Aalborg University

MSc. in Biomedical Engineering and Informatics

Master's thesis

10402

Using a CNN-LSTM Architecture to Classify Chronic Pain Models based on μECoG Recordings from S1 in large animal models

Participants:

Nickolaj Ajay Atchuthan

Hjalte Færregård Clark

Mikkel Bjerre Danyar

Main supervisor:

Suzan Meijs

Co-supervisor:

Felipe Rettore Andreis

1st of June, 2023

AALBORG UNIVERSITY

STUDENT REPORT

AALBORG UNIVERSITY
STUDENT REPORT

Department of Health Science and Technology

Master's thesis

Master of Science in Biomedical Engineering and Informatics

Selma Lagerløfs Vej 249

9260 Gistrup

http://hst.aau.dk

Title:

Using a CNN-LSTM Architecture to Classify Chronic Pain Models based on μECoG Recordings from S1 in large animal models

Project:

Master's Thesis

Project period:

02/02/2023 - 01/06/2023

Project group:

10402

Participant(s):

Hjalte Færregård Clark
Mikkel Bjerre Danyar
Nickolaj Ajay Atchuthan

Main supervisor :

Suzan Meijs

Co-supervisor:

Felipe Rettore Andreis

Report pages: 107
Appendices: 4
Date of completion: 01/06/2023

Abstract:

Background and aim: Chronic pain is a major healthcare problem that affects one of every five adults in Europe. Current diagnostic methods are suboptimal, and a variety of methods have been tried to help diagnostics. We aim to classify different chronic pain models in large animals using a CNN-LSTM based on μECoG recordings from S1.

Methods and materials: In this study, we employed 16 Danish Landrace pigs to investigate the effects of high-frequency stimulation (HFS) in inducing a temporary chronic state and evoking long-term potentiation (LTP). Additionally, some of the pigs underwent spared nerve injury (SNI) to induce long-term chronic pain. Electrocorticography (ECoG) recordings were conducted before and after the intervention, with low-frequency stimulation (LFS) applied. The recorded data were transformed using the continuous wavelet transform. This spectrogram was utilized as input for a CNN-LSTM neural network architecture, aiming to identify unique patterns in the signal that could be correlated with specific pain model groups (LTP, SNI, and control). The design of the model focused on differentiating between these groups.

Results: An accuracy of 42.8% was achieved for the multiclass model, and 52% for the binary model. Test results showed only correct predictions for the control class for both models. The validation accuracy was 63.9% and 84.0% for the multiclass and binary models respectively. Model interpretability showed differences between the location and patterns of feature attribution from validation to test data.

Conclusion: The performance of the developed CNN-LSTM model was found to be unsatisfactory for both multiclass and binary classification tasks. To enhance its generalizability, it is crucial to include a larger number of subjects, particularly from the LTP and SNI groups. By incorporating a wider range of inter-subject variance in the training process, it can potentially improve the model's performance and make it more reliable for future use.

Preface

This project was the Master's thesis by group 10402 from February 2nd, 2023, to June 1st, 2023. It was completed as part of the Biomedical Engineering and Informatics program at Aalborg University. The project was carried out under the guidance of Suzan Meijs and Felipe Rettore Andreis.

Reading Guide

This project consists of a description of the problem field within classifying chronic pain and with various methods, followed by a structured literature review and a description of the experimental protocol from the data used in this project. It then details the data management, initial inspection, preprocessing, and as well as the input image for the model. The development process of the deep learning algorithm, specifically a CNN-LSTM, is then presented, including the results from the classification of the data. The project ends with a discussion and conclusion on the study.

The appendix of this project contains theoretical background on the two chronic pain models and low frequency stimulation. This is followed by model interperability, the time series results, and lastly the time plan for this project.

The project follows the Harvard citation method, with references mentioned by the last name of the author followed by the year of publication. A full list of references is included at the end of the project.

Aim

How can a CNN-LSTM architecture classify between chronic pain models based on ERP signals from μECoG recordings in S1?

Contents

1 Problem Analysis **1**

1.1 General Pain 1
1.2 Neuropathic Pain 5
1.3 Diagnosing Chronic Neuropathic Pain 7
1.4 Overview of Different Diagnostic Imaging Methods 14
1.5 Classification of Pain 14
1.6 Aim 16

2 Structured Literature Search **17**

2.1 The Initial Search 17
2.2 AND/OR Table 18
2.3 Structured Search on Machine Learning Models for EEG Classification . . . 21

3 Experimental Protocol **25**

3.1 Protocol and Procedure of the Experiment 25
3.2 Measurement Protocol 27

4 Pre-proccesing **29**

4.1 Data Management & Signal Processing 29
4.2 Initial Inspection of μECoG Data 30
4.3 Pre-processing of μECoG Signals 31
4.4 Wavelet Transform 37
4.5 Temporal Window Length 42

5 Neural Network Architecture **45**

5.1 General CNN-LSTM architecture 45
5.2 Convolutional Neural Network 45
5.3 Long Short-Term Memory 57

6 Model Interpretability, Evaluation, and Performance Metrics **63**

6.1 Model Interperability 63
6.2 Saliency 63
6.3 Integrated Gradients 64
6.4 DeepLIFT 65
6.5 Occlusion 67
6.6 Model Evaluation 67

6.7 Classification Performance Metrics 69

7 Training and Validation of the Network 77
7.1 Development Strategy of the Network 77
7.2 Separation of Data 77
7.3 Training Process 78
7.4 Structured Training Process 80
7.5 Model Chosen for the Final Testing 84
7.6 Final CNN-LSTM Architecture 85

8 Results 87
8.1 Confusion Matrix 87
8.2 Accuracy and F1-score 88
8.3 Receiver Operating Characteristic Curve 88
8.4 Model Interperability 89

9 Discussion 101
9.1 Comparison of Results 101
9.2 Methodological Considerations 103
9.3 Future Work 105

10 Conclusion 107

Bibliography 109

A Pain Models 121
A.1 Spared nerve injury (SNI) 121
A.2 Electrically evoked acute pain 123
A.3 Low-frequency stimulation (LFS) 123

B Model Interperability 125
B.1 Multiclass model 126
B.2 Binary model 132

C Time Series 137

D Activity- and Time-plan 139

1 Problem Analysis

This chapter investigates the problem domain through the literature. The first two sections explain different types of pain and the problem of diagnosing pain. Afterward, different methods of diagnosing and evaluating pain will be explored. Lastly, the use of deep learning as a diagnostic tool will be considered by discussing different architectures.

1.1 General Pain

Pain is defined by the International Association for the Study of Pain (IASP) as: *"An unpleasant sensory and emotional experience associated with, or resembling that associated with actual or potential tissue damage."* [IASP, 2023b]. Pain can be categorized according to its duration as either acute or chronic, or in terms of mechanisms, such as nociceptive, inflammatory, or neuropathic pain (NP). [Bennet, 2011; King et al., 2013]. Pain is however not limited to physical pain, since psychological pain also occurs. This comes from psychological stimuli, e.g. losing a loved one, just like physical pain manifests as a response to noxious physical stimuli. [Mee et al., 2006]

1.1.1 Acute and Chronic Pain

Acute pain is short-term pain that happens suddenly with an intense or sharp feeling that serves as an early warning system for the body. Therefore, pain plays an important role in making the individual aware of impairment and responding appropriately. Response includes withdrawal from the stimulus causing the pain, to avoid further impairment and behaviors that reduce the effect of the impairment and facilitates recovery. Acute pain often lasts a few minutes up to less than 6 months and ends when the underlying cause of the pain has healed or been treated. [King et al., 2013; IASP, 2023a]

Chronic pain is persistent pain over 6 months or beyond the expected period of healing. It is caused by damage to the tissue or nervous system that continues after the body has recovered. The long-term component of chronic pain is not physiologically critical once the injury has healed, but does inform animals or patients the location of the previously injured area. Chronic pain can also spread to other areas than where the original injury first occurred. Typical effects due to chronic pain are mood disorders, loss of sleep, emotional

suffering, and cognitive impairments. [Zhuo, 2011; Mouraux and Iannetti, 2018; Davis et al., 2017]

Chronic pain can be further categorized into primary and secondary chronic pain. Primary chronic pain is a type of pain that has persisted longer than 6 months and has had significant emotional distress and/or functional disability, with no other condition that can account for the pain. Secondary chronic pain refers to the presence of pain that arises as a result of an underlying disability, such as spinal cord injury, cancer, or multiple sclerosis.[Nicholas et al., 2019; Ehde et al., 2003] The prevalence of chronic pain in European adults is 19% of moderate to severe intensity. Furthermore, 21% of chronic pain patients got diagnosed with depression, 56% were less able to sleep, and 32% were no longer able to work outside of their homes. [Breivik et al., 2006]. Since pain has a major impact on people's well-being, researchers try to develop novel treatments to reduce suffering and improve the quality of life for pain patients. [Mouraux and Iannetti, 2018]

1.1.2 Nociceptive and Inflammatory Pain

There are three main pain mechanisms, nociceptive, inflammatory, and NP. Nociceptive pain provides neural feedback from nociceptors in the skin, muscles, and joints, which allows the central nervous system (CNS) to detect and avoid potentially damaging stimuli. Nociceptive pain may persist due to tissue damage after surgery that does not restore properly and as well as ongoing inflammation. Nociceptive pain can also develop over time, either in parallel or independent from acute inflammation. [Finnerup et al., 2022; Armstrong and Herr, 2019]

Nociceptors are branched-free nerve endings that detect heat, cold, mechanical force, or chemical stimulation-induced pain. Primary afferent nerves are typically classified into three types: Aβ, Aδ, and C-fibers, as seen in Table 1.1. [Armstrong and Herr, 2019; Bell, 2018]

Table 1.1. Classification of primary afferents by diameter, myelin, conduction velocity, and sensory function. [Bell, 2018]

Classification	**Diameter(μm)**	**Myelin**	**Conduction velocity(m/s)**	**Sensory function**
Aβ	6–12	Yes	>35 m/s	Touch
Aδ	1–5	Thin	5–35 m/s	'Fast' pain
C	0.2–1.5	No	<2.0 m/s	'Slow' pain

Most Aβ fibers are low-frequency mechanoreceptors that react to touch, pressure, or hair movement but do not signal nociceptive stimuli. The nociceptors are divided into Aδ and C-fibers. Aδ are medium-diameter myelinated afferents responsible for fast pain. This type of pain is also well-localized meaning that if a person stubs the toe, a sharp intense sensation is focused specifically on the toe. Aδ have two subgroups, type I and type II,

where type I primarily is responsible for mechanical and chemical stimuli, whereas type II primarily is responsible for thermal stimuli. C-fibers are unmyelinated slow pain fibers with poor spatial localization. C-fibers carry information regarding mechanical, thermal and/or chemical nociception, and become stimulated secondary to Aδ, and only if the painful stimuli is persistent. Pain due to Aδ activation has been described as "pricking" pain, while pain due to C-fibers activation is "dull" or "pressing" pain. Once the nociceptive stimulus has been transduced it is transmitted to the central nervous system in the spinal cord as trains of action potentials. The first synapse is in the dorsal horn of the spinal cord, which is a site where the stimuli are processed and integrated. Projection cells cross the midline (see Figure 1.1), up through the medulla to the thalamus, where the signal is sent to the primary somatosensory cortex (S1). [Armstrong and Herr, 2019; Bell, 2018]

Inflammatory pain is a spontaneous hypersensitive pain in response to inflammation and tissue damage, e.g. a surgical wound or inflamed joint. To reduce further risk of damage and promote recovery, the body heightens the sensitivity in the injured area. Normally, most C-fibers have little to no spontaneous activity and are only activated by intense physical stimuli. When an injury occurs, these terminals are sensitized, resulting in both increased responsiveness to normal input and spontaneous activity. This can cause the activity of moderate physical stimuli to an allodynic state of heightened sensitivity. [Woller et al., 2017; Woolf et al., 2010]

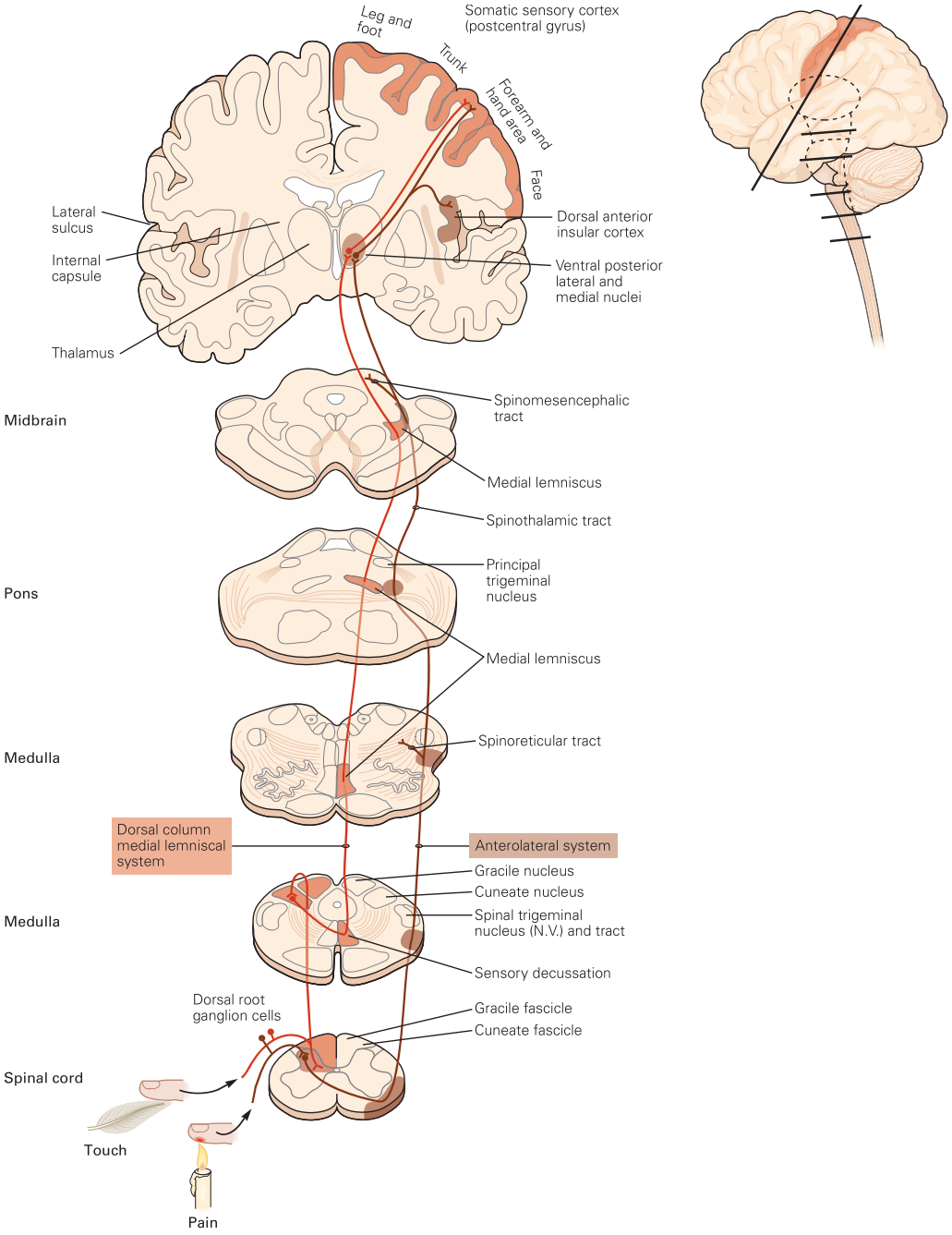


Figure 1.1. Illustration of the pathway from peripheral stimuli, either non-nociceptive (orange) or nociceptive (brown), to the primary somatosensory cortex

Source: [Kandel et al., 2021]

1.2 Neuropathic Pain

NP is defined by IASP as "*pain caused by a lesion or disease of the somatosensory nervous system*" [IASP, 2022]. Normally, pain is a warning about tissue damage signaled by receptors and fibers from the periphery to the brain. When the pathway is damaged, the immediate consequence is loss or reduction of function including pain. [Finnerup et al., 2020].

Classic symptoms of NP are widespread pain not otherwise explainable, evidence of sensory deficit, burning pain, pain to light stroking of the skin, and attacks of pain without seeming provocation. A challenging aspect of NP is that the liability for pain appears to be different from person to person, between males and females, from nerve to nerve, and with age. So the same lesion can cause no pain for one individual, but for another, it can cause severe pain. [Campbell and Meyer, 2006; Bennett, 2010] Due to the fact that NP can occur at many different locations of the body with specific changes, the primary focus will be on peripheral lesions and what peripheral and central changes would arise. A starting point for understanding NP is what happens with injury to non-neural tissue. Skin injury produces ongoing pain, but also two types of hyperalgesia; primary and secondary. Primary hyperalgesia occurs on the spot of the injury, and it is partly caused by the sensitization of the primary afferent nerves. This results in increased sensitivity to external stimuli e.g. heat stimuli. Secondary hyperalgesia is located around the injured area, but not at the site of the injury. Here there is an increased sensitivity to mechanical stimuli, but not heat stimuli. This is due to sensitization in the central nervous system. Secondary hyperalgesia is comparable to that seen in NP patients, where two types of mechanical hyperalgesia are observed; pain to light-stroking stimuli which is known as allodynia, and increased level of pain when exposed to punctate stimuli. NP can be developed immediately after the injury or can have a delayed onset, typically no longer than 6 - 12 months. [Campbell and Meyer, 2006; Bennett, 2010]

Some of the peripheral changes following the lesion is reduced thresholds in receptors by pH changes. This can lead to a neuron constantly being partially or fully depolarized. Therefore, the normal high-threshold neuron will fire with less intense stimuli. Another peripheral change is the sodium channels which are responsible for spontaneous discharges in damaged neurons, leading to spontaneous pain. Furthermore, abnormal neuronal sprouting can lead to a growth of the receptive field which feeds the abnormal pain transmission. These nerve endings can cross-talk with healthy nerves and alter them to become abnormal and chaotic. The dorsal horn modulates normal pain signals and is vital to the onward transmission of abnormal chaotic input from damaged and adjacent non-damaged neurons. Normally Aδ and C fibers terminate predominantly in lamina I and II of the dorsal horn, also Aδ in lamina V, and non-noxious Aβ fibers in lamina III.

Different hypotheses have been made, but it appears that C fibers take characteristics of and express receptors that are normally confined to low-threshold Aβ fibers. [Campbell and Meyer, 2006; Bennett, 2010]

Some of the changes in the spinal cord are the development of wind-up, central sensitization as well as expansion of the receptive fields. Wind-up is an increase in pain intensity over time due to repeated stimuli over a certain threshold to group C nerve fibers and lasts from seconds to minutes. Central sensitization is a neurophysiological process characterized by an enhancement of the functioning of neurons and circuits in the nociceptive pathways. It is caused by an increase in membrane excitability, synaptic efficacy, and a reduction in inhibitory control. Central sensitization has a several-hour duration and can be elicited by high-frequency stimulation. The receptive field refers to the specific region in the sensory periphery where stimuli can activate the sensory cell. This expansion of the receptive fields means that a larger area of the body is perceived as painful or sensitive. Other changes also occur like changes to transmitters and receptor systems within the spinal cord. Here, neuropeptides are released abnormally after nerve injury, which can affect the ion channels and growth of new axons. This contributes to long-term changes in the excitability of dorsal horn cells in combination with central sensitization. [Price et al., 1977; Navarro et al., 2007; Latremoliere and Woolf, 2009]

Also, the brain seems to change due to NP, which is called neuroplasticity. The S1 seems to undergo plastic changes due to peripheral inflammation or NP. This change is related to the communication from the thalamus to S1. When looking into resting state electroencephalography (EEG) changes, NP patients seem to have a statistically significant increase in power for the theta (4-7 Hz) band in the left hemisphere, compared to healthy controls. For the alpha (8-12 Hz) and beta (13-30 Hz) bands, controversial results were found. [Mussigmann et al., 2022]. According to Pricope et al. [2022], increased activity in the pain matrix, consisting of anterior cingulate cortex (ACC), amygdala (AMG), brainstem, hippocampus (HPC), insula (INS), periaqueductal gray (PAG), prefrontal cortex (PFC), posterior parietal cortex (PPC), S1, secondary somatosensory cortex (S2), supplementary motor area (SMA), thalamus, and the cerebellum, has been strongly endorsed.

The changes in the central nervous system vary, depending on the condition and symptoms. In unilateral NP, a decrease in activity was found in the thalamus, while for mechanothermal allodynia, an increase was detected in the thalamic activity. Changes outside the pain matrix due to NP are unique for each patient and most commonly affect the dorsolateral frontal cortex, parietal association, and some of the brainstem nuclei which lie in the pain modulatory systems. Furthermore, structural brain changes in grey and white matter have been reported in ACC, PMC, S1, thalamus, PFC, SMA, dorsal midbrain, and cingulate cortex. [Pricope et al., 2022] A summary of peripheral and central

changes that occur after a peripheral nerve injury can be seen in Figure 1.2.

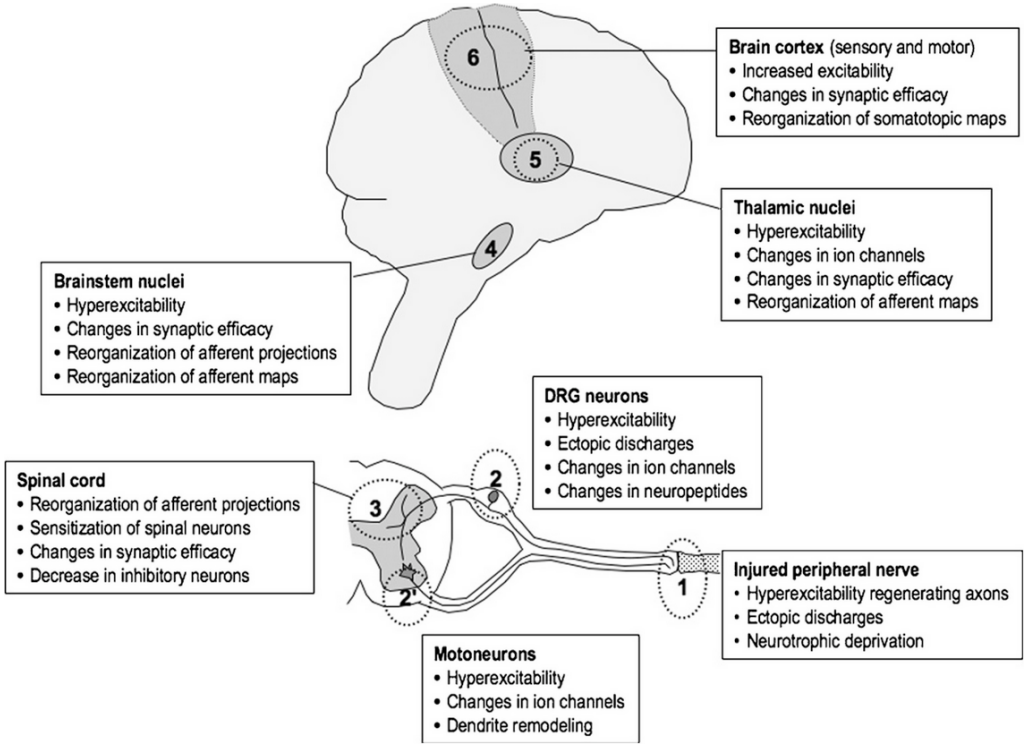


Figure 1.2. Summary of the main plastic changes that can occur in the peripheral and central nervous system after peripheral nerve injury. Examples of changes in (1) the injured peripheral nerve, (2) the Dorsal Root Ganglion (DRG) neurons and ventral horn motor neurons, (3) the spinal cord, and (4-6) in the brainstem, thalamic nuclei as well as the brain cortex.

Source: [Navarro et al., 2007]

1.3 Diagnosing Chronic Neuropathic Pain

1.3.1 Methods Used in Clinical Settings

In the existing literature, various approaches are employed to discern biomarkers that can elucidate the underlying reasons and presence of chronic NP in individuals. Nevertheless, the evaluation of pain poses a formidable task due to its inherently subjective nature, and regrettably, there are presently no biomarkers accessible for the objective quantification of pain. Consequently, the diagnosis heavily relies on subjective measures encompassing pain scales, behavioral assessments, questionnaires, and quantitative sensory testing (QST) [Wagemakers et al., 2019].

The self-report method remains widely regarded as the gold standard for assessing pain intensity, commonly utilizing numerical rating scales (NRS) or visual analog scales (VAS). Through these approaches, patients provide a subjective rating of their pain perception on a scale of 0 - 10 or indicate the severity by marking a line between two points and measuring its length. These methods are commonly employed during pain anamnesis, including the evaluation of the effectiveness of current pain treatments. In Denmark, as outlined

by Sundhedsstyrelsen [2019], pain anamnesis entails gathering comprehensive information about the patient's pain experience. This includes factors such as the pain's onset, location, intensity (measured using NRS/VAS scales), type, patterns (continuous, intermittent, diurnal variations), triggers, aggravating and alleviating factors, psychosocial conditions, whether the pain aligns with current diagnostic findings, and the effects and potential side effects of ongoing treatments. However, it is important to recognize that relying solely on patients' self-reports may introduce certain limitations, including interobserver bias and subjectivity. It should also be noted that pain intensity assessment methods cannot be used to directly compare pain levels among different patients [Davis et al., 2017; Elsayed et al., 2020; Sundhedsstyrelsen, 2019].

QST is an additional tool used to diagnose chronic NP in clinical settings and is defined by The Neuropathic Pain Research Consortium (NPRC) as a psychophysical method that measures subjective experiences, such as loss or gain of sensation, in response to particular thermal, mechanical, or vibratory stimuli. QST is a valuable tool in assessing sensory abnormalities and NP, as it provides indirect information about underlying sensory function abnormalities. It is designed to be non-invasive, using only small, portable tools and requiring less time than other protocols. [Toth, 2013]

However, QST has a few caveats when it comes to diagnosing sensory abnormalities. There is an overlap in some markers for different pain conditions, such as the loss of sensation to touch or pinprick in areas that can be reported in non-NP such as muscle pain. Similar overlaps are seen when comparing nociceptive pain and sunburn, reporting brush and heat allodynia and heat hyperalgesia. Furthermore, pressure allodynia is common in both neuropathic and nociceptive pain. The current evidence is inadequate to establish the test-retest reliability and variance over time, and as a result, it cannot be relied upon as a monitoring tool for documenting changes over time. The testing requires active participation and directed attention on behalf of the patient, and both the examiner and the patient require instruction and training in the testing procedures of QST. [Toth, 2013]. This can be especially difficult when dealing with patients who either have disorders of consciousness, speech impairments, or those who are simply unable to communicate, like infants [Schnakers and Zasler, 2007]. Without any objective indicator for pain, physicians rely on self-report as the only means to evaluate and manage pain [Woolf et al., 2010]. However, using self-report alone can lead to inadequate pain management, misjudgments, and miscommunications. [Elsayed et al., 2020; Wager et al., 2013]. Another challenge is that NP is not always peripheral meaning that tests such as QSTs are not always applicable since the manifestation of chronic pain could be due to alterations in the central nervous system. See Section 1.2.

Incorporating objective measures of pain as complementary tools to self-report may improve pain assessment and provide a better understanding of pain mechanisms. However,

this requires identifying pain through the activity of the brain itself [Elsayed et al., 2020].

1.3.2 Functional Magnetic Resonance Imaging

Most studies use functional magnetic resonance imaging (fMRI) with a high spatial resolution, allowing for detecting changes in brain activity in specific brain regions. It is based on the same technology as magnetic resonance imaging (MRI), which uses a strong magnetic field and radio waves to create detailed images of the body. But instead of creating images of organs and tissues like MRI, fMRI looks at blood flow in the brain to detect areas of activity. This is because when a brain area is more active, it consumes more oxygen, and blood flow increases to meet this increased demand. [Logothetis, 2008]. However, fMRI has a low temporal resolution, meaning that it is not able to accurately detect changes in brain activity when looking at a narrow time scale. Furthermore, it should be noted that hemodynamic techniques, such as blood oxygen level-dependent (BOLD), do not directly quantify neuronal activity, but instead, they capture the changes in blood flow dynamics that are related to neural activity. [Zolezzi et al., 2022].

In many studies, fMRI has been used to analyze the neural response of NP models and to investigate the mechanism of NP. Hubbard et al. [2015] observed the longitudinal brain changes associated with NP in rats. They found increased activity in somatosensory regions; S1 and ventral posterolateral nucleus (VPL) in the early stages of the NP model. Later on, there were changes in the activity of ACC (increased) and PAG (decreased). Furthermore, there was a correlation between cold hypersensitivity and changes observed via fMRI. A study has been conducted on non-human primates, before and after capsaicin application, and significant increases in BOLD signals were observed in brain regions such as the somatosensory, frontal, and cingulate cortices, as well as the cerebellum [Asad et al., 2016].

Komaki et al. [2016] found that mice experiencing allodynia showed significantly higher BOLD signals in the ACC and thalamus. The intact mice only exhibited activation in S1. They did this by surgically injuring the L4 spinal nerve root using spinal ligation (SNL), sensory nerve fibers were selectively stimulated, manifesting allodynia.

Furthermore, it has been shown that in humans, NP shows a distinct fMRI pattern. A study by Baliki et al. [2006] on chronic back pain patients using fMRI found that increasing pain activated classic pain matrix areas (see Figure 1.3) while high spontaneous pain activated the PFC and ACC, suggesting different spatiotemporal neuronal mechanisms for subjective pain compared to acute experimental pain.

The data that is currently accessible on the discriminative capabilities of fMRI is promising. In a research study that involved individuals suffering from chronic lower back pain, resting-state fMRI was used to differentiate them from healthy individuals, and the results

indicated an overall accuracy of 68%. [Mano et al., 2017]

While 68% accuracy may be promising, it is not particularly high for a diagnostic test in a clinical setting. Ideally, a reliable diagnostic test should have a higher accuracy to minimize false positive and false negative results. A 68% accuracy suggests that there is a significant overlap in the brain activity patterns between individuals with chronic lower back pain and healthy individuals, making it challenging to distinguish between the two groups with certainty.

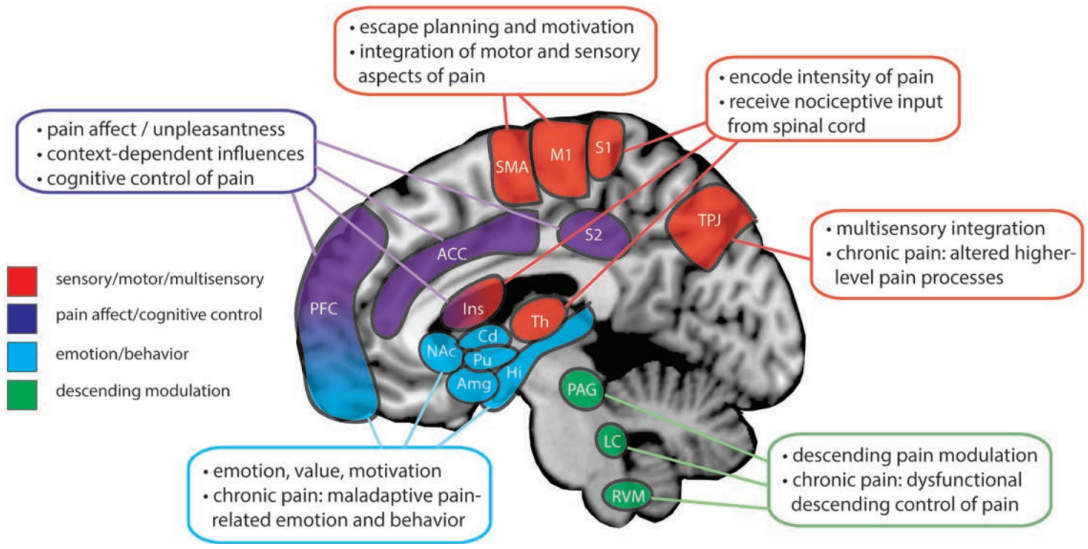


Figure 1.3. Categorisation of the different pain matrix areas and their specific functionalities.

Source: [Martucci and Mackey, 2018]

Different analysis/ feature extraction methods are used on top of BOLD in fMRI, these include region of interest (ROI) and generalized linear model (GLM). ROI analysis in fMRI involves selecting specific areas of the brain associated with a state, measuring the neural response to the state, and comparing activity between regions or groups. It identifies patterns of brain activity related to specific states [Hubbard et al., 2015; Song et al., 2021]. GLM is used in fMRI as a method for the statistical relation between brain regions and specific states, which can provide valuable insights into the neural processes underlying those states. [Song et al., 2021]

Because fMRI is a non-invasive technique it is suitable for clinical use. However, it is expensive and requires complex methodological planning when considering the development of an experiment. [Luck, 2014]

1.3.3 Positron Emission Tomography

Positron emission tomography (PET) has properties similar to fMRI when looking at the resolution in both spatial and temporal domains. PET studies have also been employed to investigate the mechanism of allodynia in NP, by measuring regional cerebral blood flow (rCBF). rCBF is a method used in PET that measures how much blood is flowing to different parts of the brain. This reflects the level of neuronal activity in different brain regions since neurons need oxygen and glucose from the blood to function. This reflection of blood flow is achieved by radioactive tracers. PET can measure rCBF by injecting a tracer that emits positrons when it decays, such as $H_2^{15}O$. The positrons then collide with electrons and produce gamma rays that can be detected by a scanner. By measuring how much tracer accumulates in different brain regions over time, PET can estimate how much blood flow those regions receive. The main advantage of PET is its capability of using highly specialized radiopharmaceutical probes tailored for specific indications [Pricope et al., 2022; de Natale et al., 2018].

Witting et al. [2006] found that patients with long-standing clinical brush-evoked allodynia following upper or lower extremity traumatic nerve injury had abnormal thermal and tactile sensitivity, hyperalgesia to pressure stimuli, and increased temporal summation (von Frey hair). Using rCBF it was seen that allodynic stimulation activated the contralateral orbitofrontal cortex (OFC) and ipsilateral insular cortex (IIC), while there was an absence of activation in S1, ACC, and thalamus. The study suggests that the cortical network responsible for sensory-discriminative processing of nociceptive pain is less active in NP, while the OFC and IIC show increased activity. This could be explained by the emotional impact of NP and the complexity of processing a mixed sensation of stimuli (brush) and pain. [Witting et al., 2006].

The use of PET scans can be very expensive since it requires the need for injecting radioactive tracers into the subject, thus also being a more invasive method. Similar to fMRI, PET imaging also demands meticulous methodological planning when designing an experiment. [Zolezzi et al., 2022].

1.3.4 Electrophysiological Neuroimaging

Scalp Electroencephalography

When trying to monitor NP one could use EEG, to measure the neural activity. The main advantage of this modality is its temporal fidelity, which comes at a cost of a lower spatial resolution. This is particularly important when studying pain, as neural activity related to pain can occur very quickly. Temporal fidelity enables researchers to look at time-critical event-related potential (ERP) responses in the brain, related to different pain models. [Luck, 2014; Zolezzi et al., 2022]. Its spatial precision, resolution, and accuracy are

relatively low compared to high-spatial-resolution imaging techniques such as fMRI. The spatial precision of EEG can be improved by spatial filters such as the surface Laplacian or adaptive source-space-imaging techniques. The spatial accuracy of EEG is low since its activity recorded from one electrode does not reflect only activity from neurons directly below that electrode, but rather, from a complex mixture of activities from multiple brain regions close to and distant from that electrode. [Cohen, 2014]. It is estimated that EEG obtains signals from a summation of potentials of 10,000 - 50,000 neurons [Murakami and Okada, 2006]. There are advantages when looking at the method's complexity and cost. EEG is rather cost-effective and is easier to implement in clinical and experimental settings. It should be noted that PET and fMRI are advantageous if trying to examine the subcortical structures which can be fundamental in NP research. [Luck, 2014; Zolezzi et al., 2022].

Hemodynamic methods, such as PET and fMRI, measure neural activity indirectly, whereas EEG directly measures electrical activity associated with neural potentials, making it more appropriate as a direct measurement tool. EEG offers greater flexibility when it comes to feature extraction. Since EEG signals have high dimensionality, non-invasive nature, high temporal resolution, and multiple frequency bands, it provides many possibilities for feature extraction. Thus a variety of signal-processing methods can be employed to investigate specific aspects of neural activity. Segmenting the frequency bands into classical categories such as delta, theta, alpha, beta, and gamma is one of the most commonly used methods for extracting features from EEG signals. Of these bands, the gamma frequency range has received considerable attention in the field of pain research, especially when looking at S1. [Zhao and Wan, 2018]. For NP models there is evidence for an increase in power and displacement of theta and delta frequencies [Zolezzi et al., 2022].

EEG has been applied in a variety of pain research with the aim of extracting useful biomarkers. Different pain models have been used, these include; thermal pain [Nuñez-Ibero et al., 2021; Savignac et al., 2022], chronic NP [Di Pietro et al., 2018; Alshelh et al., 2016; Zolezzi et al., 2022], laser evoked pain [Liberati et al., 2016; Mouraux and Iannetti, 2009], hyperalgesia, and central sensitization [Baroni et al., 2020]. EEG has also been used to compare modality-specific biomarkers, such as GABA content and resting cortical oscillations in an NP model, finding an altered brain rhythm and thalamic inhibitory neurotransmitter release [Di Pietro et al., 2018]. The specificity of laser-evoked local field potentials (LFP) in relation to pain has been a topic of discussion. However, for the purposes of this study, it is not a concern as pain models involving either injury or electrical stimulation will be utilized. [Liberati et al., 2016; Mouraux and Iannetti, 2009]

Electrocorticography and Microelectrode Arrays

Some of the limitations of classic scalp electrode setups can be bypassed by the use of electrocorticography (ECoG) or microelectrode arrays (MEA). This setup increases the spatial resolution and allows for the recording of neuronal activity at a much finer scale, making it possible to detect features that cannot be detected with traditional EEG. One of the main advantages of ECoG is that it provides a higher spatial resolution than scalp EEG. ECoG electrodes are placed on the dura or the brain and can therefore record from a much smaller area of the cortex than scalp electrodes. This increased resolution allows for the detection of more precise and localized neural activity, making it particularly useful for studies of specific cortical regions in relation to e.g. chronic pain. [Fattahi et al., 2014]

One study compared acute and chronic pain in an animal rat model, and found increased ECoG power in the theta range, with shorter effects after capsaicin injection and longer-lasting effects after nerve injury. [LeBlanc et al., 2014]. ECoG typically records from populations of neurons, detecting changes in the LFP. LFPs reflect the synchronous activity of a large number of neurons and can provide information about overall patterns of neural activity and has a range of $<$200 Hz. [Cohen, 2014; Fattahi et al., 2014]

MEA, on the other hand, provides even higher spatial resolution than ECoG. MEAs consist of an array of tiny electrodes that can be implanted directly into the brain tissue. This allows for the recording of the activity of individual neurons or small groups of neurons. One potential downside of ECoG and MEA recordings is that they are invasive, requiring the placement of electrodes directly on or in the brain which carries a risk of harming brain tissue. [Keller et al., 2016; Fattahi et al., 2014]

Additionally, ECoG and MEA recordings are mostly performed in animal models, which limits their applicability to human studies. Another important difference between ECoG and MEA recordings is the type of neuronal activity that is recorded. MEA recordings can also detect multiple unit activity (MUA) from individual neurons, allowing for a more detailed understanding of the activity of specific neurons, and enabling high-frequency feature extraction methods, such as spike counting. Spikes range from 250 - 10000 Hz with some studies differentiating between low MUA (300 - 2000 Hz) and high MUA (2000 - 10000 Hz). The spiking activity is associated with the action potentials of individual neurons or small groups of neurons (2 - 3 neurons). [Keller et al., 2016; Tøttrup, 2020]

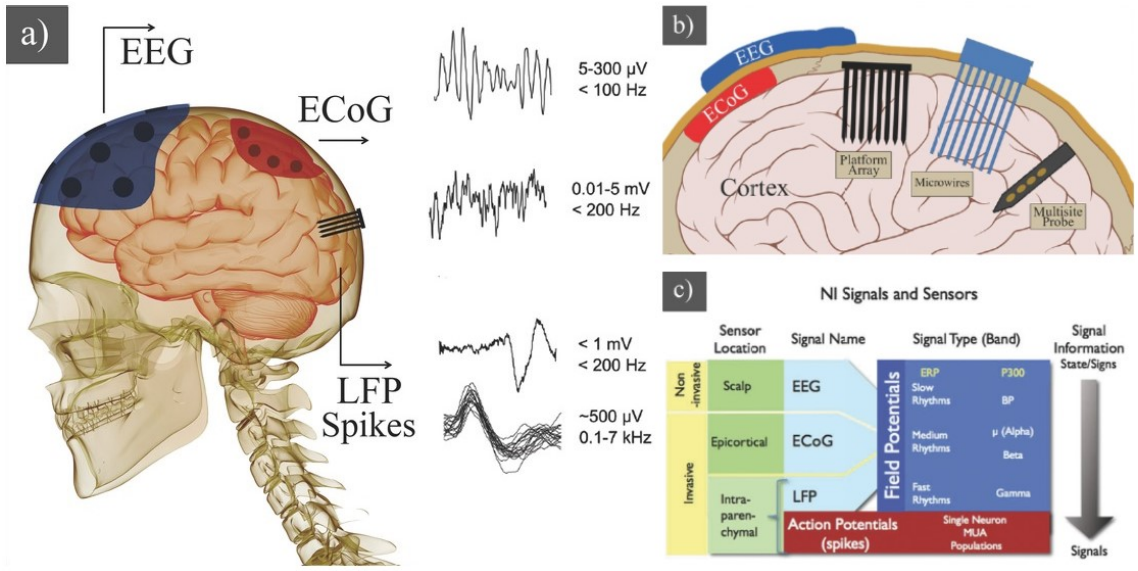


Figure 1.4. Overview of EEG, ECoG and MEA and their respective measurement resolutions, signal characteristics, and invasiveness.

Modified from: [Fattahi et al., 2014]

1.4 Overview of Different Diagnostic Imaging Methods

To condense all the specifications, advantages, and disadvantages of the different modalities, a table was created for easier comparison between EEG, ECoG/MEA, fMRI, and PET (see Table 1.2).

