsNS,5,4);
subds = {subds1, subds2, subds3, subds4};
%%
%%
imds1 = imageDatastore(cat(1,imdsS.Files,subds{1}.Files));
imds1.Labels = cat(1,imdsS.Labels,subds{1}.Labels);
imds2 = imageDatastore(cat(1,imdsS.Files,subds{2}.Files));
imds2.Labels = cat(1,imdsS.Labels,subds{2}.Labels);
imds3 = imageDatastore(cat(1,imdsS.Files,subds{3}.Files));
imds3.Labels = cat(1,imdsS.Labels,subds{3}.Labels);
imds4 = imageDatastore(cat(1,imdsS.Files,subds{4}.Files));
imds4.Labels = cat(1,imdsS.Labels,subds{4}.Labels);
imds = {imds1, imds2, imds3, imds4};
%%
% TRAIN TEST SPLIT
imgsTrain={};
imgsValidation={};
for i = 1:length(imds)
   [imgsTrain{i},imgsValidation{i}] = splitEachLabel(imds{i},0.8,0.2,"randomized");
end
%%
%% TO RESIZE IMAGES
augimgsTrain = augmentedImageDatastore([299 299],imgsTrain);
augimgsValidation = augmentedImageDatastore([299 299],imgsValidation);
%% CALLING CNN
net = vgg16;
lgraph = layerGraph(net);
newDropoutLayer = dropoutLayer(0.65,"Name","new_Dropout");
lgraph = replaceLayer(lgraph,"drop7",newDropoutLayer);
numClasses = numel(categories(imgsTrain{1}.Labels));
newConnectedLayer = fullyConnectedLayer(numClasses,"Name","new_fc", ...
```

```
   "WeightLearnRateFactor",2,"BiasLearnRateFactor",2);
lgraph.Layers(end-4:end)
lgraph = replaceLayer(lgraph,"fc8",newConnectedLayer);
newClassLayer = classificationLayer("Name","new_classoutput");
lgraph = replaceLayer(lgraph,"output",newClassLayer);
lgraph.Layers(end-4:end)
   for j = 1:4

       miniBatchSize=32;
       valFreq = floor(numel(imgsTrain{j}.Files)/miniBatchSize);
       Epochs = 10;
       valPat =10;
       newDropoutLayer = dropoutLayer(0.65 + .1*j,"Name","new_Dropout");
       lgraph = replaceLayer(lgraph,"new_Dropout",newDropoutLayer);
       options = trainingOptions("sgdm", ...
           LearnRateSchedule = "piecewise",...
           LearnRateDropPeriod =2,...
           LearnRateDropFactor=.5,...
           shuffle="every-epoch",...
           MiniBatchSize=miniBatchSize, ...
           MaxEpochs=Epochs, ...
           InitialLearnRate=(5e-5), ...
           ValidationData=imgsValidation{j}, ...
           ValidationPatience=valPat,...
           ValidationFrequency=valFreq,  ...
           OutputNetwork="best-validation-loss",...
           Verbose=1);

       trainedVGG{i}= trainNetwork(imgsTrain{j},lgraph,options);
       lgraph = layerGraph(trainedVGG);
   end
%%
events = {'Ictal', 'Inter_ictal0'};
electrodes= {'\T3', '\T5', '\F9', '\CP5'};
root = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT4\';
files = {'\PN06_1'};
fileLocations1 = getLocs(root, files, electrodes, events);
testImages2 = imageDatastore(fileLocations1, ...
   "IncludeSubfolders",true, ...
   "LabelSource",'foldernames');
%%
T = countEachLabel(testImages2);
ns_to_s = T{2,2} / T{1,2};
% SPLITTING EVENLY
numIctal = T{1,2};
imdsS = splitEachLabel(testImages2, numIctal, 'randomized', "Include",'Ictal');
imdsNS = splitEachLabel(testImages2,numIctal,'randomized','Exclude','Ictal');
testImages = imageDatastore(cat(1,imdsS.Files,imdsNS.Files));
testImages.Labels = cat(1,imdsS.Labels,imdsNS.Labels);
%% IF NO STACKING
net = trainedVGG{4};
%%
testImages = testImages2;
[YPred, scores] = classify(net, testImages);
accuracy = sum(YPred == testImages.Labels)/numel(testImages.Labels);
disp(accuracy)
%%
hf = figure;
confusionchart(hf,testImages.Labels,YPred,"RowSummary","row-normalized","ColumnSummary","column-normalized")
%% STACKING
trainedNets = trainedVGG;
testImages2 = testImages;
YPred = {};
scores = {};
accuracy = {};
for i = 1:length(trainedNets)
   [YPred{i}, scores{i}] = classify(trainedNets{i}, testImages2);
   accuracy{i} = sum(YPred{i} == testImages2.Labels)/numel(testImages2.Labels);
   disp(accuracy{i})
end
int = {};
ic = {};
```

```
A = cell2mat(scores);
for i = 1:3
   int{i} = A(:,(2*i));
   ic{i} = A(:,(2*i-1));
end
inter_score = sum(cell2mat(int),2)';
ictal_score = sum(cell2mat(ic),2)';
YPred2 = categorical;
for j = 1:length(testImages2.Labels)
   if ictal_score(j) > inter_score(j)
       YPred2(j) = 'Ictal';
   else
       YPred2(j) = 'Inter_ictal0';
   end
end
accuracy3 = sum(YPred2' == testImages2.Labels)/numel(testImages2.Labels)
scores = cell2mat({ictal_score',inter_score'});
%% CONFUSION MATRIX
hf = figure;
confusionchart(hf,testImages2.Labels,YPred2,"RowSummary","row-normalized","ColumnSummary","column-normalized")
%%
classNames = trainedVGG{1}.Layers(end).Classes;
roc = rocmetrics(testImages2.Labels,scores2,classNames);
figure
plot(roc)
title("ROC Curve: VGG16")
%% ASSESSING INACCURACIES
figure
%YPred = YPred2';
k = 1;
for i = 1:numel(testImages2.Labels)
   if YPred(i) ~= testImages2.Labels(i)
       subplot(15,15,k)
       file = testImages2.Files{i,1};
       I = imread(testImages2.Files{i,1});
       imshow(I)
       label = YPred(i);
       title(string(label))
       k = k+1;
   else
   end
end
%%
aucVGG
%%
aucVGG4 = roc.AUC;
%%
aucVGG3 = roc.AUC;
aucVGG1 = roc.AUC;
aucVGG2 = roc.AUC;
%% PLOTTING AUC VALUES
figure
bar([aucVGG1; aucVGG2]')
xticklabels(classNames)
legend(["Non-Focal","Focal"],Location="southeast")
title("AUC")
%% MAJORITY VOTING
YPred2 = categorical(true);
for i = 1:length(testImages.Files)
   for j = 1:length(YPred)
       YPred2(i) = mode(YPred{j}(i,1));
   end
end
accuracy2 = sum(YPred2' == testImages.Labels)/numel(testImages.Labels)
%% FUNCTIONS
function folders = getFolders(root)
   s=dir(root);
   idx=~startsWith({s.name},'.') & [s.isdir];
   folders={s(idx).name};
end
function fileLocations = getLocs(root, files, electrodes, events)
k = 1;
```

```
for n = 1:length(files)
   file{n} = strcat(root, files{n});
   for i = 1:length(electrodes)
       eFile{i} = strcat(file{n}, electrodes{i});
       for j = 1:length(events)
           fileLocations{k} = fullfile(eFile{i}, events{j});
           k = k+1;
       end
   end
end
end
function retainIctal(i)
for i = 1:length(allImages.Labels)
   if allImages.Labels(i,1) ~= char('Ictal')
       allImages.Labels(i,1)=categorical(erase(string(allImages.Labels(i,1)), (string(allImages.Labels(i,1))))
...
           + repmat('Non-Ictal', size(allImages.Labels(i,1), 1), 1));
   else
   end
end
End
```

Topographical maps:

