UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA PER LE COMUNICAZIONI MULTIMEDIALI E INTERNET

"Analysis of EEG signals for the prediction of epileptic seizures"

Relatore: Prof. / Dott TESTOLIN ALBERTO

Laureanda: MEDVEDEVA NATALYA

ANNO ACCADEMICO 2023 – 2024

Data di laurea 24/04/2024

Abstract

Epilepsy, a chronic neurological disorder characterized by recurrent seizures, poses significant challenges to patients' quality of life and healthcare systems. Automatic seizure prediction algorithms offer promise in enhancing patient safety and care by providing timely warnings. Traditional machine learning (ML) models for seizure prediction often rely on manual feature extraction, which is time-consuming and requires expertise due to the complexity of electroencephalogram (EEG) signals.

Deep learning (DL) models present an alternative, demonstrating effectiveness in automatically extracting features from complex data like EEG signals. This study investigates the performance of three DL models (EEGNet, DeepConvNet, and ResNet68) for seizure prediction using a dataset containing epileptic events recorded from 10 patients. The models were evaluated individually for accuracy, sensitivity, and specificity.

The findings suggest that the DL models did not achieve superior performance compared to classical ML models. EEGNet demonstrated acceptable accuracy (65.78%) and specificity (82.94%), but lacked sensitivity in capturing preictal events, potentially due to its insufficient complexity in capturing subtle patterns within EEG signals. DeepConvNet exhibited high sensitivity (83.45%), but low specificity, leading to a high false alarm rate. ResNet68, despite its complexity, requires further optimization and potentially a larger dataset to improve performance (average accuracy: 63.80%, sensitivity: 69.71%, and specificity: 61.42%).

Limitations in dataset representativeness may have contributed to the observed weaknesses across all models. Strategies such as using more diverse datasets and advanced optimization techniques may improve the strength and ability of deep learning models in seizure prediction.

In conclusion, while DL models hold promise for seizure prediction, their performance in this study raises questions about their applicability in clinical settings. Further research exploring techniques such as data augmentation and deeper architectures is needed to refine DL models for improved seizure prediction and patient care.

Contents

1 Introduction **6**

1.1 Epilepsy: A Challenging Neurological Disorder 6

1.2 Diagnosis 7

1.3 EEG signal 8

1.4 Problem Definition and Necessity of a Timely Prediction 10

2 Artificial Intelligence Applications to Epilepsy **12**

2.1 Epileptic Seizure Prediction with Classical Machine Learning 12

2.2 Deep Learning Approach 16

3 Materials and Methods **24**

3.1 EEG Dataset 24

3.2 Preprocessing 25

3.3 Tools and Libraries 26

3.4 Deep Learning Models Architecture 34

4 Experimental Design **45**

4.1 Training Process 45

4.2 Evaluation Metrics 49

5 Results **52**

5.1 EEGNet Results 52

5.2 DeepConvNet Results 55

5.3 ResNet68 Results 56

6 Conclusions **61**

7 List of Acronyms **63**

Bibliography **65**

Chapter 1

Introduction

1.1 Epilepsy: A Challenging Neurological Disorder

Epilepsy is a chronic neurological disorder that affects the lives of approximately 50 million individuals worldwide, disproportionately impacting those in low- and middle-income countries. It is the fourth most common neurological disorder, affecting about 1% of the population at all ages [1]. Characterized by recurrent seizures – which are typically sudden episodes of uncontrolled convulsions or changes in sensations – this condition interferes with the normal electrical activity in the brain. The severity can vary, from momentary lapses in awareness to strong, involuntary body movements. The unpredictable nature of these seizures significantly impacts various aspects of life, posing major challenges and demanding multifaceted therapeutic approaches.

A diagnosis of epilepsy necessitates the occurrence of at least two unprovoked seizures, each a storm of electrical activity within specific brain regions [2]. These episodes can have diverse and far-reaching consequences, affecting not only motor skills and consciousness but also orientation, sensory perception, emotional state, and cognitive functions. The burden of epilepsy, however, extends beyond the immediate effects of seizures. Individuals with epilepsy are more susceptible to physical complications, including fractures, bruises, and other injuries sustained during seizures [3]. The erratic nature of epilepsy and its potential impact on daily life can also lead to psychological burdens, with anxiety, depression, and social stigma being common challenges [1]. In resource-limited settings, where access to proper healthcare and social support is scarce, increased mortality rates are observed among individuals with epilepsy [4].

The origins of epilepsy are diverse and encompass a complex interplay of factors. Structural abnormalities arising from prenatal or perinatal brain damage, congenital malformations, or tumors can pave the way for seizures. In some cases, genetic mutations play a role, with specific gene alterations contributing to the disorder's development. Infections

that breach the brain's defenses can also trigger epilepsy, alongside metabolic imbalances affecting blood sugar, electrolytes, or other chemicals. In rare cases, despite advancements in research, the cause of epilepsy remains unelucidated for some individuals.

Managing epilepsy effectively requires a comprehensive approach. Anti-seizure medications offer a powerful tool, significantly reducing seizure frequency and severity, thereby improving quality of life and reclaiming a sense of control.

1.2 Diagnosis

Epilepsy, while posing a significant burden on individuals and healthcare systems, also presents a unique challenge in its diagnosis. Despite advancements, pinpointing its origins and accurately predicting seizure events remain challenging. This chapter describes the intrinsic difficulties of diagnosing epilepsy, highlighting the limitations of current methods and emphasizing the critical role of seizure prediction in managing this complex disorder.

The cornerstone of diagnosis lies in a detailed history, capturing patient experiences and witness accounts. Neurological examinations provide further insights, while ancillary tests like neuroimaging and EEG offer glimpses into the brain's activity. However, the interpretation of electroencephalography (EEG) signals represents a significant obstacle. Manually reviewing hours or even days of data from ambulatory or continuous EEG recordings is time-consuming and prone to subjectivity due to varying levels of expertise among readers [5].

Further complicating matters are interfering artifacts within the EEG signal, obscuring potential abnormalities and hindering accurate identification. Additionally, the inherent variability of seizures across individuals necessitates a nuanced approach, as no single "fingerprint" defines all epileptic events. Misdiagnosis and confusion with other episodic conditions remain significant concerns, highlighting the need for more precise diagnostic tools.

In effective epilepsy management, accurately detecting seizures outside clinical settings remains a key challenge. While traditional methods like surface electromyography (EMG) reliably capture the forceful movements of generalized tonic-clonic seizures, they struggle with other, subtler seizure types. This limitation calls for innovative, practical "multimodal" approaches that combine diverse data sources. It is not the topic of this work to go into detail of possible integration of measurements of physiological changes during seizures, like heart rate fluctuations, but is interesting to notice that this combined approach holds immense potential for accurately identifying a wider range of seizures, even beyond the clinic walls. Therefore, it's a crucial step towards personalized care and transformative research in epilepsy management [6].

Diagnosing epilepsy is a complicated journey, filled with personal interpretations and limitations in current methods. However, the possibility of predicting seizures brings hope,

allowing timely interventions and improving the lives of those facing this challenge. By adopting advanced technology and encouraging research collaboration, it is possible to make strides in uncovering the mysteries of epilepsy, ultimately helping individuals take control of their lives.

Considering the aforementioned complications of epileptic diagnosis, a multifaceted approach is required for effective seizure prediction and improved epilepsy management. This includes collaborative efforts among neurologists, engineers, and data scientists, leveraging advancements in machine learning and artificial intelligence to analyze complex EEG data and identify subtle preictal signatures [7]. This work highlights the importance of in-between-seizure brainwave recordings for revealing underlying problems and predicting future seizure risk.

1.3 EEG signal

Accurate acquisition, robust preprocessing, and insightful analysis of the EEG signal are crucial for achieving optimal and efficient seizure prediction. Therefore, a thorough understanding of the inherent characteristics of EEG, its acquisition methods and the challenges posed by noise from various sources is essential.

The EEG records the synchronized electrical activity of neuronal populations in the brain, offering a non-invasive glimpse into its inner workings. In the conventional EEG representation, the x-axis signifies time, capturing the temporal evolution of the electrical signals, while the y-axis represents different channels or electrodes strategically placed on the scalp, reflecting the spatial distribution of neural activity. So in simple words, the EEG signal is essentially a 2D matrix of real values, where each row corresponds to a specific channel, and each column corresponds to a discrete time point. The amplitude of the recorded electrical potentials at each channel and time point reflects the magnitude of neuronal activity in the corresponding region of the brain at that particular moment.

These electrical potentials, often measured in microvolts, originate from the summation of post-synaptic potentials of large populations of neurons. For the patients with epilepsy disorder the electrical activity of neurons varies across distinct states relevant to epilepsy:

- Interictal state, which is the period between seizures, characterized by regular rhythms reflecting wakefulness, sleep, and cognitive function.
- Preictal state, which occurs up to 90 minutes before seizure onset and potentially exhibits subtle shifts in amplitude, frequency, or specific waveform patterns like spikes or sharp waves. Identifying these preictal signatures is crucial for seizure prediction.
- Ictal state, which is the seizure itself, marked by high-amplitude, rhythmic discharges

often localized to specific brain regions.

- Postictal state, which immediately follows the seizure and lasts for a few minutes, exhibiting transient changes in EEG patterns.