Table 1.2. Table overview of capabilities of different systems. Colors indicate the relative advantages and disadvantages of the different modalities, from the perspective of imaging neural activity. [Zolezzi et al., 2022; Seixas et al., 2013; Logothetis, 2008; Luck, 2014; Cohen, 2014; Murakami and Okada, 2006]

	fMRI	**PET**	**EEG**	**ECoG/MEA**
Neural Representation	Indirect (Blood flow Oxygenation)	Indirect Metabolic activity Neurotransmitter receptors	Direct Electrical activity	Direct Electrical activity
Temporal Resolution	Moderate to high (seconds)	Low (seconds)	Very high (milliseconds)	Very high (milliseconds)
Spatial Resolution	High resolution (1-2 mm)	Moderate to low resolution (4-6 mm)	Low resolution (10-20 mm)	Moderate
Degree of Invasiveness	Low	High	Low	High
Cost	Expensive	Very expensive	Inexpensive	Expensive
Features	BOLD	rCBF	δ to $\gamma - bands$	LFP and MUA

1.5 Classification of Pain

In the search for an objective pain biomarker, machine learning and deep learning algorithms could be a useful tool. If given pain-related data, the algorithms are capable of learning complex features specific to a known class. This enables them to identify cases of the same pain-related data, even when introduced to new instances. This makes it very useful when working with EEG signals that have a complex nature and are therefore not intuitively interpreted. In this case the algorithm can be seen as a tool for data

exploration, where the assumption is that if a feature is repeatedly used by the machine, then it is probably important for the related problem. [Lötsch and Ultsch, 2018]

Deep learning is a subset of machine learning based on artificial neural networks. It works by using multiple layers of interconnected nodes to analyse data, similar to how the human brain functions. Its main advantage to regular machine learning is its ability to extract patterns and features from large amounts of data without any manual feature engineering. Because of this, many neuroscience problems are being tackled with deep learning with the purpose of finding new information from the data. [Pathak and Kashyap, 2021]. Typically one of three algorithms are being used when trying to decode EEG signals; Convolutional Neural Network (CNN), Deep Belief Network (DBN) or Recurrent Neural Network (RNN) [Livezey and Glaser, 2021; Vallabhaneni et al., 2021; Craik et al., 2019].

All three models work differently but have the same purpose of extracting patterns in the data. CNN uses convolutional layers to learn spatial features in the data, which are then downsampled through pooling layers to reduce the complexity. In the context of EEG signals, the CNN is often used to learn features from time-frequency plots such as spectrograms or raw EEG signals. [Craik et al., 2019]. DBN is a neural network consisting of multiple restricted Boltzmann machines, which is a type of unsupervised learning model. This gives the model an advantage when working with unlabeled data. The RNN is normally used to process sequential data. It has both a feed forward and backwards connection, which makes the model robust and very suitable for time series data. [Vallabhaneni et al., 2021]

Choosing which algorithm to use can be difficult since they all show good results in decoding EEG signals. When looking at review studies, it is clear that CNN is by far the most used algorithm. This makes sense because of its ability to detect localized features and translation invariant features and is good at handling noise and variability in the data. It is also a more computationally efficient model than most, because of its use of sparse interactions and parameter sharing. [Livezey and Glaser, 2021]. The downside of CNN is that because it is translation invariant, the model does not have any temporal memory of where the features are extracted from the signal. This can be a problem when working with time-dependent data such as EEG, where temporal patterns could be seen as an important feature. This can be taken into account by combining CNN with other models. A good combination could be CNN and the RNN model called Long Short-Term Memory (LSTM) since the CNN is great at spatial features while LSTM is good at finding temporal dependencies in the data. [Amrani et al., 2021; Ghosh et al., 2021]

A combination model like the CNN-LSTM model could reduce some of the shortcomings of one model alone and therefore be better suited for the complex problem of classifying chronic pain. However, the drawbacks of the combination are a furthering in the typical

concerns with deep learning algorithms. The computational complexity will be increased, with added training time and a large number of parameters. The interpretability of the model gets worsened, since it becomes harder to interpret what features the model learned and how it uses them to make a classification.

1.6 Aim

The aim of this project is to address the challenges associated with diagnosing chronic pain patients by developing objective biomarkers. Specifically, the project aims to differentiate between chronic pain models. Given the complexity of analyzing pain-related ERPs, the project intends to employ a deep learning model to identify and establish reliable biomarkers for this purpose.

How can a CNN-LSTM architecture classify between chronic pain models based on ERP signals from μECoG recordings in S1?

Structured Literature Search 2

This chapter provides insights into the process of structuring two literature searches: one on diagnosis of chronic pain and another on machine learning models for EEG State Classification. It delves into the initial search, keyword structuring, and the screening process, elucidating the methodology employed to conduct both searches. Furthermore, it presents the results obtained from the screening process for each search.

2.1 The Initial Search

Before conducting the structured literature search, an initial research question was devised as follows: *What problems are associated with the diagnosis of chronic pain?* This question was chosen since the purpose of this literature search was to get a broader understanding of the problem domain, that is diagnosing chronic pain.

To start with, an initial search was done to get a general overview of the available literature on the topic. This search was performed as a free search in scientific databases such as PubMed and IEEE, where the following keywords were used: EEG, MRI, biomarker, diagnostics, neuropathic pain, and chronic pain. The most relevant studies were used to identify common search terms/keywords for the AND/OR table used in the structured literature search. With new information gained from the initial search, a new research question was made for the structured literature search: *How can brain activity be used as a biomarker for chronic pain?* With this new question, the objective of the structured literature search was to explore the known biomarkers for chronic pain or NP that can be found when solely looking at brain activity.

2.2 AND/OR Table

From the initial search, an AND/OR table was constructed with four categories: 'Neuroscience' (the area of interest), 'Physiology', 'Type of experiment', and 'Type of measurement'. The AND/OR table was then used on the databases PubMed and Embase using their specific search syntaxes. The database IEEE was also tried, but no new articles were found and therefore the database were excluded. The AND/OR table can be seen in Table 2.1.

Table 2.1. The AND/OR table shows the combination of keywords used in the structured literature search.

Time of search: 10-02-2023	**AND**			
OR	Neuroscience (area of interest)	Physiology	Type of experiment	Type of measurement
	Diagnostic[tw]	"Neuropathic pain"[tw]	"Peripheral Nerve Injuries" [MeSH]	Electroencephalography [MeSH]
	Biomarkers[Mesh]	"Chronic pain" [MeSH]	Evoked potentials [MeSH]	eeg[tw]
	"Neural activity" [tw]	"Nociceptive Pain" [MeSH]	Evoked response [tw]	Magnetic Resonance Imaging [MeSH]
	Neuronal oscillation [tw]	"Chronic neuropathic pain" [tw]	Electrical stimulation [MeSH]	Magnetoencephalograph [tw]
		Nociception [tw]	"pain-eliciting stimulus" [tw]	"Electrocorticography" [MeSH Terms]
		"Hyperalgesia" [MeSH]	"pain stimulation intensity" [tw]	"microelectrodes" [MeSH]
		"Allodynia" [tw]	"Long-Term Potentiation" [MeSH]	
			"Somatosensory evoked potential" [tw]	
			Laser-Evoked Potentials [MeSH]	
			Nerve injury[tw]	
NOT:	Diabetes Mellitus, Drug			
Filters:	2000-2023, english, full-text available, NOT conference abstract. Conference papers from 2020 - 2023			
Results Pubmed:	150			
Results Embase:	122			

To guarantee the inclusion of relevant, accessible, and readable studies, various screening criteria were used. Initially, a full-text filter was utilized to only include articles that could be accessed in their entirety. Additionally, an English filter was implemented to only incorporate articles written in English. Furthermore, the time frame of the search was restricted to journal articles published between 2000 and 2023, and conference papers published between 2020 and 2023. The search was carried out on the PubMed and Embase databases, resulting in the identification of 150 and 122 articles, respectively.

2.2.1 Screening Process

Screening the studies found using the keywords using the AND/OR table was carried out in two parts, the first was an abstract screening, and the second full-text reading. The search

yielded a total of 272 articles, of which 17 duplicates were identified and subsequently eliminated. As a result, 255 articles remained for abstract screening. The full screening process can be seen in Figure 2.1.

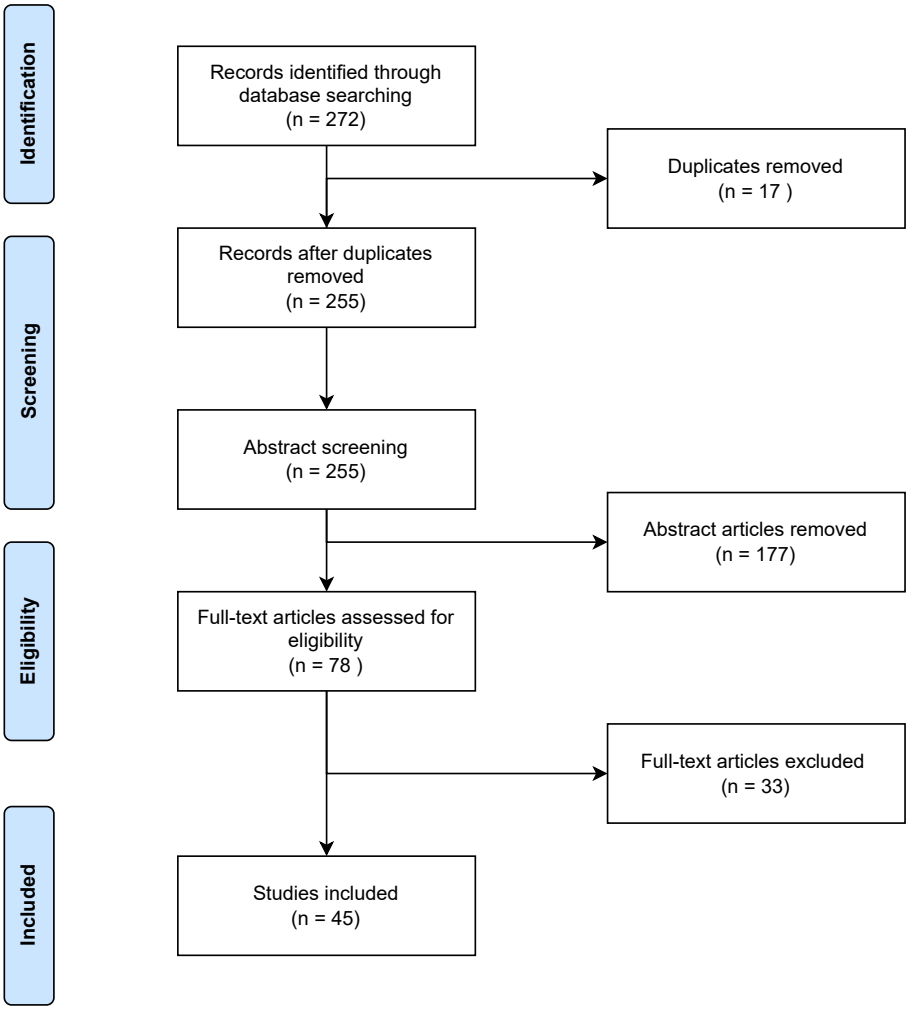


Figure 2.1. PRISMA-flowchart showing the screening process. The left column shows the steps and the number of articles left, whereas the right column shows the number of articles and at which step in the process they where removed.

To ensure all articles were judged on the same basis, inclusion and exclusion criteria were used throughout the screening process. These criteria can be seen in Table 2.2. Furthermore, to ensure thoroughness and minimize bias, a minimum of two reviewers were required to agree on whether an article should be included or excluded. In case of disagreement, a third reviewer was brought in to make the final decision.

Table 2.2. Table of inclusion and exclusion criteria for the screening process. I and E denote inclusion and exclusion criteria, respectively.

Criteria Index	Criteria
I1	Biomarkers for chronic/neuropathic pain
I2	Diagnosis of chronic/neuropathic pain
I3	Chronic/neuropathic pain model
E1	Studies from before 2000
E2	Non-English studies
E3	Duplications
E4	No full-text access
E5	Drug effect
E6	Solely regarding a sickness
E7	BCI studies
E8	Out of scope
E9	Case report/study
E10	Stimulation as treatment
E11	Receptor focus
E12	Behaviour/empathy studies
E13	Spinal cord/nerve imaging
E14	Migraine
E15	Acupuncture
E16	Anticipatory pain
E17	Transcranial magnetic stimulation

After the abstract-based screening, 126 articles were excluded where the primary reasons were: solely regarding a sickness, stimulation as treatment, migraine, and out of scope. This left a total of 78 articles to be considered for the full-text screening. During the full-text screening process, each reviewer had access to the complete article to make an informed judgment of it. Of the 78 articles, 33 were excluded where the main reasons for exclusions were: out of scope, not biomarkers for chronic/neuropathic pain, no full-text access, and transcranial magnetic stimulation. The majority of the excluded studies based on out-of-scope were studies that focused on visceral pain, pain perception without stimulation, and a single study that was retracted. The remaining 45 studies were included. During the process of full-text screening, each reviewer wrote a summary of the articles they thought to be included, where they noted the following: title, aim, biomarkers and pain model, and an argument for why the article should be included. The template for the full-text screening table is shown in Figure 2.2.

Title	Resume (aim of the study)	Applied methods	Argument for inclusion

Figure 2.2. Full text table for included articles that was written for each included article.

Reused from: [Clark et al., 2022]

2.3 Structured Search on Machine Learning Models for EEG Classification

After finishing the first structured literature search, a new question emerged about what types of algorithms, and how they are used to classify based on EEG signals. A search question was formulated as: *What machine learning models are used in the classification of different EEG states?* The reason for looking into "EEG states" instead of e.g. "EEG in pain research" was because the use of machine learning in pain research is fairly limited when comparing it to the use of it in EEG research in general. Another argument is that EEG is a very complex signal, so if one field of EEG research has great experience with a type of model, it could also be valuable in other fields. The AND/OR table constructed for the literature search can be seen in Table 2.3. In this search, only reviews were wanted, since the purpose of the search was to get a general overview of which types of models could be useful for classifying EEG signals. Furthermore, only reviews from the last 10 years were wanted, since it is a field that is developing a lot.

Table 2.3. The AND/OR table shows the combination of keywords use in the structured literature search.

Time of search: 03-03-2023	**AND**		
OR	Neuroscience (area of interest)	Type of experiment	Type of measurement
	Electroencephalography [MeSH]/[Emtree]/[Index]	"Machine Learning" [tw] "Deep Learning" [tw] Artificial Intelligence [MeSH] "Artificial Neural Network" [tw] "Neural Network" [tw]	Classification [tw] "Signal analysis" [tw] Features [tw] Decoding [tw]
NOT:	Survey		
Filters:	2013-2023, english, full-text available, conference papers from 2020-2023, review		
Total results PubMed:	64		
Total results IEEE:	104		
Total results Embase:	174		

2.3.1 Screening process

The screening process was similar to one from Section 2.2.1, with first an abstract screening and then afterward a full-text screening. In total 342 studies were found through the search of which 45 were duplicates. This gave a total of 297 studies to be screened. The full process can be seen in Figure 2.3.

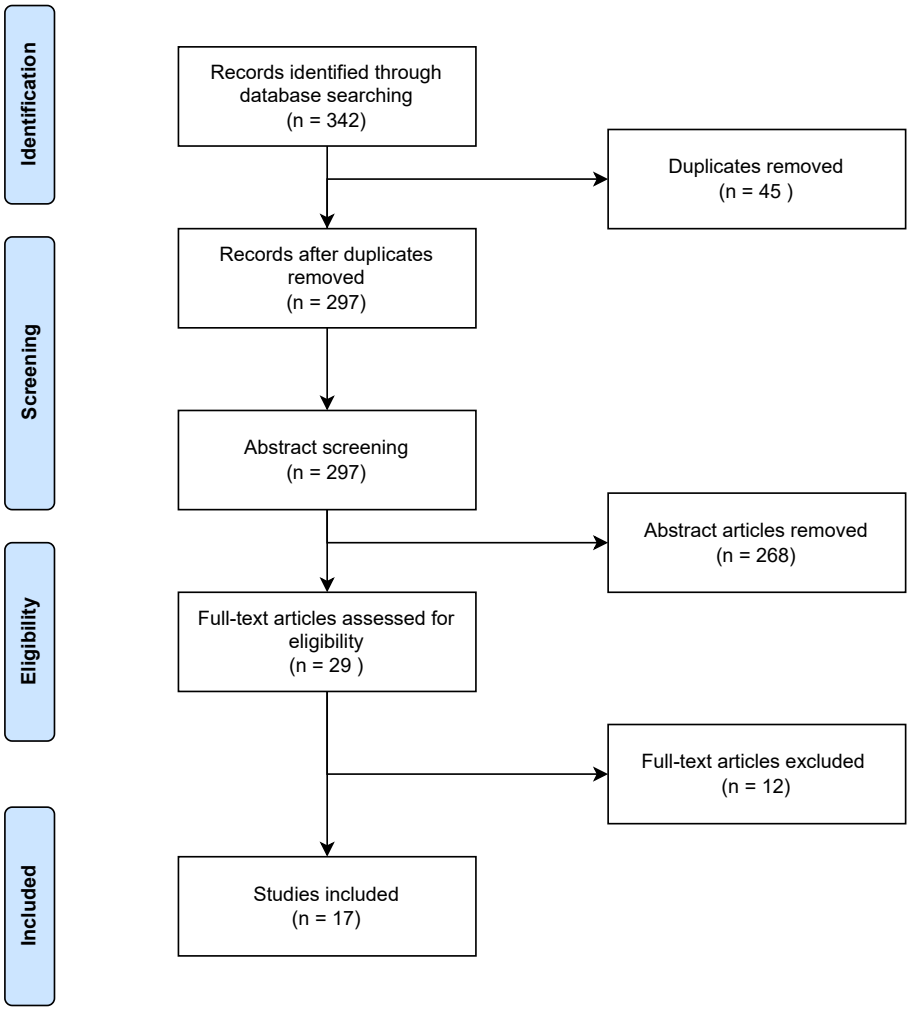


Figure 2.3. PRISMA-flowchart of the screening process. The left column shows the steps and the number of articles left, whereas the right column shows the number of articles and at which step in the process they were removed.

After the abstract screening, 268 articles were excluded, leaving a total of 29 articles to be considered for the full-text screening. From the full-text screening of the 29 articles, 12 were excluded and the remaining 17 were included. The exclusion of the studies were based on the inclusion and exclusion criteria seen in Table 2.4. The main reasons for exclusion were BCI, epileptic, or seizure studies.

Table 2.4. Table of inclusion and exclusion criteria for the screening process. I and E denote inclusion and exclusion criteria, respectively.

Criteria Index	Criteria
I1	Review on EEG classification with machine learning
E1	Studies from before 2013
E2	Non-English studies
E3	Duplications
E4	No full-text access
E5	Survey study
E6	Focus on a specific disease/disorder/state
E7	Imagery
E8	BCI
E9	Sleep classification
E10	Speech classification
E11	Out of scope
E12	Emotion
E13	MRI, CT or PET

Experimental Protocol 3

In this chapter the protocol and procedure of the experiment will be explained from which the data used in this project were acquired. This includes the experimental setup and stimulation protocol.

3.1 Protocol and Procedure of the Experiment

The experiment was conceived and carried out by the Center for Neuroplasticity and Pain (CNAP) at Aalborg University. It had approval from the Danish Veterinary and Food Administration under the Ministry of Environment and Food of Denmark (protocol number: 2020-15-0201-00514).

The experiment involved 16 Danish landrace pigs that were subjected to different conditions. Some encountered a spared-nerve injury (SNI) model, others were exposed to high-frequency stimuli (HFS), and a subset served as the control group. Before commencing the surgical procedures, measurements were set up using the Synapse software developed by Tucker-Davis Technologies (TDT).

Once anesthetized and ready for surgery, a cranial window was carefully formed via an incision just above the S1 region. Anatomical landmarks; bregma, midline suture, and coronal suture were used to locate S1. The μECoG electrode was subsequently positioned on the dura mater. Figure 3.1 provides a visual representation of the cranial window's placement over the S1 area.

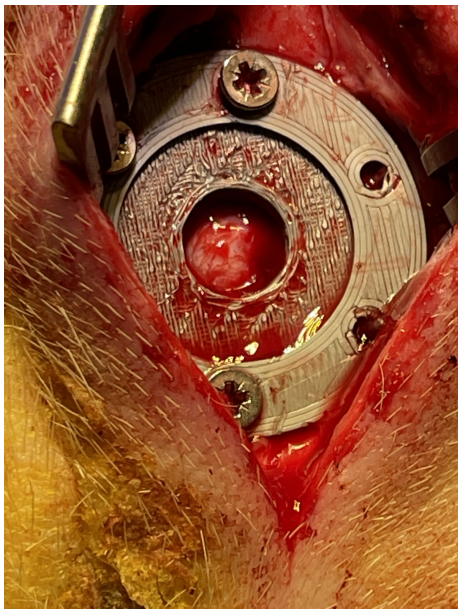


Figure 3.1. Cranial window placed above S1 in one of the subjects.

The ulnar nerve (n. ulnaris) of the pig was subjected to electrical stimulation through a pair of cooner wires, implanted subcutaneously. Before placing these wires, an identification process was performed to accurately locate and orient the nerve. During the phase of continuous stimulation, the wires were situated over the nerve.

Ascertaining the motor threshold involved a systematic modification of the stimulation amplitude. The process began at 50 μA, followed by stepwise increases of 200 μA until a distinct movement was noticeable. Subsequently, the amplitude was decreased in 50 μA steps until the absence of movement, after which it was boosted by another 50 μA to confirm the threshold.

The stimulation pattern used was an asymmetrical, rectangular, biphasic pulse with charge balance, where the second phase amplitude was 10% that of the primary phase. A programmable stimulator (STG4008, Multichannel Systems, Reutlingen, Germany) was employed for the delivery of stimulation.

The stimulation amplitudes fell into two categories: non-noxious and noxious. The non-noxious stimulation was set at twice the motor threshold, while the noxious stimulation was defined as ten times the motor threshold. The experiment also included a procedure to induce long-term potentiation (LTP). This procedure involved setting the stimulation amplitude (maintaining a 1:1 biphasic ratio) at 20 times the motor threshold, applied at a frequency of 100 Hz for 1 second, repeated 10 times.

3.2 Measurement Protocol

Each experimental trial was structured into four separate measurement periods and an intervention phase. Each of these measurement segments spanned 30 minutes and was subdivided into three data collection sets, separated by three pauses. In each data collection set, 100 instances of both non-noxious and noxious stimulations were applied, over a duration of 6 minutes, followed by a pause of 4 minutes. At the onset of each measurement, a 30-second baseline recording was made. The participants who were part of the intervention group received the LTP or SNI directed at the radial nerve during the intermission between the initial two measurement segments. The collected data was sampled at a rate of 6.1 kHz.

An illustration of the different measurement phases can be seen in Figure 3.2.

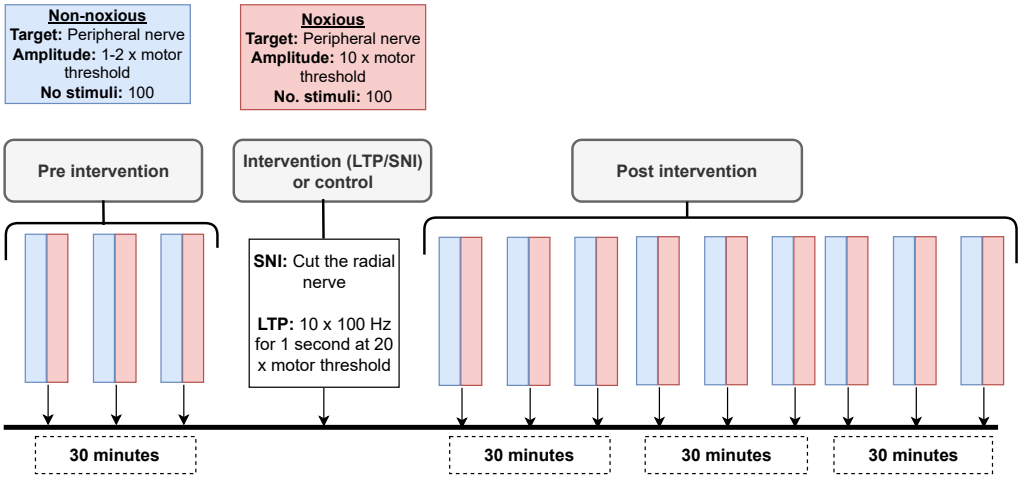


Figure 3.2. Protocol for stimulation for pre intervention, intervention/control, and post intervention. The blue and red squares are non-noxious and noxious stimuli, respectively.

Pre-proccesing 4

This chapter will include a description of how much data was available and how it was stored. Afterward, the steps of the pre-processing is explained. The purpose of this process is to convert the raw data files into spectrograms based on continuous wavelet transforms.

4.1 Data Management & Signal Processing

The data was in folders of measurements in a file format from TDT. The μECoG recordings were sampled with a frequency of 6103 Hz, using 32 electrode array. The data was then opened in Matlab (Version 2019b), to extract the time series data and onset timer for the stimuli for each experiment and saved as a .mat file. In order to minimize the size of the data, it was downsampled by a factor of three, resulting in a sampling frequency of 2034 Hz. The factor of three was found based on the highest expected frequency of interest (200 Hz), where 10 times the highest frequency of interest would be more than sufficient to accurately represent the signal without introducing aliasing, thereby satisfying the Nyquist theorem.

The .mat files were saved as a dictionary with 2 elements. First the time series data: Stimuli type (NonnoxERP or NoxERP), channels (0-31), blocks (0-3), and a 2D array containing the stimulations and time series data. The second element was the onset timers of when in the time signal, the stimulations were given.

Data from a total of 22 experiments were included in this project, where the majority had four blocks of data. The full list of data can be seen in Table 4.1. A total of 49806 epochs each measured with 32 channels gives a total of approximately 1.6 million epochs. Note that for the LTP and SNI experiments, the first 600 epochs were only used as control data since the intervention was performed after the first block of measurements.

Table 4.1. Overview of the available data for the three groups: LTP, control, and SNI. Note that the first block in LTP and SNI experiments, which contains 600 epochs, was control and was not used as an intervention group.

Experiment	Number of epochs available
LTP1	2400
LTP2	2400
LTP3	2400
LTP4	2400
LTP5	2462
LTP6	2400
Control1	2400
Control2	2400
Control3	2400
Control4	1800
Control5	2400
Control6	2400
Control7	2400
Control8	1800
Control9	2400
SNI1	2400
SNI2	2400
SNI3	2400
SNI4	2400
SNI5	2400
SNI6	2400
Total	**49806**

4.2 Initial Inspection of μECoG Data

To perform the initial inspection of the μECoG data the dictionary was loaded into Python (version 3.10.6) with scipy (version 1.9.3) function 'scipy.io.loadmat()'. To plot the time series data, the Python package matplotlib's 'pyplot' was used. As seen in Figure 4.1, the data had extensive noise, which needed to be removed. To this end, a frequency analysis of the initial inspection was performed using Magnetoencephalography and Electroencephalography (MNE) Python (version 1.2.2) which is an open-source Python package for analyzing, visualizing, and processing EEG and MEG data. [MNE, 2023a]. To plot the power spectral density (PSD) of the original time series data, MNE's 'mne.viz.plot_raw_psd' function, as seen in Figure 4.2, was used. The majority of the noise was found at 50 Hz including harmonics.

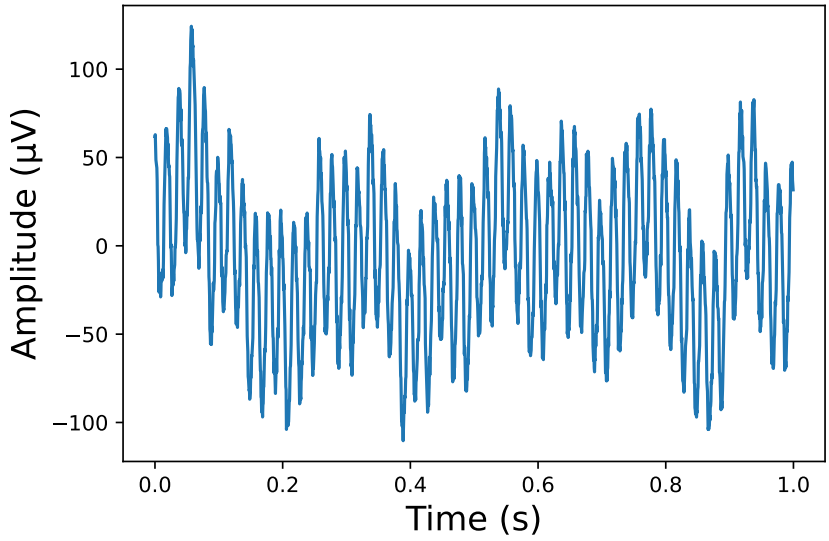


Figure 4.1. Time series signal of the original data before filtering. The noise in the signal is mostly 50 Hz.

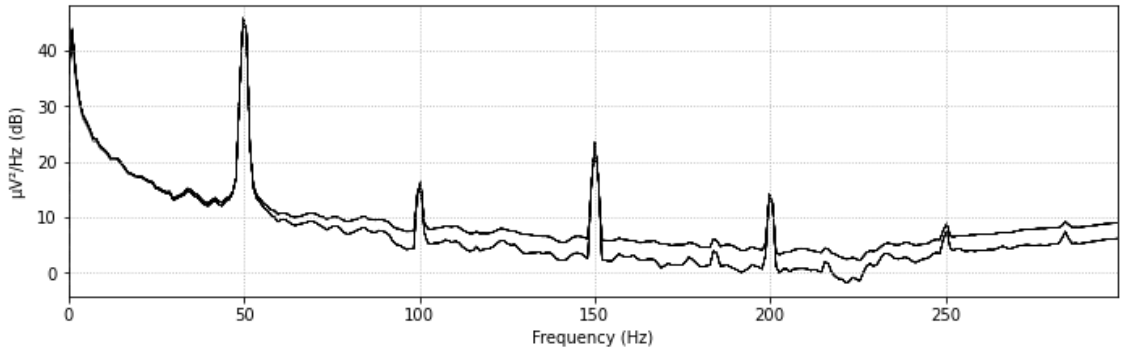


Figure 4.2. Power spectral density of the original unfiltered data from 0 - 300 Hz. Note the extensive noise at 50 Hz and the following harmonics every 50 Hz.

4.3 Pre-processing of μECoG Signals

One of the challenges when working with μECoG signals is noise and artifacts because of the low amplitude of the brain signals. Typically these include environmental, instrumental, and physiological noise which is noise from the body that is not relevant to the experiment like muscle, eye, and heart signals. For this purpose, filters are used. Filters are a function that can attenuate unwanted frequencies or parts of a signal by weighting several inputs to produce an altered output. [de Cheveigné and Nelken, 2019] The equation for a digital filter is seen in 4.1.

$$y(t) = \sum_{n=0}^{N} h(n)x(t-n) \tag{4.1}$$

Where t is the analysis point in time, and h(n) is the impulse response of the filter for each point from n to N.

There are typically four different types of filters; low-pass, high-pass, band-pass, and notch, which can be seen in Figure 4.3

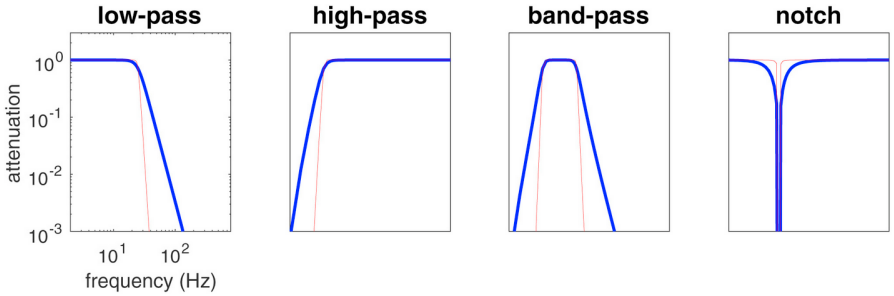


Figure 4.3. Examples of low-pass, high-pass, band-pass and notch filters. The low-pass, high-pass, and bandpass filters are 4th order (blue) or 16th order (red) Butterworth filters. The notch filters are second-order filters with Q-factors of 1 (blue) and 10 (red).

Modified from: [de Cheveigné and Nelken, 2019]

The low-pass filter attenuates higher frequencies than the band allows. This is often used on EEG signals since high-frequency variance can often be seen as irrelevant to the information in the signals. Low-pass filters can also be used to smoothen the signal so the longer-term trends in the signal become explicit. The high-pass filter attenuates lower frequencies than the band allows. This is relevant for neuroelectrical signals since they are susceptible to DC shifts and low drifts. The band-pass filter is a combination of a low-pass and high-pass filter and attenuates frequencies outside the band. This is practical to isolate a specific band or to attenuate noise outside of a targeted band. A notch filter attenuates frequencies in a narrow band. This is often used in signal-processing to attenuate 50 or 60 Hz power line noise and harmonics of this. Notch filters have a Q-factor which is a ratio of bandwidth to center frequency. It is used to determine how many Hz around the center frequency to attenuate. Notch filters are ineffective in removing noise at the end of a signal, which is why applying the notch filter on the whole time series signal is recommended rather than on epoched data. To ensure that the phase of the signal is not changed during filtering, the filter can be applied forwards and backwards, in which the effective filter order is doubled. This way, the filter becomes a zero-phase filter. [de Cheveigné and Nelken, 2019]

4.3.1 Filter Implementation

To design the band-pass filter, MNE's filter method was used. The band-pass filter is a zero-phase symmetric linear-phase FIR filter, where the FIR design was made with 'scipy.signal.firwin'. The FIR design had a windowed time domain, where the hamming window was chosen. This window type was chosen based on its stable nature and low spectral leakage [Gupta and Panghal, 2012]. The band-pass cutoff frequency was chosen

to be 1 - 200 Hz since the expected signal frequencies in an μECoG on dura are less than 200 Hz. [Fattahi et al., 2014]. The notch filter is based on the same method as MNE's filter method. The center frequency was chosen to be 50 Hz with a 2 Hz cutoff frequency on each side, making the stop-band of the notch filter from 48- 52 Hz. Since the initial inspection showed noise in harmonics of 50 Hz, the same filter was applied at 100, 150, and 200 Hz with the same cutoff around the center frequencies. As seen in Figure 4.4, the filters had the desired effect since the noise is attenuated notably compared to the remaining frequencies.

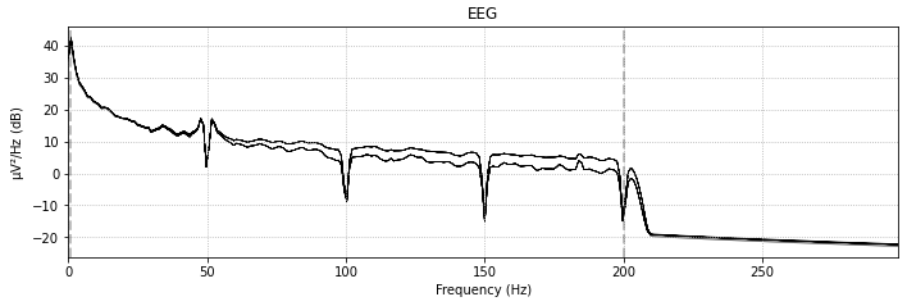


Figure 4.4. The power spectral density of the signal after filtering. The 50 Hz including harmonics have been attenuated, as well as the frequencies outside the bandpass from 1 - 200 Hz.

In order to check if the stimulation response was clear in the time series data, a mean of 20 epochs was calculated after band-pass and notch filtering. This can be seen in Figure 4.5, where there is a clear response after 200 ms.

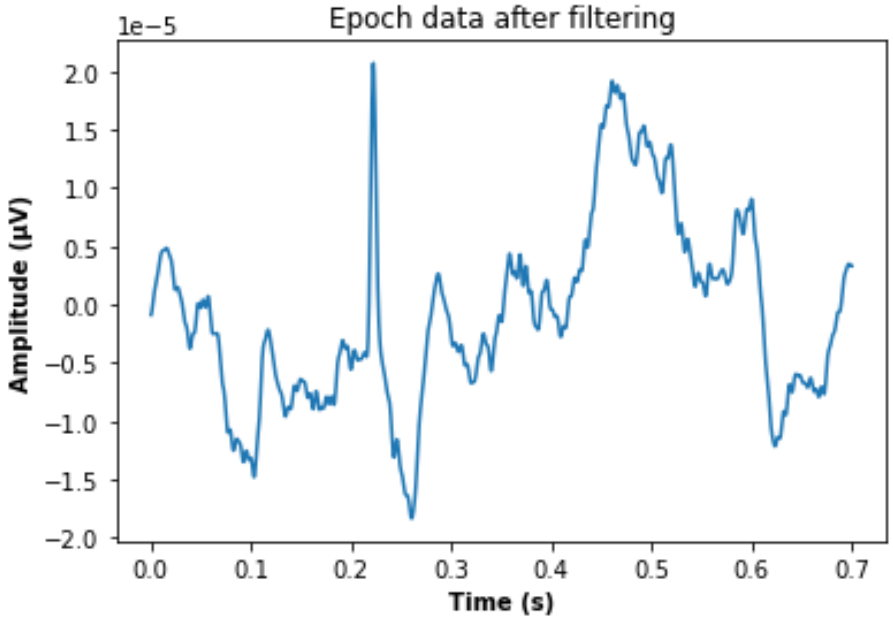


Figure 4.5. An example of 20 epochs averaged together after band-pass and notch filtering. The stimulation response can be seen just after 200 ms in the epoch.

4.3.2 Removal of Noisy Channels & Epochs

After filtering the data, it was epoched into 200 ms before the stimulation and 500 ms after stimulation, resulting in a 700 ms signal for each epoch. The identification of noisy channels and epochs were needed to ensure that only good quality data was given to the network. This was done by plotting the 32 channels using MNE's visualization tool. The channels were plotted on the y-axis, whereas the epochs were plotted on the x-axis. This way, only the noisy channels, and epochs were removed while maintaining as much of the data as possible. An example of two noisy channels can be seen in Figure 4.6 whereas noisy epochs can be seen in Figure 4.7

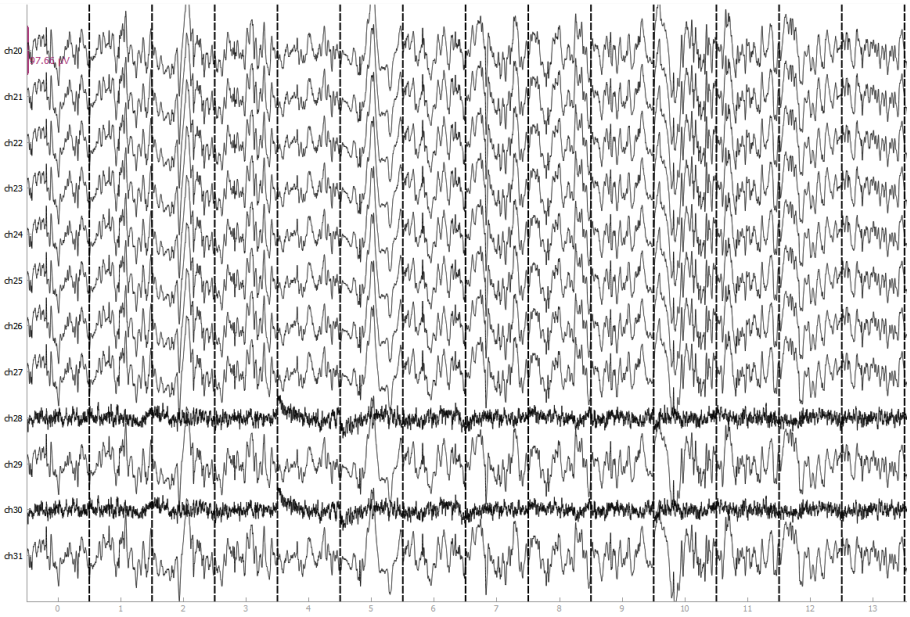


Figure 4.6. Examples of noisy channels. Channels 28 and 30 were noisy, which can be seen as spiky even after filtering the signals.

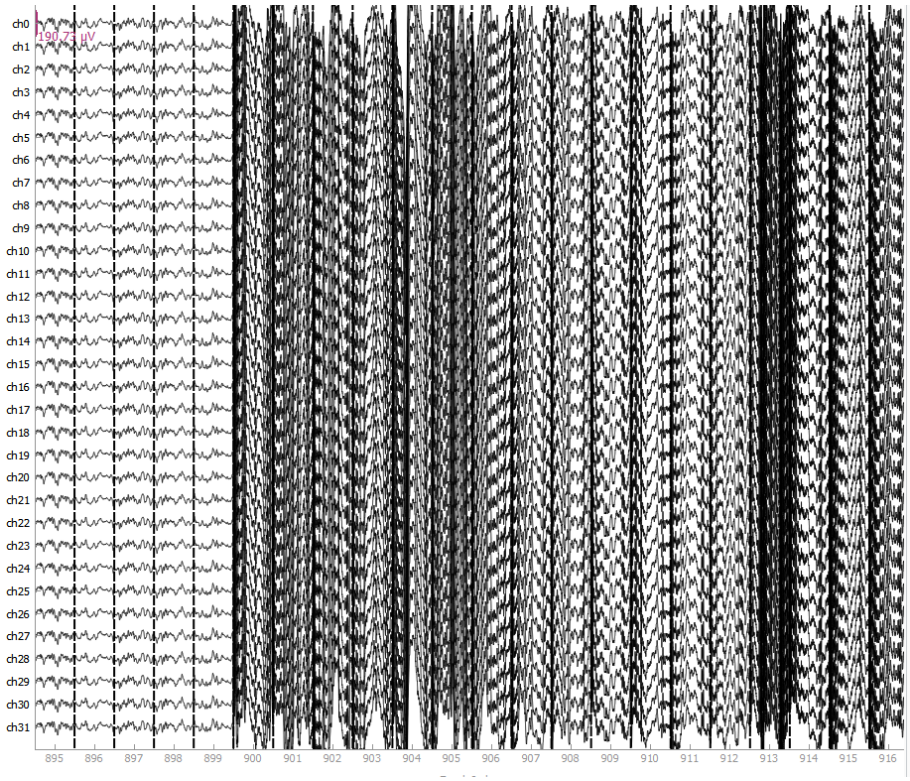


Figure 4.7. Examples of noisy epochs from epoch number 900 and forwards. The noise can be seen in the spikes of the signal that occur after filtering.