```
clc
load("C:\Users\Radi\MATLAB\Projects\Seizure
Forecasting\Visualizations\plot_topography\plot_topography\Standard_10-20_81ch.mat", 'locations');
%%
addpath('C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Visualizations\plot_topography')
T = readtable('polar_coordinates');
theta = table2array(T(:,2));
radius = table2array(T(:,3));
labels = {'Fp1','F3','C3','P3','O1','F7','T3',...
   'T5','Fc1','Fc5','Cp1','Cp5','F9','Fz','Cz','Pz',...
   'FP2','F4','C4','P4','O2','F8','T4','T6','Fc2','Fc6','Cp2','Cp6', 'F10'}';
labels0 = {'Fp1','F3','C3','P3','O1','F7','T3',...
   'Fc1','Fc5','Cp1','Cp5','F9','Fz','Cz','Pz',...
   'FP2','F4','C4','P4','O2','F8','T4','Fc2','Fc6','Cp2','Cp6', 'F10'}';
locations = table(labels, theta, radius);
%%
%% PREDICTION DATA
movie_root = "C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Visualizations\Movie Images\PN06-1.SZ";
filenames = "PN06-1.SZ";
net = trainedVGG;
labels = {'FC1','F3','C3','P3','O1','F7','T3',...
   'T5','FC1','FC5','CP1','CP5','F9','FZ','CZ','PZ',...
   'FP2','F4','C4','P4','O2','F8','T4','T6','FC2','FC6','CP2','CP6', 'F10'};
events = {'Ictal'};
root = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT4\';
files = {'\PN06_1'};
%
YPredo = {};
scoreso = {};
%
for i = 1:length(labels)
   idxer = {strcat('\',labels{i}) };
   fileLocations = getLocs(root, files, idxer, events);
   testImages = imageDatastore(fileLocations, ...
   "IncludeSubfolders",true, ...
   "LabelSource",'foldernames');
   [YPredo{i}, scoreso{i}] = classify(net, testImages);
end
ic = {};
A = cell2mat(scoreso);
for i = 1:29
  ic{i} = A(:,(2*i-1),:);
end
ictals = double(cell2mat(ic));
%% PREDICTION FRAMES
clc
loops = length(ictals);
F(loops) = struct('cdata',[],'colormap',[]);
for i = 1:loops
   chunk = ictals(i,:);
   h = plot_topography('all',chunk,0,locations,1,0,1000);
   F(i) = getframe(h);
   movieImg = frame2im(F(i));
   imFileName = filenames+"_"+num2str(i)+".png";
   imwrite(movieImg,fullfile(movie_root,imFileName));
end
%%
tt = edfread("C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Patient
Data\Clean_Data\Common2\PN06-1.common2.edf");
%%
X = cell2mat(table2array(tt));
fs = 512;
inter0 = cell(1,29);
pre = cell(1,29);
ictal = cell(1,29);
for i = 1:29
   col = X(:,i);
   [inter0{i}, pre{i}, ictal{i}, ~, ~] = get_Times(col, 7060, 8860, 8929, 8989, fs);
end
```

```
%%
movie_root = "C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Visualizations\Movie Images\PN06-1.SZ.actual";
filenames = "PN06-1.SZ.actual";
II = rescale(downsample(cell2mat(inter0), fs*2), -1, 1);
P = rescale(downsample(cell2mat(pre), fs*2), -1, 1);
SZ = rescale(downsample(cell2mat(ictal), fs), -1, 1);
loops = length(SZ);
F(loops) = struct('cdata',[],'colormap',[]);
for i = 1:loops
   chunk = SZ(i,:);
   h = plot_topography('all',chunk,0,locations,1,0,1000);
   F(i) = getframe(h);
   movieImg = frame2im(F(i));
   imFileName = filenames+"_"+num2str(i)+".png";
   imwrite(movieImg,fullfile(movie_root,imFileName));
end
function h = plot_topography(ch_list, values, make_contour, system, ...
   plot_channels, plot_clabels, INTERP_POINTS)
   % Error detection
   if nargin < 2, error('[plot_topography] Not enough parameters.');
   else
       if ~iscell(ch_list) && ~ischar(ch_list)
           error('[plot_topography] ch_list must be "all" or a cell array.');
       end
       if ~isnumeric(values)
           error('[plot_topography] values must be a numeric vector.');
       end
   end
   if nargin < 3, make_contour = false;
   else
       if make_contour~=1 && make_contour~=0
           error('[plot_topography] make_contour must be a boolean (true or false).');
       end
   end
   if nargin < 4, system = '10-20';
   else
       if ~ischar(system) && ~istable(system)
           error('[plot_topography] system must be a string or a table.');
       end
   end
   if nargin < 5, plot_channels = true;
   else
       if plot_channels~=1 && plot_channels~=0
           error('[plot_topography] plot_channels must be a boolean (true or false).');
       end
   end
   if nargin < 5, plot_clabels = false;
   else
       if plot_clabels~=1 && plot_clabels~=0
           error('[plot_topography] plot_clabels must be a boolean (true or false).');
       end
   end
   if nargin < 6, INTERP_POINTS = 1000;
   else
       if ~isnumeric(INTERP_POINTS)
           error('[plot_topography] N must be an integer.');
       else
           if mod(INTERP_POINTS,1) ~= 0
               error('[plot_topography] N must be an integer.');
           end
       end
   end

   % Loading electrode locations
   if ischar(system)
       switch system
           case '10-20'
               % 10-20 system
               load("C:\Users\RadiMATLAB\Projects\Seizure
Forecasting\Visualizations\plot_topography\plot_topography\Standard_10-20_81ch.mat", 'locations');
           case '10-10'
               % 10-10 system
```

```
                load('Standard_10-10_47ch.mat', 'locations');
            case 'yokogawa'
                % Yokogawa MEG system
                load('MEG_Yokogawa-440ag.mat', 'locations');
            otherwise
                % Custom path
                load(system, 'locations');
        end
    else
        % Custom table
        locations = system;
    end

    % Finding the desired electrodes
    ch_list = upper(ch_list);
    if ~iscell(ch_list)
        if strcmpi(ch_list,'all')
            idx = 1:length(locations.labels);
            if length(values) ~= length(idx)
                error('[plot_topography] There must be a value for each of the %i channels.', length(idx));
            end
        else, error('[plot_topography] ch_list must be "all" or a cell array.');
        end
    else
        if length(values) ~= length(ch_list)
            error('[plot_topography] values must have the same length as ch_list.');
        end
        idx = NaN(length(ch_list),1);
        for ch = 1:length(ch_list)
            if isempty(find(strcmp(locations.labels,ch_list{ch})))
                warning('[plot_topography] Cannot find the %s electrode.',ch_list{ch});
                ch_list{ch} = [];
                values(ch)  = [];
                idx(ch)     = [];
            else
                idx(ch) = find(strcmp(locations.labels,ch_list{ch}));
            end
        end
    end
    values = values(:);

       HEAD_RADIUS     = 5*2*0.511/8;  % 1/2  of the nasion-inion distance
    HEAD_EXTRA      = 1*2*0.511/8;  % 1/10 of the nasion-inion distance
    k = 4;                          % Number of nearest neighbors for interpolation

    % Interpolating input data
        % Creating the rectangle grid (-1,1)
        [ch_x, ch_y] = pol2cart((pi/180).*((-1).*locations.theta(idx)+90), ...
                                locations.radius(idx));     % X, Y channel coords
        % Points out of the head to reach more natural interpolation
        r_ext_points = 1.2;
        [add_x, add_y] = pol2cart(0:pi/4:7*pi/4,r_ext_points*ones(1,8));
        linear_grid = linspace(-r_ext_points,r_ext_points,INTERP_POINTS);         % Linear grid (-1,1)
        [interp_x, interp_y] = meshgrid(linear_grid, linear_grid);

        % Interpolate and create the mask
        outer_rho = max(locations.radius(idx));
        if outer_rho > HEAD_RADIUS, mask_radius = outer_rho + HEAD_EXTRA;
        else,                       mask_radius = HEAD_RADIUS;
        end
        mask = (sqrt(interp_x.^2 + interp_y.^2) <= mask_radius);
        add_values = compute_nearest_values([add_x(:), add_y(:)], [ch_x(:), ch_y(:)], values(:), k);
        interp_z = griddata([ch_x(:); add_x(:)], [ch_y(:); add_y(:)], [values; add_values(:)], interp_x,
interp_y, 'natural');
        interp_z(mask == 0) = NaN;
        % Plotting the final interpolation
        colormap("turbo")
        pcolor(interp_x, interp_y, interp_z);
        shading interp;
        hold on;