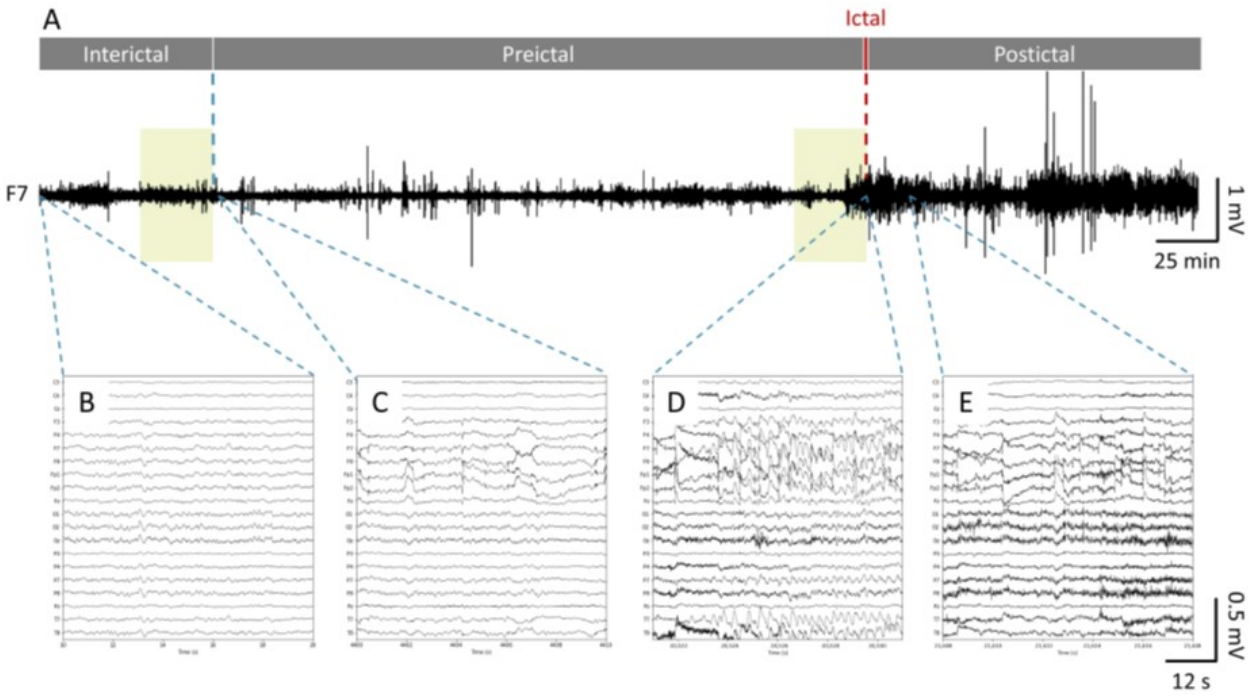


Figure 1.1: Standard segmentation for the EEG recording of an epileptic patient. (A) The trace depicts 480 min of recording from the F7 channel during a seizure divided into four main stages: interictal, preictal, ictal, and postictal. The green areas represent the 30 min from the preictal and interictal states used for the binary prediction task. Panels (B–E) illustrate magnifications of 20 s of recordings from 20 channels at the beginning of the interictal, preictal, ictal, and postictal states, respectively. Figure adapted from [8].

EEG acquisition involves placing electrodes, functioning as sensors, on the scalp, with their number varying from dozens to hundreds.

These electrodes capture local electrical fields, providing a multichannel representation of brain activity. However, this method of signal collection faces diverse challenges. Firstly, the skull acts as a filter, attenuating and blurring underlying neural signals, thereby limiting spatial resolution. Secondly, various sources can introduce unwanted electrical activity into the EEG, generating undesirable noise. These sources encompass physiological artifacts or subject-related artifacts, such as eye blinks, muscle contractions, and cardiac signals, as well as environmental artifacts like temperature and air humidity. These factors interact with noise sources such as line noise or other forms of electromagnetic interference [9].

Another critical factor is that EEG recordings inherently contain noise and lack precise localization to specific brain regions. This is because each scalp electrode captures activity

Figure 1.2: The image represents the standard electroencephalogram (EEG) monitoring process, with electrodes placed directly on the patient's head (sometimes the electrodes are integrated into a wearable cap). This examination is completely painless and requires the patient's cooperation, as they must avoid sudden movements and remain relaxed. By placing electrodes in specific areas of the scalp (along with 2 reference electrodes on the ears), the EEG can identify any abnormalities and the brain regions involved.

from its surrounding area, resulting in blurred information rather than pinpointing exact locations (spatial resolution limited to several centimeters). This phenomenon, known as "channel crosstalk," poses a significant challenge for accurately interpreting the brain's electrical activity from scalp EEG recordings [10].

Careful denoising and preprocessing of the raw EEG signal have been the focus of several studies. Various approaches have been developed to initiate signal cleaning, such as filtering to directly remove undesired frequencies or artifact removal, which includes manual removal, automatic removal, or no cleaning. Despite its subjectivity, manual removal remains prevalent, and in other cases, automatic methods such as independent component analysis (ICA) and discrete wavelet transformation (DWT) are employed [11].

1.4 Problem Definition and Necessity of a Timely Prediction

Despite receiving appropriate treatment and adhering to medication, and despite the introduction of newer antiepileptic drugs, around 20%–30% of individuals diagnosed with

epilepsy continue to experience seizures, a condition referred to as drug-resistant epilepsy [12]. This highlights the crucial need for developing accurate seizure prediction methods, enabling proactive management strategies that go beyond solely reactive interventions. The ability to predict seizures before their occurrence represents a transformative shift in epilepsy management. Early warnings, even a few minutes prior to a seizure, can empower individuals to take precautions and potentially prevent associated injuries or even the seizure itself.

Current clinical practices often rely on trained neurologists visually interpreting electroencephalography (EEG) recordings to diagnose and classify seizure types. However, this approach suffers from limitations, including time-consuming analysis, inherent subjectivity, and susceptibility to errors. Therefore, developing automated, objective, and reliable methods for interpreting and analyzing EEG signals for seizure prediction is essential.

The distinct signatures of pre-seizure and interictal states in the EEG offer a promising avenue for developing seizure prediction algorithms. These pre-seizure signatures often manifest as significant changes in amplitude and frequency compared to the baseline interictal state [13]. These alterations in brain activity preceding a seizure enable the classification of the preictal state up to 30 minutes before the onset of an actual seizure.

This work focuses on training deep learning models using EEG signals to achieve timely predictions of epileptic seizures, a critical aspect in epileptic treatment and diagnostic studies. Automated seizure detection algorithms, which incorporate automated seizure-detection features, are the subject of numerous studies aiming to enhance their accuracy and specificity for clinical reliance. Achieving high accuracy is the key for acceptance by clinicians, making this area one of the most widely explored in the field of epileptic treatment and diagnostic studies, including the present study.

Chapter 2

Artificial Intelligence Applications to Epilepsy

2.1 Epileptic Seizure Prediction with Classical Machine Learning

From the analysis of numerous scientific papers it can be observed that Artificial Intelligence (AI), driven by Machine Learning (ML) techniques and Deep Learning (DL) models, has offered an alternative and promising approach in the field of healthcare, including the prediction of epileptic seizures. However, while these techniques represent significant progress, challenges remain in achieving consistently high levels of accuracy and reliability in real-world applications.

The effectiveness of machine learning lies in its capability to discern patterns within extensive datasets using statistical methods. The growing availability of large-scale biomedical data presents new avenues for healthcare researchers to harness Machine Learning, enhancing the diagnosis and treatment processes.

The basic two steps involved for seizure detection in the various classical ML methods proposed in the literature are feature extraction and classification [14]. The first step involves extracting relevant features from the raw EEG signal. These features encapsulate essential characteristics of the brain activity and serve as the foundation for subsequent seizure prediction. The process can be likened to characterizing a complex language by analyzing its individual components (words). Similarly, for EEG signals, feature extraction aims to identify informative components that can differentiate seizure-related activity from normal brain function. Following feature extraction, the ML model enters a classification phase. Here, the model undergoes a supervised learning process, utilizing a pre-labeled dataset of EEG recordings where seizures have been meticulously identified by experts. Through this process, the model learns to distinguish between EEG patterns associated with upcoming seizures and

those representing normal brain activity.

Feature extraction techniques essentially analyze the EEG signal from various perspectives, extracting features that hold potential for seizure prediction. The commonly used methods are: Fast Fourier Transform (FFT), Wavelet Transformations (WT) or Independent Component Analysis (ICA). FFT decomposes the signal into its constituent frequencies, enabling the identification of characteristic frequency patterns associated with seizure. WT offers a more nuanced approach by allowing the analysis window to adapt to the frequency content of the signal. This flexibility makes WT particularly well-suited for capturing transient changes in EEG signals, which are often indicative of seizure activity. ICA separates the EEG signal into independent sources, potentially revealing hidden patterns or underlying processes masked by the raw EEG recording [15].

By effectively combining signal preprocessing and feature extraction techniques, classical ML algorithms can be trained to identify robust patterns within EEG signals. Classical ML approaches for epileptic seizure prediction rely on various classification techniques, which offer several advantages compared to more complex models that will be discussed in the next chapter. For instance, traditional classifiers require less computational power and time for implementation compared to deep learning models. This allows for faster model development and deployment in clinical settings. Also, classical ML techniques are generally less prone to overfitting, which occurs when a model becomes overly tuned to the training data and performs poorly on unseen data. It leads to potentially more generalizable models for seizure prediction.

Here's an overview of some commonly employed classification techniques and their specific applications in seizure prediction:

- Support Vector Machines (SVMs): SVMs aim to construct a hyperplane that maximizes the separation between seizure and non-seizure data points in the feature space. This approach focuses on identifying the most robust decision boundary for seizure classification. While achieving good accuracy, SVMs can be less interpretable due to their complex decision boundaries. Additionally, tuning their hyperparameters requires expertise, and they can be sensitive to outliers in the EEG data [16].

- Naïve Bayes: Naïve Bayes is particularly well-suited for scenarios where features are conditionally independent. In seizure prediction, it can be used to classify EEG features based on their likelihood of belonging to either seizure or non-seizure classes [17]. However, the assumption of feature independence may not always hold true for complex EEG data.

- k-Nearest Neighbors (k-NN): This non-parametric classification method classifies a new data point based on the majority class of its k nearest neighbors in the feature space.

In seizure prediction, k-NN can be used to classify new EEG recordings by comparing them to previously labeled seizure and non-seizure examples in the feature space [18]. However, k-NN can be less robust to noisy data and may struggle with high-dimensional feature spaces.