The noisy channels and epochs identified from the available data can be seen in Table 4.2. Note that the total number of epochs at the end of the table was for each channel that was not removed. The cumulative count of noisy epochs entailed eliminating the individual epochs across all 32 channels. The total number of epochs used in the project was:

- LTP intervention: 232.203 epochs
- Control: 530.712 epochs
- SNI intervention: 199.176 epochs

Table 4.2. Overview of the noisy channels and epochs in the available data. At the end of the table is the total number of removed channels and epochs, as well as the included epochs in each group (Control, LTP intervention, and SNI intervention. Note that the total number of included epochs is for each of the included channels for the respective experiment.

Experiment	Noisy channels/epochs	Number of epochs used(NI/I)
LTP1	Noisy channels: None Noisy epochs: 280	Control: 599 Intervention: 1521
LTP2	Noisy channels: 26,28,30 Noisy epochs: 692	Control: 87 Intervention: 1631
LTP3	Noisy channels: All Noisy epochs: All	Control: 0 Intervention: 0
LTP4	Noisy channels: 1,13,17,19,22 Noisy epochs: 0	Control: 600 Intervention: 1800
LTP5	Noisy channels: 1,13,16,17,18,19,20,22,25 Noisy epochs: 3	Control: 600 Intervention: 1859
LTP6	Noisy channels: 13,16,17,18,19,20,22,25 Noisy epochs: 4	Control: 600 Intervention: 1795
Control1	Noisy channels: All Noisy epochs: All	Control: 0 Intervention: 0
Control2	Noisy channels: 15,17,19,22,28 Noisy epochs: 234	Control: 2166
Control3	Noisy channels: 17,19,22 Noisy epochs: 52	Control: 2348
Control4	Noisy channels: 11,15,17,19,22 Noisy epochs: 7	Control: 1793
Control5	Noisy channels:0:15,17,19,22 Noisy epochs: 2	Control: 2398
Control6	Noisy channels: 16,17,18,19,20,22,25 Noisy epochs: 1	Control:2399
Control7	Noisy channels: 16,17,18,19,20,22,25 Noisy epochs: 2	Control:2398
Control8	Noisy channels: 15,17,19,22 Noisy epochs: 3	Control: 1797
Control9	Noisy channels: 8,13,15:31 Noisy epochs: 0	Control:2400
SNI1	Noisy channels: 16,17,18,19,20,22 Noisy epochs: 399	Control: 200 Intervention: 1799
SNI2	Noisy channels: 16,17,18,19,20,22 Noisy epochs: 6	Control: 595 Intervention: 1798
SNI3	Noisy channels: 14,15,16,17,19,22,28 Noisy epochs: 232	Control: 445 Intervention: 1723
SNI4	Noisy channels:13,16,17,18,19,20,22,25 Noisy epochs: 23	Control: 600 Intervention: 1633
SNI5	Noisy channels:All Noisy epochs: All	Control: 0 Intervention: 0
SNI6	Noisy channels: 0:15,17,19,22 Noisy epochs: 0	Control: 600 Intervention: 1800
Total	**Noisy channels: 236** **Noisy epochs: 9140**	**Control: 22626** **LTP Intervention: 8606** **SNI Intervention: 8752**

4.4 Wavelet Transform

The wavelet transform is one of the most commonly used spectrogram methods when using EEG signals in conjunction with a CNN architecture. It transforms a time-domain signal into a time-frequency signal in the form of a spectrogram, which is a suitable input for the CNN. The wavelet has the advantage that it can have a high frequency resolution at low frequencies and a high time resolution at high frequencies. [Saeidi et al., 2021] This is an important property for signals like the EEG, since it contains high frequency components within short time periods and low frequency components within long time periods. [Mane et al., 2015]

An example of a wavelet is the Morlet wavelet. A Morlet is defined as a sine wave tapered by a Gaussian. For time-frequency analysis, a complex Morlet is used where the real-valued Gaussian tapers a complex-valued sine wave. [Cohen, 2019]

$$\psi_0(\eta) = \pi^{-1/4} e^{i\omega_0\eta} e^{-\eta^2/2} \tag{4.2}$$

Where ω_0 is the nondimensional frequency [Torrence and Compo, 1998].

The continuous wavelet transform (CWT) of a sequence x_n is the sequence convoluted with a scaled and translated version of the wavelet.

$$W_n(s) = \sum_{n'=0}^{N-1} x_{n'} \psi * \left[\frac{\left(n' - n\right)\delta t}{s} \right] \tag{4.3}$$

Where s is wavelet scale, n is a localized time index, x_n is a discrete sequence and ψ is the wavelet [Torrence and Compo, 1998].

The scale parameter is a term for the width of the wavelet. This means a large scale corresponds to a wide wavelet, which is advantageous for low frequencies, while a small scale means a narrow wavelet which is advantageous for high frequencies. The translation is used to determine the location of the wavelet along the time axis. Shifting the translation positively or negatively, the wavelet can be moved forward or backwards in time to analyze the signal. By using different translations the signal can be analyzed in different time resolutions [Torrence and Compo, 1998]. Visualization of the scale and translation effect on the wavelet can be seen in Figure 4.8.

The wavelet does however have limitations for how well the resolution can be in both time and frequency. This is because of the uncertainty principle, which states that a trade-off always has to be made between the frequency resolution and the time resolution. This means that a wavelet with a high frequency resolution will have a poor time resolution and vice versa. This is important to remember when choosing the scales. [Woyczynski, 2011]

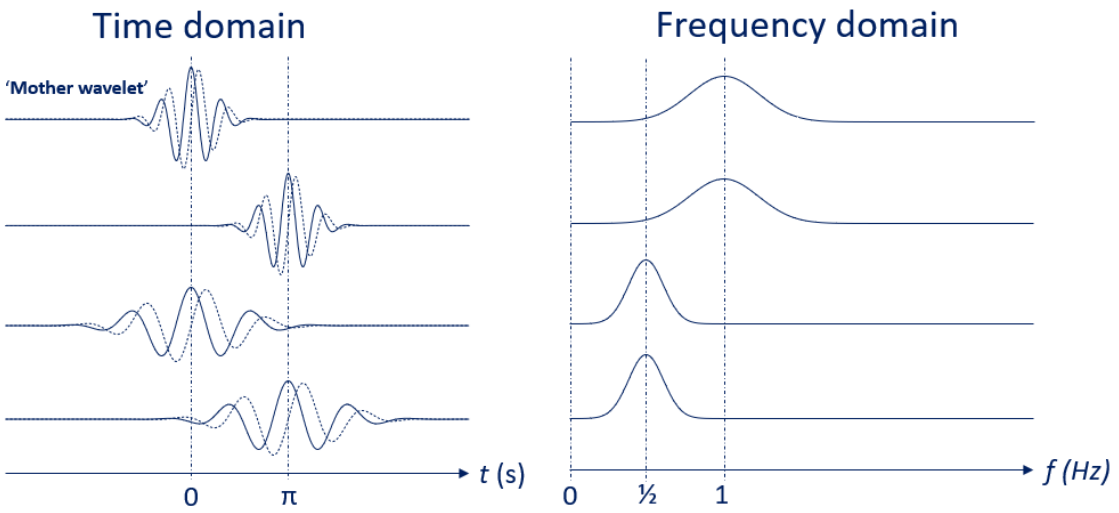


Figure 4.8. The figure shows a Morlet wavelet and the properties of scaling and translation. When the translations changes in time the frequency remains the same. When the scale changes the frequency domain changes while the time domain remains the same.

The Morlet wavelet has some advantages that make it a good choice for time-frequency analysis. For one it is Gaussian shaped in the frequency domain, which means there are not any sharp edges to create ripples. Secondly, the Morlet wavelet convolution retains the temporal resolution of the signal. Thirdly it is not as computationally heavy as some of the other wavelets, which is advantageous when working with large datasets. [Cohen, 2019]

4.4.1 Implementation of Continuous Wavelet Transform

The MNE package to Python was used to create the Morlet wavelets, through the function "mne.time_frequency.tfr_morlet(data, freqs, n_cycles)". The function is based on Mike Cohen's description of the temporal smoothing parameter of the wavelet called full width at half-maximum (FWHM) [Cohen, 2019]. This temporal smoothing parameter is defined as:

$$FWHM = \frac{n\_cycles \times \sqrt{2ln2}}{\pi \times freq} \tag{4.4}$$

Where n_cycles is the number of cycles and freq is the center frequency of the wavelet.

To implement the CWT for the project, the frequency range of interest and the number of cycles for each frequency were defined. The CWT was initially plotted with a frequency range of 0.5 - 200 Hz in steps of 1 Hz, aligning with the bandpass filter's settings of 0.5 - 200 Hz. This can be seen in Figure 4.9.

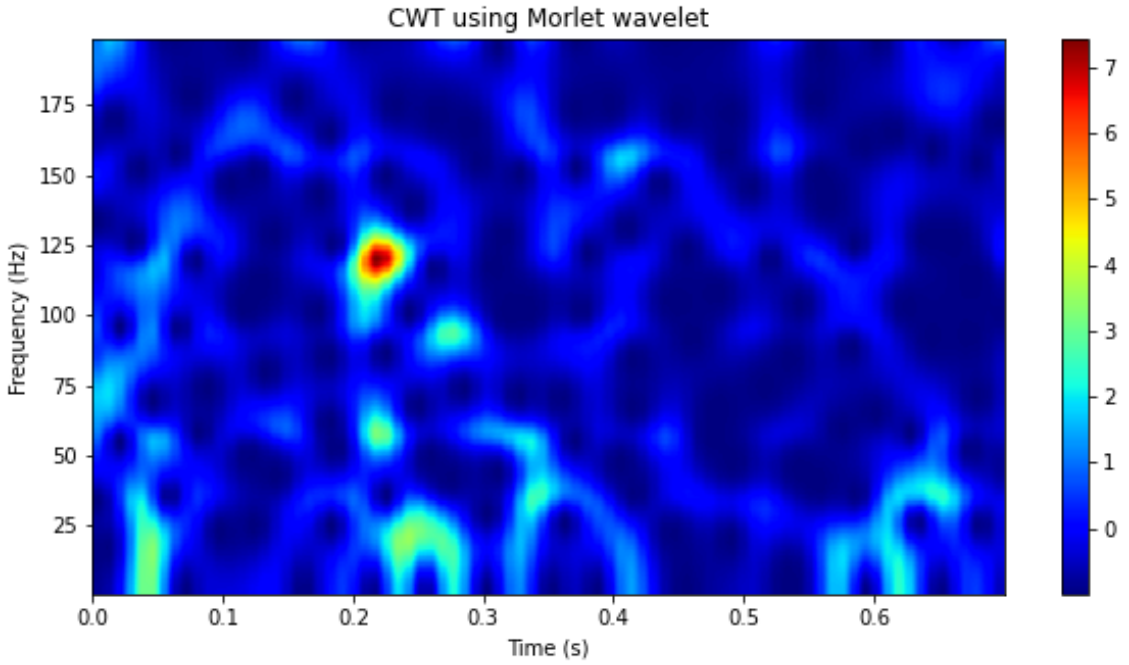
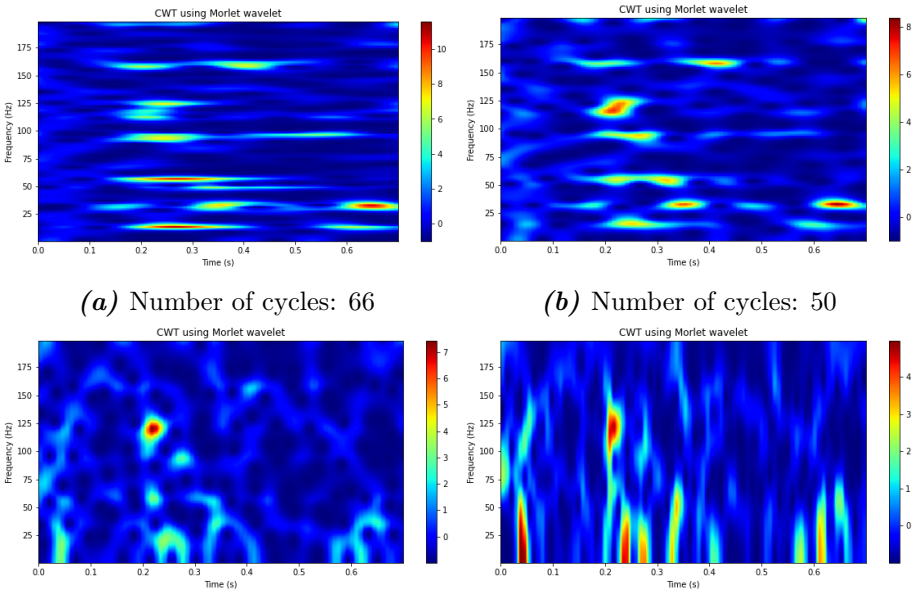


Figure 4.9. Shows a plot of the CWT with a frequency range from 0.5 - 200 Hz. Baseline (before the stimulation) of the signal is from 0 - 0.15 s. The colour bar shows the values of the pixels in the image.

The number of cycles is the trade-off parameter between the temporal and frequency resolution. A smaller number produces are higher temporal resolution while a higher number will result in high frequency resolution. To determine the number of cycles, multiple values were compared, as seen in Figure 4.10.



(a) Number of cycles: 66 ***(b)*** Number of cycles: 50

(c) Number of cycles: 20 ***(d)*** Number of cycles: 10

Figure 4.10. The figures shows plots of CWT where the number of cycles are different and thereby changes the time and frequency resolutions. The colour bars shows the values of the pixels in the images

By trying different number of cycles this trade-off between the time and frequency resolution could be evaluated. The number of cycles were therefore chosen to be 20, which can be seen in Figure 4.10a.

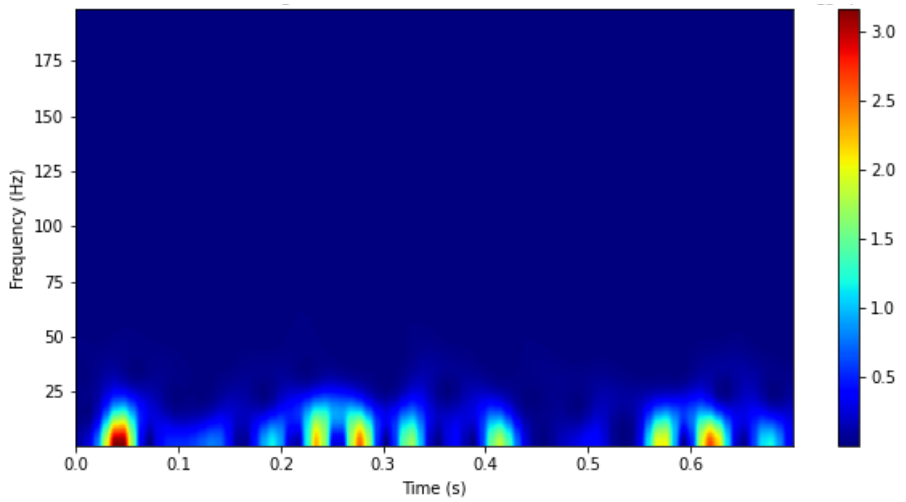
The resulting CWT was normalized with 1/f by using the baseline before the stimulation (the first 150 ms of the signal) to calculate a mean frequency vector. There are various methods for 1/f normalization. The chosen implementation was percentage change from the mean of the baseline, as this is one of the two methods recommended by Cohen [2014]. Percentage normalization can be calculated as seen in Equation (4.5).

$$\text{Percent Change} = \frac{B_f - X_f}{B_f} \tag{4.5}$$

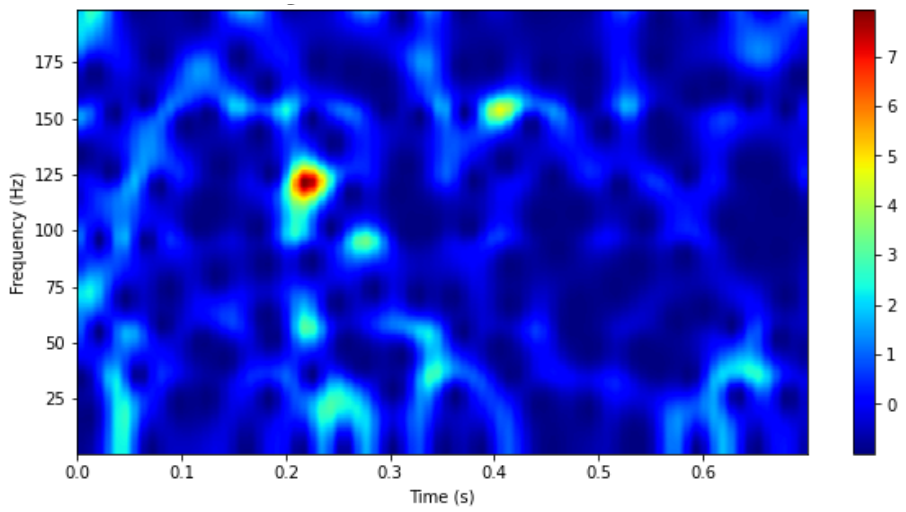
Where X_f is the value in the CWT for a given time and frequency value, and B_f is the average of the baseline for a given frequency.

This was implemented with MNEs' "apply_baseline" function using the percent baseline method.

The 1/f normalization's effect can be seen in Figure 4.11.



(a) CWT before using the baseline as normalization

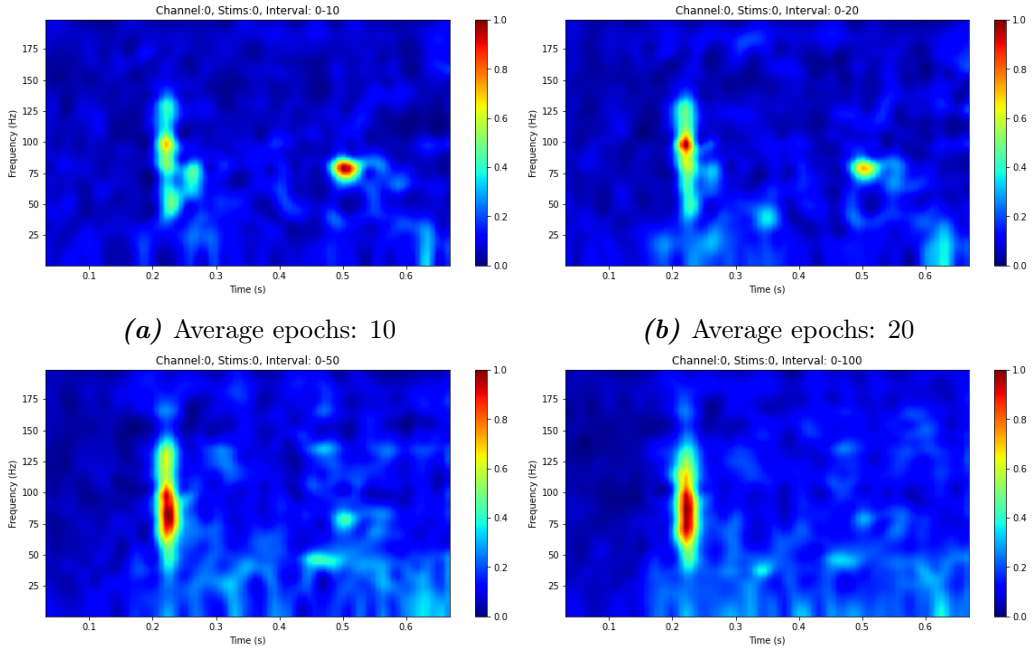


(b) CWT after using the baseline as normalization

Figure 4.11. The figures shows the effect of using the baseline to normalise the frequencies in the rest of the CWT. The colour bars shows the values of the pixels in the images

The CWTs are then normalized further between 0 - 1, to simplify the image and downsampled to reduce the array size. The downsampling was necessary since the data files would be too large for the computers used, to be able to train the deep learning algorithm. This resulted in a reduced frequency scale of factor 2, and the time scale was reduced with a scale of 16, resulting in an 82 x 100 image.

To further reduce the training time of the network and highlight the stimulation peak, the average of epochs was used. Figure 4.12 shows different numbers of epochs used for the averaging. The one used as the input was chosen to be the CWT where 20 epochs were averaged. This was chosen since there was a trade-off between the number of means that reduced the amount of data available and a stimulation response.



(a) Average epochs: 10

(b) Average epochs: 20

(c) Average epochs: 50

(d) Average epochs: 100

Figure 4.12. The figures shows different plots of a CWT of a channel with different averages of epochs. The colour bars shows the values of the pixels in the images

4.5 Temporal Window Length

As written in Section 4.4, wavelet has the advantage that it can have high frequency resolution at low frequencies and high time resolution at high frequencies. This can be implemented by decreasing the length of the temporal window. This can be calculated as seen in Equation (4.6), where the n_cycles is divided by the center frequency of the wavelet freq.

$$\text{Window length} = \frac{n\_cycles}{freq} \tag{4.6}$$

If the window length is constant, this means that the time and frequency resolution remains constant, whereas if the window length decreases, the time and frequency resolutions are scaled throughout the spectrogram. This can be seen in Figure 4.13

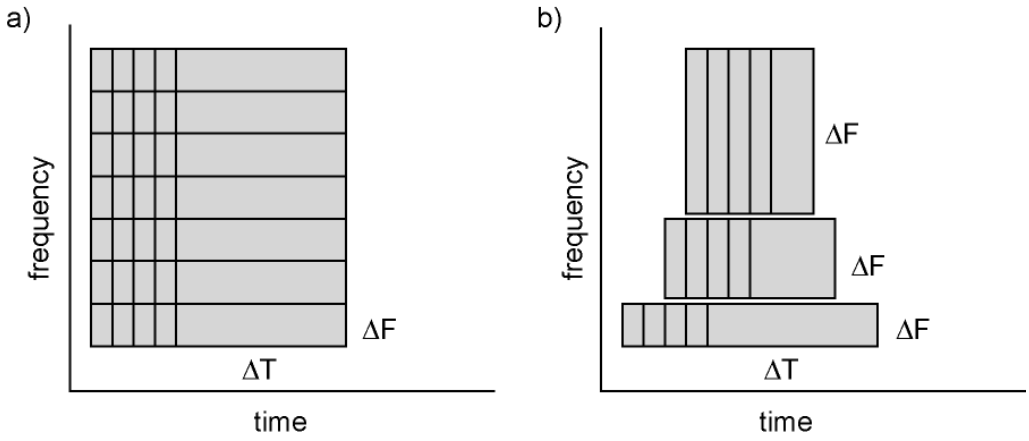
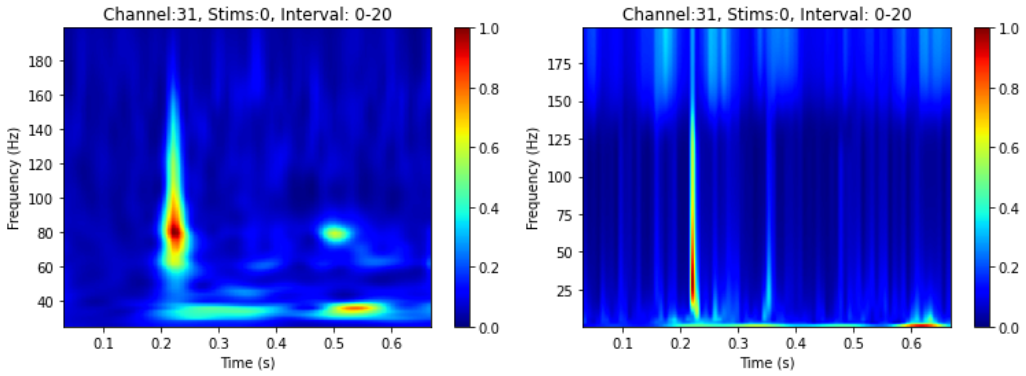


Figure 4.13. Illustration of fixed (a) or decreasing length (b) of the temporal window of the CWT.

Source: [MNE, 2023b]

The lowest frequency chosen for the wavelets was set to 0.5 Hz, which means that it takes the wavelet 2 seconds to complete one cycle. However since the epoch length was 0.7 seconds, this could not be calculated, since the whole cycle cannot be completed. Therefore, one of two options could be used. Either choose the lowest frequency possible for the signal to complete the specified number of cycles, or choose a lower number of cycles. Here decimal numbers can be used as the number of cycles. Both of these options were tested on the data.



(a) Center frequency from 25 - 200 Hz ***(b)*** n_cycles = 0.2

Figure 4.14. This figure shows the two different methods for implementing decreasing window length in the CWT. (a) shows the method where the center frequency ranges from 25 - 200 Hz. (b) shows where n_cycles was set to 0.2. The color bars show the values of the pixels in the images

Both of these options were tested on the data, as seen in Figure 4.14. In Figure 4.14a, the decreasing window length improved the frequency resolution at the lower frequencies and increased time resolution in higher frequencies which was preferred. However, the lowest frequency was 25 Hz, which cuts the majority of the traditional frequency bands off. In

Figure 4.14b, the number of cycles was 0.2, which resulted in better frequency resolution at the lowest frequencies, however at the cost of poor frequency resolution from around 20 Hz and upwards. This indicates that a decimal n_cycle results in an unstable transition in resolution between frequency and time.

As both methods resulted in suboptimal results, this was not implemented in this project.

Neural Network Architecture 5

This chapter will introduce the general idea of the final network. Furthermore, an overview of the fundamental concepts related to CNN and LSTM will be provided. To begin with, the overall architecture will be described, followed by a comprehensive explanation of the various layers within the network. The chapter also includes considerations in regards to the current literature of ERP based CNN applications.

5.1 General CNN-LSTM architecture

The study aims to construct a CNN-LSTM architecture, as described in Section 1.5, which leverages spectrograms as input to extract relevant features using convolution and achieve classification through fully connected layers. The following sections will provide a detailed explanation of the various modules comprising the overall architecture (CNN, LSTM, and fully connected). The proposed general network architecture is depicted in Figure 5.1.

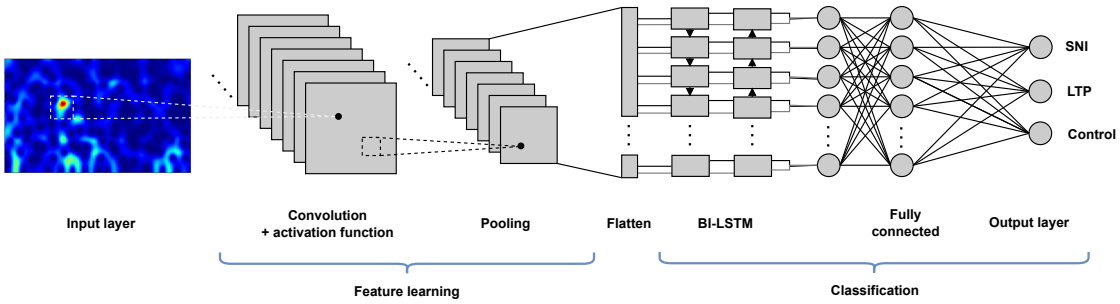


Figure 5.1. The general architecture of the CNN-LSTM model, showing the CWT as input, convolutional, LSTM and fully connected layers.

5.2 Convolutional Neural Network

5.2.1 General CNN Architecture

Designing the network architecture is a crucial step in developing any neural network as it significantly affects its classification performance. However, there is no one-size-fits-all solution for creating an optimal CNN architecture since it heavily depends on the application and the input to the neural network. Hence, it is crucial to evaluate various

hyperparameters and select the architecture that best fits the problem. [Duda et al., 2000; Nielsen, 2018]

The architecture of a CNN has three primary parts: the input layer, hidden layers, and output layer. In a CNN, the hidden layers serve two primary functions: they extract features from the input data through the use of convolutional blocks, and they classify these extracted features via fully connected layers. The design of the network's structure starts at the input layer. This is followed by a chain of convolutional blocks, which lead to a flattening layer. Following this, a number of fully connected layers are incorporated, all converging to the output layer. The design of this output layer is tailored to match the particular classification problem being addressed, which involves having an amount of neurons equivalent to the number of required classifications. Figure 5.2 illustrates the general CNN architecture. [Duda et al., 2000; Nielsen, 2018]

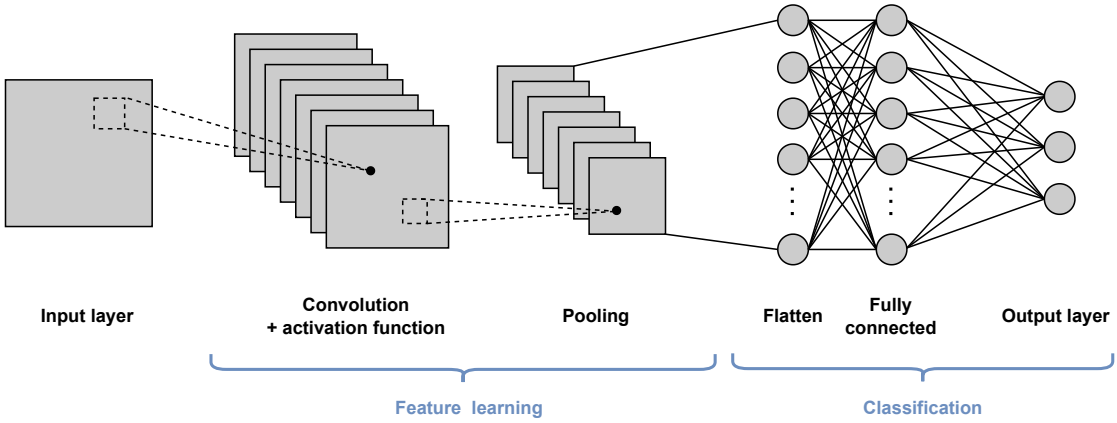


Figure 5.2. General CNN architecture that serves as the basis for the final design. This setup comprises of an input layer, convolutional layer(s), pooling layer(s), flattening layer, fully connected layer(s), and an output layer.

Reused from: [Clark et al., 2022]

5.2.2 Feature Learning

The feature learning of the CNN consist of three blocks: Convolutional layer, activation function, and pooling. The convolutional layer applies learned filters to the input image to create feature maps that summarize the presence of those features. The activation function and pooling layer are used to downsample the feature maps and make them more robust to changes in the position of the features. A review of deep learning for EEG classification tasks suggests that two convolutional blocks are commonly used for ERP and CNN classification. [Craik et al., 2019]

Convolutional Layer

After the input is given to the network, the next layer is the convolutional layer. The convolutional layer is the layer that learns features by using a local receptive field. The local receptive field is an n x m sized mask that is slid over the image with trainable weights and biases. This way, the input image's values are cross correlated with the weights and biases, creating a hidden neuron. These weights and biases are shared weight and biases, meaning that these values are not changed when the local receptive field is slid over the image, but are only updated in the optimization process. The local receptive field is then moved with a given stride, which is how many spaces it is moved before the next hidden neuron is calculated. When the local receptive field has been slid around the whole input, all the hidden neurons for the given receptive field are now a feature map that contains features for the specific values in the receptive field. For each convolutional layer, several different feature maps are usually made, each with different weights and biases. A CNN typically has multiple convolutional layers to detect higher order of features. [LeCun et al., 1998]

An example of an input image and a local receptive field with size of 3 x 3 pixels can be seen in Figure 5.3

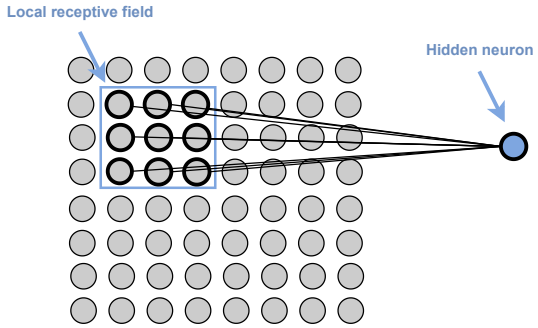


Figure 5.3. Representation of the convolutional approach with a 3 x 3 size receptive field. At each step, the mask performs an element wise multiplication with the corresponding pixels of the input, sums to a single output value. After each stride, the sums are added as hidden neurons, which results in a feature map.

Reused from: [Clark et al., 2022]

Activation Functions

The activation function determines the output of a neuron in an artificial neural network, based on its input or a set of inputs. Non-linearity is introduced by the activation function, which is essential for the neural network to learn complex patterns. Without non-linearity, the network would just be a complex linear model. Depending on the role of the layers, different activation functions can be used throughout the network. Usually, the hidden layers and the output layer have other functions, because the goal in the hidden layers is to enhance training, while the goal in the output is typically to perform classification.

Several activation functions are often used in deep neural networks, such as Sigmoid, Tanh, Softmax, and Rectified Linear Unit (ReLU), which will be described in the following sections. [Duda et al., 2000; Nielsen, 2018]

ReLU The ReLU activation function is a simple nonlinear function that outputs the input if it is positive and zero otherwise. It is widely used in deep neural networks because it is easy to implement, computationally efficient, and can overcome the vanishing gradient problem. [Nielsen, 2018] The vanishing gradient problem is a problem that arises when the gradient in the loss function approaches zero. This causes the parameters of the network, which is updated using the gradient in order to take the next step, to stagnate. This happens primarily to the layers closest to the input since backpropagation starts at the last layer and works its way backward from there. This also means that the more complex network, meaning more layers, the vanishing gradient problem worsens. [Roodschild et al., 2020]

The ReLU activation function is illustrated in Figure 5.4, which shows how it filters out negative values and preserves positive values.

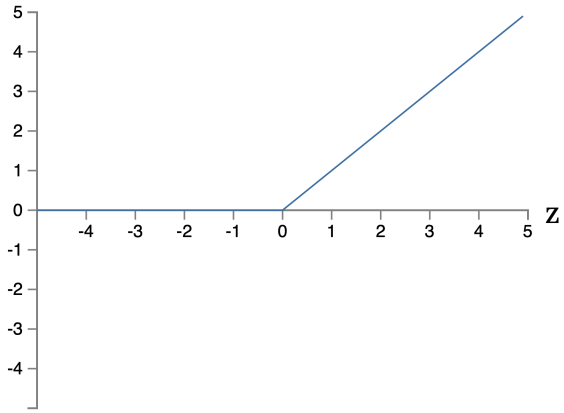


Figure 5.4. Rectified Linear Unit (ReLU) activation function. The input is seen on the z-axis, where only values above zero are positive, and input values below zero are zero.

Source: [Nielsen, 2018]

The ReLU activation function can be expressed mathematically as:

$$ReLU(x) = max(0, x) \tag{5.1}$$

In Equation (5.1) x denotes the net activation. The ReLU function is mostly used on the hidden layers of both the convolutional and linear layers. [Nielsen, 2018]

Sigmoid The sigmoid function is a mathematical function that maps any input value to a corresponding output value within the range of 0 to 1. When this function is plotted, it takes on the characteristic shape of an "s". This distinct visual feature is illustrated in

Figure 5.5, where the sigmoid curve can be seen gradually increasing from 0 to 1, with its steepest slope occurring at the midpoint of the curve.

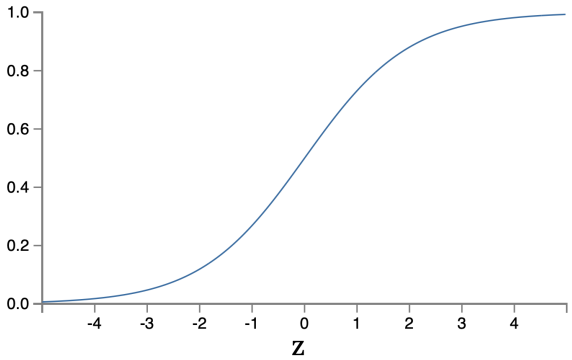


Figure 5.5. Sigmoid activation function. It is shown for all input values, the output of the sigmoid is between 0 and 1.

Source: [Nielsen, 2018]

The sigmoid function is mathematically defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

The variable x in Equation (5.2) represents the net activation, and the sigmoid function can be used in both hidden and output layers of neural networks. However, its use in hidden layers can sometimes lead to numerical instability which can generate exploding or vanishing gradients during learning, as noted in [Nielsen, 2018]. This has led to the development and adoption of alternative activation functions, such as ReLU.

Tanh The hyperbolic tangent function (tanh) shares similarities with the sigmoid activation function, but it has a distinct interval ranging from -1 to 1, with its center at 0. The function is visualized in Figure 5.6.

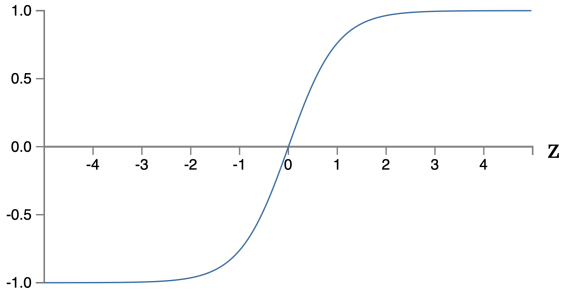


Figure 5.6. Tanh activation function. It is shown for all input values, the output of the tanh is between -1 and 1.

Source: [Nielsen, 2018]

Mathematically, the tanh function is defined as:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5.3}$$

In Equation 5.3 , the variable x represents the net activation. Moreover, a visual representation of the Tanh activation function can be observed in Figure 5.6. [Nielsen, 2018]

Softmax The softmax activation function computes the posterior probabilities by normalizing the output. Employing an exponential function effectively constrains the output values within the desirable range of 0 to 1 for every class. Moreover, it ensures that the sum of all class probabilities always equals 1, making it a vital component in the output layer of neural networks. [Nielsen, 2018]

The softmax function is mathematically defined as:

$$softmax(x) = \frac{e^x}{\sum_{m=1}^{e^{x_m}}} \tag{5.4}$$

In Equation 5.4 [Nielsen, 2018], the variable x represents the net activation, and m denotes the number of neurons.

Pooling

After the convolution layer and activation function, pooling of the feature maps is often used. Pooling is a simplification of the feature maps so the needed calculations in later layers are lowered. This is done by sliding a window function over the feature maps with a given stride, which condenses the information. Since the information is condensed, the size of the feature map after a pooling is reduced, relative the to size of the window. A common method is max-pooling, in which the pooling layer's output is the window's maximum value. An example of max-pooling with a window size of 2 x 2 can be seen in Figure 5.7. Another common method is L2 pooling, in which the squared root of the sum of the window is calculated. Common for each method is pooling to see whether the network has found a certain feature in the image. It then throws away the exact location of this, but the relative location is still known. [Nielsen, 2018]

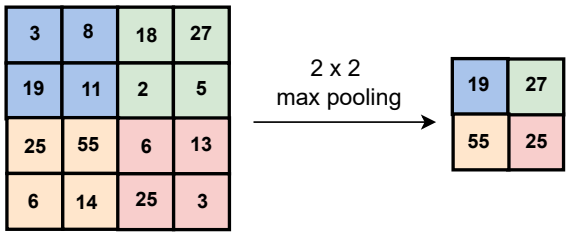


Figure 5.7. Max pooling with a window size of 2 x 2 of the input on the left, and the output of the max-pooling on the right. The stride in this example is 2.

Reused from: [Clark et al., 2022]

5.2.3 Classification

Once the features have been extracted from the input, the subsequent step is the classification. The classification segment of the network consists of a flattening layer, one or more fully connected layers, and lastly, the output layer as seen in Figure 5.8.

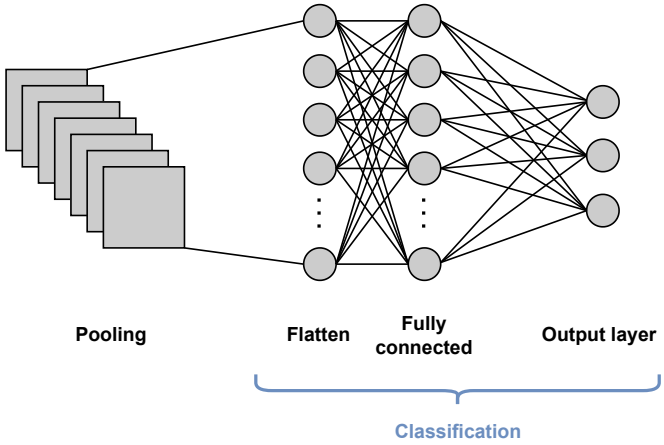


Figure 5.8. The classification segment of the CNN network consists of the flattening layer, fully connected layer(s), and the output layer.

Reused from: [Clark et al., 2022]

Flattening

The flattening layer is the layer that connects the pooling layer and the fully connected layer. The pooling layer is an n x m sized array of intensities, whereas the fully connected layer is a one column vector. In order to connect these layers, a flattening method is used. If the pooling layer is three feature maps, each with 2 x 2 pixels, then the flattening layer would be a column vector with 12 values, as seen in Figure 5.9.

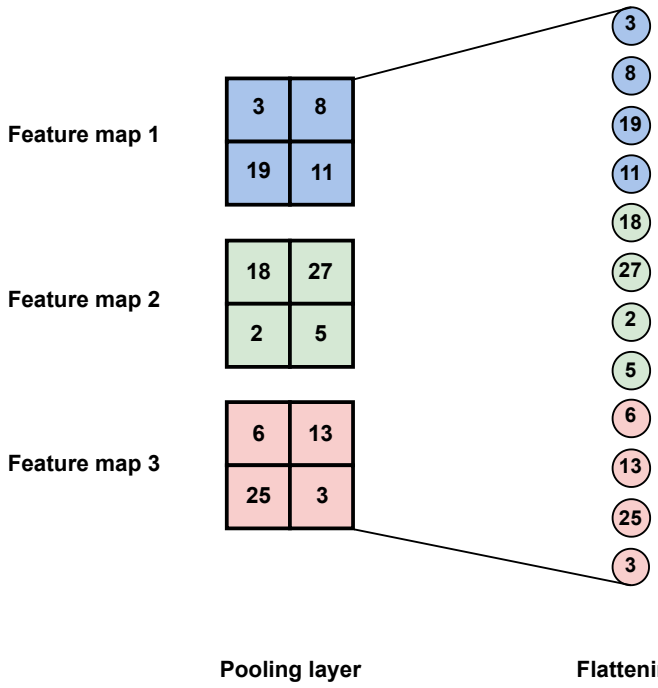


Figure 5.9. Flattening approach of three feature maps with size 2 x 2, resulting in 12 entries in the flattening layers.