        % Contour
```

```
        if make_contour
            [~, hfigc] = contour(interp_x, interp_y, interp_z);
            set(hfigc, 'LineWidth',0.75, 'Color', [0.2 0.2 0.2]);
            hold on;
        end
    % Plotting the head limits as a circle
    head_rho    = HEAD_RADIUS;                      % Head radius
    if strcmp(system,'yokogawa'), head_rho = 0.45; end
    head_theta  = linspace(0,2*pi,INTERP_POINTS);   % From 0 to 360Â°
    head_x      = head_rho.*cos(head_theta);        % Cartesian X of the head
    head_y      = head_rho.*sin(head_theta);        % Cartesian Y of the head
    plot(head_x, head_y, 'Color', 'k', 'LineWidth',4);
    hold on;
    % Plotting the nose
    nt = 0.15;      % Half-nose width (in percentage of pi/2)
    nr = 0.22;      % Nose length (in radius units)
    nose_rho   = [head_rho, head_rho+head_rho*nr, head_rho];
    nose_theta = [(pi/2)+(nt*pi/2), pi/2, (pi/2)-(nt*pi/2)];
    nose_x     = nose_rho.*cos(nose_theta);
    nose_y     = nose_rho.*sin(nose_theta);
    plot(nose_x, nose_y, 'Color', 'k', 'LineWidth',4);
    hold on;
    % Plotting the ears as ellipses
    ellipse_a = 0.08;                               % Horizontal exentricity
    ellipse_b = 0.16;                               % Vertical exentricity
    ear_angle = 0.9*pi/8;                           % Mask angle
    offset    = 0.05*HEAD_RADIUS;                   % Ear offset
    ear_rho   = @(ear_theta) 1./(sqrt(((cos(ear_theta).^2)./(ellipse_a^2)) ...
        +((sin(ear_theta).^2)./(ellipse_b^2))));    % Ellipse formula in polar coords
    ear_theta_right = linspace(-pi/2-ear_angle,pi/2+ear_angle,INTERP_POINTS);
    ear_theta_left  = linspace(pi/2-ear_angle,3*pi/2+ear_angle,INTERP_POINTS);
    ear_x_right = ear_rho(ear_theta_right).*cos(ear_theta_right);
    ear_y_right = ear_rho(ear_theta_right).*sin(ear_theta_right);
    ear_x_left  = ear_rho(ear_theta_left).*cos(ear_theta_left);
    ear_y_left  = ear_rho(ear_theta_left).*sin(ear_theta_left);
    plot(ear_x_right+head_rho+offset, ear_y_right, 'Color', 'k', 'LineWidth',4); hold on;
    plot(ear_x_left-head_rho-offset, ear_y_left, 'Color', 'k', 'LineWidth',4); hold on;
    % Plotting the electrodes
    % [ch_x, ch_y] = pol2cart((pi/180).*(locations.theta(idx)+90), locations.radius(idx));
    if plot_channels, he = scatter(ch_x, ch_y, 60,'k', 'LineWidth',1.5); end
    if plot_clabels, text(ch_x, ch_y, ch_list); end
    if strcmp(system,'yokogawa'), delete(he); plot(ch_x, ch_y, '.k'); end

    % Last considerations
    max_height = max([max(nose_y), mask_radius]);
    min_height = -mask_radius;
    max_width  = max([max(ear_x_right+head_rho+offset), mask_radius]);
    min_width  = -max_width;
    L = max([min_height, max_height, min_width, max_width]);
    xlim([-L, L]);
    ylim([-L, L]);
    colorbar;   % Feel free to modify caxis after calling the function
    clim([0 1])
    axis square;
    axis off;
    hold off;
    h = gcf;
end
r
function add_val = compute_nearest_values(coor_add, coor_neigh, val_neigh, k)

    add_val = NaN(size(coor_add,1),1);
    L = length(add_val);

    for i = 1:L
        % Distances between the added electrode and the original ones
        target = repmat(coor_add(i,:),size(coor_neigh,1),1);
        d = sqrt(sum((target-coor_neigh).^2,2));

        % K-nearest neighbors
        [~, idx] = sort(d,'ascend');
        idx = idx(2:1+k);
```

```
        % Final value as the mean value of the k-nearest neighbors
        add_val(i) = mean(val_neigh(idx));
    end

end
function fileLocations = getLocs(root, files, electrodes, events)
k = 1;
for n = 1:length(files)
    file{n} = strcat(root, files{n});
    for i = 1:length(electrodes)
        eFile{i} = strcat(file{n}, electrodes{i});
        for j = 1:length(events)
            fileLocations{k} = fullfile(eFile{i}, events{j});
            k = k+1;
        end
    end
end
end
function [inter0,pre,ictal,post, inter1] = get_Times(X, pre_ictal_start, ictal_start, ictal_end,
post_ictal_end, fs)
    inter0 = X(1 : (pre_ictal_start-1)*fs);                 % (before seizure)
    pre = X(pre_ictal_start*fs : (ictal_start)*fs);         % pre-ictal
    ictal = X(ictal_start*fs+1 : ictal_end*fs);             % ictal
    post = X(ictal_end*fs+1 : post_ictal_end*fs);           % post-ictal
    inter1 = X(post_ictal_end*fs+1: length(X));             % (after seizure)
    % where X = dataset
end
```

Probability Distributions + Likelihood estimations:

```
net = trainedVGG;
events = {{'Inter_ictal0'},{'Pre_ictal'},{'Ictal'},{'Post_ictal'},{'Inter_ictal1'}};
root6 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT5';
root17 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT3';
root01 = 'C:\Users\Radi\MATLAB\Projects\Seizure Forecasting\Features\Sem2\CWT5';
files6 = {{'\PN06_1'}, {'\PN06_2'}, {'\PN06_3'}, {'\PN06_5'}};
files17 = {{'\PN17_1'}, {'\PN17_2'}};
files01 = {'\PN01_2'};
files12 = {{'\PN12_3'}, {'\PN17_4'}};
files13 = {{'\PN13_1'}, {'\PN13_2'}, {'\PN13_3'}};
files14 = {{'\PN14_1'}, {'\PN14_2'}, {'\PN14_4'}};
files09 = {{'\PN09_1'}, {'\PN09_2'}, {'\PN09_3'}};
files16 = {{'\PN16_1'}, {'\PN16_2'}};
files5 = {{'\PN05_2'}, {'\PN05_3'}, {'\PN05_4'}};
%%
events = {{'Inter_ictal0'},{'Pre_ictal'},{'Ictal'},{'Post_ictal'}};
data92 = getActivations(events,net,root01,files09{2}, 0.7);
%%
clc
root = root01;
files = files17{3};
threshold = 0.7;
labels = getFolders(strcat(root,char(files{1})));
YPredo = {};
scoreso = {};
signal = {};
for p = 1:3
   event = cell(events{p});
   for i = 1:length(labels)
       idxer = {strcat('\',labels{i})};
       fileLocations = getLocs(root, files, idxer, event);
       testImages = imageDatastore(fileLocations, ...
       "IncludeSubfolders",true, ...
       "LabelSource",'foldernames');
       ordered = natsort(testImages.Files);
       testImages.Files = ordered;
       [~, scoreso{i}] = classify(net, testImages);
   end

   ic = {};
   A = cell2mat(scoreso);

   for j = 1:29
      ic{j} = A(:,(2*j-1),:);
   end
   ictals = cell2mat(ic);
   ictals = gpuArray(ictals);

   flags = {};

   for q = 1:length(ictals)
       for b = 1:length(labels)
           if ictals(q,b) > 0.7
               flags{q,b} = 1;
           else
               flags{q,b} = 0;
           end
       end
   end

   signal173{p} = sum(cell2mat(flags),2);
end
%%
sz1 = 478;
sz2 = 886;
sz3 = 627;
pt64 = [signal64{1}; signal64{2}; signal64{3}];
figure;
plot(movmean(pt61,30))
```