- Random Forest (RF): This ensemble method combines multiple decision trees, where each tree votes on the class of a new data point. The final prediction is made based on the majority vote of these trees. This approach helps to reduce overfitting and improve the generalization of the model. In seizure prediction, Random Forests can leverage the combined power of multiple decision trees to classify EEG features and identify patterns indicative of seizures [18].

- XGBoost: Another ensemble method, XGBoost employs gradient boosting to sequentially add decision trees, focusing on improving the model's performance for previously misclassified instances. This strategy leads to a more robust model capable of handling complex seizure prediction tasks [19]. However, XGBoost, like Random Forest, can be less interpretable due to the ensemble nature and may require careful hyperparameter tuning for optimal performance.

There are other classifiers like Linear Discriminant Analysis (LDA), Hidden Markov Models (HMMs) etc., but the listed ones have been the most efficient over past years. Evaluating the effectiveness of seizure prediction models relies on a trio of key metrics: accuracy, sensitivity, and specificity. Accuracy reflects the overall correctness of the model's predictions. A high accuracy indicates the model's ability to accurately classify both seizure and non-seizure EEG segments. Sensitivity assesses the model's ability to correctly identify preictal/seizure EEG segments. High sensitivity translates to a model's effectiveness in capturing and predicting upcoming seizures. Specificity evaluates the model's ability to correctly classify interictal (non-seizure) EEG segments. High specificity indicates the model's proficiency in accurately identifying normal brain activity. Mathematically, they are calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Sensitivity} = \frac{TP}{TP + FN}$$
$$\text{Specificity} = \frac{TN}{TN + FP}$$

where TP, TN, FP, and FN stand for true positive (correctly identified seizure signals), true negative (correctly identified non-seizure signals), false positive (incorrectly marked as seizure signals), and false negative (incorrectly marked as non-seizure signals), respectively.

Many recent Machine Learning (ML) techniques have reportedly achieved impressive

results, with accuracy, sensitivity, and specificity often exceeding 90%, and in some cases, even reaching near-perfect performance (close to 100%) on well-preprocessed datasets [18, 19, 20].

The point of departure for this project is the study "Methodological Issues in Evaluating Machine Learning Models for EEG Seizure Prediction: Good Cross-Validation Accuracy Does Not Guarantee Generalization to New Patients" by Shafiezadeh et al [8] . Even though the present thesis is based on deep learning approaches, the final results are compared to the results presented in this paper in order to observe the improvements or weak point of the modern DL models respect to the classical ML methods.

The paper mentioned above aims to evaluate seizure prediction models based on standard machine learning algorithms by comparing two cross-validation methods. The study utilizes two EEG datasets: the CHB-MIT benchmark and a new dataset collected by the Eugenio Medea IRCCS Hospital. The primary machine learning models considered include support vector machines, decision trees, k-nearest neighbors, logistic regression, naive Bayes, random forests, and gradient boosting. The best-performing model, XGBoost, achieves an average precision of 79% on the CHB-MIT dataset and 82% on the new dataset. However, when evaluated using a leave-one-patient-out validation scheme, the model's performance drops to around 50%, indicating the challenge of generalization. The EEG datasets contain multichannel recordings processed with notch and high-pass filters. Feature extraction yields 53 features, including time and frequency domain features. The study highlights the importance of robust validation methodologies, favoring leave-one-patient-out over randomized cross-validation. While randomized cross-validation may lead to overly optimistic results, leave-one-patient-out validation provides a more realistic assessment of model performance. The paper concludes by emphasizing the need for stringent evaluation criteria in seizure prediction and suggests exploring advanced machine learning techniques like deep neural networks in future research.

In the later chapters the dataset provided by the Eugenio Medea IRCCS Hospital will be discussed more in details since in this project the same dataset is used. The model demonstrates overall good performance, as indicated by the average accuracy of 81.6%. This suggests that the model can effectively differentiate between preictal (seizure) and interictal (non-seizure) states in EEG recordings. The variation in accuracy across different patients (ranging from 73.24% to 86.23%) suggests that the model's performance may be influenced by individual patient characteristics or the underlying EEG patterns associated with seizures. The variance in sensitivity between patients is even more noticeable, ranging from 49.8% to 83.4%. Factors such as the frequency and severity of seizures or unique EEG signatures may contribute to this variability. However, the consistently high specificity across most patients (ranging from 88.82% to 100%) suggests that the model is adept at correctly identifying non-seizure states, which is crucial for minimizing false alarms and ensuring patient safety.

The average sensitivity (63.47%) being lower than specificity (95.92%) implies that the model may be more proficient at correctly identifying non-seizure states than seizure states. This could be attributed to the inherent challenge of detecting seizures, which may manifest in diverse EEG patterns and vary in severity among patients.

Even if a lot of traditional ML techniques have demonstrated high accuracy, incorporating advanced machine learning techniques, such as deep neural networks, may offer opportunities to improve the model's performance and robustness, particularly in capturing complex patterns within EEG data.

2.2 Deep Learning Approach

The field of epileptic seizure prediction is witnessing a growing preference for Deep Learning (DL) models. This shift stems from the desire to reduce human intervention in the model development process.

The success of Deep Learning extends far beyond epileptic seizure prediction. DL architectures have made significant contributions in various medical domains, including: (1) Clinical Imaging, to analyze medical images like X-rays, MRIs, and CT scans for disease detection and diagnosis [21]; (2) Genomics and Proteomics, to identify patterns in genetic and protein data to understand disease mechanisms and develop personalized medicine approaches [22]; (3) Computational Biology, to extract insights from complex biological data to advance our understanding of health and disease [23]; (4) Disease Prediction: to predict the risk of developing various diseases, including neurological conditions [24]. These successes highlight the remarkable ability of DL algorithms to detect intricate patterns in high-dimensional data, making them particularly well-suited for analyzing EEG signals.

Deep Learning builds upon the foundation of Machine Learning (ML) but takes it a step further. It utilizes complex neural networks with multiple interconnected layers [25]. These layers work together to progressively process and analyze raw data, ultimately leading to a desired outcome. Unlike traditional ML approaches, which often rely on handcrafted features, Deep Learning has the remarkable ability to automatically extract relevant features from the data itself. Deep Learning eliminates this step by automatically learning informative features directly from the raw EEG data. This not only reduces human intervention but also potentially allows the model to discover subtle patterns that might be missed by human experts. Furthermore, the ability to learn features directly from data leads to more robust and generalizable models. This is particularly important for seizure prediction, where a model needs to perform well on unseen EEG recordings beyond the training data [26].

In the world of Deep Learning (DL), several neural network architectures have emerged as powerful tools for various tasks. Here, four most frequently used architectures will be discussed: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs),

Long Short Term Memory (LSTM) and Residual Neural Network (ResNet). All these DL architectures share a common training process. The models are "fed" with labeled EEG data (normal/interictal vs. pre-seizure or seizure) and progressively learn to identify features and patterns that distinguish these states. The success of this training process hinges on finding the right balance between model complexity and the amount of available training data. The following sections will delve deeper into the specifics of each architecture, exploring their strengths and limitations in the context of epileptic seizure prediction from EEG signals.

Convolutional Neural Networks

CNN structure resembles the interconnected neurons in the human brain. CNNs excel at extracting features from multi-dimensional data like EEG signals. They achieve this through a process called convolution, which acts like a filter that slides across the data, identifying and extracting informative patterns. The essential building blocks of CNN are: Convolutional Layer (Conv Layer), Activation Functions and Pooling Layers.

Convolutional Layer utilizes filters, also known as kernels, which are essentially small matrices that slide across the input EEG signal. The size of the kernel determines the extent of the EEG signal analyzed at once. Stride, on the other hand, controls the movement of the kernel. A stride of 1 indicates the kernel moves one step at a time, while a stride of 2 signifies a jump of two steps between consecutive analyses. The success of training a DNN depends on the balance between the number of parameters to be trained and the number of examples in the training dataset [27].

By performing convolution across the entire EEG signal with various kernels, the network captures diverse features embedded within the data. Mathematically, it involves calculating the weighted sum of the element-wise product between the kernel and a specific portion of the EEG signal. In simpler terms, the kernel's values are multiplied with corresponding data points in the signal, and the products are summed. This process is repeated as the kernel slides across the entire signal. By employing various kernels of different sizes and orientations, the Conv Layer can capture diverse spatiotemporal patterns embedded within the data. These patterns might include subtle changes in frequency, amplitude, or other characteristics that could indicate an impending seizure. The convolution operation is:

$$y[n] = \sum_{k=0}^{N-1} x[k] \cdot h[n-k]$$

where x is signal, h is filter, and N is the number of elements in x. The output vector is y.

After the convolution operation, activation functions introduce non-linearity into the network. This is crucial because without them, CNN would simply learn linear relationships,

limiting its ability to model complex patterns in EEG signals. Activation function is an operation which maps an output to a set of inputs. An activation function is an operation that maps an output to a set of inputs, and in CNNs, the ReLU (Rectified Linear Unit) is a popular choice due to its reduced likelihood of the gradient vanishing problem.

The gradient vanishing problem occurs when the gradient of the loss function becomes too small during backpropagation, preventing effective learning in the network. The ReLU activation function helps address this by maintaining a constant gradient, allowing only positive values to pass through and setting negative values to 0:

$$f(x) = \max(0, x)$$

Another commonly used activation function is Tanh, which is similar to the sigmoid function but ranges from -1 to 1. Tanh maps negative inputs strongly to negative values, positive inputs strongly to positive values, and zero inputs to values near zero.

The choice of activation functions depends on the specific layer's role in the network. For instance, the output layer often uses a sigmoid activation function in binary classification problems:

$$f(x) = \frac{1}{1 + e^{-x}}$$

paired with binary cross-entropy as the loss metric. In cases with more than two classes, a softmax activation function is employed at the output layer. This function calculates the probability distribution of the k output classes, ensuring that the output values ('p') fall between 0 and 1 and collectively sum to 1:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

where x is the net input. The corresponding loss metric for multi-class scenarios is categorical cross-entropy.