Reused from: [Clark et al., 2022]

Fully Connected Layers

The purpose of the fully connected layers is to classify the right label of the sample. This is done by connecting all the neurons from one layer to the next layer. [Duda et al., 2000] The architecture of a fully connected layer that is generally used can be seen in Figure 5.10.

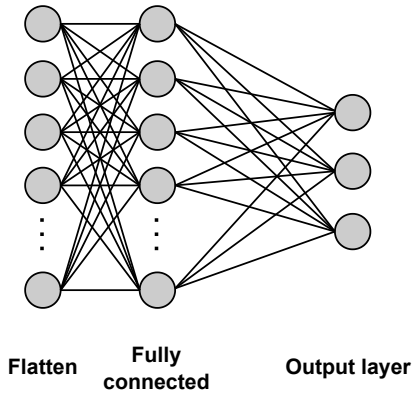


Figure 5.10. Fully connected layers with the flattening layer, fully connected layer(s), and the output layer with three neurons.

Reused from: [Clark et al., 2022]

The output of the fully connected layers depends on different parameters; The value of the input neuron, weights, biases, and activation function. The weight is a parameter

that scales the input value of a neuron. The greater the weight, the more influence the corresponding neuron's value has in the network's computations. The bias shifts the activation functions x-axis, in which a positive bias would increase the output of the activation function, whereas a negative bias would decrease the output of the activation function. Lastly, the activation function results in a non-linear modelling between the layers. A graphical illustration of the parameters can be seen in Figure 5.11. The weights and biases for each hidden neuron is a trainable parameter, meaning that by doing backpropagation, the model can change the values of each weight and bias. [Duda et al., 2000; Nielsen, 2018]

The number of hidden neurons in each of the fully connected layers is dependent on how well separated the patterns in features are. The more complex of a separation, the more neurons are needed. Since this is unknown before the training process, the number of hidden neurons is decided based on trial and error [Duda et al., 2000]. The number of fully connected layers that is recommended for ERP-based classification in a CNN model is 1-2 layers [Craik et al., 2019].

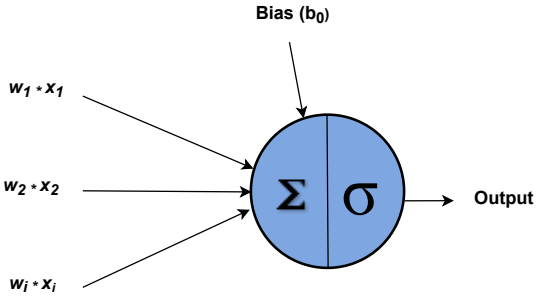


Figure 5.11. The calculation of the hidden neurons output is determined from the weights (w_i), the value of the input neuron (x_i), the bias (b_0), and the activation function (σ).

Reused from: [Clark et al., 2022]

5.2.4 Dropout

The CNN contains non-linear hidden layers, that enable them to decipher complex correlations and patterns. This can lead to something called overfitting, where the model learns to fit too close to the training data, which makes it fail to generalize to unseen data. Several methods have been made to try and prevent overfitting, with dropout being one of them. Dropout is a method where neurons are dropped, which can include neurons in both hidden and visible layers. When a neuron is dropped, it means that the neuron is temporally removed (the value is set to zero) from the network. This help overfitting, since the model can not rely on a few neurons, but is forced to use other neurons to base the classification on. The choice of which neurons are dropped for each training is random, which is often implemented with a fixed percentage of all the neurons in a specific layer. The dropout rate is a hyperparameter that can be changed between the training runs, but

around 0.5 seems close to optimal for most networks and tasks. [Srivastava et al., 2014]. A visual illustration of dropout can be seen in Figure 5.12

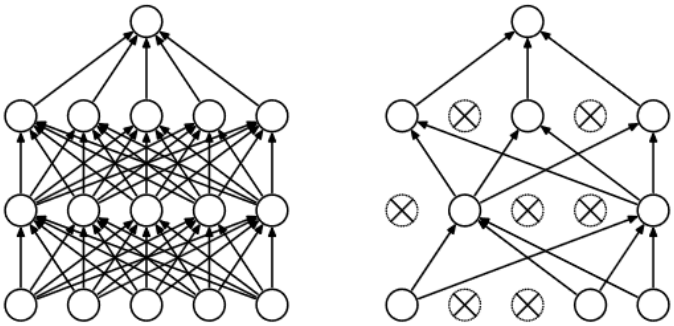


Figure 5.12. The effect of dropout on a network. (a) a standard neural network with 2 fully connected hidden layers. (b) is the same size as (a), but with a thinned number of neurons due to dropout.

Source: [Srivastava et al., 2014]

5.2.5 Output

The output layer is the last layer in the network and is where the classification between the classes happens. However, this process does not imply absolute certainty about the predicted class. To handle this uncertainty, a softmax activation function is used to make the prediction into a probability. As described in Section 5.2.2, the softmax ensures that the predicted probabilities are non-negative and sum up to one. This way, softmax ensures that the network outputs probabilities for each class, in which the highest probability is the class the network has predicted. [Grandini et al., 2020]

5.2.6 Backpropagation

Backpropagration is a method used when training artificial neural networks. It is part of the training that is called the backward pass, where the output of the network is compared to the desired output quantified by the loss function. The parameters are then adjusted by the gradients which are calculated for each neuron by applying the chain rule in the direction from the output to the input, implemented using an optimizer.

Cross Entropy Loss Function

The learning process of a neural network heavily relies on loss functions, which quantify the error between predicted and target values. In the case of classification-based architectures, a classifier loss function is used. Among them, the widely used and highly effective choice is cross-entropy loss. This loss function calculates the disparity between probability distributions of predicted and target values, providing insights into the model's performance. [Hart et al., 2000]

The mathematical formulation of cross-entropy loss is depicted in Equation 5.5:

$$\text{Loss} = -\left(y \log(p) + (1-y) \log(1-p)\right) \tag{5.5}$$

Where y represents the label (0 or 1), and p denotes the predicted probability. When dealing with multiclass classification, the loss for each class label is calculated and sums up to the results, as shown in Equation 5.6:

$$\text{Loss} = -\sum_{c=1}^{M} y_c \log(p_c) \tag{5.6}$$

Here, M refers to the number of classes, which in our case is $M = 3$. The variables y_c and p_c denote the target and predicted probabilities, respectively, for class c. It is worth noting that cross-entropy loss severely penalizes confident yet incorrect predictions. For instance, when the label is 1, the second half of the function vanishes, and the same applies to the first half when the label is 0. This loss function enables us to measure the dissimilarity between the predicted probability distribution and the true distribution across all three classes.

Utilizing cross-entropy loss in this project's deep learning architecture will help guide the training process and optimize the model's performance in distinguishing between control, LTP, and SNI stimuli. [Hart et al., 2000]. To incorporate cross-entropy loss into the neural network, the default parameterization recommended by the PyTorch documentation was used [Paszke et al., 2019]. This implementation includes a softmax activation function in the input of the loss function.

5.2.7 Gradient Descent and Nesterov-accelerated Adaptive Moment Estimation Optimizer

To make use of the information provided by the loss function, an optimizer is needed to update the weights and biases of the network. The optimizer's goal is to locate the global minima in the optimization space while using the measure of the loss function to navigate the space and find the optimal or near-optimal solution. Gradient descent is widely used as an optimization method to find the minima of a loss function. The method works by taking steps in the opposite direction of the gradient since the gradient's movement is toward the maxima. When the minima are found, the parameters of the neural network are updated in the backward pass [Ruder, 2016].

The mathematical formula for gradient descent is shown in Equation (5.7).

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_{\theta_t} J(\theta_t) \tag{5.7}$$

Where, θ represents the parameter at iteration t, α is the learning rate, and $\nabla_{\theta_t} J(\theta_t)$ denotes the gradient of the objective function with respect to the parameter θ_t. This optimizer starts with an initial parameter value and updates it based on the negative gradient direction multiplied by the learning rate. While gradient descent is a fundamental optimizer, it has some limitations. It updates the parameters only after going through an entire epoch, which can be slow and resource-intensive. Additionally, it uses a constant learning rate for all parameters, which makes it less robust and slower in converging to an optimal solution [Ruder, 2016]. This can be overcome by implementing a scheduler that updates the learning rate after each iteration. Nonetheless, this still has the disadvantage of each parameter being updated at the same rate.

To address the limitations of the homogeneous parameter update rate, the Nesterov-accelerated Adaptive Moment Estimation (NAdam) optimizer is chosen. NAdam is an extension of the widely used Adaptive Moment Estimation (Adam) optimizer, commonly employed as the default optimization algorithm in neural networks. The forthcoming sections will revolve around Adam, as the fundamental theory of NAdam is rooted in this concept. Adam has several advantages, including computational efficiency, low memory requirements, and invariance to diagonal rescaling of gradients. It performs well in large-scale problems that involve a substantial amount of data and parameters, as stated in Kingma and Ba [2014].

Adam achieves more frequent parameter updates by optimizing the learning process for small batches in each epoch and for each parameter θ. It adapts the learning rate based on the parameters, allowing for smaller updates for frequently occurring features and larger updates for infrequent features. [Ruder, 2016]. To achieve this, Adam introduces additional hyperparameters, such as first and second-order moments [Choi et al., 2019; Kingma and Ba, 2014]. The adaptive parameter updates are based on both of these moments. It maintains an exponentially decaying average of past squared gradients v_t and an exponentially decaying average of past gradients m_t, similar to the concept of momentum. [Ruder, 2016]

The Nesterov accelerated gradient (NAG) is incorporated into Adam by modifying the m_t term. This modification updates the gradient one step ahead by replacing the previous moment, $\hat{m}_{t-1}$, with the current $\hat{m}_t$. The mathematical formula for NAdam is as follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}(\beta_1 \hat{m}_t + \frac{(1-\beta_1)}{1-\beta_1^t}) \cdot \nabla_{\theta_t} J(\theta_t) \tag{5.8}$$

Where

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \text{and} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{5.9}$$

In Equation (5.8), the difference between NAdam and Adam is that the gradient is updated one step ahead by replacing the previous moment, $\hat{m}_t - 1$, with the current $\hat{m}_t$. The first ($\hat{m}_t$) and second order ($\hat{v}_t$) moments have the decay rates β_1 and β_2 . $\hat{m}_t$ and $\hat{v}_t$ represent the corrected moments of m_t and v_t, respectively. The correction factor ensures unbiased estimates of the moments so that they are not biased toward zero. t denotes an iteration. The term ϵ is a small constant used for numerical stability.

By incorporating the NAG, NAdam improves upon the original Adam optimizer. It updates the gradient using the projected new position rather than the current position, resulting in accelerated convergence. The NAdam optimizer provides faster and more efficient optimization compared to simple gradient descent, as well as improved adaptability to different parameters by dynamically adjusting the learning rate.

For the implementation of the NAdam optimizer, the default parameters recommended in the PyTorch documentation were adopted. [Paszke et al., 2019]

5.3 Long Short-Term Memory

This section will provide insight into the benefits of utilizing LSTM and delves into the architectural structure of the network. Subsequently, a comprehensive explanation of the distinct gates within the LSTM cell will be presented.

5.3.1 Advantages of LSTM

LSTM is a type of RNN developed to overcome the problem of exploding and vanishing gradients. LSTM is also great for sequential data and the identification of long-term dependencies.

RNNs are flexible in handling sequential data of varying lengths due to their feedback loops. These loops create new inputs that match the length of the data. RNNs have shared weights and biases across all rows, keeping complexity constant. However, this also causes the vanishing/exploding gradient problem.

The exploding and vanishing gradient problem refers to the difficulty of training RNNs to learn long-term dependencies, where the input or output at a certain time step depends on the information from many previous or future time steps. In such cases, the gradients of the error function with respect to the network parameters can either grow or decay exponentially as they propagate through the network during backpropagation. This poses a significant challenge in maintaining stable updates to the weights and biases of the network. Therefore the LSTM was implemented to minimize the gradient problem and exploit its ability to extract long-term dependencies. LSTM networks incorporate memory cells and gating mechanisms that allow them to selectively retain or forget information

over long sequences. This helps address the vanishing gradient problem by preserving important information over time and mitigating the effects of gradient decay. [Van Houdt et al., 2020; Sak et al., 2014]

5.3.2 General LSTM Architecture

LSTM is a type of neural network that introduces a memory cell and three gates: an input gate, an output gate, and a forget gate. The memory cell stores relevant information from past and present inputs, while the gates control how much information is added or removed from the cell and how much is passed to the next cell. This way, LSTM can learn to preserve or forget long-term dependencies in the data and avoid the exploding and vanishing gradient problem. Unlike traditional RNN, LSTM has two paths: one for short-term memory and one for long-term. The long-term memory path, also known as the cell state, has a straight connection to the output and is only manipulated by multiplication and sum. Its main purpose is to avoid exploding or vanishing gradients. The short-term memory path, also known as the hidden state, includes multiple weights, biases, and activation functions, such as sigmoid and tanh, which are a part of the cell state. (see Section 5.2.2). An LSTM cell contains both a cell state and a hidden state, which are unrolled according to the size of the sequential data. [Van Houdt et al., 2020; Sak et al., 2014]

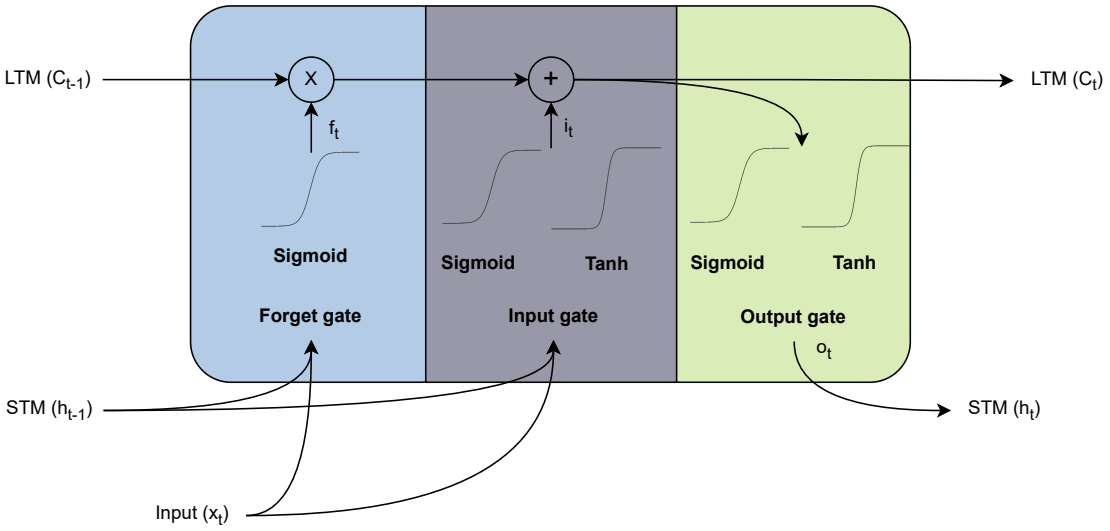


Figure 5.13. LSTM architecture with the forget (blue), input (grey), and output gate (green). LTM is the long-term memory, while STM is the short-term memory.

5.3.3 Forget Gate

The first stage in the LSTM is called the forget gate and it decides how much of the previous hidden state, which represents the long-term memory of the network, should be kept or discarded for the current input. The forget gate uses a sigmoid function, which is a mathematical function that maps any real number to a value between 0 and 1. The output of the sigmoid function can be interpreted as a probability or a percentage. For example, if the forget gate outputs 0.8 for a certain element of the hidden state, it means that 80% of that element will be retained and 20% will be forgotten. The forget gate allows the network to selectively remember or forget information that is relevant or irrelevant. [Van Houdt et al., 2020; Sak et al., 2014]

The parameters of the network are called weights and biases, which will be denoted as W and b respectively. The weights multiply the inputs and the biases are added to the multiplication. The formula for the forget gate is:

$$f_t = sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.10}$$

where f_t is the output of the forget gate with a time step t, W_f is the weight, $[h_{t-1}, x_t]$ is the concatenation of the previously hidden state output and the current time input, b_f is the bias for the forget gate, and sigmoid is the sigmoid activation function. [Van Houdt et al., 2020; Sak et al., 2014]

The output of the forget gate f_t is then used to update the cell state c_t, which is another parameter that stores information over time. The cell state is updated by multiplying it by f_t. This means that c_t is either kept or forgotten depending on the corresponding element of f_t. The formula for updating the cell state is:

$$c_t = c_{t-1} \cdot f_t \tag{5.11}$$

where c_t is the cell state at time step t, c_{t-1} is the cell state at time step $t-1$, and f_t is the output of the forget gate at time step t. The cell state c_t is then used by two other gates: the input gate and the output gate. [Van Houdt et al., 2020; Sak et al., 2014]

5.3.4 Input and Output Gate

The second stage is the input gate which is divided into two blocks. One of the blocks combines the input with the short-term memory, to create a potential long-term memory. The other block determines what percentage of the candidate should be remembered. The third stage is the output gate which is similar to the input gate, except that this updates the short-term memory instead of the long-term.

The input gate decides what new information to add to c_t based on h_{t-1} and x_t, and the output gate decides what part of c_t to output as h_t based on h_{t-1} and x_t. These two gates also use weights, biases, and sigmoid functions similar to the forget gate. The formulas for these gates are:

$$i_t = sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.12}$$

$$o_t = sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.13}$$

where i_t is the output of the input gate at time step t, o_t is the output of the output gate at time step t, W_i is the weight for the input gate, W_o is the weight for the output gate, b_i is the bias for the input gate, b_o is the bias for the output gate. $[h_{t-1}, x_t]$ is as defined in Equation (5.10).

The input gate i_t also uses the tanh activation function. The tanh function produces a candidate g_t that contains potential new information extracted from the input to add to c_t. The formula for g_t is:

$$g_t = tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{5.14}$$

where g_t is the candidate at time step t, W_g is the weight for the candidate, and b_g is the bias for the candidate. $[h_{t-1}, x_t]$ is as defined in Equation (5.10).

The input gate i_t is a percentage that decides how much of g_t to add to c_t by multiplication. The formula for updating c_t with new information is:

$$c_t = c_t + i_t \cdot g_t \tag{5.15}$$

where c_t is the updated cell state at time step t, i_t is the output of the input gate at time step t, and g_t is the candidate at time step t.

The output gate o_t decides what part of c_t to output as h_t by multiplying them element-wise after applying a tanh function to c_t. The formula for computing h_t is:

$$h_t = o_t \cdot tanh(c_t) \tag{5.16}$$

where h_t is the hidden state at time step t, o_t is the output of the output gate at time step t, and c_t is the cell state at time step t.

5.3.5 Bidirectional LSTM

A Bidirectional LSTM (BI-LSTM) (see Figure 5.14) is a type of architecture that follows the ground principles of a normal LSTM. The difference of this network is that it passes data forwards and backward so that both past and future data are considered when making the prediction. This can be useful for tasks that require context from both sides of a sequence, such as natural language processing (Sak et al. [2014]) or time series analysis (Van Houdt et al. [2020]). A BI-LSTM consists of two LSTM layers that operate in opposite directions and concatenate their outputs at each time step. Furthermore, Bi-LSTM has shown a superior ability to learn from time-series EEG data [Wu et al., 2022; Zhang et al., 2018].

The complexity of the network can be increased by implementing multiple layers. This means that the output of one LSTM layer can be fed as the input of another LSTM layer, creating a deeper network that can learn more complex patterns and dependencies in the data. [Wu et al., 2022]

The chosen architecture for this project will feature a baseline LSTM model, incorporating a bidirectional multilayer design. This approach ensures that dependencies in both directions are carefully considered, while also allowing for the implementation of multiple layers of bidirectional LSTMs. This enables the extraction of intricate features and dependencies, contributing to a comprehensive and deeper model. The baseline bidirectional multilayer LSTM module for the final architecture can be seen on Figure 5.14.

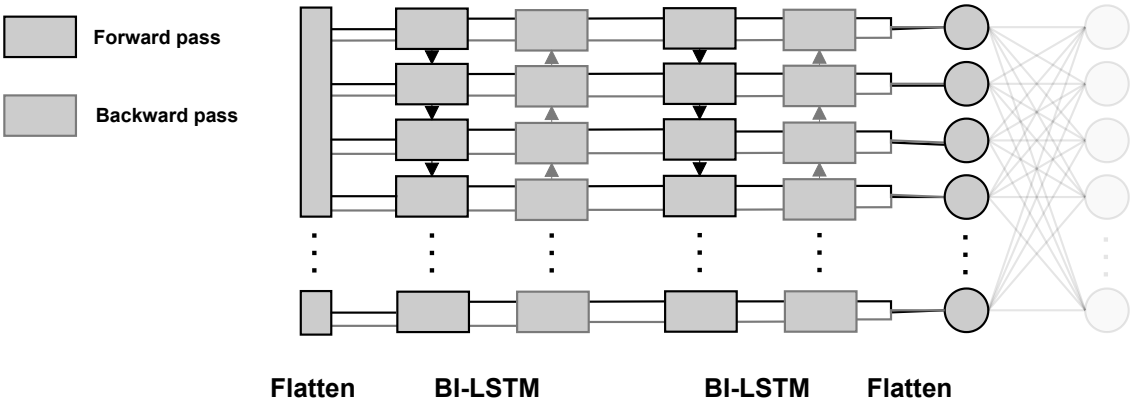


Figure 5.14. Illustration of a BI-LSTM two-layer architecture, demonstrating the interconnected feedforward and backward layers. The forward pass from the initial layer is linked to the forward pass of the subsequent layer, similarly to the backward pass. The forward pass is depicted with a solid black outline, whereas the backward pass is outlined in gray.

Model Interpretability, Evaluation, and Performance Metrics

6

This chapter will explain the general methods used to understand and assess the performance of the neural network. This includes the model interpretability, evaluation, and lastly performance metrics.

6.1 Model Interperability

In order to gain a comprehensive understanding of the underlying feature importance within the architecture, diverse interpretability methods are employed. Multiple methods are utilized to mitigate any biases that might arise from relying solely on a single approach. The focus lies on primary attribution methods, as they provide a holistic overview of the network's feature extraction and weighting. These methods are carefully selected to ensure a varied range of frameworks is utilized. The Noise Tunnel is implemented in every method. This technique serves as an additional layer for various attribution methods. By incorporating Gaussian noise into the input multiple times, Noise Tunnel computes attributions repeatedly and then combines the resulting values using the specified method.

The feature attribution methods were implemented using the Python package Captum. [Kokhlikyan et al., 2020]

6.2 Saliency

Saliency maps are a baseline method for investigating network attention. This method involves recalculating the network's gradients by backpropagating from the output to the input. By harnessing these gradients, it becomes possible to accentuate individual pixels within an image, resulting in a final score that accurately reflects the network's attention given to a specific class. During the gradient backpropagation process, each pixel of the image is assigned a relevance score, signifying its contribution to the final prediction.

One way to interpret this approach is by considering it as an approximation using a first-order Taylor expansion of the network with respect to the input. The gradients, in this case, correspond to the coefficients assigned to each feature in the linear representation of the model.

This mechanism effectively highlights the pixels with the greatest influence on the network's output, revealing where the network is directing its attention. When these relevance scores are aggregated, they form a saliency map, which visually portrays the distribution of attention across the input image. [Simonyan et al., 2013]

An example of the saliency method used on a spectrogram can be seen on Figure 6.1. This method exhibits a higher level of sensitivity to input data, as the model demonstrates strong attributes across multiple locations in the spectrogram.

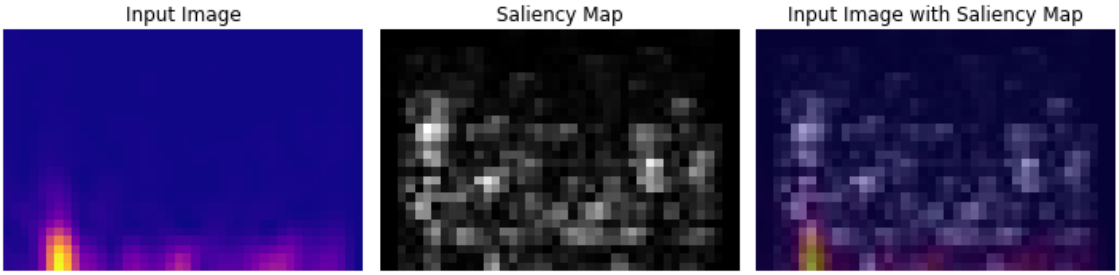


Figure 6.1. Example of a Captum-generated saliency map applied to an EEG spectrogram. Lighter pixels indicate more important features, while darker pixels indicate less important features

Input data from: [Atchuthan et al., 2023]

Saliency was implemented using the Captum function 'captum.attr.Saliency'.

6.3 Integrated Gradients

Integrated gradients is another gradient-based method similar to saliency that examines the integral of gradients from the output with respect to the input. This approach utilizes an attribution baseline, which serves as a reference point for assigning importance scores. In Integrated Gradients, the baseline is commonly selected as a point in the input space where all feature values are set to zero or a neutral value. The fundament of this method is based on the axioms; feature sensitivity and implementation invariance. Feature sensitivity refers to the importance score given to a specific feature that should reflect its actual contribution. Implementation invariance refers to the attribution score that should not be affected by the method of implementation of the architecture. This means that the attribution score should solely rely on the performance of the model and not on which layers and activation functions are implemented. This is handled by making use of continuous gradients instead

of discrete ones since these are affected by the model implementation. [Sundararajan et al., 2017]

Computationally this method approximates the integral of the gradients from the output to the input and multiplies this with the difference between the input and baseline. This is also seen in Equation (6.1)

$$\text{IntegratedGradients}(x) = (x - x_{\text{baseline}}) \times \int_{\alpha=0}^{1} \frac{\partial F(x_{\text{baseline}} + \alpha \times (x - x_{\text{baseline}}))}{\partial x} d\alpha \quad (6.1)$$

x represents the input data point of interest. x_{baseline} is a reference input used for comparison. F is the analyzed function or model. α is a weighting factor that interpolates between the baseline and the input. $\frac{\partial F(x_{\text{baseline}} + \alpha \cdot (x - x_{\text{baseline}}))}{\partial x}$ calculates how F changes with respect to x along the path from the baseline to the input. The integral $\int_{\alpha=0}^{1}$ integrates the changes in F with respect to x along the path. $d\alpha$ represents the infinitesimal change in α used in the integration.

An example of Integrated Gradients used on a spectrogram can be seen on Figure 6.2.

Figure 6.2. Example of a Captum-generated Integrated Gradients map applied to an EEG spectrogram. Lighter pixels indicate more important features, while darker pixels indicate less important features

Input data from: [Atchuthan et al., 2023]

Integrated Gradients was implemented using the Captum function 'captum.attr.IntegratedGradients', employing a zero array baseline with identical input size.

6.4 DeepLIFT

DeepLIFT is a backpropagation-based approach that calculates importance scores by tracing a path from the output to the input. Unlike integrated gradients, DeepLIFT effectively addresses the computational overhead issue by deriving importance scores from the difference between the reference activation and the input activation, without using integrals. This involves using a baseline input to establish a reference activation, which is then compared to the activation of the input. DeepLIFT employs the concept

of multipliers, to "blame" neurons for differences in the output. By multiplying the importance scores by these multipliers and propagating them backward through the network, DeepLIFT ensures that the importance scores are allocated to the input features in a way that considers the interactions and dependencies between layers. The multiplier effectively captures how changes in the input are amplified or attenuated as they pass through the network. [Shrikumar et al., 2017]

The multiplier is calculated as such:

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x} \tag{6.2}$$

x is the input neuron with a difference from reference Δx, and t is the target neuron with a difference from reference Δt. C is then the contribution of Δx to Δt.

The significance of this approach lies in its ability to generate negative activation scores for neurons, overcoming problems related to saturation. [Shrikumar et al., 2017]

In addition to its computational advantages, DeepLIFT also resolves the problem of thresholding by incorporating negative values. The thresholding problem arises when computing importance using gradients multiplied by the input, such as in saliency. In this case, even a tiny change in the input can lead to a significant shift in the gradients due to the bias term. By utilizing the difference from the reference activation, DeepLIFT achieves a more seamless and continuous importance scoring, effectively eliminating the disruptions caused by the bias term. [Shrikumar et al., 2017]

An example of DeepLIFT used on a spectrogram can be seen on Figure 6.3.

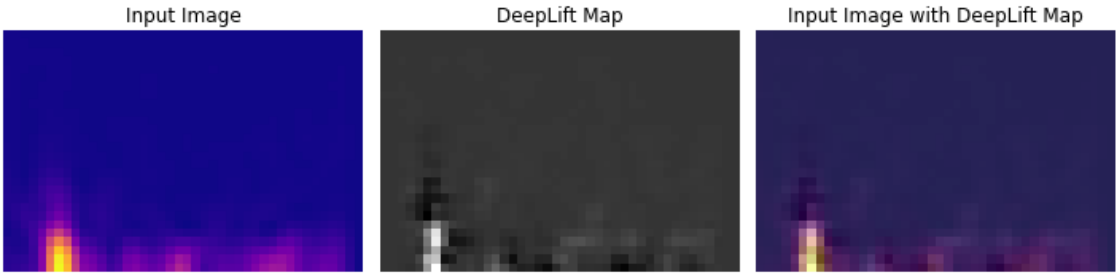


Figure 6.3. Example of a Captum-generated DeepLIFT map applied to an EEG spectrogram. Lighter pixels indicate more important features, while darker pixels indicate less important features

Input data from: [Atchuthan et al., 2023]

DeepLIFT was implemented using the Captum function 'captum.attr.DeepLift', employing a zero array baseline with identical input size.

6.5 Occlusion

Occlusion is a pertubation based technique employed to compute attribution. It employs sliding windows to selectively occlude various sections of the input with a baseline, enabling the evaluation of their impact on the network's classification accuracy. If a significant alteration in accuracy occurs, it signifies the importance of the corresponding features, which are then emphasized in the resulting attribution map. By aggregating the attribution maps obtained from each occluded image, a comprehensive representation of the model's feature attribution map is generated. [Zeiler and Fergus, 2013]

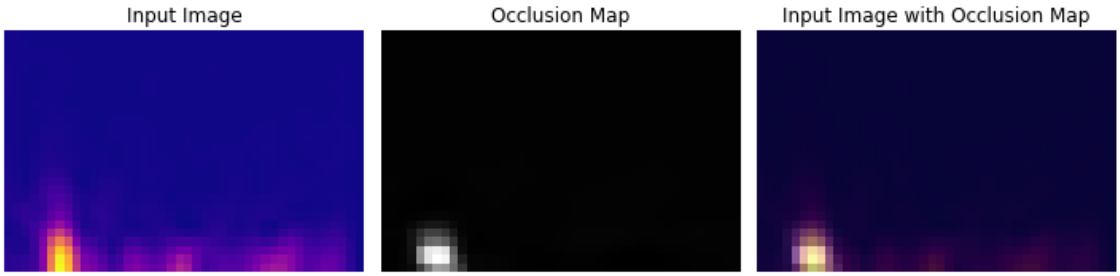


Figure 6.4. Example of a Captum-generated Occlusion map applied to an EEG spectrogram. Lighter pixels indicate more important features, while darker pixels indicate less important features

Input data from: [Atchuthan et al., 2023]

Occlusion was implemented using the Captum function 'captum.attr.Occlusion', employing a zero array baseline with identical input size, for the occluding window.

6.6 Model Evaluation

6.6.1 Learning Curves

In this project, the learning curve was used as a tool to evaluate various hyperparameters throughout each training session in order to determine the model's fit to the data. The training loss curves depict the model's performance on the training dataset, whereas the validation loss curve describes its performance on the validation dataset, indicating the model's generalizability to new data. The learning curves were generated using cross-entropy, which calculated the loss for both the training and validation datasets. The losses were plotted for each epoch, and the resulting curves for training and validation loss were visually represented based on the cross-entropy result. The curve of training loss tends to have a gentler decline, in contrast to the more fluctuating pattern seen in the validation loss curve. An illustration of different learning curve outcomes can be found in Figure 6.5, where the blue curve represents training loss, and the orange curve represents validation loss. [Brownlee, 2019]

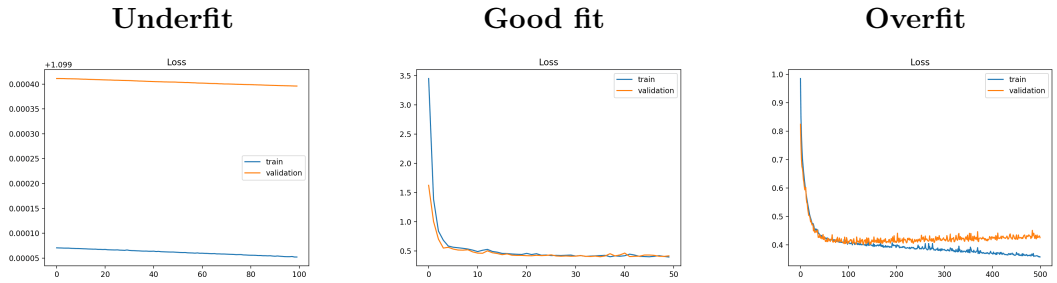


Figure 6.5. Three types of learning curves, showing underfit, good fit, and overfit. The images includes the training loss (blue) and validation loss (orange).

Source: [Brownlee, 2019]

Typically a model's performance can be categorized as overfit, underfit, or a good fit. Overfitting occurs when the model's too complex, resulting in the validation loss diverging away from the training loss curve. Underfitting is when both the model's complexity and the training duration are insufficient, causing the validation loss to not converge with the training loss curve. A good fit is when the validation and training loss curves converge and keep the same trends. [Brownlee, 2019]

To improve the model, adjustments can be made by modifying its complexity based on the learning curve assessment. For instance, if the model is underfitting due to its low complexity, additional hidden layers, larger layer sizes, or more advanced architectures can be added. Conversely, if the model is overfitting due to its high complexity, layers can be removed, layer sizes can be reduced, or less advanced architectures can be employed. This is an iterative process, where the trial-and-error approach is used. [Brownlee, 2019]

6.6.2 Model Checkpoint and Early Stopping

Throughout the training process, the performance of the model gradually improves as it learns from the training data. Model checkpoints save the model, which includes the structure of the model, as well as the weights and biases. This is done when the current validation loss is better than the previous validation loss. This way, the model that was most generalizable, which is typically based on the validation loss, gets saved for later evaluation on the test data. Early stopping is a technique that helps against overfitting a neural network to the training data. Normally during the training process, the network seems to get better and better due to its lowering training loss. But at some point, the networks start to worsen, where the error on validation data starts to increase. This is where early stopping can help stop the training process before reaching a point where the model is overfitting [Orr and Müller, 1998]. Early stopping was implemented so that the training process would break when a certain number of epochs have gone by without lowering the best performed validation loss. This parameter is called patience and was set to 30, meaning that after 30 epochs of not improving validation loss, the model would

break the training loop. A patience of 30 was chosen based on the training of 500 epochs, as seen in Figure 6.6, where the validation loss drops and then starts to rise, a patience of 30 should be sufficient in stopping the model if it is not improving.

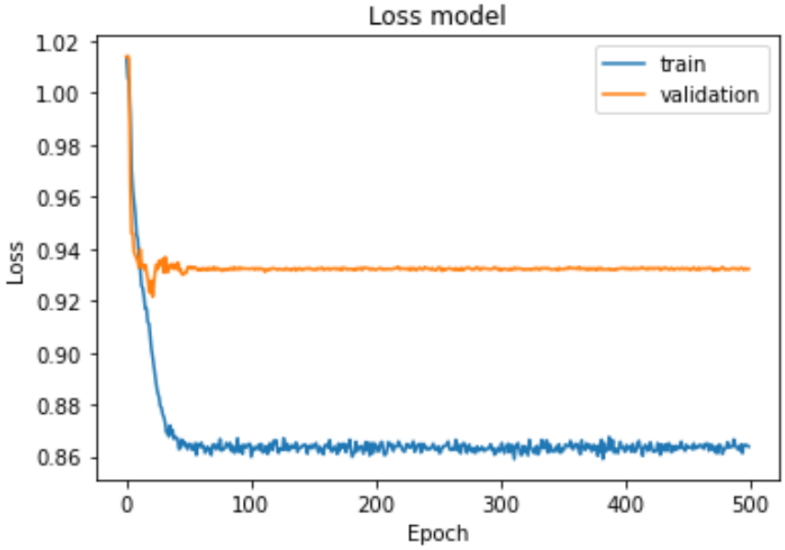


Figure 6.6. Trained model with 500 epochs where the training loss (blue) and validation loss (orange) are shown. Note that using model checkpoint, the model would have stopped around 40 epochs, before an increase in validation loss, and there preventing overfitting.

6.7 Classification Performance Metrics

In order to decide how well a classification algorithm has performed, a confusion matrix can be used. A confusion matrix is a matrix in which the predicted values of the test data are compared to the actual values. When considering a two class problem, there are four possible outcomes: if the instance is positive, and the classifier predicted positive, it is counted as a true positive (TP). If the classifier predicted negative it is counted as a false negative (FN). If the instance is negative, and the classifier predicted negative, it is counted as a true negative (TN). If the classifier predicted positive, it is counted as a false positive (FP). Then the values from the four scenarios are then plotted in a 2 x 2 matrix as seen in Figure 6.7.

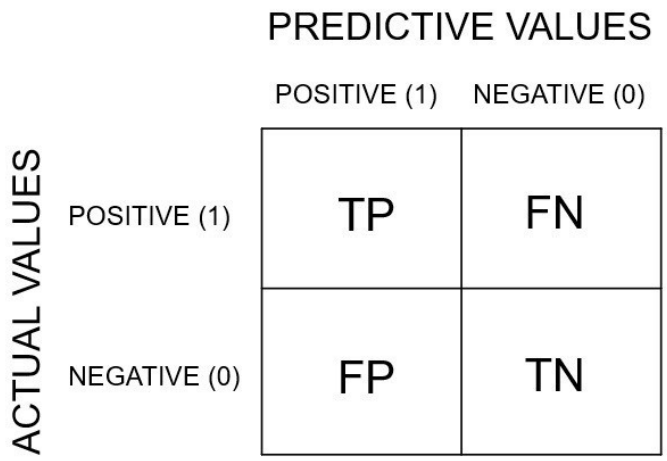


Figure 6.7. Confusion matrix for binary classification with true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

Reused from: [Clark et al., 2022]

However, in the case of multiclass classification, with three classes, the confusion matrix would be a 3 x 3 matrix, where each cell represents the count of instances for a specific combination. The class is listed in the same order in the rows and columns, and the correctly predicted instances are located in the main diagonal. According to Markoulidakis et al. [2021], when using a confusion matrix with multiclass classification, the use of TP, TN, FP, and FN is not applicable. Instead, they use TP along the diagonal for the correctly predicted instances, and intragroup mismatch (IM), which is a falsely predicted instance for instances predicted elsewhere than the main diagonal. The confusion matrix encloses relevant information about the classifier and the performance of the classification rule, which is often used for further analysis of the performance. [Grandini et al., 2020]. A confusion matrix for multiclass classification can be seen in Figure 6.8.

		Predicted class		
		C1	C2	C3
Actual class	C1	TP	IM	IM
	C2	IM	TP	IM
	C3	IM	IM	TP

Figure 6.8. Confusion matrix with three classes with true positive (TP) and intragroup mismatch (IM). The actual values are listed in the rows, whereas the predicted values are listed in the columns.

6.7.1 Accuracy

Accuracy is one of the most used performance metrics for classification and measures the number of correct predictions compared to the total amount of predictions. It is calculated from the confusion matrix by the correct prediction instances in the main diagonal divided by all instances in the confusion matrix as seen in Equation (6.3). The accuracy weights the classes equally, which means that if the amount of data in the different classes are unbalanced, this can give a wrongful conclusion about the network's performance. [Grandini et al., 2020]

$$Accuracy = \frac{TP}{TP + IM} \tag{6.3}$$

TP is the sum of all the true positives, and IM is the sum of all intragroup mismatches.

6.7.2 F1-score

An alternative to accuracy is the F1-score to measure the performance of the network which takes class imbalance into account. This is done by calculating precision and recall. Precision is a measure of the TP predictions of the network, while recall is the probability that the positive predicted values are actually true. Precision and recall measures under the concept of harmonic mean, which is a type of mean that gives a result that favors the lower values in the dataset. This way, the F1-score is low if either the precision or recall is low, but increases if both parameters are similar. [Grandini et al., 2020]

The F1-score can range from 0 to 1 where 1 means a perfect classification between the classes, and can be calculated from the confusion matrix as seen in Equation (6.4)

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{6.4}$$

The F1-score for multiclass classification has to be calculated with an adjustment. Here the TP, FP, and FN must be found for each class, and thereby calculate the individual F1-score and thereafter calculate the final F1-score. When calculating the F1-score for each class, the TP score is one of the values along the diagonal, depending on which class is calculated. Then the FN values are found by the remaining values in the row of the TP value. The FP values are found by the column of the respective TP value. [Shickel et al., 2020; Grandini et al., 2020]. This is shown for the first class C1 in Figure 6.9

		Predicted class		
		C1	C2	C3
Actual class	C1	TP	FN	FN
	C2	FP	TN	TN
	C3	FP	TN	TN

Figure 6.9. This figure shows how to find true positive (TP), false positive (FP), and false negative values from the confusion matrix when calculating the F1-score for the C1 class.

After the F1-score is calculated for each class, multiple ways of finding the average F1-score are used. Here the macro, micro, and weighted F1-score will be presented. Macro F1-score is the multiplied F1-scores of the individual classes divided by the total number of classes, as seen in Equation (6.5). This way, each class is weighted equally and is therefore often used for balanced datasets. [Shickel et al., 2020; Grandini et al., 2020]

$$\text{Macro F1-score} = \frac{F1_{c1} + F1_{c2} \cdots F1_{cn}}{K} \tag{6.5}$$

$F1_{cn}$ is the F1-score of a given class, and K is the total number of classes.