```
%%
signal6 = {};
for i = 1:4
   signal6{i} = getActivations(events,net,root6,files6{i}, 0.8);
end
%%
net = trainedNet;
signal1 = getActivations(events,net,root01,files01, 0.8);
%%
r1 = '06.25.51'; % Registration Start Time
%skip Resistration End Time
s1 = '08.10.26'; % Seizure Start Time
s2 = '08.11.08'; % Seizure End Time
times2(r1, s1, s2, 8)
%%
a91 = [signal091{1}; signal091{2}];
figure;
plot(movmean(a91',30))
%%
Data09 = [signal091{1}, signal091{2}; signal092{1}, signal091{2}; signal093{1}, signal093{2}];
Data01 = [signal012{1}, signal012{2}];
Data14 = [signal142{1}, signal142{2}; signal144{1}, signal144{2}];
Data17 = [signal171{1}; signal171{2}; signal172{1}, signal172{2}];
Data06 = [signal6{1}, signal6{2};signal62{1}, signal62{2};signal63{1}, signal63{2};signal64{1}, signal64{2}];
%%
Data14 = [cell2mat(signal142'); cell2mat([signal144{1}; signal144{2}]')]
%%
figure;
plot([signal144{1}; signal144{2}])
%%
Data14 = [signal144{1}; signal144{2}]';
%% CREATING AND PLOTTING DISTRIBUTIONS
pt6 = [signal6{1,1};signal6{1,2};signal6{1,3};signal6{1,4}];
%%
flags = [flags1T+flags2T];
figure;
sgtitle('Two flagging regimes (PN09-2)',FontSize=16)
subplot(311)
plot(flags1T, LineWidth=4);
xline(712,'-r', 'Seizure Onset', LineWidth=3)
xline(712-180, '--k', '30 min before Seizure', LineWidth=3)
subplot(312)
plot(flags2T, LineWidth=4);
xline(712,'-r', 'Seizure Onset', LineWidth=3)
xline(712-180, '--k', '30 min before Seizure', LineWidth=3)
subplot(313)
plot(flags, LineWidth=4);
xline(712,'-r', 'Seizure Onset', LineWidth=3)
xline(712-180, '--k', '30 min before Seizure', LineWidth=3)
xlabel('Frames (10 sec)')
%%
A = [signal1{1}; signal1{2}];
B = movmean(A, 60);
figure;
plot(B)
%%
sigs2 = OverallMovemean(pt6, 60);
A1 = get_Times(sigs2{1}, 378, 516, 558, 622, 682);
A2 = get_Times(sigs2{2}, 706, 844, 886, 955, 1015);
A3 = get_Times(sigs2{3}, 447, 585, 627, 669, 729);
A4 = get_Times(sigs2{4}, 298, 436, 478, 522, 582);
signal6events = [A1;A2;A3;A4];
[II0, Pre1, Pre2, Seizure, Post, II1] = obtainEvents(signal6events);
pd_II0 = fitdist(II0,'Normal');
pd_Pre1 = fitdist(Pre1,'Normal');
pd_Pre2 = fitdist(Pre2,'Normal');
pd_SZ = fitdist(Seizure,'Normal');
pd_Po = fitdist(Post,'Normal');
pd_II1 = fitdist(II1,'Normal');
x_values = 0:.01:29;
   pdf_II0 = pdf(pd_II0, x_values);
   pdf_Pre1 = pdf(pd_Pre1, x_values);
```

```
    pdf_Pre2 = pdf(pd_Pre2, x_values);
    pdf_SZ = pdf(pd_SZ, x_values);
    pdf_Po = pdf(pd_Po, x_values);
    pdf_II1 = pdf(pd_II1, x_values);


figure;
subplot(3,1,1)
plot(x_values, pdf_II0, 'black', "LineWidth",3)
hold on
plot(x_values, pdf_Pre1, 'yellow', "LineWidth",3)
hold on
plot(x_values, pdf_Pre2, 'blue', "LineWidth",3)
hold on
plot(x_values, pdf_SZ ,'red', "LineWidth",3)
hold off
legend('Interictal (30 min away from Seizure)', "Pre-Ictal (30 min before Seizure)", "10 min before
Seizure","Seizure", 'FontSize', 14);
ylabel("Probability");
title('PDFs of Seizure Events (Patients 06 + 17)');
subplot(3,1,2)
histfit(II0, 20)
hold on
histfit(Pre1, 20)
hold on
histfit(Pre2, 20)
hold on
histfit(Seizure, 20)
hold off
subplot(3,1,3)
histfit(II0, 100, 'kernel')
hold on
histfit(Pre1, 100, 'kernel')
hold on
histfit(Pre2, 100, 'kernel')
hold on
histfit(Seizure, 100, 'kernel')
xlabel('Flagged Electrodes');
hold off
%%
figure;
Y = fft(M);
Fs = 1;
L = length(M);
plot(Fs/L*(-L/2:L/2-1),abs(fftshift(Y)),"LineWidth",3)
title("fft Spectrum in the Positive and Negative Frequencies")
xlabel("f (Hz)")
ylabel("|fft(X)|")
%% PLOTTING LENGTH OF SEIZURES ----NEW NET
sigs2 = OverallMovemean(pt6, 30);
M = [sigs2{1}, sigs2{2}, sigs2{3},sigs2{4}];
t = 1:length(M);
figure;
plot(t,M)
ylim([0 15])
yline(4.32)
yline(6.75)
yline(8.31)
yline(10.6)
xline(558, LineWidth=2)
xline(622, LineWidth=2)
xline(length(sigs2{1})+886, LineWidth=2)
xline(length(sigs2{1})+995, LineWidth=2)
xline(length([sigs2{1},sigs2{2}])+628, LineWidth=2)
xline(length([sigs2{1},sigs2{2}])+670, LineWidth=2)
A = length(M) - length(sigs2{4});
xline(A+478, LineWidth=2)
xline(A+522, LineWidth=2)
hold on
plot(M,'LineWidth',2)
yregion(0,4.32,"FaceColor","#0072BD");
yregion(4.32,6.75,"FaceColor","#EDB120");
```

```
yregion(6.75,8.31,"FaceColor","#D95319");
yregion(8.31,10.6,"FaceColor",'r');
yregion(10.6,29,"FaceColor","#A2142F");
xlabel('time')
ylabel('Flagged Electrodes')
title('Patient 6 Seizures')
hold off
%% PLOTTING ONE SEIZURE
M = sigs2{2};
t = 1:length(M);
figure;
yline(4.32)
yline(6.75)
yline(8.31)
yline(10.6)
xline(886, LineWidth=2)
xline(995, LineWidth=2)
hold on
plot(t,M,'black','LineWidth',2)
yregion(0,4.32,"FaceColor","#0072BD");
yregion(4.32,6.75,"FaceColor","#EDB120");
yregion(6.75,8.31,"FaceColor","#D95319");
yregion(8.31,10.6,"FaceColor",'r');
yregion(10.6,29,"FaceColor","#A2142F");
xlabel('time')
ylabel('Flagged Electrodes')
title('Patient 06-2 Seizure')
%% OBTAINING DATA SET FOR REGRESSOR
PN061 = [signal6{1}{1};signal6{1}{2}];
PN062 = [signal6{2}{1};signal6{2}{2}];
PN063 = [signal6{3}{1};signal6{3}{2}];
PN065 = [signal6{4}{1};signal6{4}{2}];
Data1 = {PN061;PN062;PN063;PN065};
Data1{1} = paddata(flip(Data1{1}),886);
Data1{3} = paddata(flip(Data1{3}),886);
Data1{4} = paddata(flip(Data1{4}),886);
Data2 = {};
for i = 1:4
   Data2{i} = movmean(Data1{i},60);
end
%% PLOTTING THE EVENTS LEADING TO SEIZURE
figure;
t = length(Data2{1}):-1:1;
plot(t,Data2{4})
hold on
plot(t,Data2{3})
hold on
plot(t,Data2{2})
hold on
plot(t,Data2{1})
%% OBTAINING LIKELIHOOD DATA
clc
range = length(Data2{1});
SZVsPre1T = zeros(4,range);
pre2Vspre1T = zeros(4,range);
SZVsIIT = zeros(4,range);
for i = 1:4
   X = Data2{i};
   for j = 1:length(X)
       %SZVsPre1T(i,j) = log(pdf(pd_SZ,X(j))./pdf(pd_Pre1, X(j)));
       %pre2Vspre1T(i,j) = log(pdf(pd_Pre2, X(j))./pdf(pd_Pre1,X(j)));

       SZVsIIT(i,j) = log(pdf(pd_SZ,X(j))./pdf(pd_II0, X(j)));
   end
end
%% PLOTTING AND FITTING LINEAR MODEL
len = 60;
t1 = 1:61;
Mean = mean(SZVsIIT,1);
data1= Mean(1:61);
figure;
scatter(data1,log(t1))
```