There are always Pooling Layers in between convolutional layers, which serve a dual purpose: dimensionality reduction and overfitting prevention. Pooling layers downsample the output from the convolutional layer, typically by selecting the maximum value (max pooling) or average value (average pooling) within a specific window. This reduces the number of parameters in the network, making it computationally efficient and less prone to overfitting, where the model memorizes the training data too well and performs poorly on unseen data. To address the challenge of overfitting, additional strategies involve the application of regularization techniques such as l2 regularization, batch normalization or the integration of dropout layers. These techniques act to constrain the model's freedom, encouraging the optimization process to seek simpler solutions.

L2 regularization, also known as Ridge Regression or Tikhonov regularization, imposes

constraints on the model's flexibility by incorporating a regularization term into the cost function. This compels the model, while fitting the data, to maintain weights at minimal values. The regularization term, expressed as the squared 2-norm (Euclidean distance), is influenced by the parameter λ, regulating the extent of model regularization. Here is the mathematical representation of L2:

$$R(w) = \frac{\lambda}{2}\|w\|^2$$

$$\text{error} = \text{L2 regularization} + \text{loss function} \tag{2.1}$$

where m is the length of the weights vector W of the model. If λ equals 0, there is no regularization, as the added regularization term in the loss function becomes zero. Conversely, if λ is too substantial, the weights tend to approach zero, causing the model to underfit the data and impeding effective learning.

Dropout emerges as a widely favored regularization technique in deep learning [28]. Its fundamental principle is straightforward: during the training process, neurons in certain layers are randomly "dropped out," effectively set to zero and disregarded. The selection of neurons to deactivate in each layer is based on a probability parameter, denoted as p. This implies that, at each training step, each neuron has a probability p of being deactivated, with p representing the dropout rate or the fraction of nodes to be ignored. The implementation of dropout introduces a form of randomness during training, preventing the network from becoming overly reliant on specific neurons and enhancing its generalization ability to new data.

Batch normalization reduces internal covariate shift by maintaining consistent input distributions across layers, which facilitates more stable and efficient training [29]. It also acts as a regularizer by introducing slight noise during training, similar to the dropout technique, contributing to improved model generalization. Lastly, batch normalization can have a positive impact on the optimization process, allowing for the use of higher learning rates and accelerating convergence during training.

Returning to the conventional CNN architecture employed for seizure prediction, it typically comprises multiple convolutional layers stacked together. Each successive layer serves to extract increasingly intricate features from the EEG signal, facilitating the network's ability to discern relevant patterns. Following these convolutional layers, pooling layers are frequently introduced to reduce dimensionality, while dropout layers are incorporated to mitigate the overfitting problem. Finally, fully connected layers take over, integrating the extracted features and making the final prediction about the presence or absence of a seizure.

The process of training in CNN is based on backpropagation (BP). BP represents the core of the learning process of a neural network, since the model's learnable parameters undergo

iterative updates. Initially, predictions are generated through a forward pass, following which the gradient of the loss function relative to the weights is computed. The loss quantify the discrepancy between a model's predictions and the true labels (ground truth) for a given set of data. By minimizing this loss function during training, the model iteratively refines its internal parameters (weights and biases) to produce increasingly accurate predictions. The most popular loss functions in AI are MNE (Mean Squared Error), used for regression tasks and Cross-Entropy loss, which is used for classification tasks and measures the difference between the probability distribution generated by the model and the true probability distribution of the target class. It penalizes deviations between the predicted and actual distributions, with higher penalties for larger deviations. Here is the equation for cross-entropy calculation:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(p_{ik})$$

where p(ik) is the predicted probability distribution and y(ik) is the actual distribution for N data points across K classes.

Subsequently, the weights are adjusted in the opposite direction of the gradient. Mathematically, backpropagation entails the modification of weights and biases, influenced by factors such as momentum, learning rate, cost function, layer number, and regularization parameter:

$$W_l^{(t+1)} = W_l^{(t)} - \eta\frac{\partial C}{\partial W_l} - \lambda W_l^{(t)}$$

$$B_l^{(t+1)} = B_l^{(t)} - \eta\frac{\partial C}{\partial B_l}$$

where W, B, l,λ, x, n, m, t, and C represent the weight, bias, layer number, regularization parameter, learning rate, total number of training samples, momentum, updating step, and cost function respectively. The regularization parameter, denoted by λ, plays an important role in preventing data overfitting, whereas the learning rate governs the pace at which the network assimilates knowledge during training. Momentum aids in converging the data efficiently. While a typical learning rate may be initially set to

$$1 \times 10^{-2} - 1 \times 10^{-5},$$

but its optimal value is often determined through trial and error. The adjustment of parameters like momentum, regularization, and learning rate is facilitated by modern optimization algorithms such as Stochastic Gradient Descent (SGD) with Momentum, Adam, or RMSprop. An optimization algorithm assumes responsibility for dictating how the model's parameters (weights) evolve throughout the training process. Its overarching objective is to minimize

an objective function, namely the cost function, which is contingent on the model's weights. Given that the cost function is computed based on the model's predictions, determined by the weights and input features, the significance of weights in minimizing the cost function becomes apparent. Hence, as the values of weights are dictated by Gradient Descent, the optimizer's role is to refine the Gradient Descent process, ensuring its efficiency and efficacy in model optimization.

In the context of epileptic seizure prediction, CNNs have emerged as powerful tools due to their ability to identify spatiotemporal patterns within EEG signals [30]. These patterns, often subtle variations in frequency, amplitude, or other characteristics, can hold crucial information about the potential onset of a seizure. By learning these intricate patterns from labeled EEG recordings (normal, preictal or ictal), CNNs can effectively classify new, unseen data, potentially aiding in early intervention and improved patient outcomes. The training time of CNNs can vary depending on the dataset size and model complexity. While simpler models might train in hours, achieving accuracies exceeding 90% for seizure detection tasks. However, it's important to acknowledge that this remarkable accuracy might not be universally achievable and may depend on various factors, including specific datasets and model configurations. A study employing a relatively simple CNN architecture with two convolutional layers, a pooling layer, and three fully connected layers reported an average accuracy of approximately 99% for seizure detection [31]. Conversely, deeper models, such as a 13-layer deep CNN designed for automated classification of normal, preictal, and ictal EEG signals without prior feature extraction, achieved an average accuracy of 88.67%, a specificity of 90.00%, and a sensitivity of 95.00%. [5] These findings highlight the potential of CNNs for seizure prediction, with promising accuracy levels being reported in various studies.

Recurrent Neural Networks and LSTM

RNNs (Recurrent Neural Networks) differ from CNNs (Convolutional Neural Networks) in their ability to consider both current and past information, making them suitable for analyzing sequential data like EEG signals. They employ an internal memory, known as a hidden state, which updates with each new data point, capturing relevant information from past observations. However, RNNs suffer from the vanishing gradient problem, limiting their ability to learn long-term dependencies [32]. LSTM (Long Short-Term Memory) networks, which are evolution of RNN, address this issue by incorporating gates to control the flow of information. These gates, including the forget gate, input gate, cell state, and output gate, enable LSTMs to retain information over longer periods. While LSTMs offer improved performance in learning long-term dependencies, they come with increased computational costs and require careful optimization of hyperparameters for optimal results. LSTM is an evolution on the RNNs that include gates to deal with the vanishing gradient problem. Lately,

LSTM is getting used more and more often in the field of epileptic seizure prediction.

LSTMs are noteworthy in the field of seizure prediction, given their demonstrated efficacy and suitability for this application [33]. They are capable of capturing temporal dependencies in the data. This is important for EEG analysis because patterns in EEG data often depend on the previous data points in time. For different preictal window sizes and the complexity of the LSTM architectures, the sensitivity and specificity values consistently go beyond 99%. The experiments conducted on the same dataset and with the same conditions, but with classical ML models show that the LSTM network significantly improves classification performance. What's more, the key success of the LSTM classifier is maintaining sensitivity and specificity above 99%, resulting in a very low False Positive Rate (FPR) and successful prediction of every seizure. The LSTM classifier outperforms previous methods, including CNN, on the same EEG dataset with similar preictal window durations [34]. In many studies, it's common to combine CNN with LSTM, achieving accuracy, specificity, and sensitivity higher than 99%. This combination eliminates the need for certain feature extraction and selection steps [35].

Residual Neural Networks

ResNets also address the challenge of training very deep neural networks, which can become prone to vanishing gradients. ResNets introduce "shortcut connections" that allow information to flow directly between layers, facilitating better training and potentially improving model performance [36]. These skip connections allow the network to skip one or more layers.

The main characteristic of ResNet lies in its use of residual blocks. A residual block consists of two main paths: the identity path and the shortcut path. The identity path is a straightforward pass-through of the input to the next layer, while the shortcut path introduces a series of operations. Mathematically, the output of the residual block is given by:

$$\text{output} = F(x) + x$$

where F(x) represents the operations within the block, and x is the input. The essence of ResNet's innovation is the introduction of shortcut connections, which enable the network to skip one or more layers during forward and backward passes. This alleviates the vanishing gradient problem encountered in training deep networks. The skip connections facilitate the flow of gradients, allowing for the successful training of exceedingly deep networks by preserving information from earlier layers.

To balance computational efficiency and expressiveness, ResNet often adopts a bottleneck architecture within each residual block. This involves a sequence of three operations: 1x1

convolution (dimension reduction), 3x3 convolution, and another 1x1 convolution (dimension restoration). This configuration reduces the computational load while maintaining the expressive power of the network. ResNet achieves its depth by stacking residual blocks on top of each other. The architecture often follows a pattern of increasing depth while decreasing spatial resolution, typically using pooling layers or strides in convolutional layers to downsample the spatial dimensions.

Following the stacking of residual blocks, ResNet typically concludes with a global average pooling layer and a fully connected layer for the final prediction. The combination of identity and shortcut paths in residual blocks enables the network to simultaneously focus on capturing low-level and high-level features, crucial for discerning subtle patterns in EEG signals indicative of seizures. Using ResNet for seizure classification allows to achieve state-of-the-art results with minimal data preprocessing and without extracting any hand-crafted features [37]. In the latest studies, the use of network architecture combining convolutional, batch-normalization, residual connections, and drop-out layers provides good convergence and has demonstrated the highest performance among numerous network structures that we used in previous years.