The idea behind the micro F1-score is to calculate all the different classes together without considering possible differences between the classes. This is done by taking the TP for a specific class divided by the column of the TP value. This is done for all the classes and ends up giving the same result as accuracy, as shown in Equation (6.6). [Shickel et al.,

2020; Grandini et al., 2020].

$$\text{Micro F1-score} = \frac{\sum_{c=1}^{K} TP_c}{\sum_{c=1}^{K} \text{Total Column}_c} \tag{6.6}$$

K is the total number of classes, and c is the given class.

The weighted F1-score is weighted based on the number of samples each class contributes to the whole dataset. This way, if a class containing the majority of the samples is correctly predicted, it weighs more than the minor classes. This is calculated by taking the F1-score of the individual classes and multiplying it by a weight, which is the percentage of the whole dataset that class contribute. This calculation is shown in Equation (6.7). [Shickel et al., 2020; Grandini et al., 2020]

$$\text{Weighted F1} = F1_{c1} \cdot W_{c1} + F1_{c2} \cdot W_{c2} \cdots F1_{cn} \cdot W_{cn} \tag{6.7}$$

where $F1_{cn}$ is the calculated F1-score for the specific class, and W_{cn} is the weight assigned to each class based on how much the specific class contributes to the total dataset. Based on these methods, and the fact that data used in this project is imbalanced, the weighted F1-score was chosen as the method in this project, since it weights the classes depending on their contribution to the total amount of instances in the dataset.

6.7.3 Receiver Operating Characteristic Area Under the Curve

Another common method for evaluating the performance of classifiers is the receiver operating characteristic (ROC) curve, which is a curve of sensitivity versus 1 - specificity of the test data. Sensitivity is a measure of the ability of the classifier to detect TP when it is actually a TP instance. Specificity is a measure of how well the classifier can detect TN in all instances of TN. [Mandrekar, 2010b,a]

Sensitivity and specificity can be calculated from the confusion matrix as seen in Equations (6.8) and (6.9).

$$TPR = \frac{TP}{TP + FN} \tag{6.8}$$

$$TNR = \frac{TN}{TN + FP} \tag{6.9}$$

The ROC curve is made by plotting the sensitivity on the y-axis and 1 - specificity on the x-axis, where each point on the ROC curve corresponds to different threshold values.

If the algorithm classifies based on pure chance, it will follow the diagonal line from the lower left corner to the top right corner which is called the line of no discrimination. An example of the ROC curve can be seen in Figure 8.2

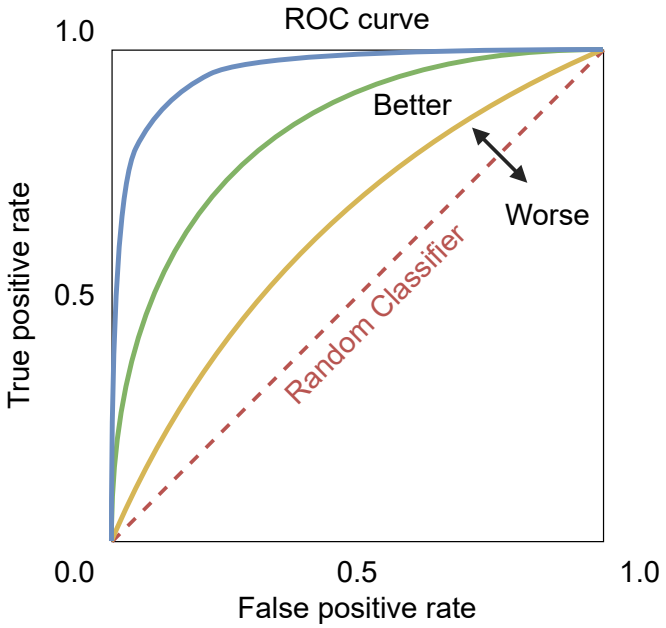


Figure 6.10. Example of ROC curves with a line of no discrimination along the diagonal. Classifiers with a ROC curve close to 0,1 is better than classifiers closer to 1,0.

Reused from: [Clark et al., 2022]

ROC curves from different classifiers can be difficult to compare. For this purpose, the area under the curve (AUC) is often used which is a method that is calculated by numerical integration. This way, a value between 0 and 1, where 1 is a perfect classification, can be used to compare classifiers. An AUC score of 0.5 suggests that the model does not discriminate between classes effectively. A score in the range of 0.7 to 0.8 is deemed acceptable, a score from 0.8 to 0.9 is viewed as excellent, and a score exceeding 0.9 is regarded as exceptional in terms of the model's classification performance. [Mandrekar, 2010a].

When using multiclass classification, a problem arises regarding the AUC since there are multiple classes, and getting one AUC for each class can only inform about the individual class, but not about the overall performance. In order to calculate an overall AUC, two methods are often used: macro and micro AUC. Macro AUC is a method in which the percentage of the correctly predicted samples for each class is divided by the number of classes as seen in Equation (6.10). This method is often used on balanced data since it weights the classes equally.

$$MacroPR = \frac{Pr_a + Pr_b + Pr_c}{n_{classes}} \tag{6.10}$$

Pr_n is the percentage of correctly predicted samples for class n.

If the dataset is imbalanced, the micro method is often used, since it takes the number of correctly predicted samples from each class divided by the total number of samples from all classes, as seen in Equation (6.11). Due to the imbalanced nature of the dataset in this project, with the control group comprising approximately 55% of the total data, the micro method was employed to calculate the average AUC.

$$MicroPR = \frac{Cp_a + Cp_b + Cp_c}{n_p} \tag{6.11}$$

Cp_n is the number of correctly predicted samples for class n.

Training and Validation of the Network 7

This chapter starts with an introduction of the overall strategy for the model development process. This is then followed by a structured description of the different optimizations used on the model to improve the performance.

7.1 Development Strategy of the Network

The dataset was initially partitioned into training, validation, and testing subsets. The training and validation set were used to train and validate the models through an iterative process, where different architectures and model parameters were tested. The models with the best validation curves and accuracy were saved as candidates for the final valuation with the test set. The model's performance was evaluated by using the classification performance metrics discussed in Section 6.7. Due to difficulties with training a model with acceptable performance, the planned duration for the training process was extended in hopes of finding a more robust model. The time planning for this project can be read in Appendix D.

7.2 Separation of Data

The first step of the process was the separation of the data, which was chosen to be a 60/20/20 split, where 60% of the data were used as the training set and 20% were used for both the validation and test sets. For the separation of the data, it was ensured that data belonging to the same experiment always was in the same dataset. This ensured that data from the same experiment was not used for both training and validation. One drawback of this method was the inability to achieve a perfect 60/20/20 split due to the allocation of the entire experiment into either the training, validation, or test set. Consequently, the resulting groups comprised 62.2%, 20.7%, and 17.1% respectively. In Table 7.1, the number of experiments from each class in the training, validation, and test split is shown.

Table 7.1. Division of dataset into training, validation, and test. Each row consist of how many subjects were in each of the data splits. Note that for the control, the value included the first measurement block of both the SNI and LTP subjects in the given split.

	Training	**Validation**	**Test**
Control	11	4	3
LTP	3	1	1
SNI	3	1	1

7.3 Training Process

Following the data partitioning, an iterative training cycle was performed, where the architecture and parameters of the CNN-LSTM were adjusted to achieve the best converging and accurate model. To start with some general architectures with 1-3 convolutional layers, 1-3 fully connected layers, max pooling, and ReLU for the CNN were used to find a good starting point. [Craik et al., 2019; Atchuthan et al., 2023]. The LSTM was tested with 2 layers with a size between 50-200 neurons for each layer. An overview of the iterative training process can be seen in Table 7.2.

Table 7.2. Plan for training the CNN. The training parameters have been divided into input layer, hidden layers, and backpropagation. The second column represents the range of changes to parameters that were performed.

Plan for iterative training process	
Input layer	
Image	CWT as input → 1-200 Hz (82 x 100 pixels)
Gaussian noise	On/Off
Hidden layers	
Convolutional layers	
No. of convolutional layers	1, 2, 3
No. of filters (features)	8, 16, 32, 64
Activation function	Convolutional layer: ReLU
No of pooling layers	Number of poolings = number of convolutional layers
No. of fully connected layers	1, 2, 3
Kernel size	2x2, 3x3, 4x4, 5x5, 6x6
Padding	0, 1 and 2
Pooling method	Max pooling
Pooling size	2x2, 3x3, 4x4, 5x5
Pooling stride	2,3,4,5
Dropout	Between 0.2-0.8
Fully connected layer	
No. of hidden neurons in fully connected layer	2 - 200
Dropout	0.2-0.8
Backpropagation	
Optimizer	NAdam
Loss function	Cross Entropy
L2 regulation	On/Off
Batch size	6 - 64
Learning rate	0.02, 0.002, 0.005, 0.001, 0.0001
Scheduler	On/Off

From this iterative training process, the best performing model was chosen to be a base model, from which a structured optimization was made. The base model consisted of three 3 x 3 kernel size convolutional layers with 8, 32, and 64 feature maps, respectively. To this, one max pooling with 5 x 5 kernel with stride 5, and 2 max pooling layers with kernel size 2 x 2 with stride 2. The dropout was 0.7 for all layers, and the LSTM was bidirectional and had two layers each with a hidden size of 75. There were two fully connected layers with input sizes 150 and 75, respectively. Finally, Gaussian noise was added to the input, and L2 regulation was added to the loss function. The confusion matrix of the base model can be seen in Figure 7.1, where the accuracy was 63.57%. Additionally, the model was best at predicting the control group, whereas the SNI group was the hardest to predict.

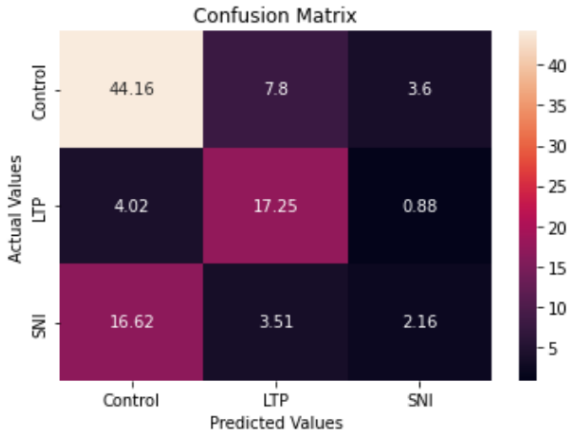


Figure 7.1. Confusion matrix showing the predictions for the base model on validation data. The values in the confusion matrix are shown in percentages.

From the learning curve, which can be seen in Figure 7.2, it can be seen that the training and validation loss follows each other until around epoch 15. Here the validation loss hit a plateau while the training loss kept dropping.

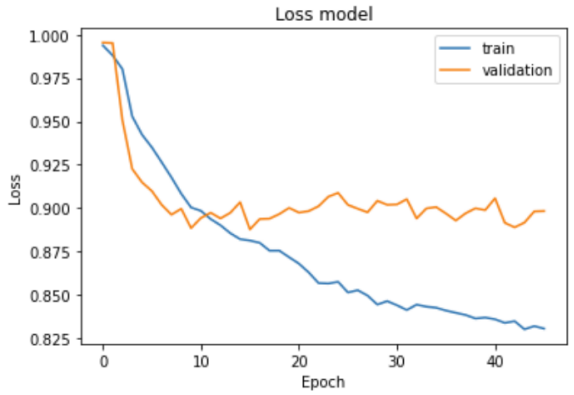


Figure 7.2. Learning curve of the base model, with the blue line being the training loss and the orange line the validation loss.

7.4 Structured Training Process

The base model served as the fundamental framework upon which the network's final tuning was built. A myriad of iterations were carried out for optimizing the model. This section will highlight the various transformative enhancements, including modifications in complexity, input variations, the incorporation of model binarization, and regularization and scheduling.

7.4.1 Changing the Complexity of the Network

Based on the observations in the base model, where it overfitted to the training data, changing the complexity of the network seemed intuitive. In order to reduce the complexity, two models were made. One where the number of convolutional layers was reduced to two, and another where the LSTM was removed. As seen in Figure 7.3, the confusion matrix showed that the network with two convolutional layers predicted SNI more than the base model, which resulted in fewer correctly predicted control instances. The learning curve showed the training loss dropped the more epochs it went through, whereas the validation loss dropped some in the beginning, but then started to slowly increase. When decreasing the complexity of the model, it resulted in including all three classes, however the training and validation loss did not follow each other and the accuracy dropped. Another approach to reduce the complexity was to remove the LSTM. As seen in Figure 7.3, the confusion matrix for the network without LSTM exclusively predicted the control class which was not desired. The learning curve showed little to no improvement for the training and validation loss. Since decreasing the complexity of the network did not improve the learning curve or accuracy, an increase in the complexity was tested. As seen in Figure 7.3, four convolutional layers were tested. The confusion matrix showed that the network was unable to predict SNI, as well as a weaker performance of differentiating between control and LTP. The learning curve converged validation and training loss in the beginning, but then they started to diverge after the 5th epoch.

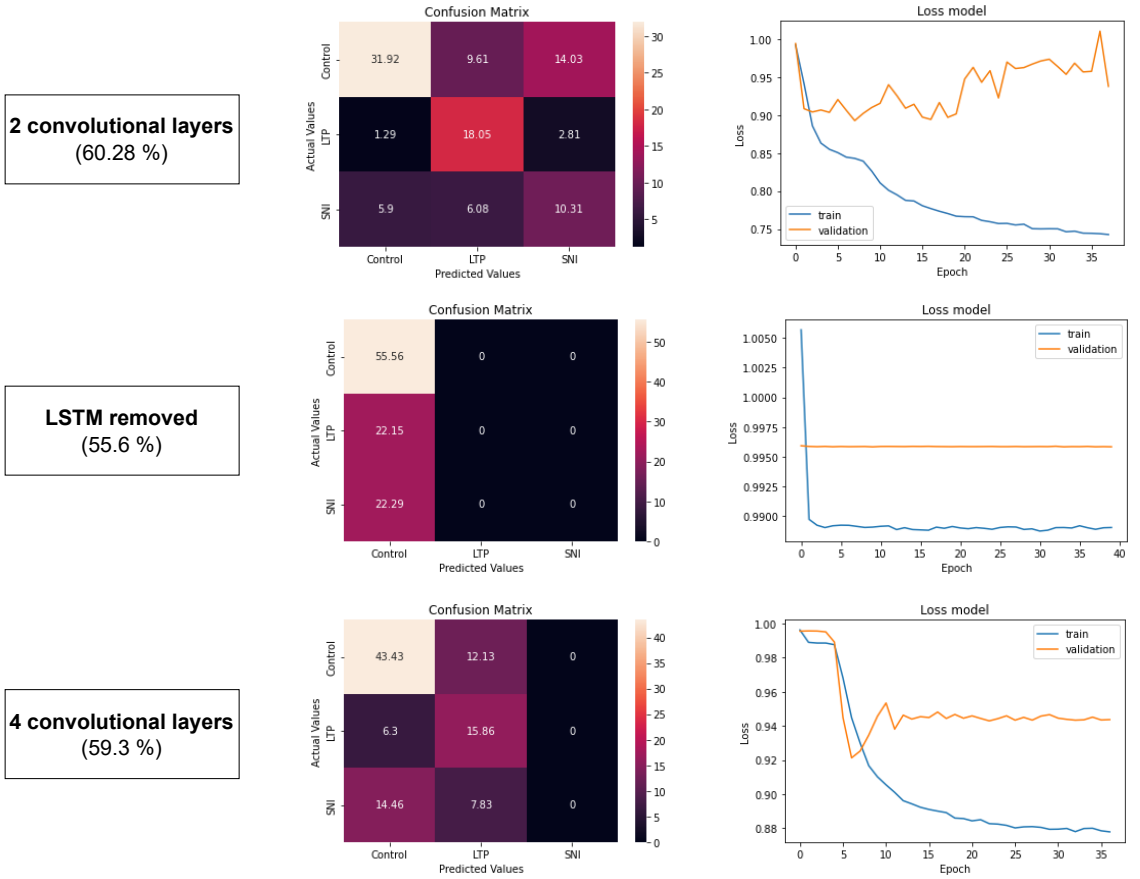


Figure 7.3. Learning curve of the base model, with the blue line being the training loss and the orange line the validation loss. The accuracies of the two models are written underneath the respective model names. The values in the confusion matrix are shown in percentages.

7.4.2 Balanced Classes

Based on the analysis of the confusion matrices, it became evident that there was a notable disparity in the classification of the different classes. The model exhibited a tendency to favor the control class, presumably due to its higher representation in the dataset. Additionally, the number of epochs for the LTP and SNI classes were approximately equal. In order to address this issue, a weighting scheme was implemented, taking into account the distribution of each class in the datasets. By modifying the cross-entropy loss function and assigning weights to each class, the model's bias towards underrepresented classes were mitigated. This approach effectively balances the datasets while preserving their inherent variance, by not having to remove any data to balance it. Evening out the dataset was also done for comparison. On Figure 7.4 the confusion matrix and the learning curve can be seen for both methods.

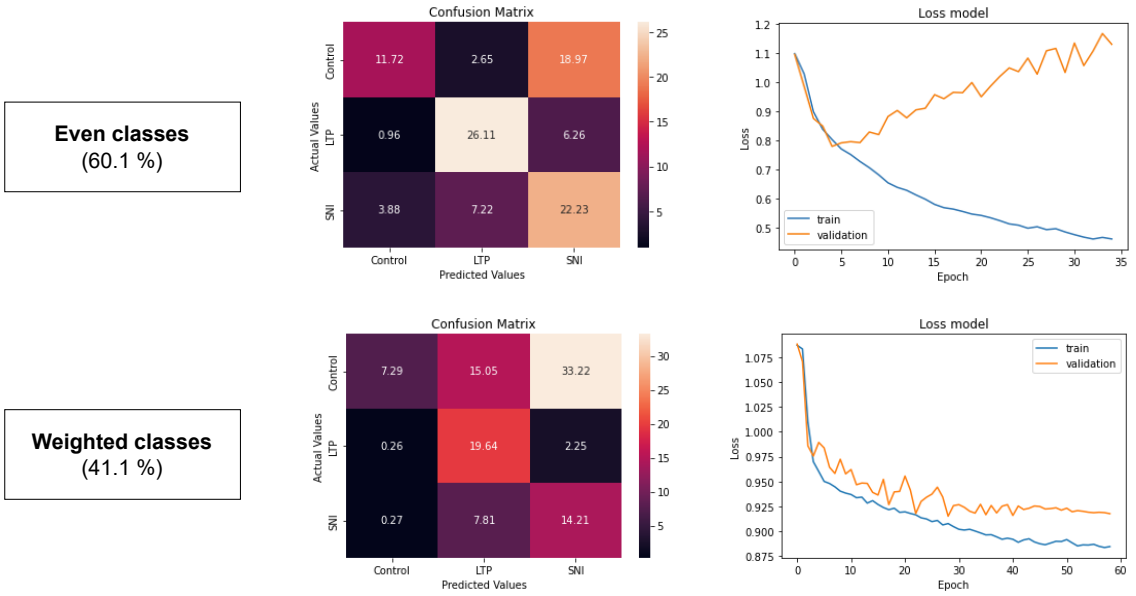


Figure 7.4. Confusion matrix and learning curve for both class balancing methods. The blue line represents the training loss curve, while the orange line represents the validation loss curve. The accuracies of the two models are written underneath the respective model names. The values in the confusion matrix are shown in percentages.

Both classes exhibited a decline in performance when evaluating accuracy. Additionally, the confusion matrix analysis confirmed the initial hypothesis that the model tended to predict the overrepresented class. Employing the methods of even dataset or class weighting resulted in a notable increase in predictions for the SNI class, while fewer predictions were made for the control class. However, this improvement came at the cost of both models where control were predicted as either LTP or SNI. When examining the learning curves, it became evident that the evened dataset started to diverge at around 5 epochs, indicating a pronounced overfitting to the training data. On the other hand, the weighted method demonstrated learning curves that better aligned with each other, converging at approximately 50-60 epochs, albeit with generally higher loss values.

Based on these findings, it was determined that neither of these models were suitable for further utilization in the subsequent stages of the structural training process.

7.4.3 Regularization & Scheduling

In the search for a better model, the L2 regularization and the scheduler for the learning rate were looked into. Therefore an attempt was made using the base model without the L2 regularization and one with the scheduler added. As seen in Figure 7.5, the model without L2 regularization had a higher accuracy of 63.9% which was 0.3% points more than the base model. The correct predictions for each class were similar to the base model, in which the model could not predict SNI correctly. The learning curve of the training and validation loss follows each other better than the base model until around epoch 65 where

they start to diverge. The model with scheduler achieved an accuracy of 63.0% accuracy and had the same classification problems as the base model, where SNI was predicted as control. The learning curve showed a convergence until epoch 30, where the validation and training loss started to diverge. These two models were both better than the majority of the tested models. However, the model without L2 regularization performed the best, which is why this was chosen for further consideration.

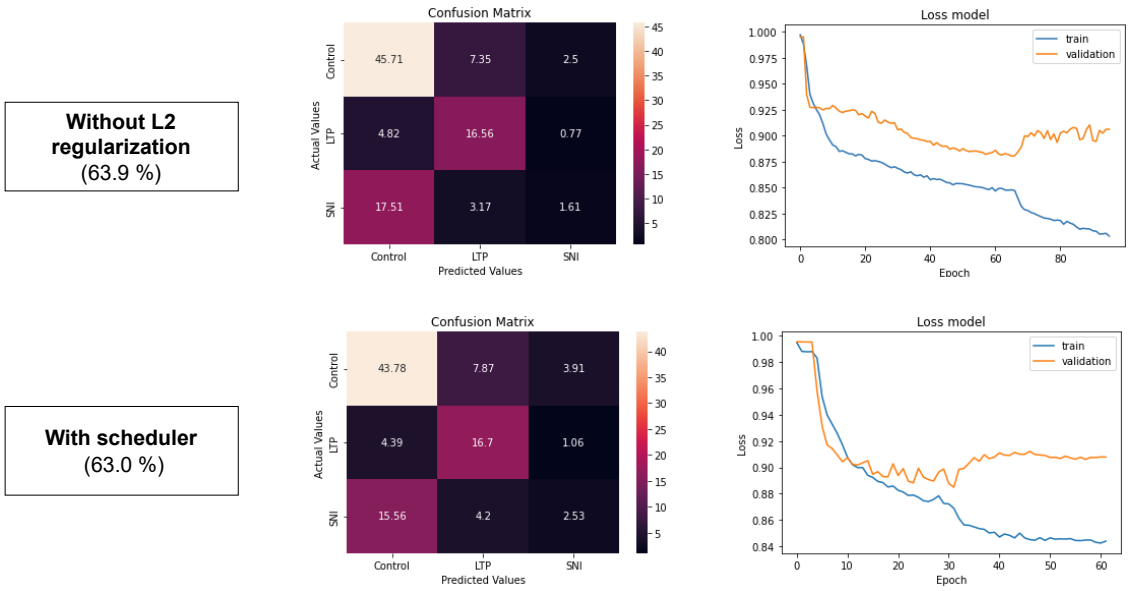


Figure 7.5. Learning curve and confusion matrix of the base model without L2 regularization and one with a scheduler. The blue and orange lines are the training and validation loss, respectively. The accuracies of the two models are written underneath the respective model names. The values in the confusion matrix are shown in percentages.

7.4.4 Binary Classification

Since multiclass classification did not perform sufficiently, it was needed to test whether binary classification was a possibility. Here a one versus one approach was tested, which is a binary classification between each pair of groups. It was clear from the previous networks that it was possible to classify between LTP and control, but not between SNI and control. To this end, only the control versus LTP model was included in the project. Additionally, a binary classification model between control versus intervention was also tested. Both of these models were tested with the base model structure with Gaussian noise, L2 regularization, scheduler on, as well as the output layer had two neurons instead of three. For this purpose, the data was structured with respect to the desired binary classification. As seen in Figure 7.6, the confusion matrix for control versus LTP had a good classification rate between the classes with an accuracy of 81.7%. The learning curve of this model was promising compared to earlier models since the validation and training loss followed each other for a while throughout the training process until the validation loss flattened out. The confusion matrix for the model with control versus intervention, which

can be seen in Figure 7.6, did not predict sufficiently. The accuracy of 62.9% was lower than the base model with three classes. The learning curve exhibited diverging trends, with the validation loss increasing while the training loss decreased.

Based on these findings, it was determined that control versus LTP was suitable for further utilization in the subsequent stages of the structural training process. However, the control versus intervention model was not explored further.

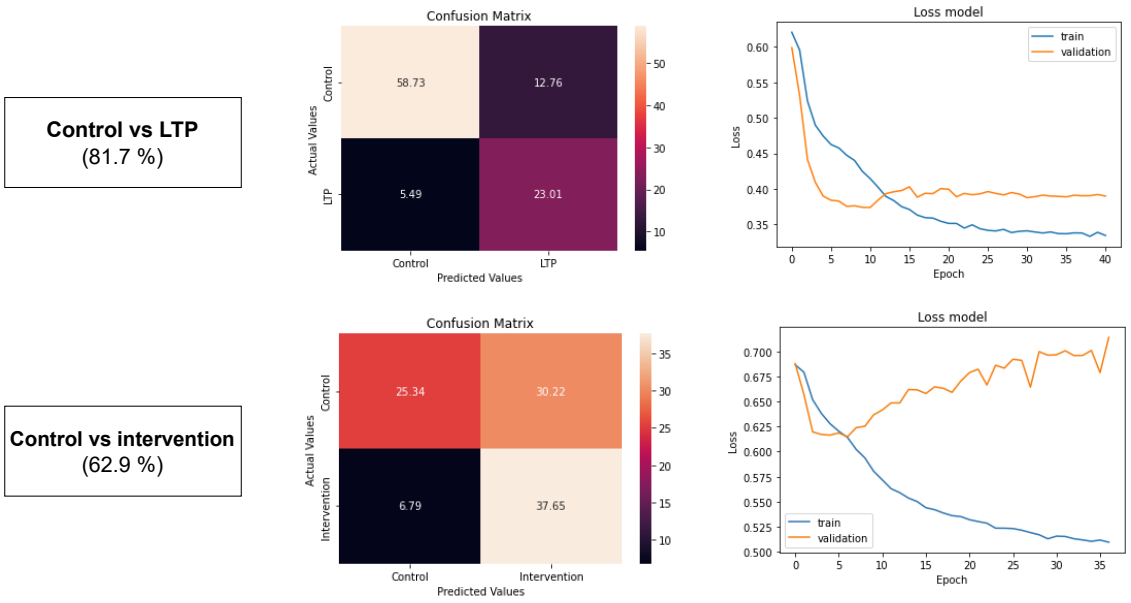


Figure 7.6. Learning curve and confusion matrix of control vs LTP and control vs intervention. The blue and orange lines are the training and validation loss, respectively. The accuracies of the two models are written underneath the respective model names. The values in the confusion matrix are shown in percentages.

7.5 Model Chosen for the Final Testing

After extensive evaluation, the best-performing models were identified for further utilization. Among the various approaches and modifications explored, two models have emerged as the top performers: the base model without L2 regularization as well as the binary model between control and LTP. The best models using these architectures were determined by employing a comprehensive training methodology. Each architecture underwent ten training iterations, with different initialization schemes. This allowed us to asses, models performance across multiple trials and mitigate the potential bias introduced by only training once. After the ten runs were completed, performance from the different models was analyzed based on the confusion matrix and learning curve to assess accuracy and robustness. The best version of each model based on the validation data is shown in Figure 7.7, where the model was saved with the lowest validation loss due to the model checkpoint. For the model without L2 regularization, the best performing model was the original model. For the binary model between control and LTP, a higher performing model

with 84.0% accuracy was achieved, as well as a learning curve with better convergence. The chosen instances of the multiclass and binary model were used for further evaluation of the test data.

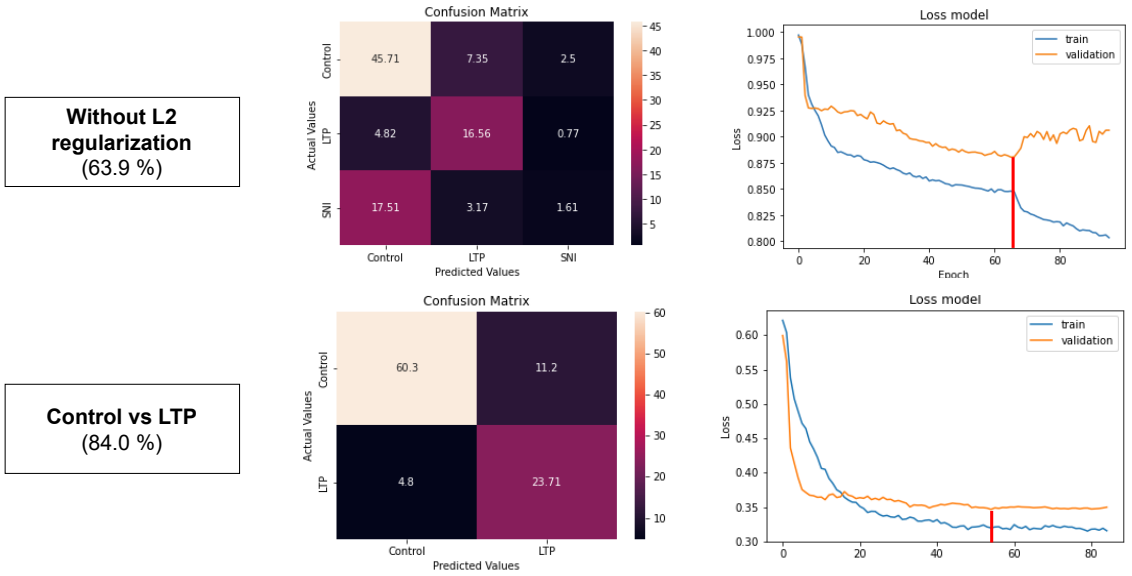


Figure 7.7. Learning curve and confusion matrix of the base model without L2 regularization and one with a scheduler. The blue and orange lines are the training and validation loss, respectively. The accuracies of the two models are written underneath the respective model names. The red lines indicate when the model was saved with the model checkpoint. The values in the confusion matrix are shown in percentages.

7.6 Final CNN-LSTM Architecture

The final CNN-LSTM architectures for the multiclass classification model and for the binary can be seen in Figure 7.8. The model consisted of 3 convolutional layers with kernel size 3 x 3 with ReLU as activation function and dropout of 70%. Each of the convolutional layers was connected to max pooling layers where the first max pooling layer had a kernel size of 5 x 5, where the rest had 2 x 2 kernel size. This was followed by a flattening layer with, two layer bidirectional LSTM with 75 cells and dropout of 50%, two fully connected layers with 50% dropout on the input layer, and finally the output layer with softmax activation function that had either 3 classes for the multiclass classification or 2 classes for the binary model.

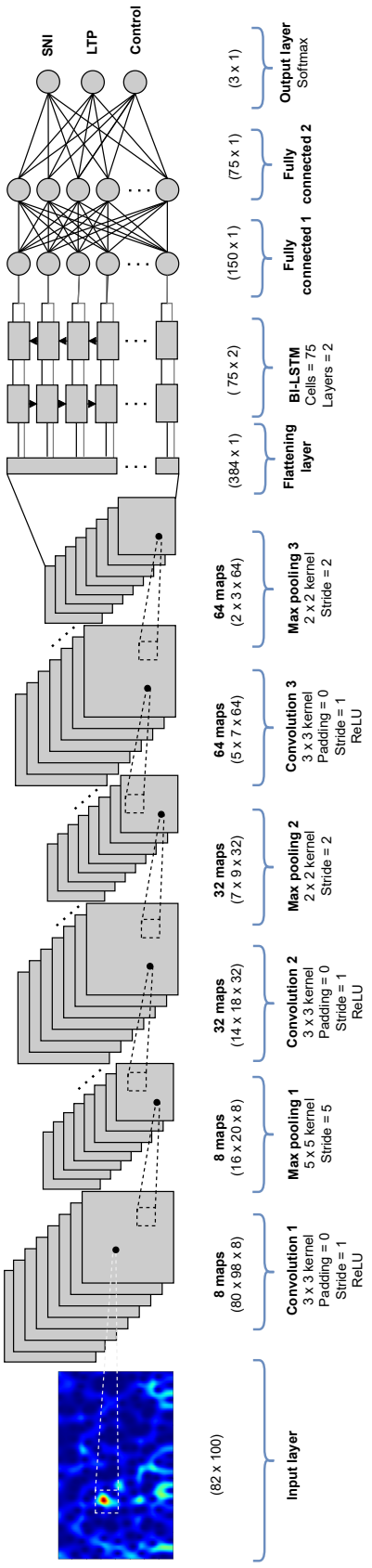


Figure 7.8. Illustration of the final architecture of the CNN-LSTM. The model consisted of 3 convolutional layers each with max pooling layers afterwards. This was followed by a flattening layer, two layer bidirectional LSTM, two fully connected layers, and finally the output layer with softmax activation function that had either 3 classes for the multiclass classification or 2 classes for the binary model.

Results 8

This chapter shows the results of the two chosen CNN-LSTM models, which were performed on the test data. The results are shown in the form of performance metrics, which quantify the performance of the CNN-LSTM model. Furthermore, the results of model interpretability is also presented for both models

8.1 Confusion Matrix

The confusion matrix in Figure 8.1 shows how well the models were able to predict on the test data. For the multiclass classification, it was clear that the model was unable to learn generalizable features based on the training data since the model only had correct predictions from the control class. The SNI class did not receive a single prediction, which indicates that the SNI data in the training set did not have similar features as the SNI data from the test set. The LTP class was predicted multiple times, however, no correct predictions were made. For the binary classification, the model predicted LTP but did not get any correct predictions. Although the control group was classified correctly more often than not, it did predict all the LTP instances as control. To this end, the same problem as the multiclass occurred where it predicted LTP but did not get any instances right.

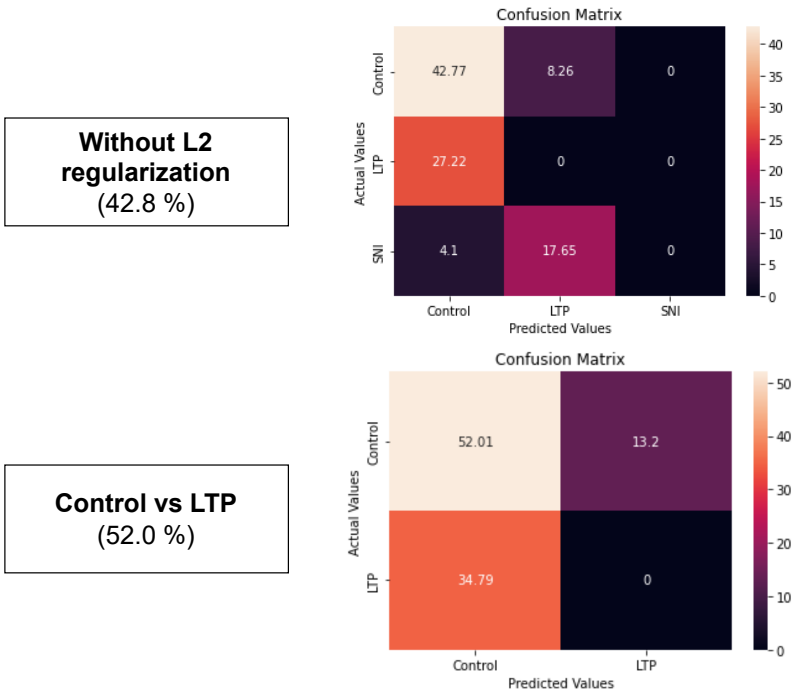


Figure 8.1. Confusion matrix for the model chosen for multiclass classification and binary classification. The accuracy for each model is listed beneath their respective names. The values in the confusion matrix are shown in percentages.

8.2 Accuracy and F1-score

For the multiclass classification only the control class had correct predictions resulting in an accuracy of 42.8%. Since the test data had an imbalance of the three classes, where 50% of the data were from the control class, the F1-score is a more suitable measure of the accuracy. Calculated based on Equation (6.7) the F1-score of 34.9% was achieved. For the binary classification, an accuracy of 52.0% was achieved. The F1-score was also calculated since the binary classification had an imbalance between the two classes, where the control class constituted 65% of the total dataset. The F1-score score of 44.6% was achieved.

8.3 Receiver Operating Characteristic Curve

The ROC curve would normally be used as another performance metric of the model, but is not a suitable metric for this instance, since two of the three classes did not have correct predictions for the multiclass classification, and no correct predictions for LTP in the binary classification. Therefore the ROC curve was only calculated for the control class in the multiclass classification. As seen in Figure 8.2, and as for the accuracy shown above, the model was poor at predicting correctly with an AUC of 0.48, which is below the line of no discrimination.

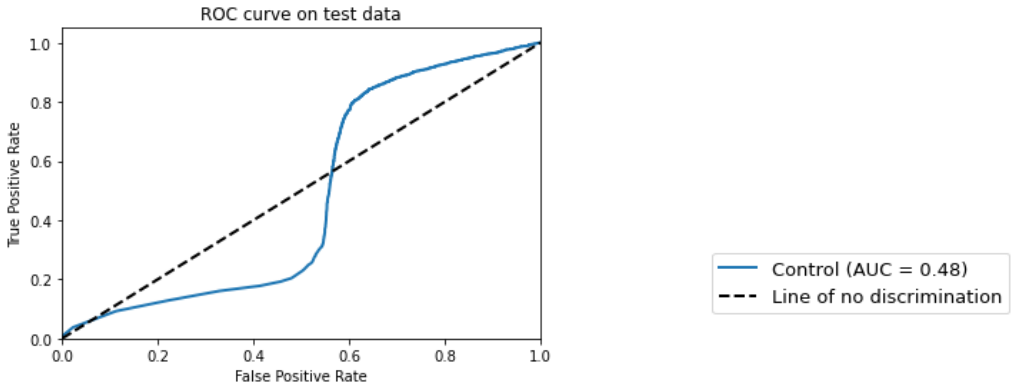


Figure 8.2. ROC curve with a line of no discrimination along the diagonal. The blue line is the ROC curve of the control class, with an AUC of 0.48

8.4 Model Interperability

This section aimed to present the outcomes obtained by applying diverse model interpretability methods. The primary objective was to compare the classification results of the validation and test datasets to identify significant discrepancies. This was because it was observed that a substantial disparity between the classification results of the test and validation sets. Additionally, the analysis primarily focused on comparing the control and LTP classes, as the SNI class exhibited poor performance in both the validation and test sets. Lastly, a comparison was made with the binary model highlighting the key differences in feature attribution between the two models.

It was worth mentioning that the occlusion method failed to provide any meaningful attribution and instead presented a noisy map devoid of any correlation with the actual response. This observation suggested that the model may not be relying on specific features that significantly impact its classification performance. An example of the noisy occlusion map can be seen in Figure 8.3.

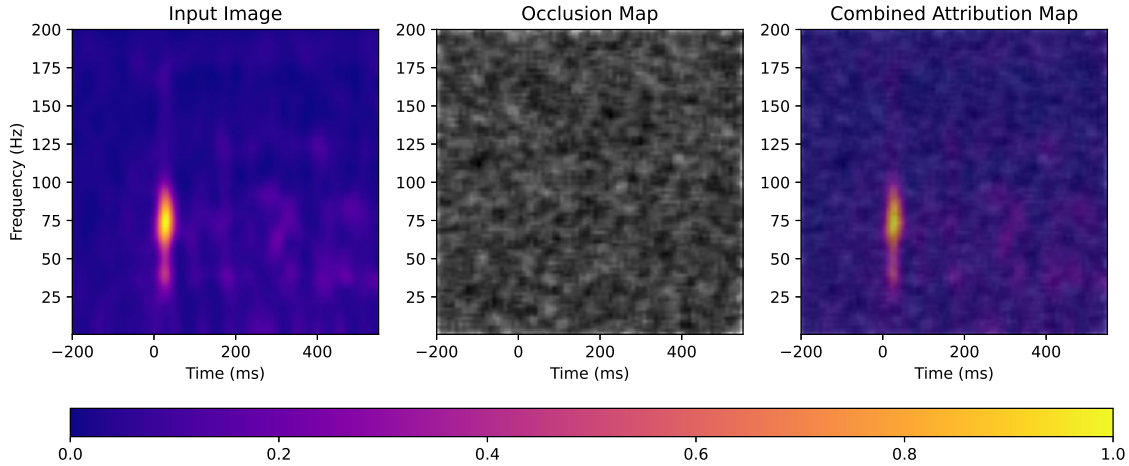


Figure 8.3. An example of the occlusion method applied to the validation set + control class reveals that the resulting attribution map exhibits no discernible correlation with the actual input.

The occlusion method was not further presented in this part of the results but can be found in Appendix B.

8.4.1 Multiclass Model

Control

The final results for the control class were computed using Saliency, Integrated Gradients, and DeepLIFT. These results can be seen on Figures 8.4 to 8.6 respectively.

When analyzing the input image alone, a significant disparity became apparent between the two types of control. The validation and test control classes exhibited displacement in both time and frequency when compared. Validation had a greater high-frequency span of 30 - 100 Hz shortly after stimulation at around 0 ms. Conversely, the test control had a shorter span of 0 - 25 Hz later in the signal at 200 ms. However, there was a subtle response in the test control that had a resemblance to the more pronounced response observed in the validation control at the same time point. In summary, there was a notable distinction between the two control groups.