```
%[p, gof] = fitlm(Mean', log(t)', 'poly3','Normalize','on','Robust','LAR');
mdl1 = fitlm(data1', log(t1)', 'RobustOpts','on');
hold on
plot(mdl1)
hold on
%%
figure;
t2 = 60:180;
data2= Mean(60:180);
scatter(data2,t2)
%[p, gof] = fitlm(Mean', log(t)', 'poly3','Normalize','on','Robust','LAR');
mdl2 = fitlm(data2', t2', 'RobustOpts','on');
hold on
plot(mdl2)
%% DOUBLE Y AXIS PLOT OF LIKELIHOODS AND ELECTRODES BASED ON TIME TO ONSET
figure;
for i = 1:4
   x = Data2{i};
   t = 1:length(x);
   yyaxis left
   SZVsPre1 = pdf(pd_SZ,x)./pdf(pd_Pre1, x);
   pre2Vspre1 = pdf(pd_Pre2, x)./pdf(pd_Pre1,x);
   SZVsII = pdf(pd_SZ,x)./pdf(pd_II0, x);
   scatter(x,t)
   title('30 minutes To Seizure')
   ylabel('Time to Onset')
   xlabel('# of Flagged Electrodes')
   hold on

   hold on
   yyaxis right

   scatter(x, SZVsPre1, 'r')
   hold on
   yline(0.9)
   scatter(x, pre2Vspre1,'black')
   hold on
   scatter(x, SZVsII, 'b')
   legend('Seizure vs. Pre-Ictal', "Seizure vs. Interictal",'Location','northwest');
   hold on
   ylabel('Ratio of Likelihood')
   yscale log

end
hold off
%%
figure;
for i = 1:4
   x = Data2{i};
   t = 1:length(x);
   SZVsII = log(pdf(pd_SZ,x)./pdf(pd_II0, x));

   scatter(SZVsII, log(t))
   hold on
   xlabel('Ratio of Likelihoods')
   ylabel('Time from Seizure Onset (10sec chunks)')
   %yscale log
   title("Basis of Regressor")
end
%% PLOTTING MODEL BACK ONTO SEIZURE FILES
sigs3 = OverallMovemean(pt6, 30);
%sigs2 = OverallMovemean(pt6, 90);
M = sigs3{1};
t = 1:length(M);
figure;
subplot(2,1,1)
plot(t,M)
%xline(length(sigs2{1})+886)
%xline(length(sigs2{1})+995)
%xline(length([sigs2{1},sigs2{2}])+628)
%xline(length([sigs2{1},sigs2{2}])+670)
A = length(M) - length(sigs2{4});
```

```
%xline(A+478)
%xline(A+522)
yline(4.32)
yline(6.75)
yline(8.31)
yline(10.6)
xline(558, LineWidth=2)
xline(622, LineWidth=2)
ylim([0 5])
hold on
plot(t,M,'black','LineWidth',2)
yregion(0,4.32,"FaceColor","#0072BD");
yregion(4.32,6.75,"FaceColor","#EDB120");
yregion(6.75,8.31,"FaceColor","#D95319");
yregion(8.31,10.6,"FaceColor",'r');
yregion(10.6,29,"FaceColor","#A2142F");
subplot(2,1,2)
X = M;
R3= log(pdf(pd_SZ,X)./pdf(pd_II0, X));
ypred3 = predict(mdl1, R3');
hold on
plot(ypred3)
xline(558, LineWidth=2)
ylim([-50 50])
yline(-5)
yline(200)
yline(0)
ylabel('Time To Onset')
xlabel('log(Ratio of Likelihoods)')
%%
figure;
plot(558-ypred1,R1)
%%
ypred2 = predict(model2, R2');
hold on
plot(t+50,ypred2)
ypred3 = predict(model3, R3');
hold on
plot(t+50,ypred3)
%plot(poly2, t, R2, 'b')
%yscale log
%xlim([0 200])
%%
%%
X = sigs3{1}(500);
R1 = log(pdf(pd_SZ,X)./pdf(pd_Pre1, X));
predict(model1, R1')
%%
Xx = signals06_3{1};
R1 = log(pdf(pd_SZ,Xx)./pdf(pd_Pre1, Xx));
yf = predict(model1, R1', 50);
figure;
plot(R1, yf)
%%
p = 3;
PriorMdl = bayeslm(p,'ModelType','semiconjugate','VarNames',["R1" "R2" "R3"]);
fhs  = 10;
X = [data1;data2;data3];
x = X(:,1:(end-fhs))';
xf = X(:,(end - fhs + 1):end)';
y = 1:90-fhs;
yft = 50:90;
PosteriorMdl = estimate(PriorMdl,x,y,'Display',false);
%%
figure;
t = 90:-1:1;
plot(X,t)
hold on
plot(xf,yF)
%%
tt = 1:10;
[yF,YFCov] = forecast(PosteriorMdl,xf);
```

```
cil = yF - norminv(0.975)*sqrt(diag(YFCov));
ciu = yF + norminv(0.975)*sqrt(diag(YFCov));
figure;
hold on
h=gca;
plot(xf,yF)
plot(tt,[cil ciu],'k--')
hp = patch([X(end - fhs + 1) X(end) X(end) X(end - fhs + 1)],...
   h.YLim([1,1,2,2]),[0.8 0.8 0.8]);
uistack(hp,'bottom');
legend('Forecast horizon','True GNPR','Forecasted GNPR',...
   'Credible interval','Location','NW')
title('Real Gross National Product: 1909 - 1970');
ylabel('rGNP');
xlabel('Year');
hold off
%%
predict(exp1,1)
%%
f1 = fitlm(t',data1');
f2 = fitlm(t',data2');
%%
X = zeros(1,90);
X = signals06{1};
%%
clc
figure;
for i = 1:4
   x = signals06_1{i};
   preVsSZ = log(pdf(pd_Pre, x)./pdf(pd_SZ,x));
   preVsII = log(pdf(pd_II0, x)./pdf(pd_Pre,x));
   SZVsII = log(pdf(pd_II0, x)./pdf(pd_SZ,x));

       plot(x, preVsII, 'black')
       hold on
       plot(x, preVsSZ, 'blue')
       hold on
       plot(x, SZVsII,'red')
       hold on
       ylabel("Ratio of Likelihood")
       xlabel("# of Flagging Electrodes")
       hold on
end
hold off
%%
figure;
plot(signals06)
%%
figure;
plot(signal11);
hold on
plot(signal21)
hold on
plot(signal31)
hold on
plot(signal41)
hold off
%%
clc
sigs2 = OverallMovemean(pt6, 60);
%%
cell2mat(sigs2{1}')
%%
ind = 1:length()
t1 = find(pdf(pd_Pre,x)==pdf(pd_SZ,x));
disp(find(preVsSZ==1))
disp(find(SZVsII==1))
%%
figure;
subplot(3,1,1)
plot(x_values, pdf_II0, 'black', "LineWidth",3)
hold on
```

```
plot(x_values, pdf_Pre, 'blue', "LineWidth",3)
hold on
plot(x_values, pdf_SZ ,'red', "LineWidth",3)
hold off
legend('Interictal (30 min away from Seizure)', "Pre-Ictal (30 min before Seizure)", "Seizure", 'FontSize',
14);
ylabel("Probability");
title('PDFs of Seizure Events (Patients 06 + 17)');
subplot(3,1,2)
histfit(II0)
hold on
histfit(Pre)
hold on
histfit(Seizure)
hold off
subplot(3,1,3)
histfit(II0, 100, 'kernel')
hold on
histfit(Pre, 100, 'kernel')
hold on
histfit(Seizure, 100, 'kernel')
xlabel('Flagged Electrodes');
hold off
%% NEW DIST
%%
plot(x_values, pdf_Po, 'yellow')
hold on
plot(x_values, pdf_II1, 'black')
hold off
%%
%% CALCULATING MOVMEAN FEATURES FOR CNN ACTIVATIONS   / LSTMS
clc
data0 = {PN06_1'; PN06_2'; PN06_3'; PN06_5'};
data = {};
for i = 1:4
   data{i} = movmean(data0{i},120);
end
movmean_data = data';
%% Viewing individual activations
JJ = cell2mat(OverallMovemean(signal172, 1));
M = cell2mat(OverallMovemean(signal172, 60));
t = 1:numel(M);
figure;
plot(t,JJ)
plot(t,M,'LineWidth',2)
hold on
xlabel("Time (10 sec)")
ylabel("# of Flagging Electrodes")
title("PN06-4")
%%
figure;
tiledlayout(2,2)
for i = 1:3
   nexttile
   stackedplot(data{i}')
   xlabel("Time Step")
end
%%
function folders = getFolders(root)
   s=dir(root);
   idx=~startsWith({s.name},'.') & [s.isdir];
   folders={s(idx).name};
end
function signal = getActivations(events, net, root, files, threshold)
labels = getFolders(strcat(root,char(files{1})));
YPredo = {};
scoreso = {};
signal = {};
for p = 1:4
   event = cell(events{p});
   for i = 1:length(labels)
       idxer = {strcat('\',labels{i})};
```