However, the introduction of skip connections results in an increased computational cost compared to shallower networks. Also, like LSTMs, ResNets introduce additional hyper-parameters associated with the residual blocks, requiring careful tuning for optimal performance.

As a result, all the described deep learning models show impressive accuracy levels, which translates to the potential for highly reliable seizure prediction. Unlike traditional ML methods, DL models can learn features directly from raw EEG data, eliminating the need for extensive manual feature engineering. However, the high complexity of the models burdens the computational requirements. Also, all deep learning networks are very time-consuming in the training process, and usually, a GPU is required. DL models tend to require more training data and careful task-dependent hyper-parameter search, to avoid overfitting.

Chapter 3

Materials and Methods

The following chapters will delineate the experimental framework employed for the prediction of epileptic seizures using deep learning models and the results that were achieved.

3.1 EEG Dataset

This study utilizes an electroencephalogram (EEG) dataset curated by the Epilepsy and Clinical Neurophysiology Unit at Eugenio Medea IRCCS Hospital, Conegliano, Italy. In particular, the dataset includes recordings from 10 patients suffering from epilepsy (4 male and 6 female), capturing a total of 40 seizure events. EEG recordings were acquired using a standard 20-channel montage, as illustrated in Figure 3.1. This comprehensive electrode placement provides rich spatial information about brain activity. The EEG signal is segmented into

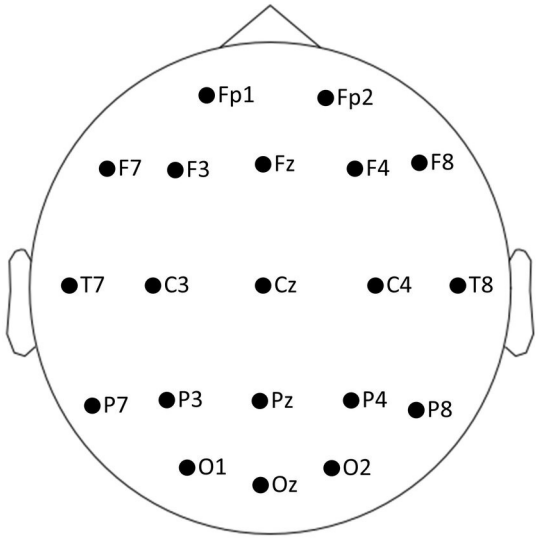


Figure 3.1: Scalp positioning of the 20 common EEG channels used in our data set. Figure adapted from [8].

four distinct states, described in detail within the "EEG Signal" chapter: interictal, preictal, ictal, and postictal. Seizure onset (ictal state) was meticulously marked by clinicians based on electroclinical and video-EEG data. This study focuses on a binary classification task: differentiating between the preictal state (immediately preceding a seizure) and the interictal state, characterized by normal brain activity. Figure 2 visually depicts this classification objective. To achieve this, two classes were defined based on the temporal proximity to a seizure: 1) class 0, which contains EEG signals randomly selected from interictal periods, serving as the baseline (normal) class; 2) class 1: which contains the EEG signals sampled within a 0-30 minute window before seizure onset [8].

3.2 Preprocessing

The first step is the dataset cleaning and preprocessing in order to shape it to the right form to be the input of machine learning and deep learning models for the different prediction tasks.

The whole project is done with Python. The following libraries are used for the preprocessing of the dataset in the initial stage of the project: NumPy, pandas, and MNE-Python.

NumPy is a fundamental Python library that provides powerful tools for numerical computing. In fact, it introduces efficient multidimensional arrays (ndarrays) that are ideal for storing and manipulating large datasets of EEG signals. It forms the foundation for many scientific Python libraries, including MNE-Python [38].

Pandas complements NumPy by providing high-level data structures like Series (one-dimensional labeled arrays) and DataFrames (two-dimensional labeled data structures) [39].

MNE (Minimum Norm Estimation) serves as a cornerstone for various tasks crucial to preparing EEG data for deep learning models. It was specifically designed for the analysis of magnetoencephalography (MEG) and electroencephalography (EEG) data. MNE provides numerous functions for loading EEG data from various file formats commonly used in neuroimaging research, including those generated by different EEG acquisition systems. It allows for the application of various filtering techniques to remove unwanted noise from the EEG signal. This includes band-pass filtering to focus on the frequency range of interest for seizure prediction and eliminating power line interference. MNE also offers functionalities for identifying and removing artifacts from the EEG signal that were mentioned several times in previous chapters [40].

In this work, the EEG signal is segmented into non-overlapping time windows of 5 seconds each. The data is converted from microvolts to volts (V) by multiplying it with a factor of 1e-6. This scaling ensures compatibility with the units expected by MNE. Then a notch filter is applied to specifically remove line noise at frequencies of 50 Hz and 100 Hz, which are common electrical interferences. These frequencies are irrelevant to brain activity and can potentially

mask seizure-related patterns. After that, a band-pass filter allows only frequencies within a specific range to pass through, effectively removing unwanted low-frequency noise (below 1 Hz) and high-frequency components (above 125 Hz) from the signal. The chosen band-pass range (1 Hz to 125 Hz) encompasses the relevant frequency bands associated with brain activity of interest in seizure prediction tasks.

The sampling frequency of the EEG signal is reduced with the process, called downsampling, which aims to achieve a balance between retaining essential information and reducing computational cost. The new sampling frequency is set to 250 Hz. The automatic zero-padding during downsampling is set, ensuring signal integrity.

Following data preprocessing, two distinct labels are assigned to each sample for supervised learning purposes. Samples belonging to the preictal state (between 0 and 30 minutes before a seizure) are designated with a label of "0," while interictal state samples receive a label of "1." Subsequently, all samples undergo reshaping into a uniform matrix format for compatibility with DL models. The resulting matrix dimensions are (20, 1280), where "20" represents the number of EEG channels included in the analysis. Each channel within this matrix holds 1280 data points, corresponding to the temporal resolution of the EEG recording. In the end, the two classes are balanced to have equal number of samples. Following the completion of the initial data preprocessing steps, the resulting dataset, approximately 5 GB in size, is prepared for further analysis. The preparation of the dataset involve additional steps like data normalization, but before that the preprocessed dataset is uploaded to the virtual machine (VM) where the core training of the deep learning models will take place.

3.3 Tools and Libraries

Prior to delving into the specific architectures and training processes of the employed deep learning models, it's crucial to discuss the software and hardware environment that defines this research.

Processing Units

Deep neural networks require significant computational resources to effectively handle the complexity of seizure prediction tasks. The size and dimensionality of the EEG dataset, coupled with the intricate structures of the models themselves (including numerous layers and a vast array of parameters), necessitate processing power beyond that offered by traditional CPUs. This limitation translates to significantly longer training times, hindering the exploration of various model configurations and hyperparameter tuning strategies. To accelerate the training process and facilitate efficient experimentation, leveraging the processing power of Graphics Processing Units (GPUs) is absolutely necessary in projects involving deep

learning. GPUs are specifically designed for parallel processing tasks, making them ideal for handling the computationally intensive operations involved in training DL models.

This project utilizes a Virtual Machine (VM) environment for enhanced flexibility and resource management. Setting up the VM involves installing compatible drivers, configuring the CUDA environment, and ensuring compatibility with the available GPU. Here's a breakdown of the specific software and hardware employed:

- CUDA Version: 12.2
- NVIDIA Driver Version: 535.104.12
- GPU: Tesla V100-SXM2-16GB

The availability and utilization of the GPU are constantly monitored throughout the project. All models, datasets, and computations are moved to the available device, ensuring efficient utilization of GPU resources.

Libraries

The seizure prediction task necessitates the utilization of powerful software libraries. One of the most used deep learning frameworks is PyTorch. It adopts a dynamic computational graph, allowing for the creation and modification of the graph during runtime. This flexibility empowers researchers to experiment with different network architectures and loss functions on the fly. Compared to certain other deep learning frameworks, PyTorch boasts a relatively simpler and more intuitive syntax, making it easier to learn and understand for developers. This clarity in code structure enhances debugging and model interpretation. Also, PyTorch integrates with NVIDIA's CUDA framework, enabling efficient training on GPUs [41].

The core library torch provides fundamental building blocks for constructing deep learning models. It includes functionalities for defining tensors (multidimensional arrays), performing mathematical operations on them, and implementing various neural network layers (e.g., convolutional layers, recurrent layers). This submodule torch.nn houses pre-built neural network modules like nn.Conv2d (convolutional layer), nn.LSTM (Long Short-Term Memory layer), and nn.Linear (fully-connected layer). This torch.optim offers a collection of optimization algorithms, essential for training deep learning models. Algorithms like optim.Adam and optim.SGD (Stochastic Gradient Descent) update the model's weights and biases based on the calculated loss function, progressively improving the model's performance during training. torch.nn.functional: This torch.nn.functional provides various activation functions (e.g., ReLU, Sigmoid) and loss functions (e.g., CrossEntropyLoss). The torch.utils.data facilitates data management for deep learning tasks. Classes like Dataset and DataLoader streamline data loading, shuffling, and batching processes, ensuring efficient

training. The torch.cuda.amp enables Automatic Mixed Precision (AMP) training. AMP leverages a mixed-precision approach to accelerate training without compromising accuracy, particularly beneficial for resource-intensive tasks like seizure prediction with large datasets.

While PyTorch provides the core functionalities for deep learning, Torchvision, a companion library, offers specific tools and functionalities particularly useful for computer vision tasks. However, the concept of "vision" can encompass tasks beyond traditional image recognition. In the context of seizure prediction, EEG data can be visualized as a 2D matrix, where rows represent channels and columns represent time points. Therefore, certain functionalities within Torchvision can be leveraged even for seizure prediction tasks involving EEG data. For instance, torchvision.transforms can be used for data augmentation techniques, which artificially create variations of existing data samples. This can help the model generalize better and prevent overfitting, a common challenge in deep learning. Even though, PyTorch is a popular choice, the deep learning community is diverse, and it is possible to achieve similar results with other frameworks. In fact there can be many other alternatives: Tensorflow, Keras, Google Colab etc. The choice of framework often depends on your specific project requirements and personal preferences.