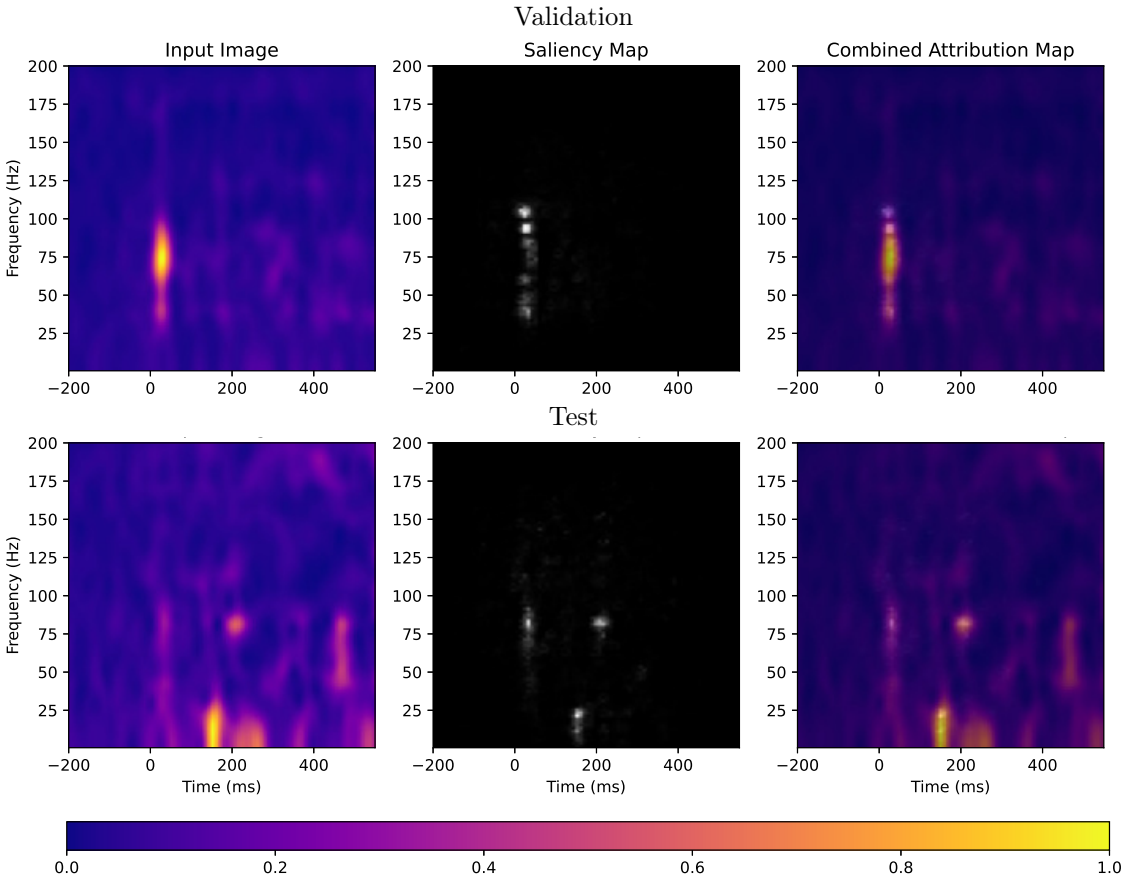


Figure 8.4. Saliency maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

Upon examining the saliency map (see Figure 8.4), differences in the contributing features were evident. The model's attention appeared to be concentrated at a specific time point, spanning across the entire active frequency band of the input, with a heightened focus on higher frequencies on the control validation. Conversely, in the control test, attention was more widely distributed both in terms of time and frequency. However, some correspondence was observed between the two sets when closely inspecting the response after 0 ms at 75 Hz. The remainder of the attention on the control test was centered around 200 ms, encompassing both the low-frequency range of 0 - 25 Hz and the high frequency of 75 Hz.

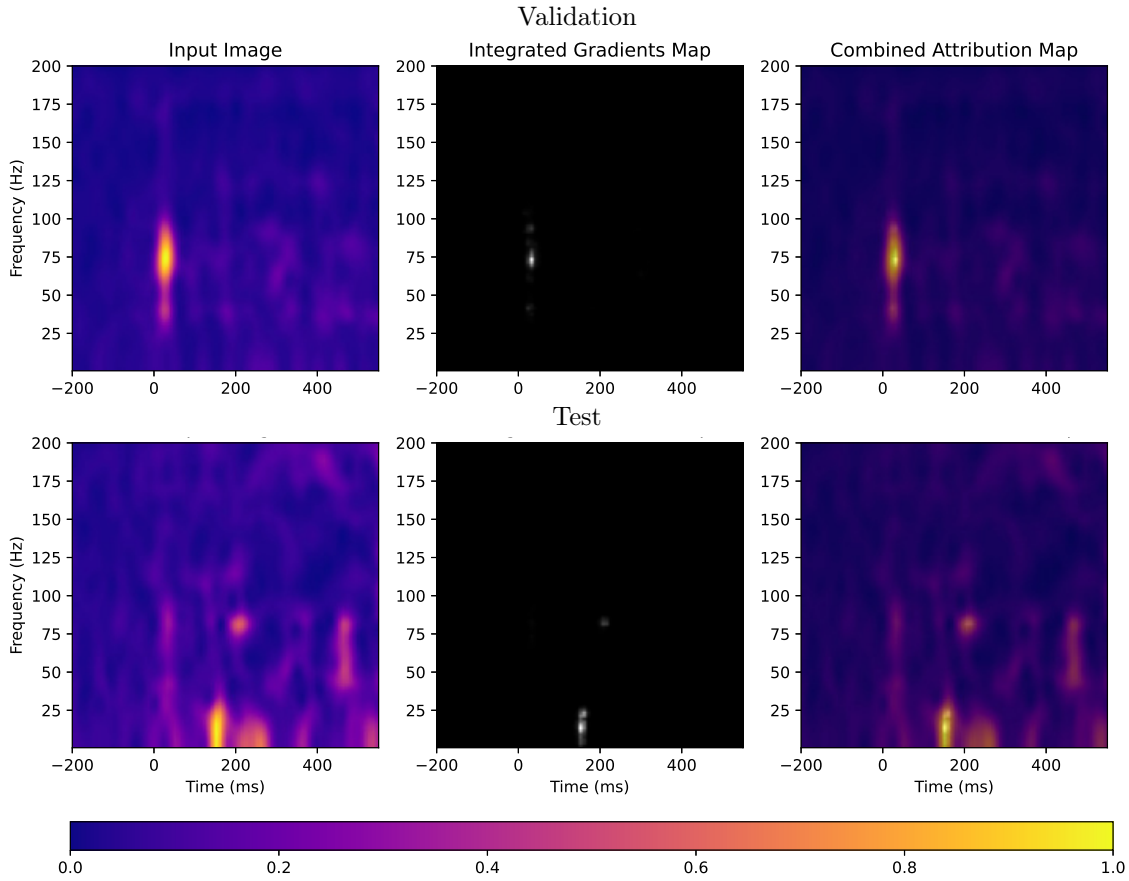


Figure 8.5. Integrated Gradients maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

On the integrated gradients map (see Figure 8.5), the differences in the contributing features were less prominent compared to saliency. Similar to the saliency map, in the control validation, the model's attention appeared to be concentrated at a specific time point after 0 ms, primarily focused on 75 Hz, with some attribution at 90 Hz both. Conversely, in the control test, the attention was less dispersed both in terms of time and frequency compared to saliency. However, unlike saliency, there were no apparent similarities between the validation and test control. In the control test, the attention was centered around 200 ms, encompassing the low-frequency range of 0 - 25 Hz with a negligible attribution at the high frequency of 80 Hz.

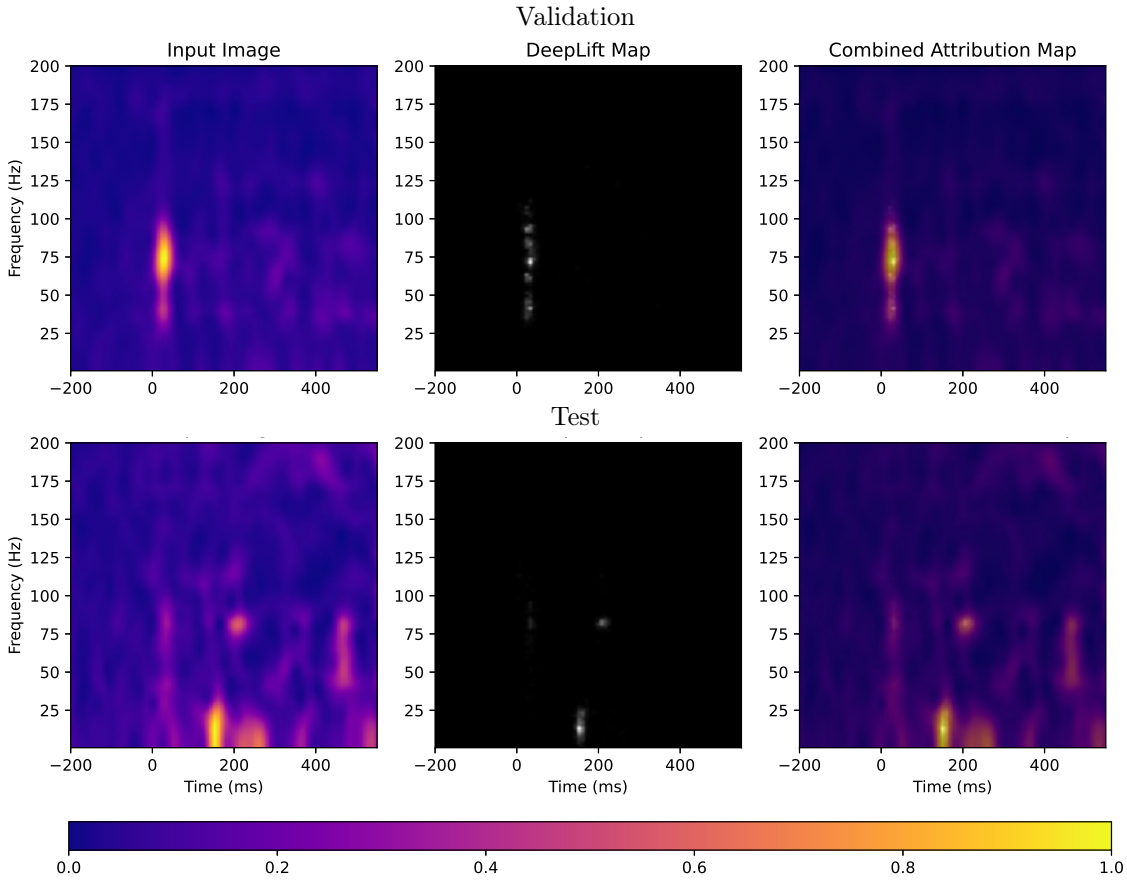


Figure 8.6. DeepLIFT maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

Lastly, the deeplift analysis (see Figure 8.6) revealed similarities to the saliency map, with a broader attribution in the frequency domain spanning from 30 to 100 Hz, while remaining highly specific to a particular time point around 0 ms. As for the control set, the deeplift map exhibited similarities to the integrated gradients, with attention centered around 200 ms. It encompasses the low-frequency range of 0-25 Hz, with minimal attribution at the high frequency of 80 Hz.

To summarize, a different attention from the model was seen when looking at the two types of datasets. Using the three methods saliency, integrated gradients and deeplift it was evident that the model was looking at different time points and frequencies. This corresponds with the input where there seemed to be some latency in the response and low-frequency activity after stimulation in the control test compared to validation.

LTP

The LTP class attribution was determined through the utilization of Saliency, Integrated Gradients, and DeepLIFT techniques. The outcomes of these computations are showcased in Figures 8.7 to 8.9 correspondingly.

Examining the input image in isolation revealed a minor variation in the frequency range. Specifically, there was a slight downward shift from the LTP validation to the test. In the case of LTP validation, the frequency range spanned from 40 - 175 Hz, with a peak around 75 Hz. Conversely, in the LTP test, the frequency range covered 40 - 140 Hz, with a peak at 40 Hz. In short, there was more resemblance between the validation and test data, although there was more activity at the higher frequencies on the validation

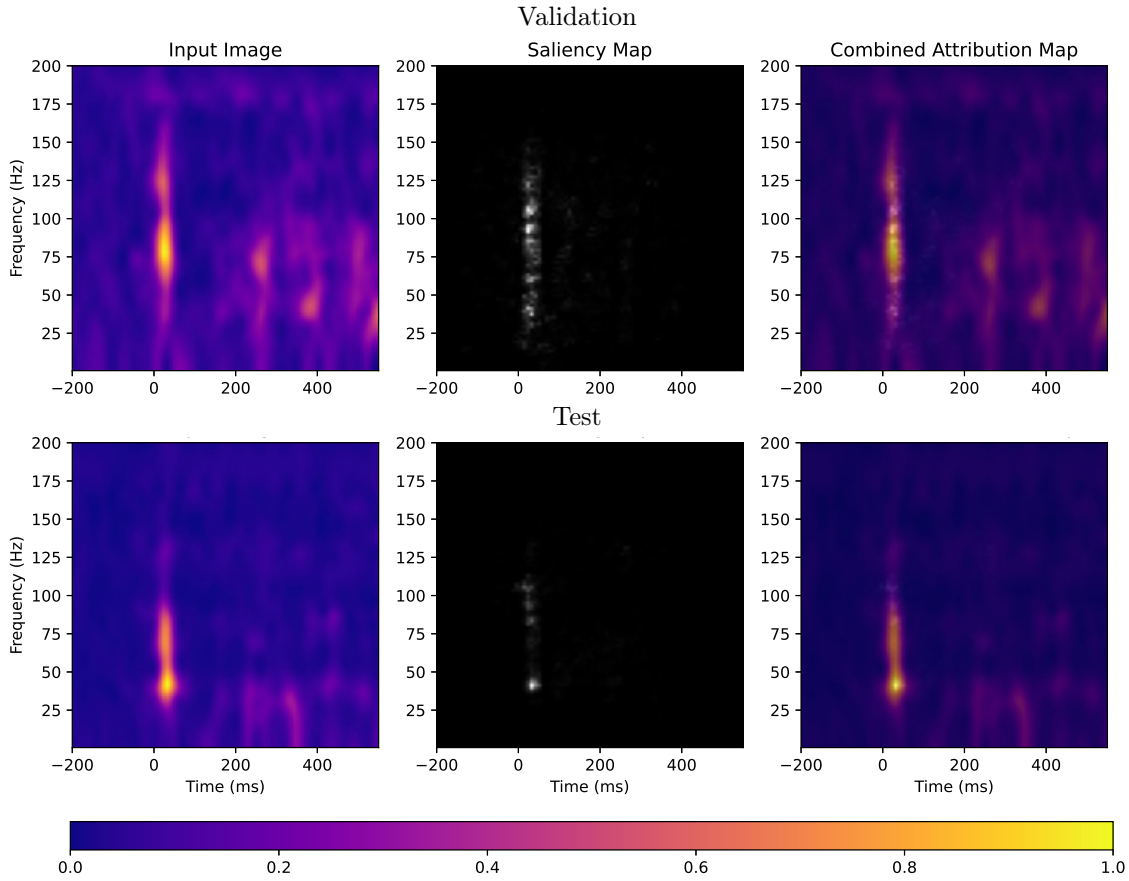


Figure 8.7. Saliency maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

When examining the saliency map (see Figure 8.7), the validation phase revealed a broad and comprehensive attribution, encompassing the complete response of the input image across a frequency range of 25 - 150 Hz. In contrast, the LTP test demonstrated a narrower attribution, with a particular focus on the prominent frequency of 40 Hz.

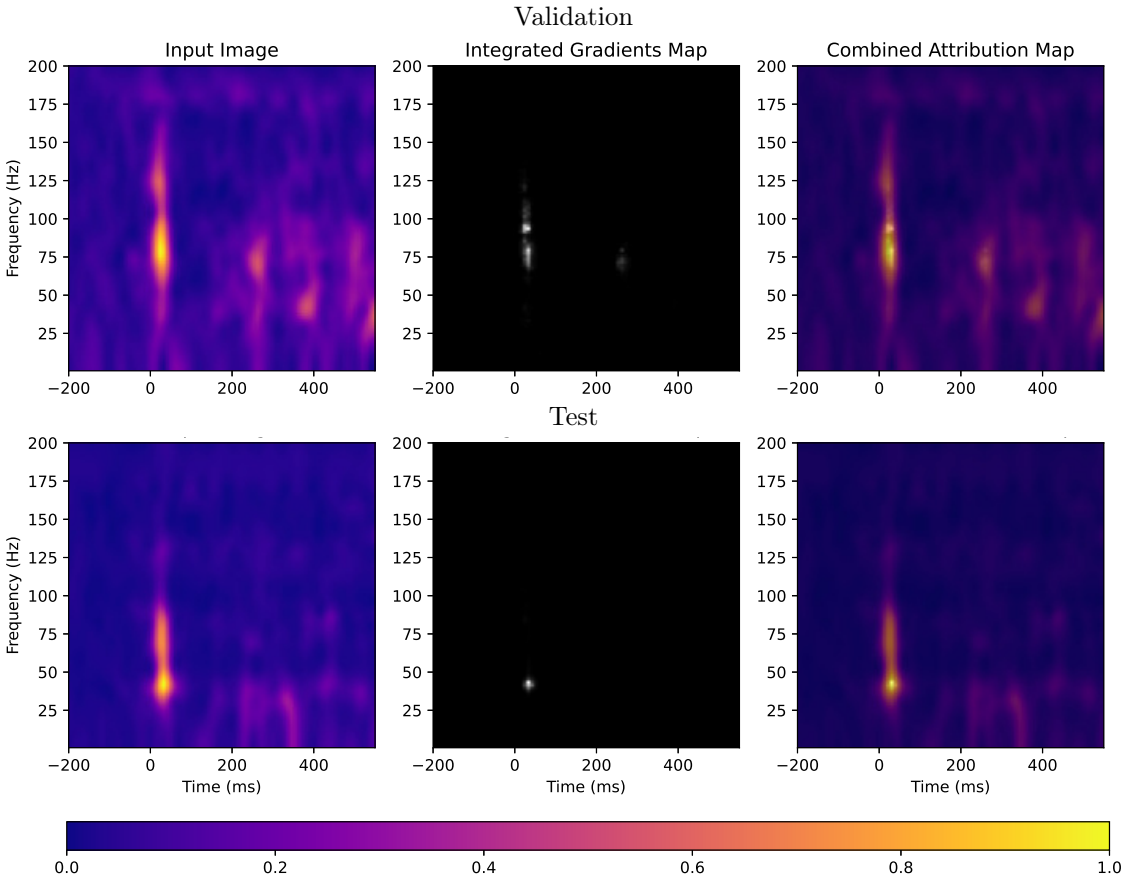


Figure 8.8. Integrated Gradients maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

The integrated gradients map (see Figure 8.8) showcased a more focused distribution of attributions. In the validation dataset, the range of attribution spanned from 75 - 120 Hz, exhibiting a narrower spread compared to other frequencies. On the other hand, the LTP test exhibited a distinct point of attribution at 40 Hz, around the 0 ms mark. Notably, both datasets demonstrated pronounced attributions that are clearly discernible from the background. However, in the validation set, there was an additional delayed attribution at approximately 280 ms and 75 Hz.

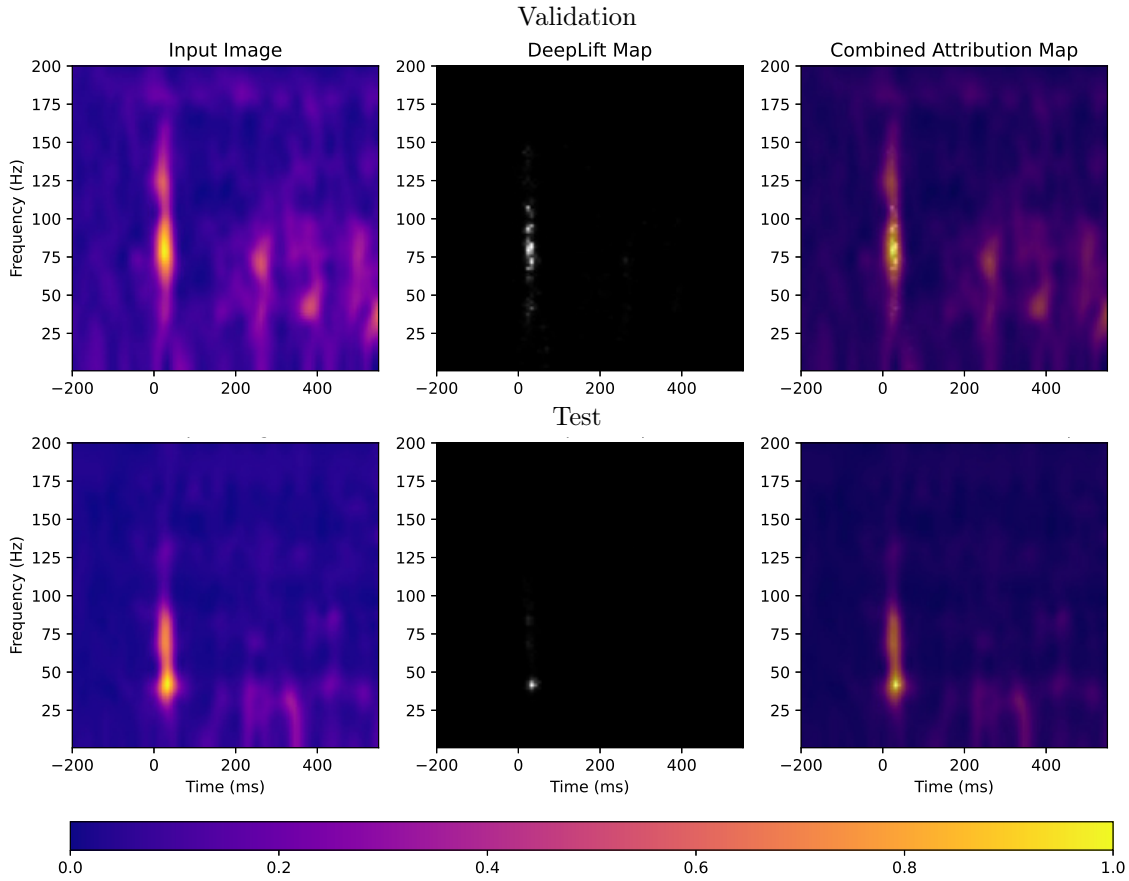


Figure 8.9. DeepLIFT maps for both the validation and test set. TLighter pixels indicate more important features, while darker pixels indicate less important features

In the deeplift analysis (see Figure 8.9), the attribution was found to be comparable to the saliency analysis. When examining the LTP validation dataset, the attribution appeared to be more widespread across a frequency range of 30 - 150 Hz, with the most prominent attribution observed between 70 - 110 Hz. On the other hand, the LTP test dataset exhibited a specific activation at 40 Hz immediately after stimulation at 0 ms. These findings indicated that both datasets showed distinct patterns of attribution in the deeplift analysis, with the validation dataset demonstrating a broader frequency distribution and the test dataset displaying a more specific and focused activation.

To summarize, a different attention from the model was seen when looking at these two types of datasets. Using the three methods saliency, integrated gradients and deeplift it was evident that the model was looking at the same time points and approximately the same frequencies. These observations align with the input data, indicating a resemblance between the validation and test datasets. However, there was a slightly heightened level of activity at higher frequencies in the validation dataset.

8.4.2 Comparison to the Binary Model

This section focused on comparing the interpretability results of the multiclass model and the binary model, specifically highlighting the significant differences in feature attribution between the two models.

Control

Upon analyzing the results using the three interpretability methods, a notable difference was observed primarily in the saliency method. However, the other interpretability methods employed for the control using the binary model can be found in Appendices B.2.1 and B.2.2. These appendices provide a comprehensive overview of the interpretability methods and their outcomes for the binary model control experiments.

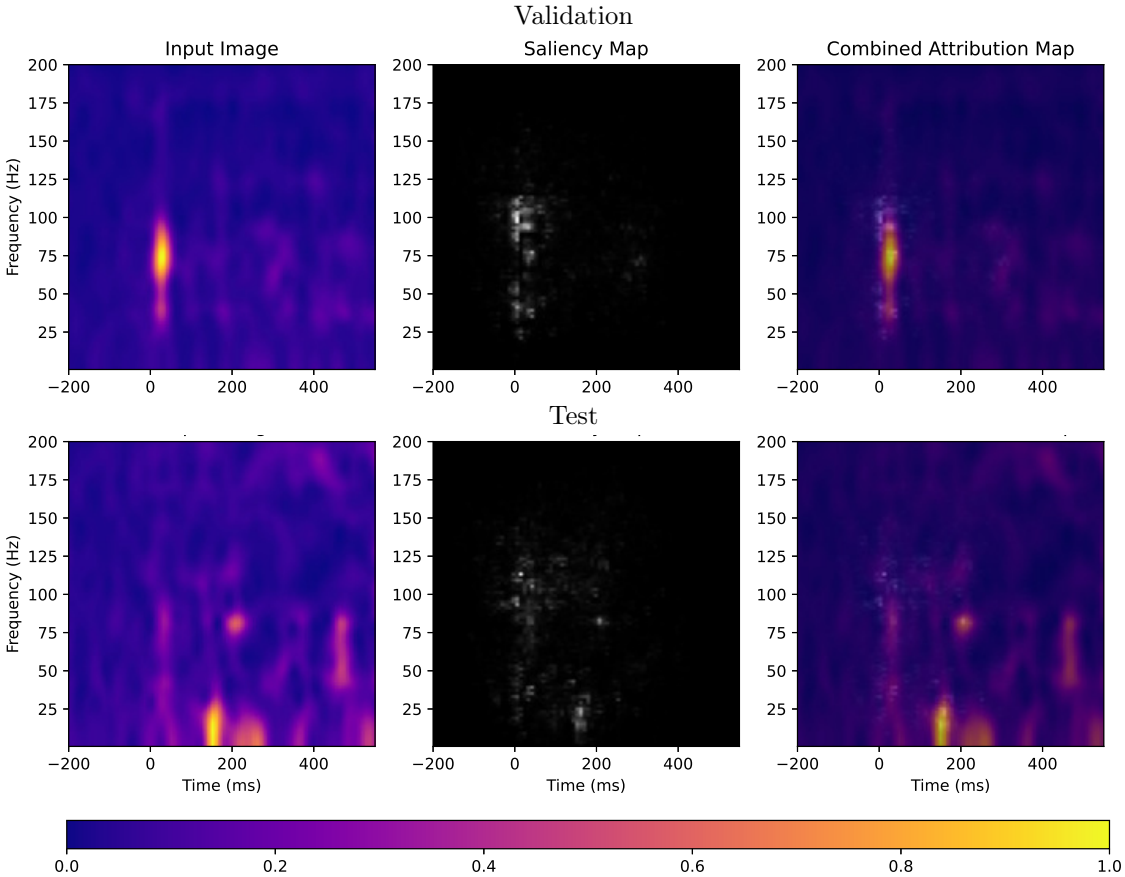


Figure 8.10. Saliency maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

The saliency maps (see Figure 8.10) for both the control validation and test datasets exhibited a distinctive speckle-like pattern that was spread across the image. In the validation dataset, this pattern appeared within a narrower window in the time domain, specifically ranging from 0 to 100 ms. Additionally, there was some residual attribution observed around the 300 ms mark. In terms of frequency, the saliency contributions ranged

from 25 - 125 Hz, with the residue attribution specifically occurring around 75 Hz. On the other hand, the test dataset demonstrated a more evenly spread pattern in both the time and frequency domains. The saliency contributions were distributed over a wider time range, ranging from 0 - 200 ms. Similarly, the frequency range spanned from 0 - 150 Hz, showcasing a broader distribution of attributions compared to the validation dataset.

It was important to note that the speckle-like pattern observed in the saliency maps of the control validation and test datasets was not evident in the saliency maps of the multiclass model attribution. This difference suggested that the binary model's interpretability results exhibited a distinct pattern that was not observed in the multiclass model's attributions for the control group.

In summary, the interpretability analysis of the binary model's control experiments showed a distinct speckle-like pattern in the saliency maps. The pattern was observed in both the validation and test datasets, with narrower time windows and specific frequency ranges. Importantly, this pattern was not seen in the saliency maps of the multiclass model.

LTP

Upon analyzing the results using the three interpretability methods, a notable difference was observed primarily in the saliency and deeplift methods. However, the other interpretability method integrated gradients employed for the LTP used on the binary model can be found in Appendices B.2.1 and B.2.2. These appendices provide a comprehensive overview of the interpretability methods and their outcomes for the binary model LTP experiments.

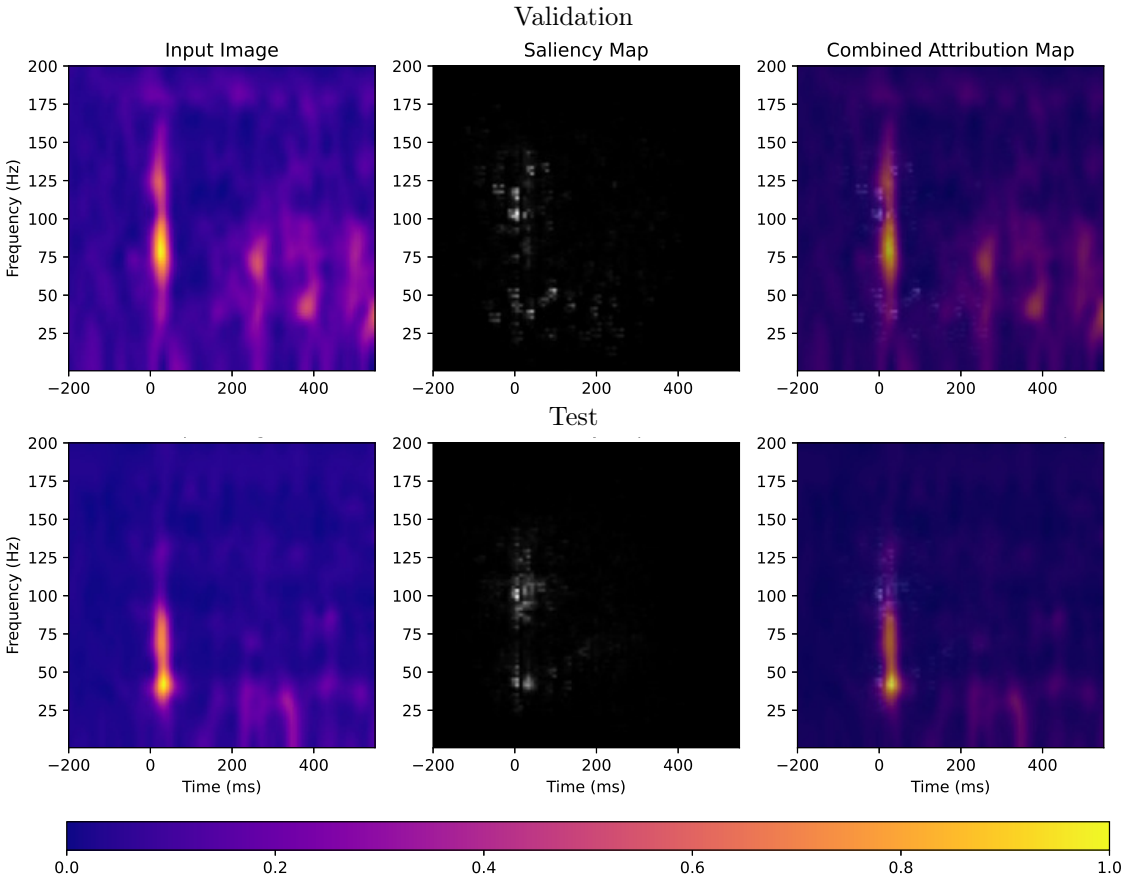


Figure 8.11. Saliency maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

The saliency maps (see Figure 8.11) for both the LTP validation and test datasets exhibited speckle patterns similar to those observed in the control set. In the validation dataset, the frequencies ranged from 25 - 150 Hz, with a broader spread observed in the lower frequencies within the timeframe of approximately 0 - 200 ms. In the test dataset, there was a narrower distribution in the time domain, and resembling the frequency span observed in the validation dataset. Notably, both datasets show the highest attribution at approximately 100 Hz precisely at the stimulation time point of 0 ms.

Deeplift (see Figure 8.12) shows a slight difference in only the validation patterns, again with the speckle-like attributions, ranging in 0 - 300 ms at frequency ranges at 25 - 140 Hz.

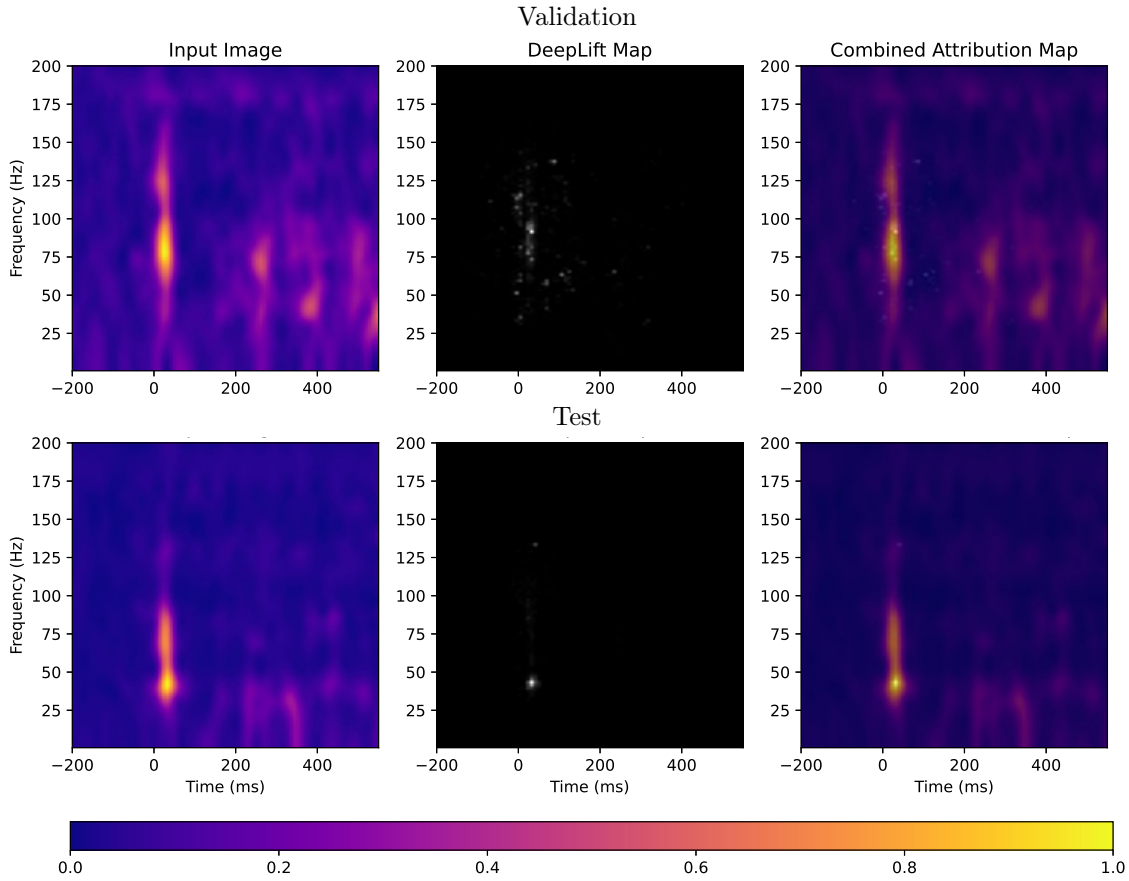


Figure 8.12. DeepLIFT maps for both the validation and test set. Lighter pixels indicate more important features, while darker pixels indicate less important features

In summary, the interpretability analysis of the binary model's LTP experiments revealed distinct patterns in the saliency and deeplift methods. The saliency maps showed speckle-like patterns in both the validation and test datasets, with a broader spread of frequencies observed in the validation dataset and a narrower distribution in the time domain in the test dataset. Notably, both datasets exhibited a prominent attribution at around 100 Hz during the stimulation time point of 0 ms. The deeplift method displayed similar speckle-like attributions, ranging from 0 - 300 ms and frequencies of 25 - 140 Hz in the validation dataset.

Discussion 9

This chapter provides a summary of the findings of the report and attempts to interpret the results. Both the results and methodological considerations will be discussed. Additionally, the chapter concludes with a discussion on the limitations and potential future directions within the methodology.

9.1 Comparison of Results

In this project, a CNN combined with an LSTM model was developed to classify between control, LTP, or SNI pain models in pigs. Continuous wavelet transform of ERPs from ECoG signals in S1 was calculated as input. To our knowledge, no papers have been published with classification between different chronic pain models in large animals. This underlines the novelty of this project, but also the complexity of the problem. Because of this novelty, a more objective solution was chosen, using a deep learning architecture, in the hopes of it detecting patterns unbeknownst to the literature.

For the multiclass classification, the model predicted 42.8% of the total predictions correctly on the test data, and an F1-score of 34.9%. The difference between accuracy and F1-score can be explained by the fact that only the control class had correctly predicted samples, and since the classes were imbalanced, the F1-score dropped notably. For the binary classification, an accuracy of 52.0% was achieved with an F1-score of 44.6%. The drop in the accuracy score to the F1-score is due to the same reasons as for the multiclass classification, where only the control group was correctly predicted. These results indicate that the features learned from the training data, which were applicable to the validation data where an accuracy of 63.9% and 84.0% was achieved for multiclass and binary classification respectively, did not seem to be transferable to the test data for both models.

Another interesting result was the absence of predictions for the SNI class from the models. This was recurrent through most of the models, that the SNI class was barely predicted from any of the models, with the exception of the balanced classes. This indicates that since the majority of the data are control, the models had a hard time predicting the SNI class. This does however not explain why the models were still capable of predicting the LTP class since the amount of data was the same in the SNI and LTP class. Perhaps the LTP and control data were not as similar as the control and the SNI classes, making it

more likely to predict the LTP class than the SNI when data was unbalanced towards the control data.

Similar studies have classified between chronic pain patients and healthy controls. Santana et al. [2019] used different convolutional neural networks with resting-state fMRI scannings from 60 patients (control = 98) as input, and achieved a balanced accuracy of 86.8% and an AUC of 0.93. When comparing these results to this project with an accuracy of 52.0% for the binary model, it is clear that their accuracy is higher. However, when comparing it to the validation accuracy of 84.0%, these results are similar. Due to the absence of a confusion matrix in Santana et al. [2019], it becomes challenging to assess whether they encountered difficulties with classifying one of their classes, like in this project where the LTP group was not correctly predicted.

By employing model interpretability techniques, it was discovered that the multiclass model exhibits different attention to time points and frequencies between validation and test, where the test is seen to have a more delayed feature attribution timewise. This observation is further reinforced when examining the input alone, as the response pattern is overshadowed by heightened activity in later time points of the signal (Figure 8.5). Notably, there are additional disparities observed in the LTP attribution regarding frequencies. Specifically, it becomes apparent that the model focuses on a singular peak when analyzing the test data, whereas the validation data exhibited a more dispersed distribution across frequencies. These two aforementioned discoveries potentially shed light on the challenges faced by the model in achieving improved performance on the test data. The significant dissimilarities in the features it attributes to may impede its ability to accurately classify the test samples. This is due to the expected patterns and characteristics of the class are not sufficiently recognized.

Moreover, it appears that the model fails to attribute complex feature relationships to the test data, as it predominantly focuses on single peaks. This limited perspective of the model prevents it from capturing intricate patterns and interdependencies within the data, potentially contributing to its inability to achieve higher performance on the test dataset. This limitation is especially undesirable since the feature it predominantly examines bears a striking resemblance to existing, less sophisticated methods like N1/P1-based classification, offering little novelty compared to established classical approaches as seen in Lenoir et al. [2020].

The misclassification of LTP as as control in the test dataset can be related to the observed similarities in attribution maps when comparing control and LTP instances. Both the control (Figure 8.5) and LTP (Figure 8.8) attribution maps exhibit a distinct point-like pattern, with the primary difference lying in the specific position of the feature. For the control class, the feature is centered around 190 ms and 20 Hz, whereas for LTP,

it is centered around 20 ms and 40 Hz. This resemblance in the attribution patterns can potentially lead to confusion for the model during the classification process, resulting in misclassifications of LTP instances as control. The proximity of the feature positions between the two classes, despite their differences, may also contribute to the model's difficulty in accurately discerning between them in the test dataset. The significant differences in the spread of frequencies between the LTP validation and test data further contributes to the model's challenge in accurately classifying LTP instances. The model's reliance on a single peak in the test data, while the validation data exhibits a broader frequency distribution, explains the misclassification of LTP as control.

9.2 Methodological Considerations

9.2.1 Data

An important part of training a neural network is the data provided to the model. Although a lot of data was available in this project, the subjects from which it originated were limited. For both the SNI and the LTP interventions only five subjects had each intervention. This meant that the 60/20/20 split for train, validation, and test data only included one subject in each validation and test set, while the train data had three subjects. This may have been enough subject if the features between the classes were clearly distinguishable, but that did not seem to be the case from the results. This makes it difficult to say whether the models' inability to correctly predict the classes were due to it not being possible or if there were too few subjects in both the intervention classes. If there are too few subjects, it can cause the variance between subjects to be greater than the variance between classes, making it difficult to classify the samples correctly. This is because there is a chance of outliers or extreme values from single experiments which, when having a few subjects in each class, can significantly impact the variance within a class which can dominate the variance between the groups. A smaller sample size can also give a skewed reflection of the true population affecting the generalizability of the findings. But how many subjects are enough? To this end, there is also an ethical aspect of deciding how many animals should be used in these experiments. The more animals there are used for a single experiment, the less every single animal matters in the overall analysis. On the other hand, it should also be insured that enough animals are used to produce statistically significant and reliable results, otherwise, the animals would be used without a meaningful outcome. [Picciotto, 2020]

Seeing the confusion matrices of the different models, it is clear that the SNI intervention group was an issue since the models very rarely predicted this. A possibility is that the SNI and control class did not have a difference that was noticeable by the model. This may be explained by the recording site of the signals, which was on top of the dura. A

study has found that following a nerve injury intervention, the peak amplitude of neuronal activity significantly increased only when reaching the third to the sixth cortical layers compared to control [Meijs et al., 2022]. It is uncertain whether the electrode on top of the dura is capable of reaching signals from the third cortical layer because of the attenuation by the layers. Therefore, the lack of noticeable differences between the SNI and control groups could very well be an issue of signal detection rather than a failure of the machine learning models themselves.

Another limiting factor for this project has been the size of the recording site since only signals from the S1 were recorded. There are multiple cortical and subcortical structures involved in different aspects of the pain experience, such as the ACC or the thalamus, which could have contained more information useful to the algorithm. [Martucci and Mackey, 2018]. In human experiments, a larger area of the brain is often looked upon using EEG caps or fMRI. The fMRI has the advantage of being able to visualize the subcortical areas, such as the limbic system, which is an area involved in the processing of pain. Using EEG could have been beneficial since it is a non-invasive alternative, which is more likely to be used on humans. It is however much less precise and the signals are dampened by the layers between the scalp and the brain. Considering the preceding discussion, the S1 emerges as the most plausible region of the brain to anticipate alterations, especially when the experiment at hand pertains to the stimulation of sensory neurons.