```
        fileLocations = getLocs(root, files, idxer, event);
        testImages = imageDatastore(fileLocations, ...
        "IncludeSubfolders",true, ...
        "LabelSource",'foldernames');
        %ordered = natsort(testImages.Files);
        %testImages.Files = ordered;
        [~, scoreso{i}] = classify(net, testImages);
    end

    ic = {};
    A = cell2mat(scoreso);

    for j = 1:29
       ic{j} = A(:,(2*j-1),:);
    end
    ictals = cell2mat(ic);
    ictals = gpuArray(ictals);

    flags = {};

    for q = 1:length(ictals)
        for b = 1:length(labels)
            if ictals(q,b) > threshold
                flags{q,b} = 1;
            else
                flags{q,b} = 0;
            end
        end
    end

    signal{p} = sum(cell2mat(flags),2);
end
end
function fileLocations = getLocs(root, files, electrodes, events)
place = 1;
for n1 = 1:length(files)
    file{n1} = strcat(root, files{n1});
    for i1 = 1:length(electrodes)
        eFile{i1} = strcat(file{n1}, electrodes{i1});
        for j1 = 1:length(events)
            fileLocations{place} = fullfile(eFile{i1}, events{j1});
            place = place+1;
        end
    end
end
end
function sig2 = OverallMovemean(signal, k)
sig = {};
sig2 = {};
    for i = 1:size(signal,1)
       sig{i} = {signal{i,1}'; signal{i,2}'; signal{i,3}'; signal{i,4}'; signal{i,5}'};
       A = cell2mat(sig{i}');
       sig2{i} = movmean(A,k);
    end
end
function  A = get_Times(X, pre1_start, pre2_start, ictal_start, ictal_end, post_ictal_end)
    inter0 = X(1 : (pre1_start-1));                 % (before seizure)
    pre1 = X(pre1_start +1: (pre2_start));                 % pre-ictal
    pre2 = X(pre2_start +1: (ictal_start));
    ictal = X(ictal_start+1 : ictal_end);             % ictal
    post = X(ictal_end+1 : post_ictal_end);           % post-ictal
    inter1 = X(post_ictal_end+1: length(X));             % (after seizure)
    % where X = dataset
    A = {inter0,pre1,pre2, ictal,post, inter1};
end
function [II0, Pre1, Pre2,Seizure, Post, II1] = obtainEvents(signal)
BS = {};
Pre1 = {};
Pre2 = {};
SZ = {};
Po = {};
AS = {};
```

```
   for i = 1:size(signal,1)
       for j = 1:6
           if j == 1
               BS{i} = signal{i, j};
           elseif j == 2
               Pre1{i} = signal{i, j};
           elseif j==3
               Pre2{i} = signal{i, j};
           elseif j == 4
               SZ{i} = signal{i, j};
           elseif j == 5
               Po{i} = signal{i, j};
           elseif j == 6
               AS{i} = signal{i, j};
           end
       end
   end
II0 = cell2mat(BS)';
Pre1 = cell2mat(Pre1)';
Pre2 = cell2mat(Pre2)';
Seizure = cell2mat(SZ)';
Post = cell2mat(Po)';
II1 = cell2mat(AS)';
end
function [t_onset, length] = times(r1, s1, s2)
   reg0 = datetime(r1, 'InputFormat', 'HH.mm.ss');
   sz0 = datetime(s1, 'InputFormat', 'HH.mm.ss');
   sz1 = datetime(s2, 'InputFormat', 'HH.mm.ss');
   t_onset = floor(seconds(time(between(reg0,sz0))));
   length = seconds(time(between(sz0,sz1)));
   t_end = length + t_onset;
   fprintf('Seizure Range: \t %g - %g seconds \n\nSeizure Length: \t %g seconds \n\n ', t_onset,t_end,length)
end
function [t_onset, length] = times2(r1, s1, s2, min)
   reg0 = datetime(r1, 'InputFormat', 'HH.mm.ss');
   sz0 = datetime(s1, 'InputFormat', 'HH.mm.ss');
   sz1 = datetime(s2, 'InputFormat', 'HH.mm.ss');
   t_onset = floor(seconds(time(between(reg0,sz0)))/10);
   length = seconds(time(between(sz0,sz1)));
   t_end = length + t_onset;
   fprintf('Seizure Range: \t %g - %g seconds \n\nSeizure Length: \t %g seconds \n\n ', t_onset,t_end,length)
   pre1_t = t_onset - 180;
   pre2_t = t_onset - min*6;
   fprintf('Pre ranges: \t %g - %g seconds ', pre1_t,pre2_t)
   post = t_end + 60;
   fprintf("Post: %g",post)
end
function signal = scrapeoutActivations(sig, time, r1, s1, s2)
   [t_onset, length] = times(r1, s1, s2);
   t_onset = round(t_onset/10);
   time_cutoff = t_onset - time;
   signal = sig(time_cutoff:t_onset-1);
end
function bounds(length_pre_ictal, length_post_ictal, seizure_start, seizure_end)
   pre_ictal_start = seizure_start - length_pre_ictal;
   post_ictal_end = seizure_end + length_post_ictal;
   fprintf('Pre-ictal Start: \t %g seconds \n\n Post-ictal End: \t %g seconds \n\n ',
pre_ictal_start,post_ictal_end)
end
function [B,ndx,dbg] = natsort(A,rgx,varargin)
% Natural-order / alphanumeric sort the elements of a text array.
%
% (c) 2012-2023 Stephen Cobeldick
%
% Sorts text by character code and by number value. By default matches
% integer substrings and performs a case-insensitive ascending sort.
% Options to select the number format, sort order, case sensitivity, etc.

fnh = @(c)cellfun('isclass',c,'char') & cellfun('size',c,1)<2 & cellfun('ndims',c)<3;
%
if iscell(A)
        assert(all(fnh(A(:))),...
```

```
                'SC:natsort:A:CellInvalidContent',...
                'First input <A> cell array must contain only character row vectors.')
        C = A(:);
elseif ischar(A) % Convert char matrix:
        assert(ndims(A)<3,...
                'SC:natsort:A:CharNotMatrix',...
                'First input <A> if character class must be a matrix.') %#ok<ISMAT>
        C = num2cell(A,2);
else % Convert string, categorical, datetime, enumeration, etc.:
        C = cellstr(A(:));
end
%
chk = '(match|ignore)(case|dia)|(de|a)scend(ing)?|(char|nan|num)[<>](char|nan|num)|%[a-z]+';
%
if nargin<2 || isnumeric(rgx)&&isequal(rgx,[])
        rgx = '\d+';
elseif ischar(rgx)
        assert(ndims(rgx)<3 && size(rgx,1)==1,...
                'SC:natsort:rgx:NotCharVector',...
                'Second input <rgx> character row vector must have size 1xN.') %#ok<ISMAT>
        nsChkRgx(rgx,chk)
else
        rgx = ns1s2c(rgx);
        assert(ischar(rgx),...
                'SC:natsort:rgx:InvalidType',...
                'Second input <rgx> must be a character row vector or a string scalar.')
        nsChkRgx(rgx,chk)
end
%
varargin = cellfun(@ns1s2c, varargin, 'UniformOutput',false);
ixv = fnh(varargin); % char
txt = varargin(ixv); % char
xtx = varargin(~ixv); % not
%
% Sort direction:
tdd = strcmpi(txt,'descend');
tdx = strcmpi(txt,'ascend')|tdd;
% Character case:
tcm = strcmpi(txt,'matchcase');
tcx = strcmpi(txt,'ignorecase')|tcm;
% Char/num order:
ttn = strcmpi(txt,'num>char')|strcmpi(txt,'char<num');
ttx = strcmpi(txt,'num<char')|strcmpi(txt,'char>num')|ttn;
% NaN/num order:
ton = strcmpi(txt,'num>NaN')|strcmpi(txt,'NaN<num');
tox = strcmpi(txt,'num<NaN')|strcmpi(txt,'NaN>num')|ton;
% SSCANF format:
tsf = ~cellfun('isempty',regexp(txt,'^%([bdiuoxfeg]|l[diuox])$'));
%
nsAssert(txt, tdx, 'SortDirection', 'sort direction')
nsAssert(txt, tcx,  'CaseMatching', 'case sensitivity')
nsAssert(txt, ttx,  'CharNumOrder', 'number-character order')
nsAssert(txt, tox,   'NanNumOrder', 'number-NaN order')
nsAssert(txt, tsf,  'sscanfFormat', 'SSCANF format')
%
ixx = tdx|tcx|ttx|tox|tsf;
if ~all(ixx)
        error('SC:natsort:InvalidOptions',...
                ['Invalid options provided. Check the help and option spelling!',...
                '\nThe provided options:%s'],sprintf(' "%s"',txt{~ixx}))
end
%
% SSCANF format:
if any(tsf)
        fmt = txt{tsf};
else
        fmt = '%f';
end
%
xfh = cellfun('isclass',xtx,'function_handle');
assert(nnz(xfh)<2,...
        'SC:natsort:FunctionHandle:Overspecified',...
```