Another powerful Python library utilized in this project is Scikit-learn, which offers a wide array of tools for machine learning tasks. For instance, the StandardScaler module removes the mean and scales the data to unit variance, ensuring uniform contribution of all features during model training. This is crucial for algorithms sensitive to feature scales. The LabelBinarizer is employed when seizure labels are categorical (e.g., "seizure," "non-seizure"), requiring conversion into binary vectors for classification algorithms. One-hot encoding is commonly utilized for this purpose.

KFold Cross-Validation is a robust method employed to mitigate overfitting by partitioning the data into folds (training and validation sets) for multiple training iterations. It furnishes a more dependable estimate of model performance on unseen data. Another widely used data splitting method provided by Scikit-learn is train-test split, which randomly divides the dataset into test and training sets based on the specified splitting percentage.

Additionally, Scikit-learn offers numerous metrics essential for accurate evaluation of model training. These metrics include confusion matrix, roc auc score, and the general classification report. The confusion matrix provides a visual representation of the model's performance in terms of true positives, true negatives, false positives, and false negatives. The ROC AUC score quantifies the model's ability to differentiate between seizure and non-seizure classes, independent of class distribution, making it preferable over accuracy, particularly for imbalanced datasets. The classification report furnishes detailed precision, recall, F1-score, and support for each class, offering a comprehensive insight into the model's performance [42].

For visualization in project the library Matplotlib is used. Matplotlib is a comprehensive

library for creating static, animated, and interactive visualizations in Python. The pyplot module provides a MATLAB-like interface for creating plots and graphs. It's useful for visualizing data, trends, and patterns, making it essential for exploratory data analysis and communicating results [43].

To maintain control over GPU usage and availability, both Gpustat and GPUtil prove to be highly effective. These tools offer real-time monitoring of GPU resources, encompassing usage, memory allocation, and temperature.

For efficient hyperparameter optimization, the Optuna library is employed.

Data splitting

The management of data is facilitated by the MyDataset class, which is tailored for the loading and preprocessing of data for training machine learning models. This class inherits from the Dataset class provided by the PyTorch library, enabling seamless integration with PyTorch's data loading utilities. It comprises various methods that process all samples, converting them into a suitable format (numpy array), and returns them alongside their corresponding labels. Optionally, data transformations specified by the transform parameter are applied to the samples. In this project, all samples are transformed into tensors. The crucial step of the project is data splitting. Imagine training a model on a single set of EEG recordings. The model might memorize specific patterns within that data, leading to high accuracy. However, when presented with a new recording with slightly different characteristics, the model might struggle because it hasn't truly learned the underlying patterns of seizures. Splitting the dataset into training and testing sets offers a solution. This is where cross-validation (CV) techniques come into play. CV includes various techniques that involve splitting the dataset into multiple subsets, known as folds, where each fold is used as a validation set while the remaining folds are used for training. Some of the most popular CV methods are:

- **K-Fold Cross-Validation:** The dataset is divided into k equal-sized folds. The model is trained k times, each time using $k-1$ folds for training and the remaining fold for validation. This process is repeated k times, with each fold used exactly once as the validation set. The performance metrics are averaged over all k iterations. The optimal value for k depends on the size and complexity of your dataset. Common choices include $k = 5$ or $k = 10$. Higher k provides a more stable estimate of performance but uses less data for training in each fold. This might be less suitable for smaller datasets. Lower k uses more data for training but leads to a less stable performance estimate.

- **Stratified K-Fold Cross-Validation:** Similar to k-fold cross-validation, but it ensures that each fold preserves the proportion of samples from each class. This is particularly useful for imbalanced datasets where certain classes are underrepresented.

- **Leave-One-Out Cross-Validation (LOOCV):** Each data point in the dataset is sequentially held out as the validation set, while the remaining data points are used for training. This process is repeated for each data point, resulting in n iterations for a dataset with n samples. LOOCV maximizes the use of available data for training and validation but can be computationally expensive for large datasets.

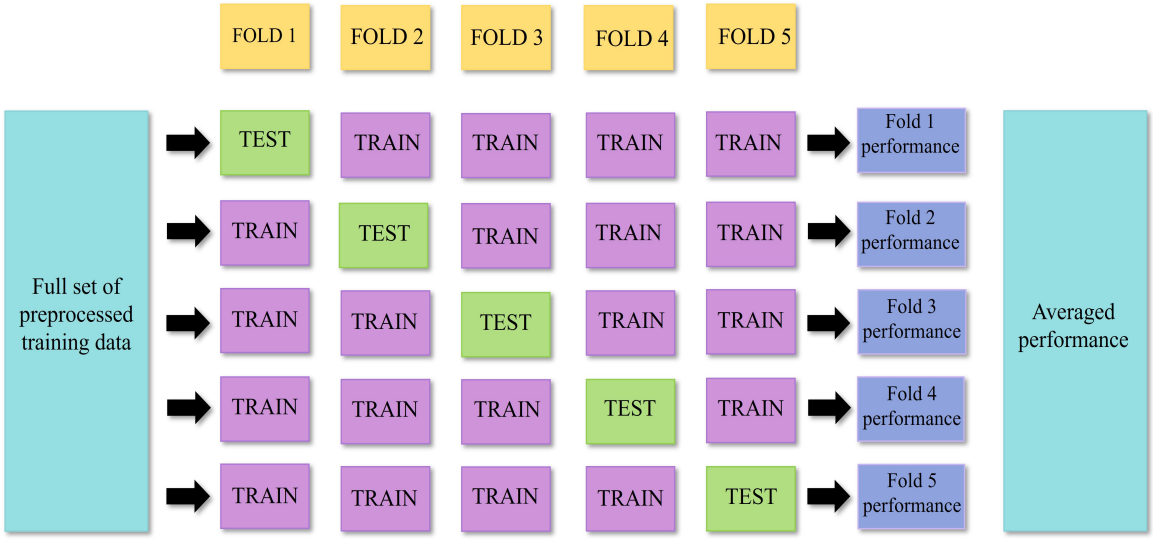


Figure 3.2: 5-fold Cross Validation data splitting

A variant of traditional cross-validation methods is Random cross-validation (RCV), which randomly selects data points for the training and validation sets without any predefined structure. One common RCV technique is random train-test split, where the dataset is randomly partitioned into a training set and a validation set based on a specified ratio. For example, researchers may choose to allocate 80% of the data to the training set and the remaining 20% to the validation set. This approach is straightforward to implement and computationally efficient, making it suitable for large datasets.

Even though, LOO offers a seemingly ideal approach for model evaluation, its computational burden often makes it impractical for large and complex datasets. That is why, for this project the 5-fold splitting is chosen. F-fold CV maximizes the usage of the valuable EEG dataset. All data points are used for both training and testing, providing a more comprehensive evaluation of the model's performance. It perfectly balances efficiency with a reasonable estimate of generalizability.

However, it is important to keep in mind that K-Fold CV might introduce a slight bias towards overfitting. The model, during each fold, encounters a subset of the entire dataset. This might lead the model to potentially learn idiosyncrasies of the training data within that fold, which might not generalize well to unseen data. Studies have shown that models trained with K-Fold CV can exhibit higher accuracy metrics compared to LOO CV, but this might not always translate to superior performance on truly unseen data.

The overfitting problem of k-fold cross validation method is strictly linked to the nature of EEG signals themselves, which are inherently time-series data. EEG signals collected from patients may exhibit temporal correlations, especially within the same class (e.g., preictal or interictal). For instance, preictal signals may show certain patterns or trends leading up to a seizure event, while interictal signals represent normal brain activity. When using k-fold CV to evaluate seizure prediction algorithms, the temporal correlation within EEG signals can violate the assumption of independence between samples in different folds. It happens when samples within classes are collected in close proximity in time without randomization with other classes. f the data is not randomized across folds, each fold may contain samples from the same patient or time period, leading to biased model evaluation. This temporal correlation can impact the performance estimates of seizure prediction models obtained through k-fold CV. The classifier may learn to exploit these temporal correlations to make predictions, leading to overly optimistic performance estimates during cross-validation. To address this issue, alternative cross-validation strategies such as block-wise or trial-wise CV can be considered. These methods ensure that samples from the same patient or time period remain together in either the training or test set, reducing the impact of temporal correlation on model evaluation [44].

By acknowledging the potential overfitting bias and employing strategies like regularization techniques or different variations of k-fold CV, it is possible to find equilibrium between efficiency and time optimization.

Normalization

Normalization ensures all features are on a common scale, allowing machine learning models to focus on the underlying patterns relevant to seizure detection rather than being swayed by differences in signal amplitude. Z-score normalization, also known as standardization, is a widely used technique for this purpose. Z-score normalization transforms each data point in an EEG signal by subtracting the mean (average) and then dividing by the standard deviation of the entire training set. This results in a new set of values with a mean of 0 and a standard deviation of 1. Here is the z-score equation:

$$z = \frac{x - \mu}{\sigma}$$

In the above formula, z stands for z-score; x is the random variable, μ stands for mean and σ represents standard deviation.

Unlike methods like min-max normalization that simply scale data to a specific range, Z-score normalization considers both the mean and standard deviation. This is crucial because EEG signals can exhibit variations in both baseline activity and overall magnitude across different subjects or recording sessions. Z-score normalization effectively addresses these

variations by transforming the data into a standard distribution.

EEG signals are often characterized by the relative differences between brain regions rather than absolute power values. Z-score normalization highlights these relative changes by removing the influence of individual subject or session-specific biases [45]. This allows the model to better identify patterns in the relationships between different EEG channels, which might be key for seizure prediction.