An important consideration for the results of this project is the time scale of the experiment. The recorded data used was pre-intervention and directly after the intervention, which is not similar to a real-life scenario with a patient. Patients will often experience pain for weeks before seeking treatment, which is enough time for neuroplasticity to occur and change neural activity. Hence, it would be more favorable to utilize a model that has been trained on data similar to a clinical setting, where patients have experienced pain for an extended duration. [Mussigmann et al., 2022; Pricope et al., 2022].

9.2.2 Architecture

The training of the neural network is an iterative process in which the model is optimized by changing the hyperparameters and architecture. The large number of possible combinations makes it impractical to try each one, necessitating the development of a well-structured training plan. This plan was an attempt to ensure that only one change was made to the model at a time and thereby allowing the researchers to note which specific parameters or architectural changes had a positive effect on the learning. However, this does not mean that a better performing model than the ones achieved in this project cannot be accomplished, it is unlikely that the persistently inaccurate predictions are solely due to the model itself. It is more probable that the issue lies within the data. A recommendation

of the most important parameters can therefore not be made based on the results, since new data added to the model could potentially have a considerable impact on which parameters would output the best results.

9.3 Future Work

Because of the novelty of this project's aim, it was unclear from the start whether it would be possible to distinguish between the pain models. Although the results did not show it to be possible, future work may improve it and increase the capabilities of the model. This work should include more subjects since the results from this project have proven that too few can result in inconclusive results. Using more animals would be beneficial both for the amount of data and also to give more trustworthy results and thereby limiting how many further experiments needed to be done. The experiment should also include more recordings at different stages after the intervention since some of the changes from the SNI model could occur as late as after 6 to 12 months. On the other hand, this does however question the feasibility of longitudinal studies with animals with implants. An additional adjustment could be to measure from multiple locations of the brain simultaneously. These areas could offer valuable insights into the underlying mechanisms of various chronic pain types, enabling the discovery of distinct characteristics across pain models. Notably, the SNI model stands out due to its long-term effects on both the peripheral and central nervous systems.

If an accurate model for predicting chronic pain in animals was developed, future studies could also focus on translation from animal models to humans, as this has the potential to greatly improve the classification of different types of chronic pain compared to the methods used today. This translation also needs to focus on determining the optimal locations for measuring brain-related physiological signals, as well as whether they would be evoked potentials or if resting-state signals will be sufficient for the classification. Using animal models is an important step when considering the ethical aspect of research. An experiment like the one performed in this project would likely not be considered ethical if conducted on human subjects. The animals used allow for more invasive and in terms life-threatening procedures. These experiments are often done on smaller animals like rodents to determine if the results are significant. By moving to larger animals that have a brain structure similar to humans, the translation of the results becomes easier. The outcomes of the experiments are further strengthened when multiple species are used in the tests, as this can be interpreted as cross-validation across different species. Overall, the use of animal models gives a stronger foundation in translation to human studies.

Conclusion 10

In conclusion, this project aimed to develop a CNN combined with an LSTM to classify between control, LTP, and SNI pain models in pigs using ECoG signals. The results showed that both multiclass and binary classification achieved low accuracy and F1-score on the test data. However, the model struggled to predict the SNI class, indicating the difficulty of distinguishing it from the control class. The model interpretability analysis revealed differences in attribution patterns between validation and test data, potentially explaining the drop in performance on the test data. Moreover, the model mainly focused on single peaks and failed to capture complex feature relationships, limiting its ability to accurately classify the test samples. The similarities in attribution patterns between control and LTP instances further contributed to misclassifications. Further research should increase the number of subjects in each class, especially LTP and SNI, as well as focus on the translation from animal to human studies.

Bibliography

Alshelh et al., 2016. Zeynab Alshelh, Flavia Di Pietro, Andrew M Youssef, Jenna M Reeves, Paul M Macey, E Russell Vickers, Christopher C Peck, Greg M Murray and Luke A Henderson. *Chronic neuropathic pain: it's about the rhythm.* Journal of Neuroscience, 36(3), 1008–1018, 2016.

Ghita Amrani, Amina Adadi, Mohammed Berrada, Zouhayr Souirti and Saïd Boujraf, 2021. Ghita Amrani, Amina Adadi, Mohammed Berrada, Zouhayr Souirti and Saïd Boujraf. EEG signal analysis using deep learning: A systematic literature review. In *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, pages 1–8. IEEE, 2021.

Armstrong, Herr, 2019. Scott A Armstrong and Michael J Herr. *Physiology, nociception.* 2019.

Asad et al., 2016. Abu Bakar Ali Asad, Stephanie Seah, Richard Baumgartner, Dai Feng, Andres Jensen, Elaine Manigbas, Brian Henry, Andrea Houghton, Jeffrey L Evelhoch, Stuart WG Derbyshire et al. *Distinct BOLD fMRI responses of capsaicin-induced thermal sensation reveal pain-related brain activation in nonhuman primates.* PloS one, 11(6), e0156805, 2016.

Nickolaj Ajay Atchuthan, Hjalte Clark, Mikkel Bjerre Danyar, Amalie Koch Andersen, Felipe Rettore Andreis and Suzan Meijs, 2023. Nickolaj Ajay Atchuthan, Hjalte Clark, Mikkel Bjerre Danyar, Amalie Koch Andersen, Felipe Rettore Andreis and Suzan Meijs. Classification of noxious and non-noxious event-related potentials from S1 in pigs using a convolutional neural network. In *2023 11th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1–4. IEEE, 2023.

Baliki et al., 2006. Marwan N Baliki, Dante R Chialvo, Paul Y Geha, Robert M Levy, R Norman Harden, Todd B Parrish and A Vania Apkarian. *Chronic pain and the emotional brain: specific brain activity associated with spontaneous fluctuations of intensity of chronic back pain.* Journal of Neuroscience, 26(47), 12165–12173, 2006.

Baroni et al., 2020. Andrea Baroni, Giacomo Severini, Sofia Straudi, Sergio Buja, Silvia Borsato and Nino Basaglia. *Hyperalgesia and central sensitization in subjects with chronic orofacial pain: Analysis of pain thresholds and EEG biomarkers.* Frontiers in neuroscience, 14, 552650, 2020.

Bell, **2018**. A Bell. *The neurobiology of acute pain.* The Veterinary Journal, 237, 55–62, 2018.

Bennet, **2011**. M Bennet. *Neuropathic Pain Vol. 2.* Neuropathic Pain, 2, 224, 2011.

Bennett, **2010**. Michael Bennett. *Neuropathic pain.* OUP Oxford, 2010.

Breivik et al., **2006**. Harald Breivik, Beverly Collett, Vittorio Ventafridda, Rob Cohen and Derek Gallacher. *Survey of chronic pain in Europe: prevalence, impact on daily life, and treatment.* European journal of pain, 10(4), 287–333, 2006.

Brownlee, **2019**. Jason Brownlee. *How to use learning curves to diagnose machine learning model performance.* Machine Learning Mastery, 2019.

Campbell, Meyer, **2006**. James N Campbell and Richard A Meyer. *Mechanisms of neuropathic pain.* Neuron, 52(1), 77–92, 2006.

Choi et al., **2019**. Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison and George E Dahl. *On empirical comparisons of optimizers for deep learning.* arXiv preprint arXiv:1910.05446, 2019.

Clark et al., **May 2022**. *Hjalte Clark, Amalie Andersen, Nickolaj Atchuthan and Mikkel Danyar.* May 2022. URL `https://projekter.aau.dk/projekter/da/studentthesis/udvikling-af-cnn-model-til-ecog-klassifikation-af-noxious-og-nonnoxious-stimulation-baseret-paa-.html`.

Cohen, **2019**. Michael X Cohen. *A better way to define and describe Morlet wavelets for time-frequency analysis.* NeuroImage, 199, 81–86, 2019.

Cohen, **2014**. Mike X Cohen. *Analyzing Neural Time Series Data: Theory and Practice.* Issues in Clinical and Cognitive Neuropsychology. The MIT Press, 1 edition, 2014. ISBN 0262019876; 9780262019873.

Craik et al., **2019**. Alexander Craik, Yongtian He and Jose L Contreras-Vidal. *Deep learning for electroencephalogram (EEG) classification tasks: a review.* Journal of neural engineering, 16(3), 031001, 2019.

Davis et al., **2017**. Karen D Davis, Herta Flor, Henry T Greely, Gian Domenico Iannetti, Sean Mackey, Markus Ploner, Amanda Pustilnik, Irene Tracey, Rolf-Detlef Treede and Tor D Wager. *Brain imaging tests for chronic pain: medical, legal and ethical issues and recommendations.* Nature Reviews Neurology, 13(10), 624–638, 2017.

de Cheveigné, Nelken, **2019**. Alain de Cheveigné and Israel Nelken. *Filters: when, why, and how (not) to use them.* Neuron, 102(2), 280–293, 2019.

de Natale et al., 2018. Edoardo Rosario de Natale, Heather Wilson, Gennaro Pagano and Marios Politis. *Chapter Seven - Imaging Transplantation in Movement Disorders*, volume 143 of *International Review of Neurobiology*. Academic Press, 2018. doi: https://doi.org/10.1016/bs.irn.2018.10.002. URL https://www.sciencedirect.com/science/article/pii/S0074774218301314.

Decosterd, Woolf, 2000. Isabelle Decosterd and Clifford J Woolf. *Spared nerve injury: an animal model of persistent peripheral neuropathic pain*. Pain, 87(2), 149–158, 2000.

Di Pietro et al., 2018. Flavia Di Pietro, Paul M Macey, Caroline D Rae, Zeynab Alshelh, Vaughan G Macefield, E Russell Vickers and Luke A Henderson. *The relationship between thalamic GABA content and resting cortical rhythm in neuropathic pain*. Human brain mapping, 39(5), 1945–1956, 2018.

Duda et al., 2000. Richard O. Duda, Peter E. Hart and David G. Stork. *Pattern classification*. 2000.

Ehde et al., 2003. Dawn M Ehde, Mark P Jensen, Joyce M Engel, Judith A Turner, Amy J Hoffman and Diana D Cardenas. *Chronic pain secondary to disability: a review*. The Clinical journal of pain, 19(1), 3–17, 2003.

Elsayed et al., 2020. Mahmoud Elsayed, Kok Swee Sim and Shing Chiang Tan. *A novel approach to objectively quantify the subjective perception of pain through electroencephalogram signal analysis*. IEEE Access, 8, 199920–199930, 2020.

Fattahi et al., 2014. Pouria Fattahi, Guang Yang, Gloria Kim and Mohammad Reza Abidian. *A review of organic and inorganic biomaterials for neural interfaces*. Advanced materials, 26(12), 1846–1885, 2014.

Finnerup et al., 2022. Nanna B Finnerup, Lone Nikolajsen and Andrew SC Rice. *Transition from acute to chronic pain: a misleading concept?* Pain, 163(9), e985–e988, 2022.

Finnerup et al., 2020. Nanna Brix Finnerup, Rohini Kuner and Troels Staehelin Jensen. *Neuropathic pain: from mechanisms to treatment*. Physiological reviews, 2020.

Ghosh et al., 2021. Lidia Ghosh, Dipayan Dewan, Abir Chowdhury and Amit Konar. *Exploration of face-perceptual ability by EEG induced deep learning algorithm*. Biomedical Signal Processing and Control, 66, 102368, 2021.

Grandini et al., 2020. Margherita Grandini, Enrico Bagli and Giorgio Visani. *Metrics for multi-class classification: an overview*. arXiv preprint arXiv:2008.05756, 2020.

Gupta, Panghal, 2012. Sonika Gupta and Aman Panghal. *Performance analysis of fir filter design by using rectangular, hanning and hamming windows methods*.

International Journal of Advanced Research in Computer Science and Software Engineering, 2(6), 2012.

Hart et al., **2000**. Peter E Hart, David G Stork and Richard O Duda. *Pattern classification*. Wiley Hoboken, 2000.

Hubbard et al., **2015**. Catherine S Hubbard, Shariq A Khan, Su Xu, Myeounghoon Cha, Radi Masri and David A Seminowicz. *Behavioral, metabolic and functional brain changes in a rat model of chronic neuropathic pain: a longitudinal MRI study.* Neuroimage, 107, 333–344, 2015.

IASP, **Apr 2022**. IASP. *Terminology: International association for the study of pain*, 2022. URL https://www.iasp-pain.org/resources/terminology/?navItemNumber=576.

IASP, **Jan 2023a**. IASP. *Acute pain*, 2023. URL https://www.iasp-pain.org/resources/topics/acute-pain/.

IASP, **Feb 2023b**. IASP. *Terminology: International association for the study of pain*, 2023. URL https://www.iasp-pain.org/resources/terminology/.

Janjua et al., **2021**. Taha Al Muhammadee Janjua, Thomas Gomes Nørgaard dos Santos Nielsen, Felipe Rettore Andreis, Suzan Meijs and Winnie Jensen. *The effect of peripheral high-frequency electrical stimulation on the primary somatosensory cortex in pigs.* IBRO neuroscience reports, 11, 112–118, 2021.

Kandel et al., **2021**. Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, Sarah Mack et al. *Principles of Neural Science*. McGraw-Hill, 6 edition, 2021. ISBN 9781259642234,1259642232.

Keller et al., **04 2016**. Corey J. Keller, Christopher Chen, Fred A. Lado and Kamran Khodakhah. *The Limited Utility of Multiunit Data in Differentiating Neuronal Population Activity.* PLOS ONE, 11(4), 1–20, 2016. doi: 10.1371/journal.pone.0153154. URL https://doi.org/10.1371/journal.pone.0153154.

King et al., **2013**. W King et al. *Acute pain, subacute pain, and chronic pain.* Encyclopedia of pain, 10, 978–3, 2013.

Kingma, Ba, **2014**. Diederik P Kingma and Jimmy Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

Kokhlikyan et al., **2020**. Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina,

Carlos Araya, Siqi Yan and Orion Reblitz-Richardson. *Captum: A unified and generic model interpretability library for PyTorch*, 2020.

Komaki et al., **2016**. Yuji Komaki, Keigo Hikishima, Shinsuke Shibata, Tsunehiko Konomi, Fumiko Seki, Masayuki Yamada, Naoyuki Miyasaka, Kanehiro Fujiyoshi, Hirotaka J Okano, Masaya Nakamura et al. *Functional brain mapping using specific sensory-circuit stimulation and a theoretical graph network analysis in mice with neuropathic allodynia.* Scientific reports, 6(1), 1–11, 2016.

Latremoliere, Woolf, **2009**. Alban Latremoliere and Clifford J Woolf. *Central sensitization: a generator of pain hypersensitivity by central neural plasticity.* The journal of pain, 10(9), 895–926, 2009.

LeBlanc et al., **2014**. Brian W LeBlanc, Theresa R Lii, Andrew E Silverman, Robert T Alleyne and Carl Y Saab. *Cortical theta is increased while thalamocortical coherence is decreased in rat models of acute and chronic pain.* PAIN®, 155(4), 773–782, 2014.

LeCun et al., **1998**. Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86 (11), 2278–2324, 1998.

Lenoir et al., **2020**. Dorine Lenoir, Ward Willaert, Iris Coppieters, Anneleen Malfliet, Kelly Ickmans, Jo Nijs, Kristl Vonck, Mira Meeus and Barbara Cagnie. *Electroencephalography during nociceptive stimulation in chronic pain patients: a systematic review.* Pain Medicine, 21(12), 3413–3427, 2020.

Liberati et al., **2016**. Giulia Liberati, Anne Klöcker, Marta M Safronova, Susana Ferrao Santos, Jose-Geraldo Ribeiro Vaz, Christian Raftopoulos and André Mouraux. *Nociceptive local field potentials recorded from the human insula are not specific for nociception.* PLoS biology, 14(1), e1002345, 2016.

Livezey, Glaser, **2021**. Jesse A Livezey and Joshua I Glaser. *Deep learning approaches for neural decoding across architectures and recording modalities.* Briefings in bioinformatics, 22(2), 1577–1591, 2021.

Logothetis, **2008**. Nikos K Logothetis. *What we can do and what we cannot do with fMRI.* Nature, 453(7197), 869–878, 2008.

Lötsch, Ultsch, **2018**. Jörn Lötsch and Alfred Ultsch. *Machine learning in pain research.* Pain, 159(4), 623, 2018.

Luck, **2014**. Steven J. Luck. *An Introduction to the Event-Related Potential Technique.* A Bradford Book, second edition edition, 2014. ISBN 0262525852,9780262525855. URL `https://mitpress.mit.edu/9780262324069/an-introduction-to-the-event-related-potential-technique/`.

Mandrekar, **2010a**. Jayawant N Mandrekar. *Receiver operating characteristic curve in diagnostic test assessment.* Journal of Thoracic Oncology, 5(9), 1315–1316, 2010.

Mandrekar, **2010b**. Jayawant N Mandrekar. *Simple statistical measures for diagnostic accuracy assessment.* Journal of Thoracic Oncology, 5(6), 763–764, 2010.

Mane et al., **2015**. Akshaya R Mane, SD Biradar and RK Shastri. *Review paper on feature extraction methods for EEG signal analysis.* Int. J. Emerg. Trend Eng. Basic Sci, 2(1), 545–552, 2015.

Mano et al., **2017**. Hiroaki Mano, Gopal Kotecha, Kenji Leibnitz, Takashi Matsubara, Aya Nakae, Nicholas Shenker, Masahiko Shibata, Valerie Voon, Wako Yoshida, Michael Lee et al. *Classification and characterisation of brain network changes in chronic back pain: A multicenter study.* bioRxiv, page 223446, 2017.

Ioannis Markoulidakis, George Kopsiaftis, Ioannis Rallis and Ioannis Georgoulas, *2021.* Ioannis Markoulidakis, George Kopsiaftis, Ioannis Rallis and Ioannis Georgoulas. Multi-Class Confusion Matrix Reduction method and its application on Net Promoter Score classification problem. In *The 14th pervasive technologies related to assistive environments conference*, pages 412–419, 2021.

Martucci, Mackey, **2018**. Katherine T Martucci and Sean C Mackey. *Neuroimaging of pain: human evidence and clinical relevance of central nervous system processes and modulation.* Anesthesiology, 128(6), 1241–1254, 2018.

Mee et al., **2006**. Steven Mee, Blynn G Bunney, Christopher Reist, Steve G Potkin and William E Bunney. *Psychological pain: a review of evidence.* Journal of Psychiatric Research, 40(8), 680–690, 2006.

Meijs et al., **2022**. Suzan Meijs, Andrew Hayward, Carsten Bjarkam and Thomas Gomes Nørgaard dos Santos Nielsen. *Spared ulnar nerve injury results in increased layer III-VI excitability in the pig primary somatosensory cortex.* 2022.

MNE, **2023a**. MNE. *Python homepage*, 2023. URL `https://mne.tools/stable/index.html`.

MNE, **2023b**. MNE. $Mne.time_frequency.tfr_morlet, 2023. URL.$

Mouraux, Iannetti, **2009**. André Mouraux and Gian Domenico Iannetti. *Nociceptive laser-evoked brain potentials do not reflect nociceptive-specific neural activity.* Journal of neurophysiology, 101(6), 3258–3269, 2009.

Mouraux, Iannetti, **2018**. André Mouraux and Gian Domenico Iannetti. *The search for pain biomarkers in the human brain.* Brain, 141(12), 3290–3307, 2018.

Murakami, Okada, **2006**. Shingo Murakami and Yoshio Okada. *Contributions of principal neocortical neurons to magnetoencephalography and electroencephalography signals.* The Journal of physiology, 575(3), 925–936, 2006. ISSN 0022-3751.

Mussigmann et al., **2022**. Thibaut Mussigmann, Benjamin Bardel and Jean-Pascal Lefaucheur. *Resting-state electroencephalography (EEG) biomarkers of chronic neuropathic pain. A systematic review.* NeuroImage, page 119351, 2022.

Navarro et al., **2007**. X Navarro, Meritxell Vivó and Antoni Valero-Cabré. *Neural plasticity after peripheral nerve injury and regeneration.* Progress in neurobiology, 82(4), 163–201, 2007.

Nicholas et al., **2019**. Michael Nicholas, Johan WS Vlaeyen, Winfried Rief, Antonia Barke, Qasim Aziz, Rafael Benoliel, Milton Cohen, Stefan Evers, Maria Adele Giamberardino, Andreas Goebel et al. *The IASP classification of chronic pain for ICD-11: chronic primary pain.* Pain, 160(1), 28–37, 2019.

Nielsen, **2018**. Michael A. Nielsen. *Neural Networks and Deep Learning*, 2018. URL `http://neuralnetworksanddeeplearning.com/`.

Nuñez-Ibero et al., **2021**. Maider Nuñez-Ibero, Borja Camino-Pontes, Ibai Diez, Asier Erramuzpe, Endika Martinez-Gutierrez, Sebastiano Stramaglia, Javier O Alvarez-Cienfuegos and Jesus M Cortes. *A Controlled Thermoalgesic Stimulation Device for Exploring Novel Pain Perception Biomarkers.* IEEE Journal of Biomedical and Health Informatics, 25(8), 2948–2957, 2021.

Orr, Müller, **1998**. Genevieve B Orr and Klaus-Robert Müller. *Neural networks: tricks of the trade.* Springer, 1998.

Paszke et al., **2019**. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al. *Pytorch: An imperative style, high-performance deep learning library.* Advances in neural information processing systems, 32, 2019. URL `https://pytorch.org/docs/stable/index.html`.

Dharmendra Pathak and Ramgopal Kashyap, *2021.* Dharmendra Pathak and Ramgopal Kashyap. A review of the classification of neuroscience problems with the help of deep learning framework. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–6. IEEE, 2021.

Picciotto, **2020**. M Picciotto. *Consideration of sample size in neuroscience studies.* J. Neurosci, 40, 4076–4077, 2020.

Price et al., **1977**. Donald D Price, James W Hu, Ronald Dubner and Richard H Gracely. *Peripheral suppression of first pain and central summation of second pain evoked by noxious heat pulses.* Pain, 3(1), 57–68, 1977.

Pricope et al., **2022**. Cosmin Vasilica Pricope, Bogdan Ionel Tamba, Gabriela Dumitrita Stanciu, Magdalena Cuciureanu, Anca Narcisa Neagu, Ioana Creanga-Murariu, Bogdan-Ionut Dobrovat, Cristina Mariana Uritu, Silviu Iulian Filipiuc, Bianca-Mariana Pricope et al. *The Roles of Imaging Biomarkers in the Management of Chronic Neuropathic Pain.* International Journal of Molecular Sciences, 23(21), 13038, 2022.

Roodschild et al., **2020**. Matías Roodschild, Jorge Gotay Sardiñas and Adrián Will. *A new approach for the vanishing gradient problem on sigmoid activation.* Progress in Artificial Intelligence, 9(4), 351–360, 2020.

Ruder, **2016**. Sebastian Ruder. *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.

Saeidi et al., **2021**. Maham Saeidi, Waldemar Karwowski, Farzad V Farahani, Krzysztof Fiok, Redha Taiar, PA Hancock and Awad Al-Juaid. *Neural decoding of EEG signals with machine learning: a systematic review.* Brain Sciences, 11(11), 1525, 2021.

Sak et al., **2014**. Haşim Sak, Andrew Senior and Françoise Beaufays. *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition.* arXiv preprint arXiv:1402.1128, 2014.

Sandkühler, **2007**. Jürgen Sandkühler. *Understanding LTP in pain pathways.* Molecular pain, 3, 1744–8069, 2007.

Sandkuhler, **2009**. Jurgen Sandkuhler. *Models and mechanisms of hyperalgesia and allodynia.* Physiological reviews, 89(2), 707–758, 2009.

Santana et al., **2019**. Alex Novaes Santana, Ignacio Cifre, Charles Novaes De Santana and Pedro Montoya. *Using deep learning and resting-state fMRI to classify chronic pain conditions.* Frontiers in neuroscience, 13, 1313, 2019.

Savignac et al., **2022**. Chloé Savignac, Don Daniel Ocay, Yacine Mahdid, Stefanie Blain-Moraes and Catherine E Ferland. *Clinical use of electroencephalography in the assessment of acute thermal pain: a narrative review based on articles from 2009 to 2019.* Clinical EEG and Neuroscience, 53(2), 124–132, 2022.

Schnakers, Zasler, **2007**. Caroline Schnakers and Nathan D Zasler. *Pain assessment and management in disorders of consciousness.* Current opinion in neurology, 20(6),

620–626, 2007.

Seixas et al., **2013**. Daniela Seixas, Guy Ebinger, Janet Mifsud, Joseph Schmucker von Koch and Sheri Alpert. *Functional Magnetic Resonance Imaging*, European Commission, 2013.

Benjamin Shickel, Scott Siegel, Martin Heesacker, Sherry Benton and Parisa Rashidi, 2020. Benjamin Shickel, Scott Siegel, Martin Heesacker, Sherry Benton and Parisa Rashidi. Automatic detection and classification of cognitive distortions in mental health text. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 275–280. IEEE, 2020.

Shrikumar et al., **2017**. Avanti Shrikumar, Peyton Greenside and Anshul Kundaje. *Learning important features through propagating activation differences.* pages 3145–3153, 2017.

Simonyan et al., **2013**. Karen Simonyan, Andrea Vedaldi and Andrew Zisserman. *Deep inside convolutional networks: Visualising image classification models and saliency maps.* arXiv preprint arXiv:1312.6034, 2013.

Song et al., **2021**. Yingchao Song, Qian Su, Qingqing Yang, Rui Zhao, Guotao Yin, Wen Qin, Gian Domenico Iannetti, Chunshui Yu and Meng Liang. *Feedforward and feedback pathways of nociceptive and tactile processing in human somatosensory system: A study of dynamic causal modeling of fMRI data.* NeuroImage, 234, 117957, 2021.

Srivastava et al., **2014**. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research, 15(1), 1929–1958, 2014.

Sundararajan et al., **2017**. Mukund Sundararajan, Ankur Taly and Qiqi Yan. *Axiomatic Attribution for Deep Networks.* 2017.

Sundhedsstyrelsen, **2019**. Sundhedsstyrelsen. *Smerteguide.* `https://www.sst.dk/-/media/Udgivelser/2019/Smerteguide.ashx`, 2019. Accessed: 2023-05-29.

Torrence, Compo, **1998**. Christopher Torrence and Gilbert P Compo. *A practical guide to wavelet analysis.* Bulletin of the American Meteorological society, 79(1), 61–78, 1998.

Cory Toth. The Clinical Presentation of Neuropathic Pain, page 1–32. Cambridge University Press, 2013. 10.1017/CBO9781139152211.002.

Tøttrup, **2020**. Lea Tøttrup. *Functional Cortical Changes in an Animal Model of Neuropathic Pain*, 2020. ISSN 22461302.

Vallabhaneni et al., **2021**. Ramesh Babu Vallabhaneni, Pankaj Sharma, Vinit Kumar, Vyom Kulshreshtha, Koya Jeevan Reddy, S Selva Kumar, V Sandeep Kumar and Surendra Kumar Bitra. *Deep Learning Algorithms in EEG Signal Decoding Application: A Review*. IEEE Access, 9, 125778–125786, 2021.

van den Broeke et al., **2010**. Emanuel N van den Broeke, Clementina M van Rijn, José A Biurrun Manresa, Ole K Andersen, Lars Arendt-Nielsen and Oliver HG Wilder-Smith. *Neurophysiological correlates of nociceptive heterosynaptic long-term potentiation in humans*. Journal of neurophysiology, 103(4), 2107–2113, 2010.

van den Broeke et al., **2013**. Emanuel N van den Broeke, Lonneke Koeslag, Laura J Arendsen, Simon W Nienhuijs, Camiel Rosman, Clementina M van Rijn, Oliver HG Wilder-Smith and Harry van Goor. *Altered cortical responsiveness to pain stimuli after high frequency electrical stimulation of the skin in patients with persistent pain after inguinal hernia repair*. Plos one, 8(12), e82701, 2013.

Van Houdt et al., **2020**. Greg Van Houdt, Carlos Mosquera and Gonzalo Nápoles. *A review on the long short-term memory model*. Artificial Intelligence Review, 53, 5929–5955, 2020.

Wagemakers et al., **2019**. Sjors H Wagemakers, Joanne M van der Velden, A Sophie Gerlich, Alinde W Hindriks-Keegstra, Jacqueline FM van Dijk and Joost JC Verhoeff. *A systematic review of devices and techniques that objectively measure patients' pain*. Pain Physician, 22(1), 1–13, 2019.

Wager et al., **2013**. Tor D Wager, Lauren Y Atlas, Martin A Lindquist, Mathieu Roy, Choong-Wan Woo and Ethan Kross. *An fMRI-based neurologic signature of physical pain*. New England Journal of Medicine, 368(15), 1388–1397, 2013.

Witting et al., **2006**. Nanna Witting, Ron C Kupers, Peter Svensson and Troels S Jensen. *A PET activation study of brush-evoked allodynia in patientswith nerve injury pain*. Pain, 120(1-2), 145–154, 2006.

Woller et al., **2017**. Sarah A Woller, Kelly A Eddinger, Maripat Corr and Tony L Yaksh. *An overview of pathways encoding nociception*. Clinical and experimental rheumatology, 35(Suppl 107), 40, 2017.

Woolf et al., **2010**. Clifford J Woolf et al. *What is this thing called pain?* The Journal of clinical investigation, 120(11), 3742–3744, 2010.

Woyczynski, **2011**. Wojbor A. Woyczynski. *A first course in statistics for signal analysis*. Springer, 2011.

Wu et al., **2022**. Fengjie Wu, Weijian Mai, Yisheng Tang, Qingkun Liu, Jiangtao Chen and Ziqian Guo. *Learning spatial-spectral-temporal EEG representations with deep attentive-recurrent-convolutional neural networks for pain intensity assessment*. Neuroscience, 481, 144–155, 2022.

Zeiler, Fergus, **2013**. Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013.

Zhang et al., **2018**. Pengbo Zhang, Xue Wang, Weihang Zhang and Junfeng Chen. *Learning spatial-spectral-temporal EEG features with recurrent 3D convolutional neural networks for cross-task mental workload assessment*. IEEE Transactions on neural systems and rehabilitation engineering, 27(1), 31–42, 2018.

Zhao, Wan, **2018**. Zi-Fang Zhao and You Wan. *Electrophysiological Signature of Pain*. Advances in Pain Research: Mechanisms and Modulation of Chronic Pain, pages 167–177, 2018.

Zhuo, **2011**. Min Zhuo. *Cortical plasticity as a new endpoint measurement for chronic pain*. Molecular pain, 7, 1744–8069, 2011.

Zolezzi et al., **2022**. Daniela M Zolezzi, Luz María Alonso-Valerdi and David I Ibarra-Zarate. *Chronic neuropathic pain is more than a perception: Systems and methods for an integral characterization*. Neuroscience & Biobehavioral Reviews, page 104599, 2022.

Pain Models A

A.1 Spared nerve injury (SNI)

The spared nerve injury (SNI) model is an animal model extensively used in research to simulate the physiological mechanisms in peripheral NP chronic pain. The SNI model involves partial ligation of two of the three terminal branches of the sciatic nerve (the common peroneal and tibial nerves), leaving the third branch (the sural nerve) intact. This model was created because of the lack of understanding of patients with partial nerve injuries, the most common group of NP patients. Furthermore, this model was created to be more reproducible, since other models involve quite precise surgery using ligation to constrict the nerves. [Decosterd and Woolf, 2000]

The SNI procedure is performed under anesthesia and involves making an incision through the biceps femoris muscle to access the sciatic nerve and its three terminal branches; sural, common peroneal, and tibial nerves (see Figure A.1). The tibial and common peroneal nerves are then tied tightly with 5.0 silk and sectioned distal to the ligation, removing 2-4 mm of the distal nerve stump. Great care is taken to avoid any contact or stretching of the intact sural nerve. The muscle and skin are then closed in two layers. This activates mechanical hypersensitivity in the lateral dorsal paw during the second week after surgery, with hypersensitivity persisting for up to 6 months. Therefore longer follow-up studies are advantageous when making use of the SNI model [Decosterd and Woolf, 2000].

Since the SNI model produces minimal variability in the degree of damage, the reproducibility and comparability between studies is effective. Furthermore, this model enables direct investigation of changes in both injured primary sensory neurons, neighboring intact sensory neurons, and neural activity in the brain.

The nerve injury model of this project was different since the radial nerve was cut instead.

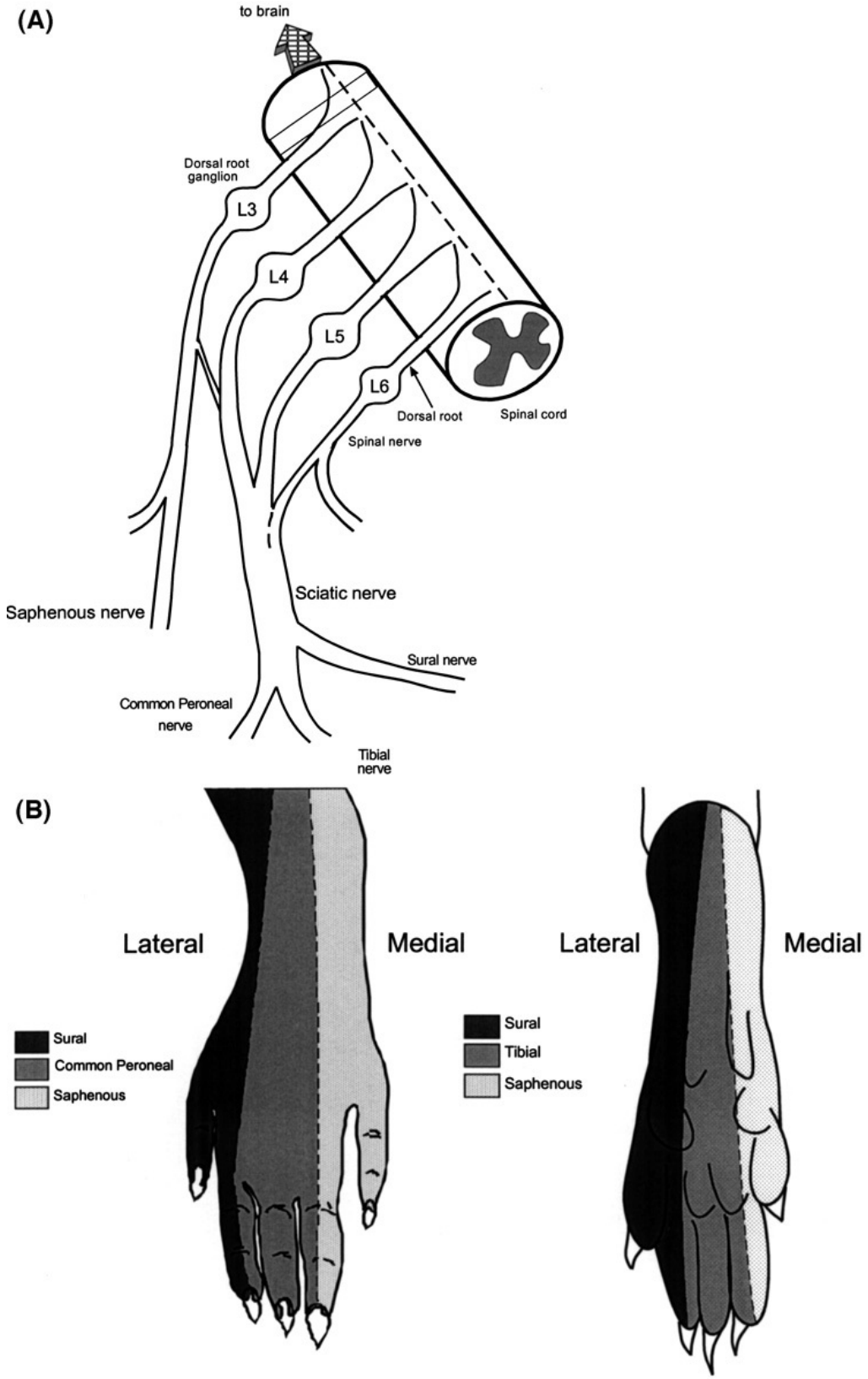


Figure A.1. (A) Picture of sciatic and saphenous nerves with their origins and little overlap between them. (B) Map of zones on a rat paw that the nerves control, with some overlap at the edges of these zones.
Source: [Decosterd and Woolf, 2000]

A.2 Electrically evoked acute pain

Long-term potentiation (LTP) is a well-documented phenomenon and is used in pain research, learning, and memory. It is generally defined as a long-lasting increase in synaptic strength, and has at least two different stages which is distinguishable based on their duration and the signal transduction pathways involved. The early phase of LTP lasts up to three hours and is independent of de-novo protein synthesis. Late-phase LTP depends on de-novo protein synthesis, may involve structural changes at synapses, and last longer than three hours, up to an animal's life span. The increase in synaptic strength refers to the level of the post-synaptic reaction such as the neurotransmitters in enhanced and/or the effect of the neurotransmitters becoming stronger. It is important to note that this does not include action potential firing.[Sandkühler, 2007] LTP is also present in the nociceptive system, where it is believed to be a key mechanism in the development and maintenance of chronic pain. [van den Broeke et al., 2010] Long-term potentiation and depression (LTD) can induce changes in the cortical map following nerve injury. This affects the size of the areas by strengthening or weakening connections between cortical areas. [Navarro et al., 2007] As a pain model, LTP can be induced by high-frequency electrical stimulation (HFS) on primary afferents, where the stimulation strength (around 100 Hz) is needed to activate c-fibers. [Sandkuhler, 2009; Janjua et al., 2021; van den Broeke et al., 2013]

A.3 Low-frequency stimulation (LFS)

To investigate the shocked system, for example, using SNI or HFS models, researchers often apply low-frequency stimulation (LFS). This method ensures the evocation of an event-related potential (ERP) that can be synchronized with a trigger. By doing this, researchers can create epochs linked to specific triggers. The advantage of this technique is that the resulting ERPs can be compared across different pain models, including SNI, HFS, and control conditions. It is theorized that the LFS-induced ERPs will be modulated differently depending on the specific pain model being used. The technique of utilizing electrical stimulation, such as LFS, aims to replicate the sensation of a person experiencing stimulation in areas affected by allodynia or hyperalgesia. It should be noted that similar outcomes could also be achieved by utilizing mechanical or thermal stimuli.[Janjua et al., 2021]

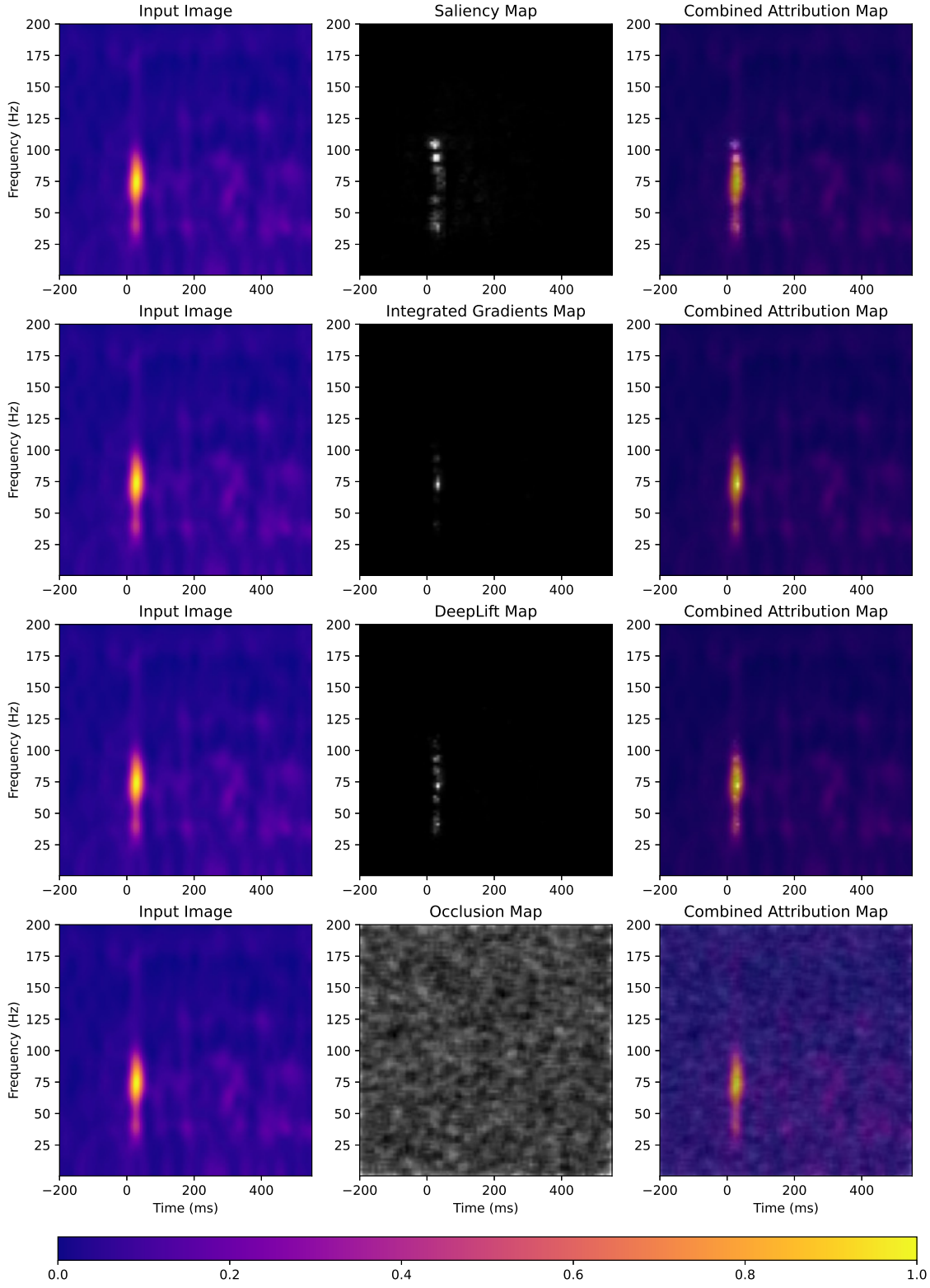
Model Interperability B

This appendix chapter contains the plots from the explainable AI from the multiclass classification model as well as the binary model. It will contain information about the models performance on validation and test data for the control, LTP, and SNI groups.