```
        'The function handle option may only be specified once.')
assert(all(xfh),...
        'SC:natsort:InvalidOptions',...
        'Optional arguments must be character row vectors, string scalars, or function handles.')
if any(xfh)
        txfh = xtx{xfh};
end
%
%% Identify and Convert Numbers %%
%
[nbr,spl] = regexpi(C(:), rgx, 'match','split', txt{tcx});
%
if numel(nbr)
        V = [nbr{:}];
        if strcmp(fmt,'%b')
                V = regexprep(V,'^0[Bb]','');
                vec = cellfun(@(s)pow2(numel(s)-1:-1:0)*sscanf(s,'%1d'),V);
        else
                vec = sscanf(strrep(sprintf(' %s','0',V{:}),',','.'),fmt);
                vec = vec(2:end); % SSCANF wrong data class bug (R2009b and R2010b)
        end
        assert(numel(vec)==numel(V),...
                'SC:natsort:sscanf:TooManyValues',...
                'The "%s" format must return one value for each input number.',fmt)
else
        vec = [];
end
%
%% Allocate Data %%
%
% Determine lengths:
nmx = numel(C);
lnn = cellfun('length',nbr);
lns = cellfun('length',spl);
mxs = max(lns);
%
% Allocate data:
idn = permute(bsxfun(@le,1:mxs,lnn),[2,1]); % TRANSPOSE lost class bug (R2013b)
ids = permute(bsxfun(@le,1:mxs,lns),[2,1]); % TRANSPOSE lost class bug (R2013b)
arn = zeros(mxs,nmx,class(vec));
ars =  cell(mxs,nmx);
ars(:) = {''};
ars(ids) = [spl{:}];
arn(idn) = vec;
%
%% Debugging Array %%
%
if nargout>2
        dbg = cell(nmx,0);
        for k = 1:nmx
                V = spl{k};
                V(2,:) = [num2cell(arn(idn(:,k),k));{[]}];
                V(cellfun('isempty',V)) = [];
                dbg(k,1:numel(V)) = V;
        end
end
%
%% Sort Matrices %%
%
if ~any(tcm) % ignorecase
        ars = lower(ars);
end
%
if any(ttn) % char<num
        % Determine max character code:
        mxc = 'X';
        tmp = warning('off','all');
        mxc(1) = Inf;
        warning(tmp)
        mxc(mxc==0) = 255; % Octave
        % Append max character code to the split text:
        %ars(idn) = strcat(ars(idn),mxc); % slower than loop
```

```
        for ii = reshape(find(idn),1,[])
                ars{ii}(1,end+1) = mxc;
        end
end
%
idn(isnan(arn)) = ~any(ton); % NaN<num
%
if any(xfh) % external text-sorting function
        [~,ndx] = txfh(ars(mxs,:));
        for ii = mxs-1:-1:1
                [~,idx] = sort(arn(ii,ndx),txt{tdx});
                ndx = ndx(idx);
                [~,idx] = sort(idn(ii,ndx),txt{tdx});
                ndx = ndx(idx);
                [~,idx] = txfh(ars(ii,ndx));
                ndx = ndx(idx);
        end
elseif any(tdd)
        [~,ndx] = sort(nsGroups(ars(mxs,:)),'descend');
        for ii = mxs-1:-1:1
                [~,idx] = sort(arn(ii,ndx),'descend');
                ndx = ndx(idx);
                [~,idx] = sort(idn(ii,ndx),'descend');
                ndx = ndx(idx);
                [~,idx] = sort(nsGroups(ars(ii,ndx)),'descend');
                ndx = ndx(idx);
        end
else
        [~,ndx] = sort(ars(mxs,:)); % ascend
        for ii = mxs-1:-1:1
                [~,idx] = sort(arn(ii,ndx),'ascend');
                ndx = ndx(idx);
                [~,idx] = sort(idn(ii,ndx),'ascend');
                ndx = ndx(idx);
                [~,idx] = sort(ars(ii,ndx)); % ascend
                ndx = ndx(idx);
        end
end
%
%% Outputs %%
%
if ischar(A)
        ndx = ndx(:);
        B = A(ndx,:);
else
        ndx = reshape(ndx,size(A));
        B = A(ndx);
end
%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%natsort
function grp = nsGroups(vec)
% Groups in a cell array of char vectors, equivalent to [~,~,grp]=unique(vec);
[vec,idx] = sort(vec);
grp = cumsum([true(1,numel(vec)>0),~strcmp(vec(1:end-1),vec(2:end))]);
grp(idx) = grp;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%nsGroups
function nsChkRgx(rgx,chk)
% Perform some basic sanity-checks on the supplied regular expression.
chk = sprintf('^(%s)$',chk);
assert(isempty(regexpi(rgx,chk,'once')),...
        'SC:natsort:rgx:OptionMixUp',...
        ['Second input <rgx> must be a regular expression that matches numbers.',...
        '\nThe provided input "%s" looks like an optional argument (inputs 3+).'],rgx)
if isempty(regexpi('0',rgx,'once'))
        warning('SC:natsort:rgx:SanityCheck',...
                ['Second input <rgx> must be a regular expression that matches numbers.',...
                '\nThe provided regular expression does not match the digit "0", which\n',...
                'may be acceptable (e.g. if literals, quantifiers, or lookarounds are used).'...
                '\nThe provided regular expression: "%s"'],rgx)
end
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%nsChkRgx
function nsAssert(txt,idx,eid,opt)
% Throw an error if an option is overspecified.
if nnz(idx)>1
	error(sprintf('SC:natsort:%s:Overspecified',eid),...
		['The %s option may only be specified once.',...
		'\nThe provided options:%s'],opt,sprintf(' "%s"',txt{idx}));
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%nsAssert
function arr = ns1s2c(arr)
% If scalar string then extract the character vector, otherwise data is unchanged.
if isa(arr,'string') && isscalar(arr)
	arr = arr{1};
end
end
```

Semaphore Flagging + Kurtosis

```
net = trainedVGG;
events = {{'Inter_ictal0'},{'Pre_ictal'},{'Ictal'},{'Post_ictal'},{'Inter_ictal1'}};
[B,ndx,dbg] = natsort(A,rgx,varargin);
%%
A = [signal142{1}; signal142{2}]';
B = [signal144{1}; signal144{2}]';
%%
Data092 = cell2mat(data92');
pi = [data92{1}; data92{2}]';
%%
% CHANGING PARAMETERS SO THAT THERE ARE NO FALSE POSITIVES
% AMOUNT OF FRAMES REQUIRED TO WARRANT A FLAG
% These seem ideal for PT6
J1 = 93;
J2 = 2;
t1 = 3;
t2 = 3;
G_t1 = 50;
G_t2 = 74;
w1 = 23;
w2 = 120;
tic
amt1 = 100;
amt2 = 100;
flag_no = 1;
A =cell(amt1,amt2);
B =cell(amt1,amt2);
%C =cell(amt1,amt2);
for k = 1:amt1
   for l = 1:amt2
       J1 = k;
       w1 = l;
       [~, ~, beyond30, in30,~] = Semaphores({pi}, J1, J2, w1, w2, t1, t2, G_t1, G_t2);