It's important to note that for robust Z-score normalization, the mean and standard deviation used for transformation should be calculated exclusively on the training set. This ensures the test set remains completely unseen by the model. Using the training set's statistics prevents the model from adapting to the test data's specific characteristics, ultimately leading to a more reliable evaluation of its generalizability to unseen EEG recordings.

In simpler terms, imagine the training set as a "practice test" for the model. The model learns the "average" and "spread" (mean and standard deviation) of EEG signals from the training data. Then, when encountering a new, unseen EEG recording (the test set), it can analyze the relative changes within that recording based on the learned "average" and "spread" from the training set, allowing for more accurate seizure prediction.

This is how it is done in this project as well: inside of the training loop of each out of 5 folds data loaders are created for the training, validation, and testing sets and the mean and standard deviation of the training data are calculated using a loop over the batches in the training data loader. The mean and standard deviation are computed separately for each feature dimension (channel) of the input data. The mean and std tensors are accumulated by summing up the mean and standard deviation of each batch across all feature dimensions. After iterating over all batches, the accumulated mean and standard deviation values are divided by the total number of batches to obtain the average mean and standard deviation across the entire training dataset. After computing the mean and standard deviation, the training data is normalized by subtracting the mean and dividing by the standard deviation. This normalization process is performed batch-wise within the training loop. Before feeding the data into the model for training, each batch is normalized using the mean and standard deviation computed from the training dataset. Similarly, the validation and test data are also normalized using the mean and standard deviation calculated from the training dataset. However, the mean and standard deviation are applied without dividing by the number of batches since normalization is done at once for the entire test set.

DataLoader

As was mentioned before, samples are loaded in small batches before being normalized and given to the model as input for training. This process is possible with Dataloader, which is a fundamental component in many DL frameworks, including PyTorch, and it allows to

manage efficiently large quantities of data. DataLoader also randomly shuffles the data, to prevent the model from learning spurious patterns based on the data order and it can apply preprocessing and transformation functions to the data before creating batches. This functionality is essential for tasks like normalization, scaling, feature extraction, and data type conversion.

Loading data in batches allows efficient use of memory resources, especially when dealing with large datasets such as EEG recordings. Instead of loading the entire dataset into memory at once, which may not be feasible due to memory constraints, batches enable processing smaller chunks of data sequentially. This memory-efficient approach enables the training process to handle datasets that exceed the available memory capacity of the hardware, preventing potential memory overflow issues and improving overall system stability.

Batch processing can significantly speed up the computation during training. By processing data in parallel across multiple batches, modern hardware, such as GPUs, can leverage parallel computing capabilities to perform computations faster compared to processing data sequentially.

Loading data in batches introduces variability and randomness into the training process. Each batch contains a different subset of data samples, allowing the model to encounter diverse examples during training. This variability helps prevent the model from memorizing specific patterns or sequences in the data, encouraging the model to learn robust features and patterns that are more likely to generalize well to new EEG recordings, including those with different patient characteristics or recording conditions.

Batching data also facilitates dynamic learning during the training process [46]. As the model updates its parameters based on each batch's loss and gradients, it can adapt and adjust its learning strategy iteratively. This dynamic learning approach allows the model to fine-tune its parameters gradually, improving its ability to capture complex patterns and relationships in the EEG data over successive batches.

Additionally, batch processing aligns well with other optimization techniques such as mini-batch gradient descent, which further enhances the training process by efficiently updating model parameters based on mini-batch gradients. The choice of the correct batch size is influenced by numerous factors such as model complexity, available time, computational power, and the characteristics of the dataset. The GPUs are able to handle larger batch sizes, which can be beneficial especially in DL models with complex architecture, while if computational resources are limited or if training needs to be performed on a CPU, smaller batch sizes are preferable to avoid memory constraints and ensure smooth processing without excessive resource consumption. Batch size also impacts training time. Smaller batch sizes may require more iterations to process the entire dataset, leading to longer training times. Conversely, larger batch sizes can accelerate training by processing more data in each iteration and can lead to smoother optimization trajectories and faster convergence, but they

may require more memory and risk overfitting if not properly regularized. So, the common approach to correctly choose the batch size is to perform empirical testing and validation with different batch sizes. As a general guideline, starting with a batch size of 32 to 128 is common for many deep learning tasks. This range balances computational efficiency and model generalization. In this project, the optimal choice for all models is a batch size of 32, which avoids memory issues and requires a reasonable amount of time with the use of a GPU.

3.4 Deep Learning Models Architecture

This chapter focuses on the core components of our seizure prediction system: the deep learning architectures used to analyze EEG signals. We will explore three distinct models, each offering strengths in feature extraction and classification for seizure prediction task. The first two models, DeepConvNet and EEGNet, represent established choices for seizure prediction from EEG data. Their convolutional neural network (CNN) architectures have proven successful in learning relevant spatial and temporal features from these signals. The third model, ResNet68, employs a different approach. Traditionally used for image classification, its residual learning architecture holds potential for adaptation to EEG signal analysis. ResNet blocks, a key component, enable the network to learn from the cumulative residuals of its inputs, facilitating the understanding of complex relationships within the data. However, adapting ResNet68 for EEG data presented a challenge, as it was originally designed for 2D image inputs.

The following sections will provide a more detailed analysis of each model's architecture. The specific configurations of DeepConvNet and EEGNet, the functionalities of ResNet68's residual learning blocks and the rationale behind the modifications made for EEG data processing will be explored. By comparatively analyzing these three architectures, it will be possible to gain valuable insights into their suitability for seizure prediction from EEG signals.

EEGNet

In the realm of seizure prediction from EEG signals, the emergence of EEGNet marked a significant advancement. Unlike generic Deep Convolutional Neural Networks (CNNs) commonly used for image analysis, EEGNet boasts a unique architecture specifically designed to excel at processing the inherent characteristics of EEG data. Let's delve into its origins, strengths, and how it optimizes seizure prediction tasks.

Original EEGNet

The EEGNet architecture development is linked to the use of CNNs in EEG-based Brain-Computer Interfaces (BCIs). BCIs allow direct communication with computers by utilizing neural activity as the control signal, commonly measured through EEG signals. Traditional BCI paradigms require specific feature extractors and classifiers, limiting their application to particular EEG control signals. CNNs, known for automatic feature extraction in computer vision and speech recognition, have been applied to EEG-based BCIs with success, but their generalization across different paradigms remains unclear. Furthermore, traditional CNNs with numerous convolutional layers can be computationally expensive to train, especially for real-time applications where faster processing is crucial.

EEGNet addresses this challenge by introducing depthwise and separable convolutions to construct a compact CNN model tailored for EEG signals. These convolutions encapsulate well-known EEG feature extraction concepts, such as spatial filtering and filterbank construction, while reducing the number of trainable parameters compared to existing approaches [47].

A standard convolution operation involves applying a filter (kernel) that slides across the input data, extracting features based on element-wise multiplication and summation. While effective, this approach can become computationally expensive, especially for processing high-dimensional data like images. Depthwise separable convolutions address this challenge by decomposing the standard convolution into two separate steps: Depthwise Convolution and Pointwise Convolution.

In depthwise convolution step a separate filter is applied to each input channel independently, extracting features specific to that region's activity. There is a depth parameter D, which determines how many spatial filters are learned for each feature map. Unlike regular convolutions, depthwise convolutions are not connected to all previous feature maps, this way they reduce the number of parameters needed to be learned.

In pointwise convolution (1x1 convolution) step a 1x1 convolution is applied across the feature maps generated by the depthwise convolution. This 1x1 kernel acts as a linear combination, reducing the number of feature maps and introducing non-linearities (through activation functions) if desired. To gain a clearer understanding of this intricate process, one can refer to Figure 3.4. In the original EEGNet model architecture there are several blocks designed for feature extraction and classification (Figure 3.5). Block 1 involves two consecutive convolutional steps, including depthwise convolution for spatial filtering and capturing frequency-specific spatial filters. Two sequential 2D convolutional steps are performed. First, F1 2D convolutional filters of size (1, 64) are applied to capture the EEG signal's band-pass frequencies. The length of the filter is chosen to be half the sampling rate, allowing capturing frequency information above 2 Hz. This step helps in initial feature extraction. A depthwise

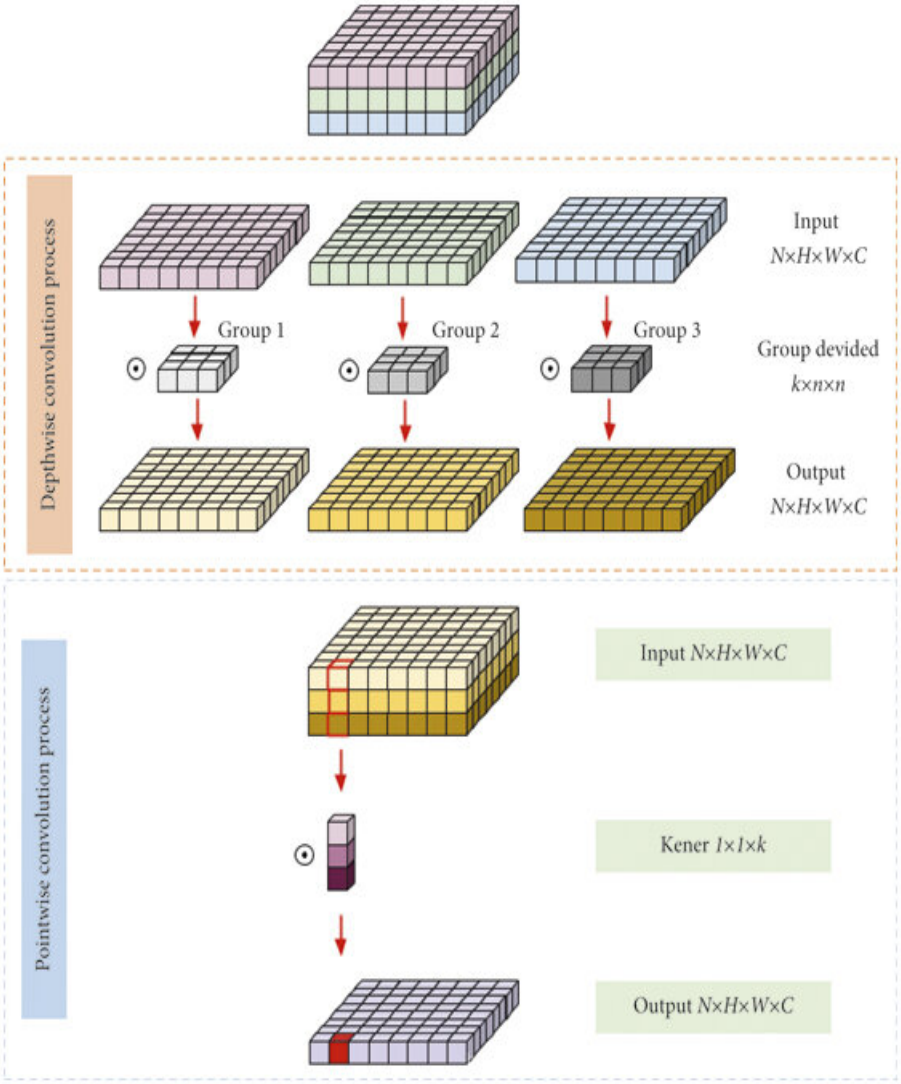


Figure 3.3: Depthwise and pointwise convolution processes.