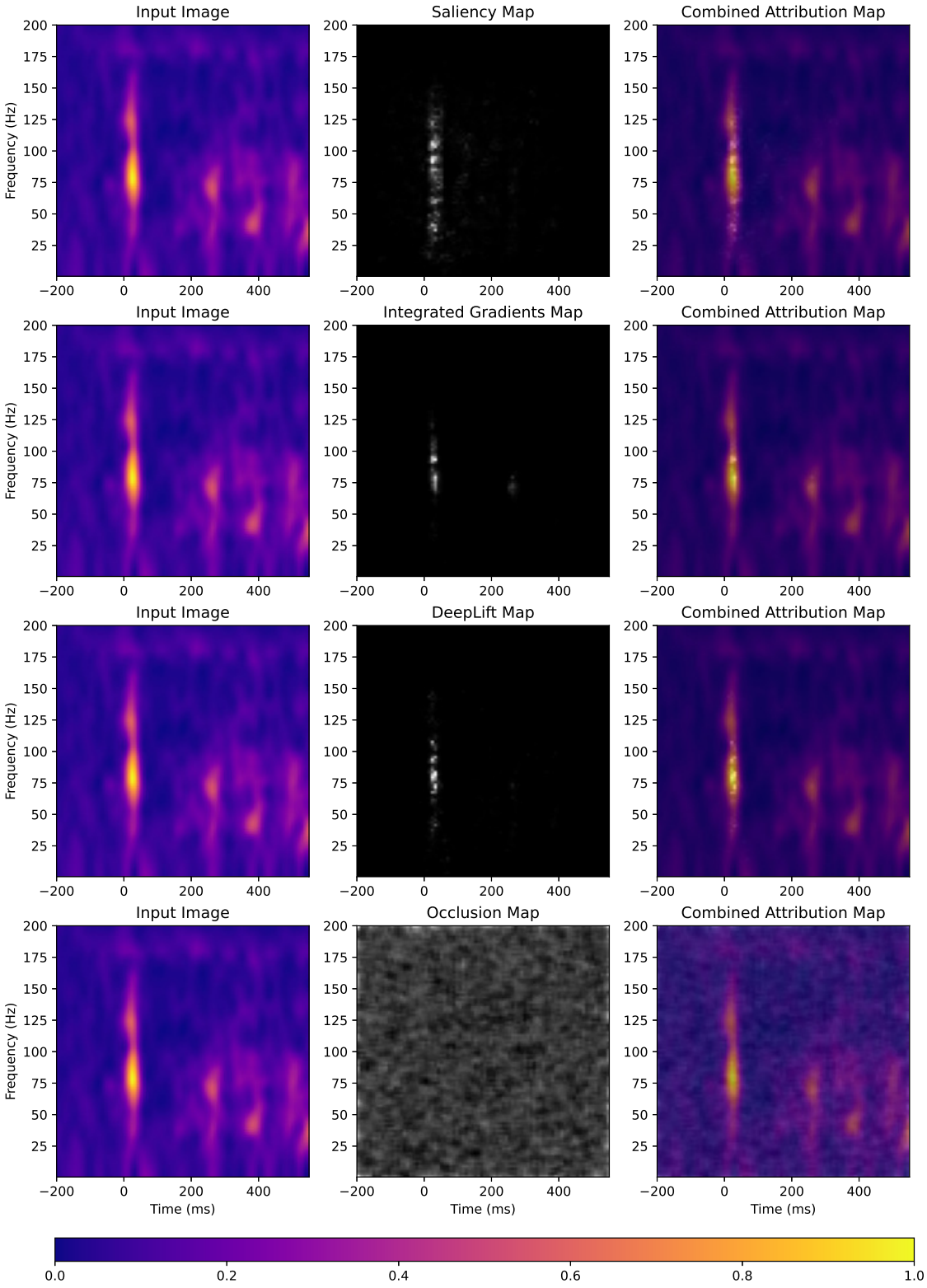
B.1 Multiclass model

B.1.1 Validation

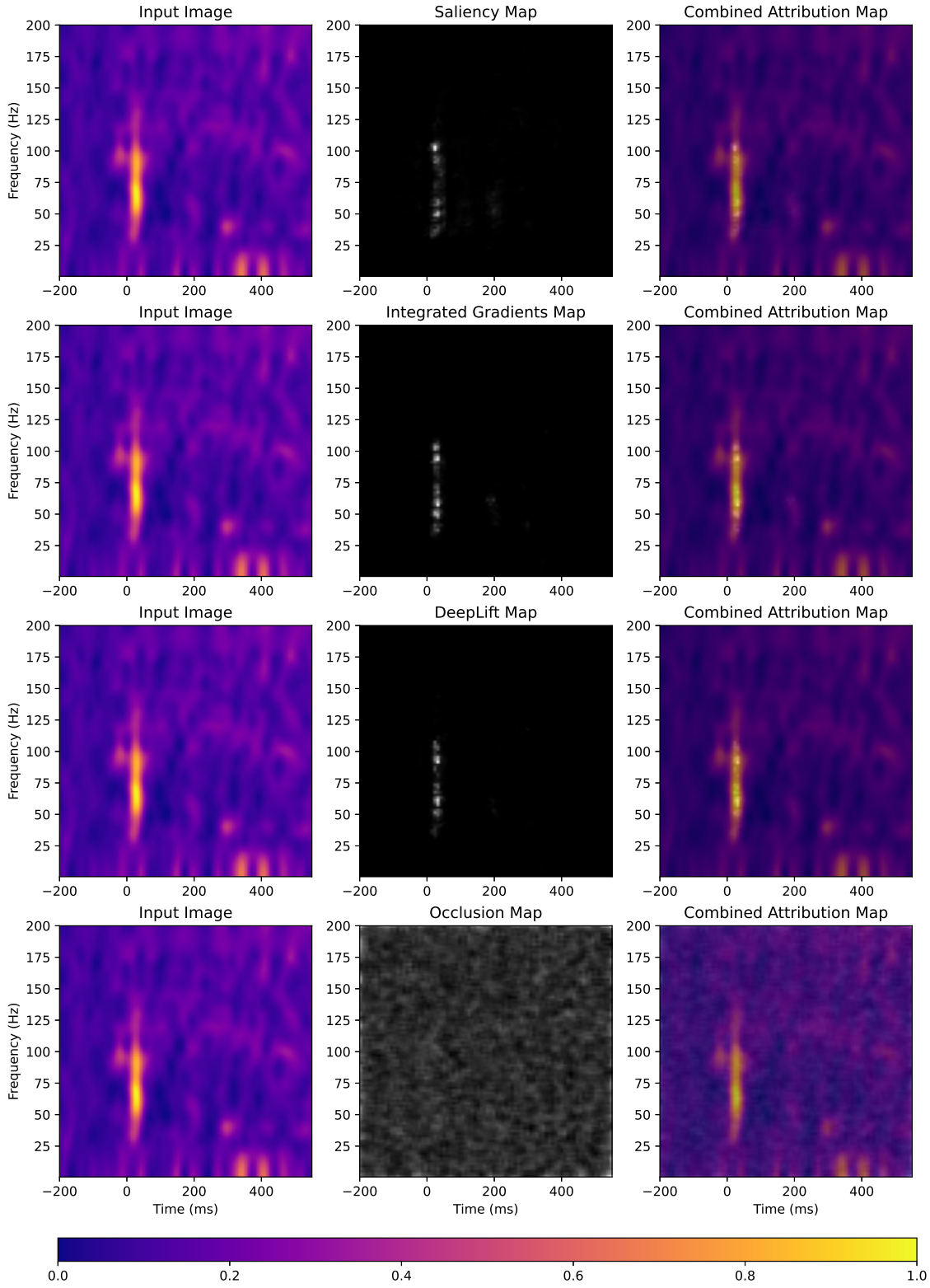
Control



LTP

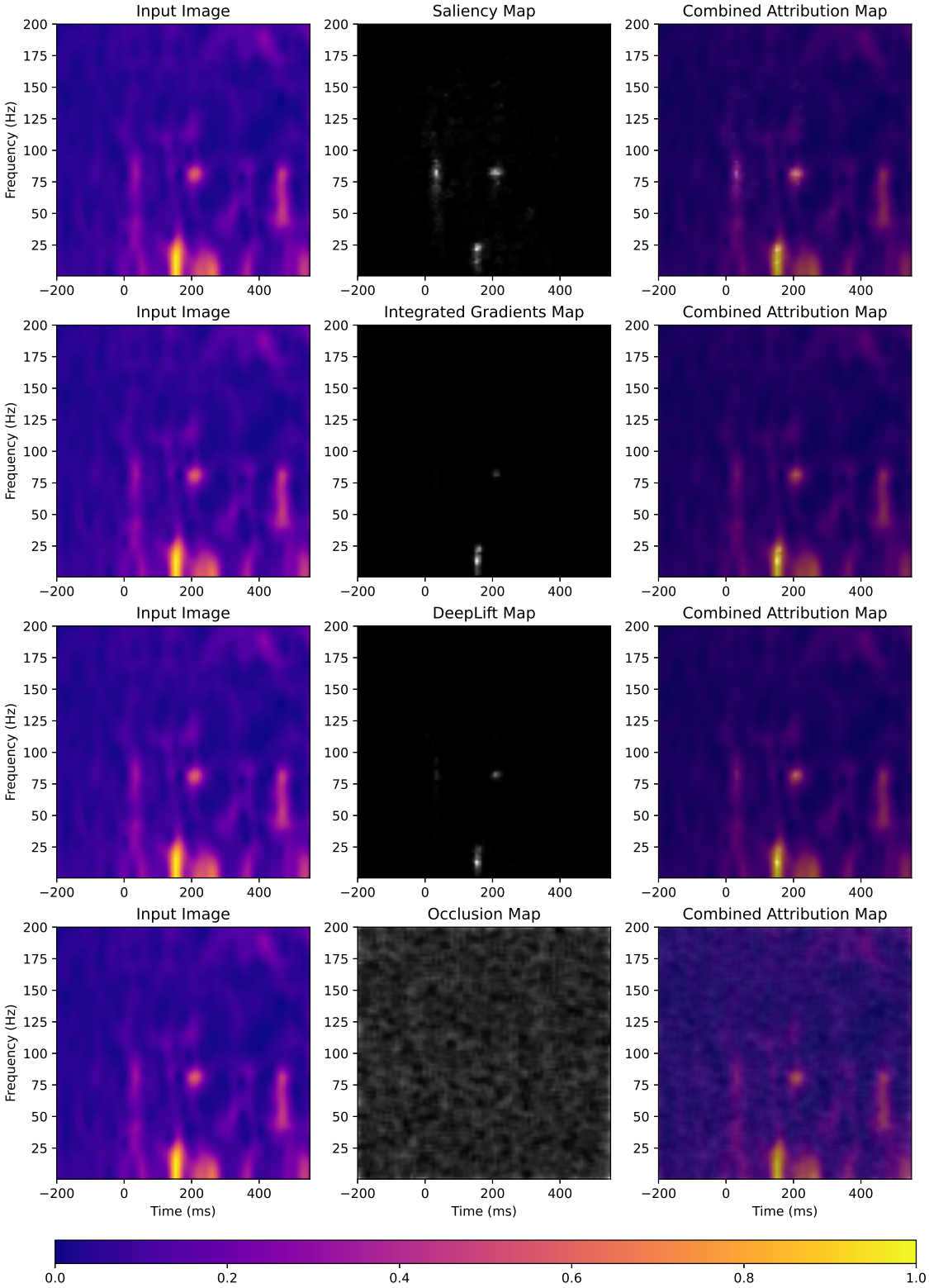

Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

SNI

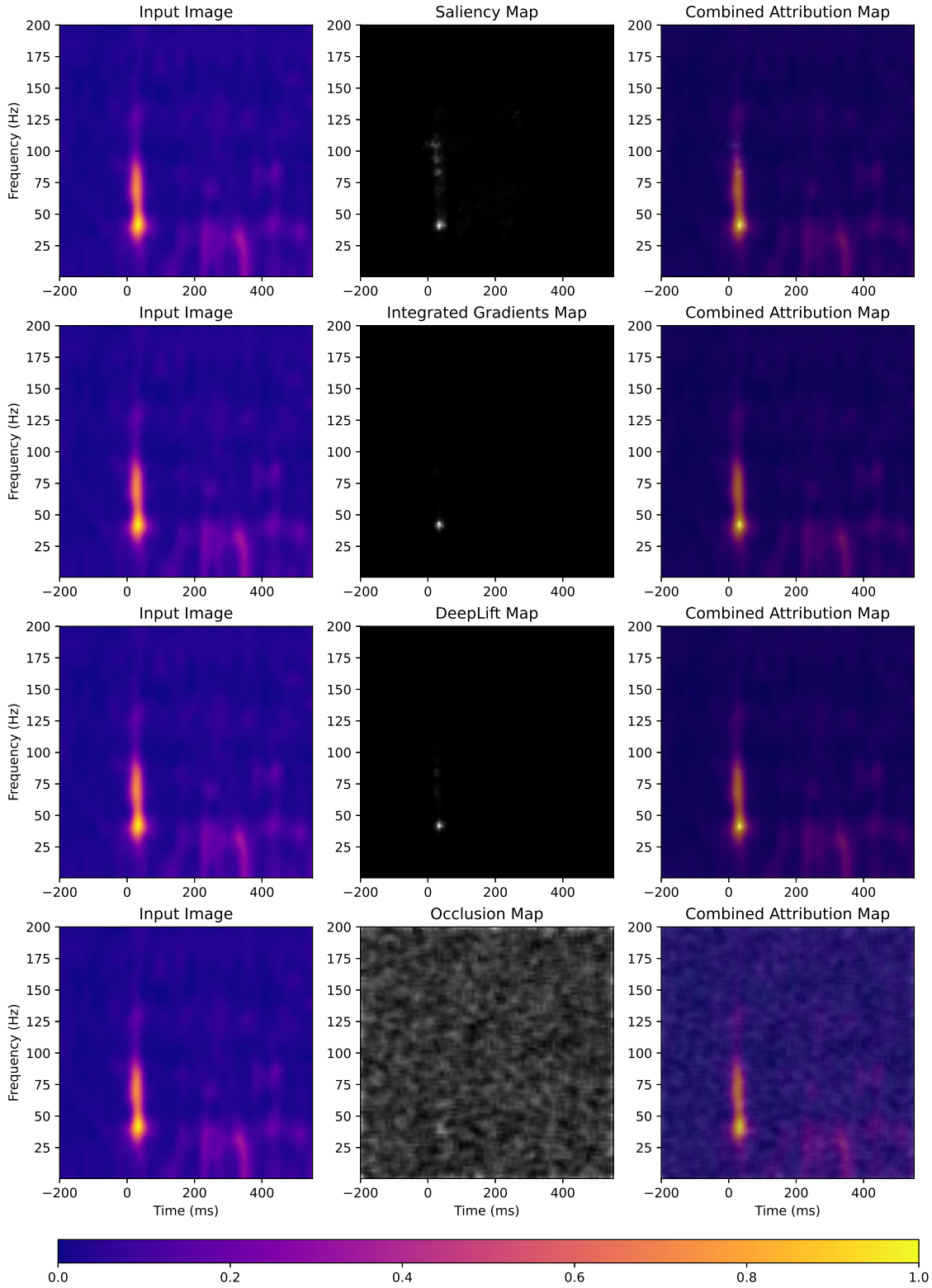

Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

B.1.2 Test

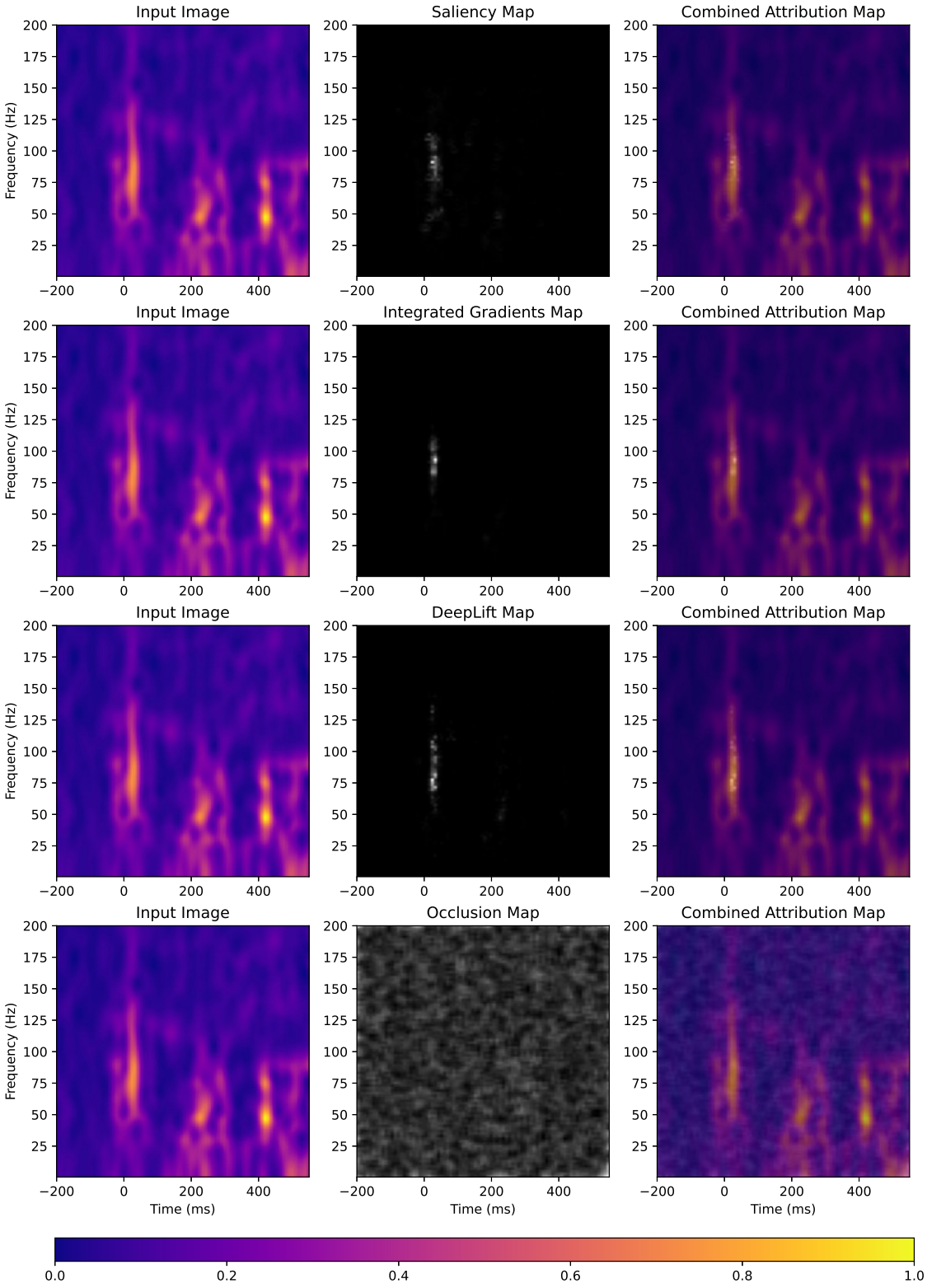
Control



LTP


Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

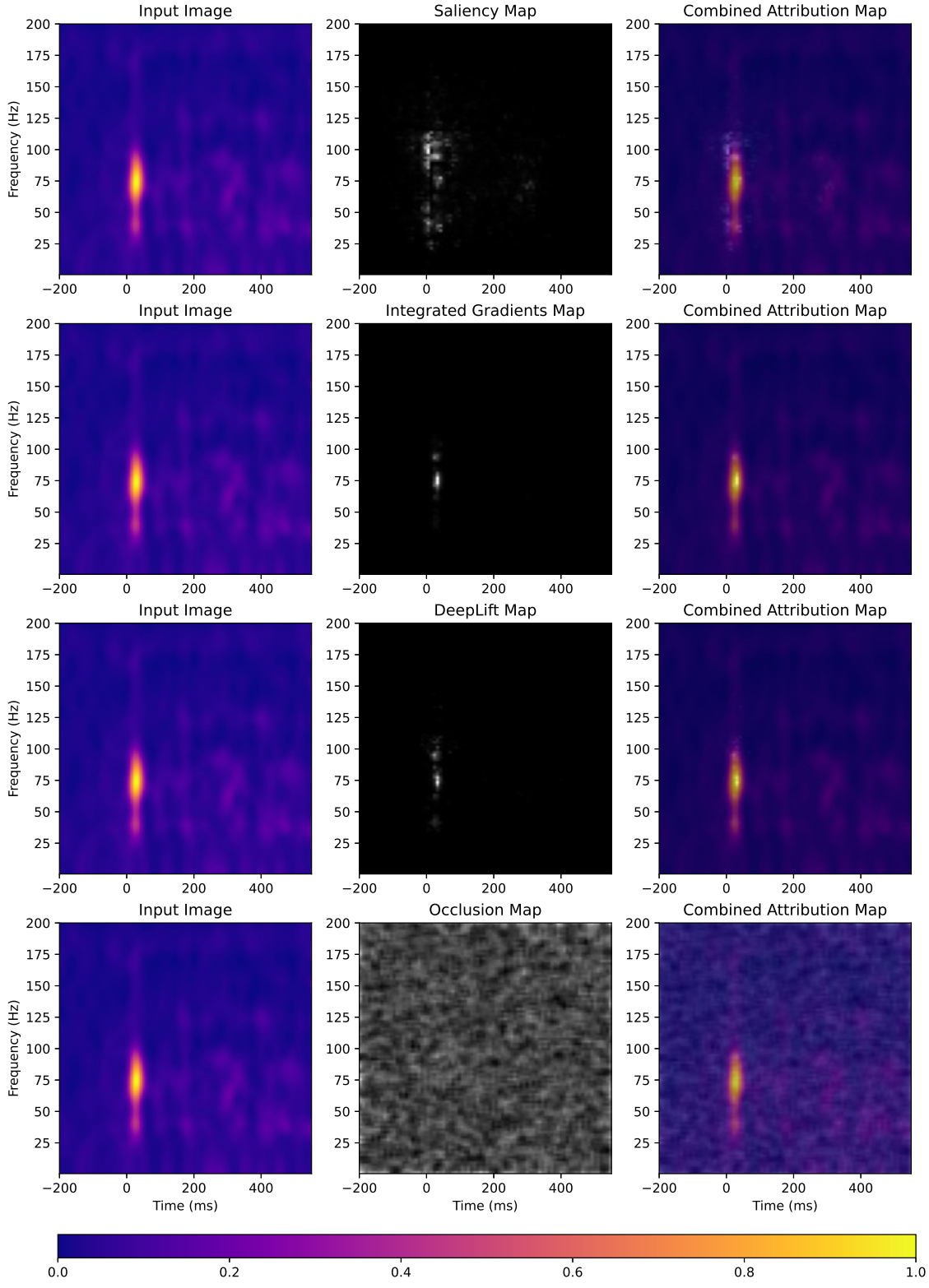
SNI


Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

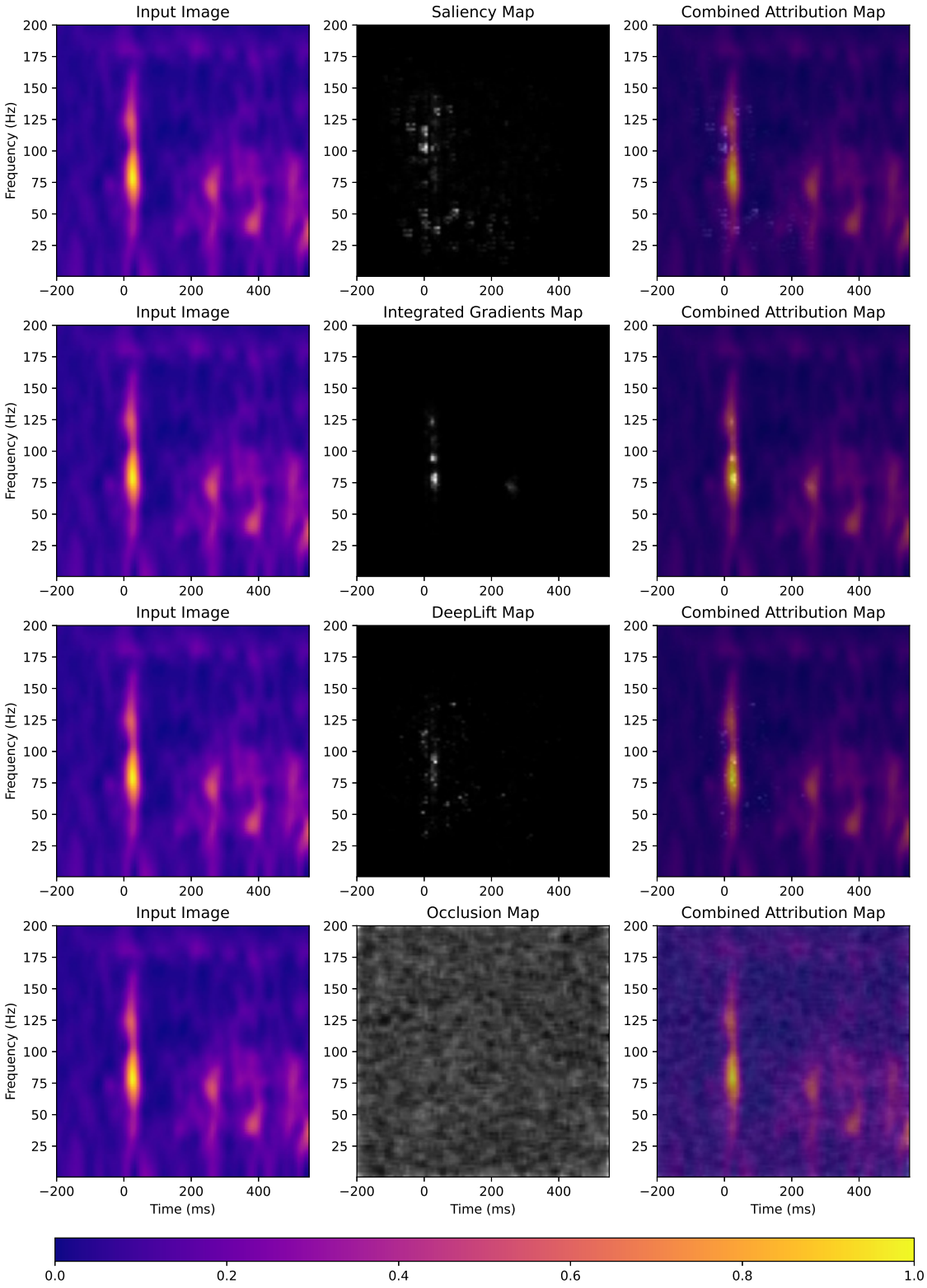
B.2 Binary model

B.2.1 Validation

Control

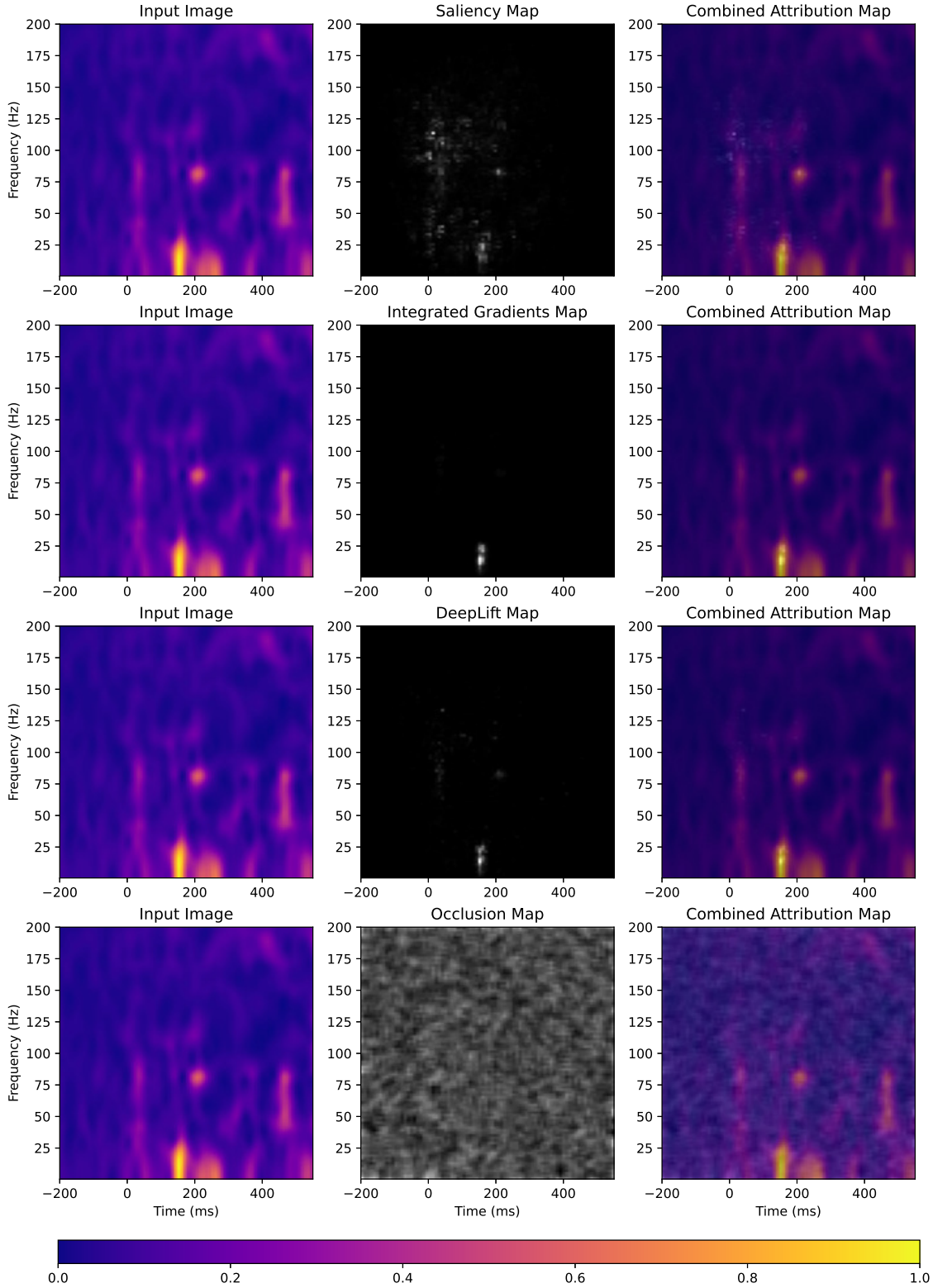


LTP

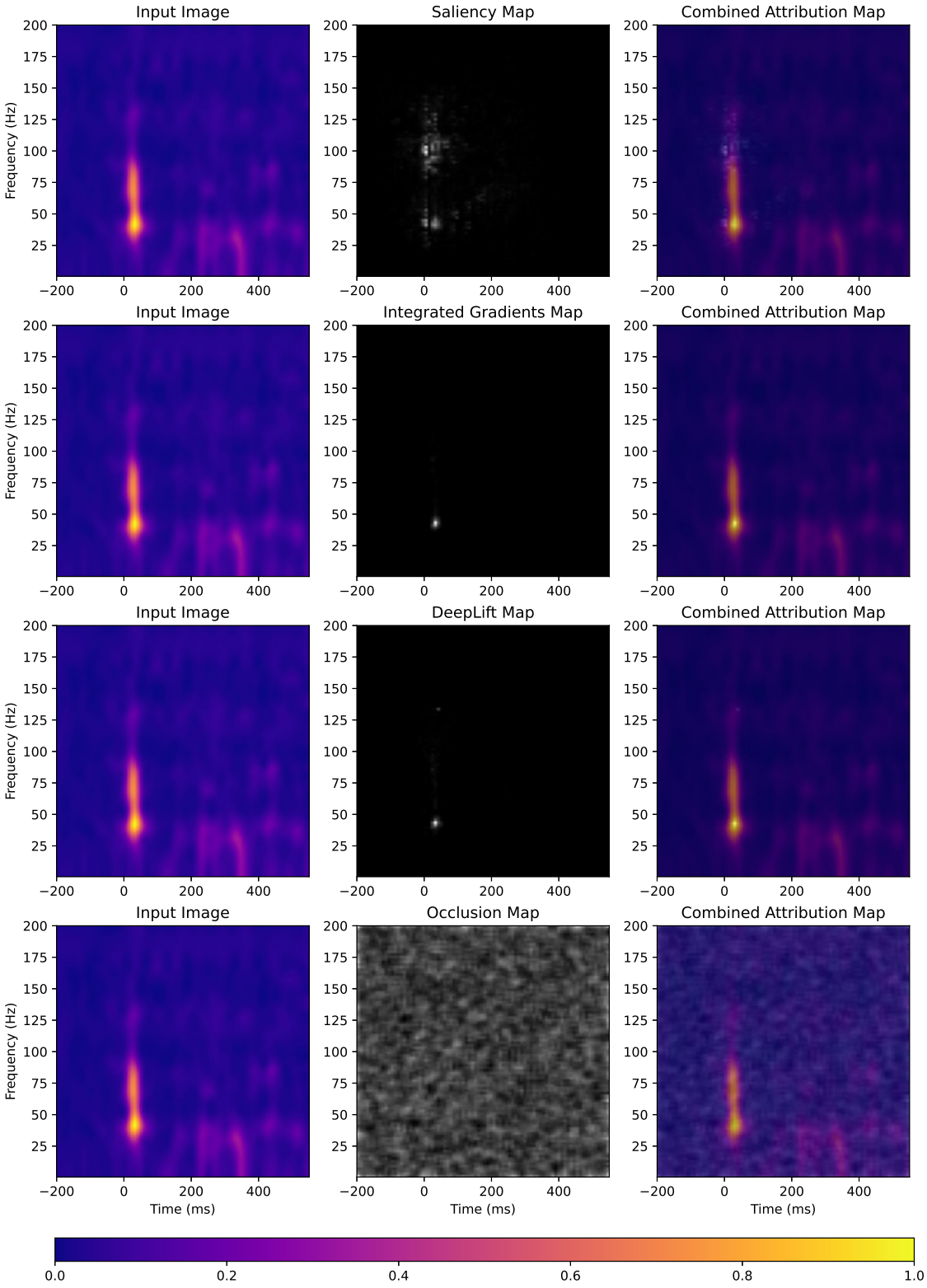

Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

B.2.2 Test

Control



LTP


Input Image
Saliency Map
Combined Attribution Map
Input Image
Integrated Gradients Map
Combined Attribution Map
Input Image
DeepLift Map
Combined Attribution Map
Input Image
Occlusion Map
Combined Attribution Map
Frequency (Hz)
Time (ms)
200
175
150
125
100
75
50
25
−200
0
200
400
0.0
0.2
0.4
0.6
0.8
1.0

Time Series C

In order to test whether time series data could be used as input as an alternative to CWT, this was tested with a 1-dimensional convolutional layer. The network consisted of the same hyperparameters as the base model, however only with one convolutional layer. Time series was tested with multiclass classification and a binary between control and intervention. As seen in Figure C.1, the confusion matrix of the time series model with multiclass classification did not predict the SNI group. Furthermore, the LTP prediction was questionable, whereas the control predictions were somewhat consistent. The learning curve showed a small drop in the validation loss with respect to the starting point but did not follow the training loss. For the binary classification, an accuracy of 63.4% between intervention and control was obtained, compared to the CWT version of the binary classification with 62.9% performing slightly better. Based on these findings, it was determined that neither of these models was suitable for further utilization.

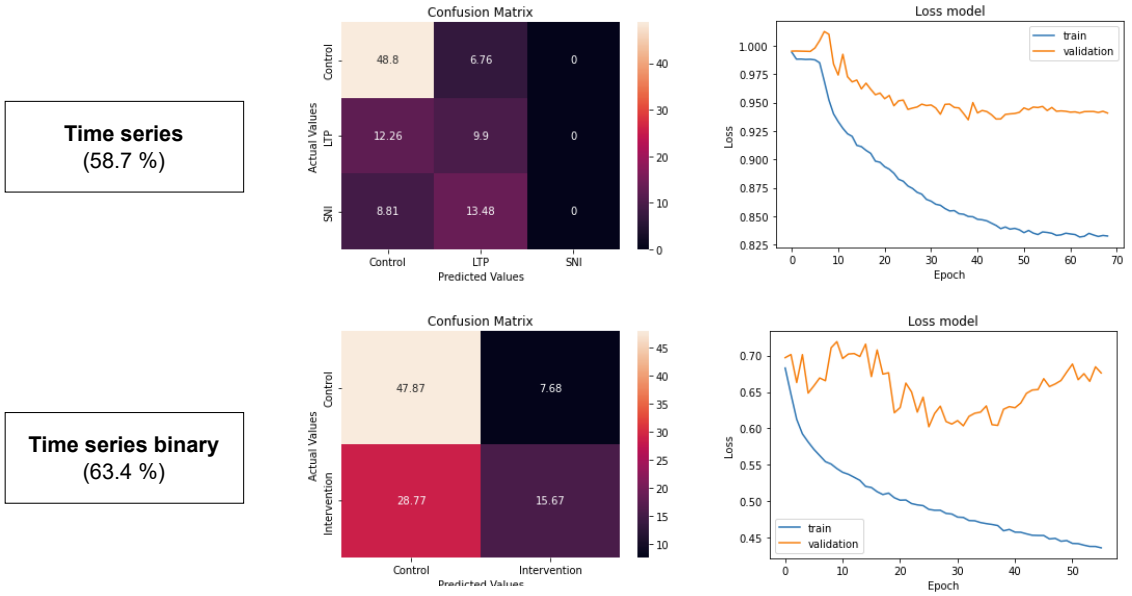


Figure C.1. Confusion matrix and learning curve for time-series models with multiclass and binary classification. The blue and orange lines in the training and validation loss, respectively. The accuracy of the two models is written underneath the respective model names.

Activity- and Time-plan D

To keep track of how much time we had available for each activity in this project an activity and time plan was made. The first iteration of this plan can be seen in Table D.1. The plan was made from the basis of us wanting to use 40 hours a week on the project. We then used backtracking to set milestones for when we should be finished with different parts of the project. We also wanted to make sure we met the requirements of the learning goals, which is why these were put on the plan as well.

Table D.1. Initial activity and time plan

Activities	February				Marts				April				May					Total	Leaning goals
	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21		
Init. prob. search	x	x	x																1,2,4,5
Struct. lit. search			x	x															1,2,3,4,5
Introduction				x	x	x													1,2,4,5
Pre-processing					x	x	x												1,2,4,5
Development of ML/DP method								x	x	x	x	x	x	x					1,2,4,5,6
Results													x	x	x				2,6
Discussion																x	x		2,4,5,6
Conclusion																	x		2
Article finalize															x	x	x		2
Appx. & PBL			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		2
Project (hours)	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	680	
Additional deadlines																			
Status seminary																			
Semester meetings																			

The time planning was an iterative process and was continuously updated as time passed. Table D.2 is an example of an iteration of the plan. An example of the change is that more time was added to the structured literature search since two weeks wasn't enough time to perform two searches, which was not planned at the start of the project. The article writing was also scrapped because the results of the projects were not suitable for an article. Two of the members of the group also got a study job this semester, which was added to the plan, so this could be accounted for in the plan. Keeping track of the extra working hours meant that the two members could slowly catch up in their free time.

Table D.2. Iterated version of the activity and time plan

Activities	February				Marts				April					May				Total	Leaning goals
	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21		
Init. prob. search	x	x	x																1,2,4,5
Struct. lit. search			x	x	x														1,2,3,4,5
Introduction				x	x	x	x	x											1,2,4,5
Pre-processing								x	x	x									1,2,4,5
Development of ML/DP method										x	x	x	x	x					1,2,4,5,6
Explainable AI											x	x	x	x					
Results															x	x			2,6
Discussion																x	x		2,4,5,6
Conclusion																	x		2
Article finalize																x	x		2
Appx. & PBL			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		2
Project (hours)	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	680	
Additional deadlines																			
Status seminary																			
Semester meetings																			
Arbejde																			
Nickolaj	6	2	5	0	4	0	6	4	0	0	0	3	0	3	0	6	6	45	
Mikkel	2	2	8	0	5	3	4	2	2	0	2	4	0	2	4	0	0	40	

The activity and time plan were used on a weekly basis to create overall goals for each week. These goals ensured that the work done each week was aligned with the plan and helped prevent falling behind. These weekly goals were assessed each Friday, and a status meeting was held at the end of the day. At these meetings, the goals would be discussed whether they were reached or more work was needed. This was also used to iterate the time plan if a task was more time-consuming than first assumed.

For the final month, we made an overall plan for each day, so we had a strict timeline for when the different sections had to be finished. This can be seen in Table D.3. The article writing was redacted from the time plan since the results were not expected and therefore not suitable for an article. Instead, the time was used to do some more experimenting with the model and the data to try and improve the results.

Table D.3. Final month of the project

Date	Focus	Deadlines
2. maj	CNN (udvikling + formidling af CNN arkitektur)	
3. maj	CNN (udvikling + formidling af CNN arkitektur)	
4. maj	CNN (udvikling + formidling af CNN arkitektur)	
5. maj	CNN (udvikling + formidling af CNN arkitektur)	**Optimering af CNN done**
8. maj	Test+validering af CNN → resultater	
9. maj	Test+validering af CNN → resultater	
10. maj	Test+validering af CNN → resultater	
11. maj	Validering-, test- & resultat-afsnit	
12. maj	Validering-, test- & resultat-afsnit	
15. maj	Artikelskrivning	
16. maj	Artikelskrivning	
17. maj	Artikelskrivning	
18. maj	Artikelskrivning	
19. maj	Artikelskrivning	
22. maj	Artikelskrivning	
23. maj	Resultater + diskussion	
24. maj	Diskussion	
25. maj	Diskussion	
26. maj	Diskussion + konklusion	
29. maj	Diskussion + konklusion + abstract	
30. maj	**Alt tekst skal være færdig**	**Rapport og artikel done**
31. maj	Sidste rettedag	**Aflevering**