       A{k,l} = beyond30(:,flag_no);
       B{k,l} = in30(:,flag_no);
       %C{k,l} = in10(:,flag_no);
   end
end
Sdiff1 = zeros(amt1, amt2);
Sdiff2 = zeros(amt1, amt2);
Sdiff3 = zeros(amt1, amt2);
clc
for i = 1:length(A)
   for j = 1:size(A,1)
       Sdiff1(i,j) = sum(A{i,j} - B{i,j});
       %Sdiff2(i,j) = sum(B{i,j} - C{i,j});
       %Sdiff3(i,j) = sum(A{i,j} - C{i,j});
   end
end
%%
figure;
title('Grid Search of Kurtosis Parameters')
surface(Sdiff1)
xlabel('Smoothing Window (movmean)')
ylabel('Kurtosis Window')
toc
%%
[R,C] = find(Sdiff1==min(Sdiff1(:)))
val = Sdiff1(R,C)
%%
sort(Sdiff1)
%%
max(R) * .1
min(C)*.2
[7,2];
%%
D = 1;
test = Data1{D};
```

```
w2 = 120;
M2w = 2;
M2 = movmean(test,M2w);
P2 = zeros(length(test),1);
for i=1:length(test)
   P2(i) = kurtosis(test(max(0,i-w2)+1:i));
end
w1 = 30;
M1w = 13;
M1 = movmean(test, M1w);
P1 = zeros(length(test),1);
for i=1:length(test)
   P1(i) = kurtosis(test(max(0,i-w1)+1:i));
end
figure;
plot(P2,LineWidth=2)
hold on
plot(P1,LineWidth=2)
xline(length(test),'-r', 'Seizure Onset', LineWidth=3)
xline(length(test)-180, '--k', '30 min before Seizure', LineWidth=3)
title('Kurtosis of Different Window Sizes (PN06)')
xlabel('time (10 sec)')
ylabel('Kurtosis')
legend('Large Window', 'Small Window', 'Location','northwest','FontSize', 16)
%%
t1 = 3;
t2 = 3;
J1 = 78;
J2 = 2;
w1 = 15;
w2 = 65;
G_t1 = 30;
G_t2 = 102;
[flags1T, flags2T, beyond30, in30, in10] = Semaphores({Data092}, J1, J2, w1, w2, t1, t2, G_t1, G_t2);
flagsT ={};
for D = 1:1
   flagsT{D} = flags1T{D} +flags2T{D};
end
%%
figure;
plot(flagsT{1})
%%
Flags = {flagsT{1}, flagsT{2}};
M = 1*movmean(cell2mat(Flags),180);
Mark = zeros(length(M),1);
for i = 1:length(M)
   if M(i)>4.3
       Mark(i) = 1;
   else
       Mark(i) = 0;
   end
end
SOZ_marks = zeros(size(Flags));
min30 = zeros(size(Flags));
for i=1:length(Flags)
   x = 0; s = length(Flags{i});
   if (i==1)
       SOZ_marks(i) = s+1;
       min30(i) = SOZ_marks(i)-180;
   else
       SOZ_marks(i) = SOZ_marks(i-1)+s+1;
       min30(i) = SOZ_marks(i)-180;
   end
end
flags1 = cell2mat(flags1T);
flags2 = cell2mat(flags2T);
%%
figure;
plot(flags1)
hold on
plot()
%%
```

```
fp = 1;
tp = 1;
data = Data9;
time = zeros(3, length(data{2}));
for i = 1:3
   for j = 1:length(data{i})
       if flags2T{i}(j) ==1 && j < (length(data{i}) - 180)
           fp = fp + 1;
       elseif flags2T{i}(j) ==1 && j >= (length(data{i}) - 180)
           tp = tp +1;
           time(i,j) = length(data{i})-j;
       end
   end
end
timeV = mean(nonzeros(time))
tp
fp
%%
figure;
plot(flags1, LineWidth=3)
hold on
xline(726)
xline(747+726)
xline(546+747+726)
line([t(SOZ_marks); t(SOZ_marks)]./10,repmat([0;
20],1,length(SOZ_marks)),'LineWidth',2,'Color','r','LineStyle','--','Marker','none');
line([t(min30); t(min30)]./10,repmat([0;
20],1,length(min30)),'LineWidth',2,'Color','k','LineStyle',':','Marker','none');
yline(4,LineWidth=2)
%%
clc
figure;
t1 = 1:length(Data1{4});
Marks = Mark(end-length(Data1{4})+1:end);
for i = 1:300
   plot(t1(i), MM1(i))
   if Mark(i) == 1
       area(MM1(i-1:i), t1(i-1:i))
   end
   hold on

end
%%
%%
Test = [signal64{1};signal64{2}; signal64{3}; signal64{4}];
SOZ = length([signal64{1}; signal64{2}]);
%%
i = 500;
t1 = 1:length(Test);
F = cell2mat(flags2T);
figure;
plot(t1(1:i), MM2(1:i))
hold on
Ls = length(nonzeros(F(1:i)));
funct = MM2;
plotFlags(Ls, i, F, funct)
plotLines(SOZ,i)
%%
xline(length(Data1{4})-180,'LineWidth',2,'Color','k','LineStyle',':','Marker','none');
%%
%%
figure;
plot(Data1{1})
%%
xline(length(Data2{D})-180, LineWidth=2)
xline(length(Data2{D})-60, LineWidth=2)
hold on
plot(Data2{D})
%fprintf("Threshold 1: %d \n\nThreshold 2: %d\n\nThreshold 3: %d\n\n", floor(mean(A)), floor(mean(B)),
floor(mean(C)))
function [flags1T, flags2T, beyond30, in30, in10] = Semaphores(Data1, J1, J2, w1, w2, t1, t2, G_t1, G_t2)

   for i = 1:1
```

```
        Z1{i} = movmean(Data1{i},J1);
        Z2{i} = movmean(Data1{i},J2);
    end
C_count1 = 0;
C_count2 = 0;
flags1T = {};
flags2T = {};
in30 = zeros(4, 2);
in10 = zeros(4, 2);
beyond30 = zeros(4, 2);
for q = 1:length(Z1)
    flags1 = zeros(1, length(Z1{q}));
    flags2 = zeros(1, length(Z1{q}));
    P1 = zeros(length(Z1{q}),1);
    for i=1:length(Z1{q})
        P1(i) = kurtosis(Z1{q}(max(0,i-w1)+1:i));
    end

    for i = 2:length(Z1{q})

        if P1(i-1) < t1 && P1(i) < t1
            C_count1 = C_count1 + 1;
            if C_count1 >= G_t1
                flags1(i) = 1;
            end
        else
            C_count1 = 0;
        end
    end
    P2 = zeros(length(Z2{q}),1);
    for i=1:length(Z2{q})
        P2(i) = kurtosis(Z2{q}(max(0,i-w2)+1:i));
    end

    for i = 2:length(Z2{q})

        if P2(i-1) < t2 && P2(i) < t2
            C_count2 = C_count2 + 1;
            if C_count2 >= G_t2
                flags2(i) = 1;
            end
        else
            C_count2 = 0;
        end
    end
    in30_1 = sum(flags1(end-180:end));
    in30_2 = sum(flags2(end-180:end));
    in30(q, 1:2) = [in30_1, in30_2];
    in10_1 = sum(flags1(end-60:end));
    in10_2 = sum(flags2(end-60:end));
    in10(q, 1:2) = [in10_1, in10_2];
    beyond30_1 = sum(flags1(1:end-180));
    beyond30_2 = sum(flags2(1:end-180));
    beyond30(q, 1:2) = [beyond30_1, beyond30_2];
    flags1T{q} = flags1;
    flags2T{q} = flags2;
end
end
function plotLines(SOZ,i)
if i >= SOZ-180
    label_1 = {'30 min'};
    xline(SOZ-180, '-.', label_1, LineWidth=4)
end
if i >= SOZ-60
    label_2 = {'10 min'};
    xline(SOZ-60, '-.', label_2, LineWidth=4)
end
if i > SOZ
    label_3 = {'Seizure Onset'};
    xline(SOZ, 'r', label_3, LineWidth=4)
end
if i > SOZ + 44
```

```
   label_4 = {'Seizure End'};
   xline(SOZ+44, 'r', label_4, LineWidth=4)
end
end
function plotFlags(Ls, i, F, funct)
x2 = zeros(Ls,4);
y2 = x2;
count = 1;
for j = 1:i
   if j == 1

   elseif F(j) == 1
       x2(count,1:4) = [j-1 j j j-1];
       y2(count,1:4) = [0 0 funct(j) funct(j-1)];
       count = count+1;

   else
   end
end
for k = 1:count-1
   fill(x2(k,:), y2(k,:),[0.6350 0.0780 0.1840],EdgeColor='none')
end
end
```