convolution of size (C, 1) is applied to learn spatial filters specific to each temporal filter. Batch normalization is applied along the feature map dimension, followed by the Exponential Linear Unit (ELU) activation function, aiding in regularization and non-linearity. Dropout regularization technique is applied to prevent overfitting. A dropout probability of 0.5 is set for within-subject classification and 0.25 for cross-subject classification. An average pooling layer of size (1, 4) is used to reduce the sampling rate of the signal to 32 Hz, aiding in dimensionality reduction. In the second block a separable convolution is applied, consisting of a depthwise convolution followed by F2 pointwise convolutions (1, 1). This operation helps in further feature extraction and dimensionality reduction, decoupling the relationship within and across feature maps. Another average pooling layer of size (1, 8) is applied for additional dimensionality reduction. In the classification block, the features extracted are

directly passed to a softmax classification layer with N units, where N represents the number of classes in the data. No dense layer for feature aggregation is used prior to the softmax layer, reducing the number of free parameters in the model.

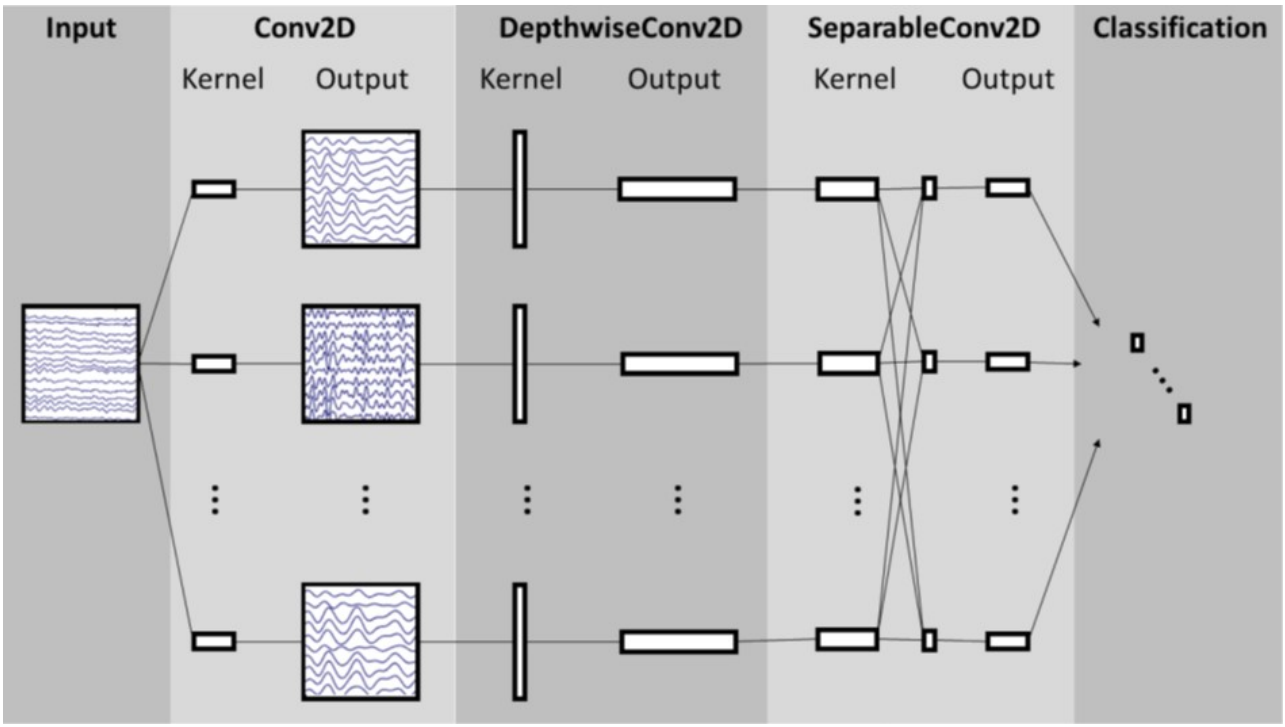


Figure 3.4: Original EEGNet architecture. The lines represent connections between the input and output feature maps formed by convolutional kernels. Initially, the network applies a temporal convolution to grasp frequency patterns. Then, it employs a depthwise convolution, linked to each feature map independently, to capture frequency-specific spatial patterns. Finally, the separable convolution combines a depthwise convolution for learning individual feature map summaries temporally, followed by a pointwise convolution to effectively blend the feature maps. Figure adapted from [47].

EEGNet Architecture Adapted for This Project

The EEGNet architecture, adapted for this project, presents some subtle differences respect to the original model, but they share the same design principles and architectural choices are characteristical for EEGNet-like models (Figure 3.6). The first convolutional layer initiates the feature extraction process by convolving input EEG data with 32 filters. It accepts 20 input channels, corresponding to EEG electrode channels, and employs a kernel size of (1, 51) with a stride of (1, 1) and padding of (0, 25). Batch normalization is applied to enhance convergence, and the layer is devoid of biases. Notably, modifications in padding are made to accommodate the input signal dimensions.

Following the first convolutional layer, the depthwise convolutional layer captures spatial relationships within EEG feature maps without increasing model complexity. It takes the

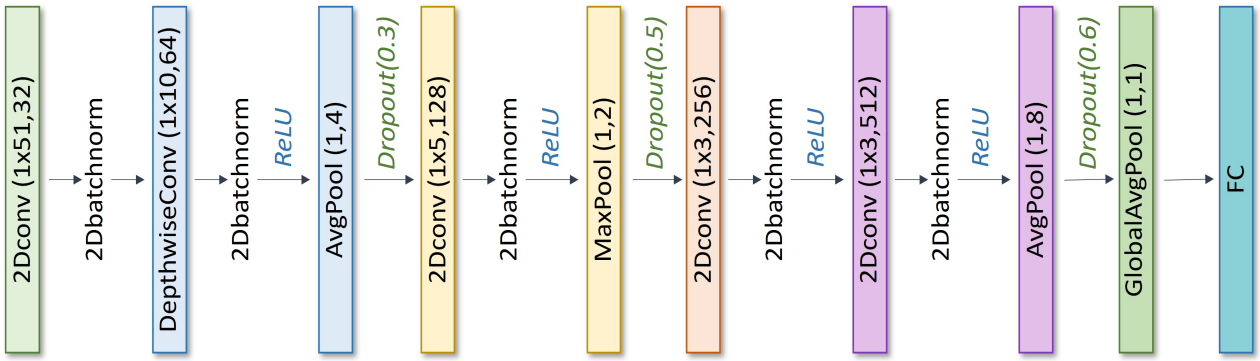


Figure 3.5: EEGNet architecture developed for this project.

output of the previous layer, consisting of 32 channels, and generates 64 output channels. The layer utilizes a kernel size of (1, 10) with a stride of (1, 1) and padding of (0, 5). Activation is introduced through ReLU, and batch normalization is applied for stable training dynamics. Additionally, dropout is incorporated to mitigate overfitting, a departure from the original EEGNet.

Subsequently, an average pooling layer downsamples the spatial dimensions of feature maps with a kernel size of (1, 4) and a stride of (1, 4), enhancing computational efficiency. A dropout layer with a dropout probability (p) of 30% follows the AvgPool.

Convolutional Layer 3 further abstracts and refines features extracted from previous layers. With an input of 64 channels, it produces 128 output channels using a kernel size of (1, 5) and padding of (0, 2). Activation, batch normalization, and dropout are employed to enhance feature representation and prevent overfitting.

A max-pooling layer downsamples feature maps, retaining only the maximum values, aiding in hierarchical feature extraction with a kernel size of (1, 2) and a stride of (1, 2). It is followed by a dropout layer with p = 50%. Convolutional layers 4 and 5 deepen the network's representation capabilities. Convolutional layer 4 accepts 128 input channels and produces 256 output channels with a kernel size of (1, 3) and padding of (0, 1). Convolutional layer 5 further expands the depth with 256 input channels and 512 output channels, utilizing the same kernel size and padding as Convolutional layer 4.

The second average pooling layer further reduces feature map dimensions with a kernel size of (1, 8) and a stride of (1, 8), focusing on essential information. Also each of the last two convolutional layers is followed by a dropout layer with p = 60%.

A global average pooling (GAP) layer aggregates spatial information across feature maps, facilitating global context understanding, resulting in an output size of (1, 1). Global Average Pooling produces a fixed-size representation regardless of the input size. This is particularly useful in scenarios where the input size can vary, as it ensures that the network's output

has a consistent dimensionality. This fixed-size representation simplifies the subsequent fully connected layers' architecture and makes the model more robust to input size variations. GAP reduces the spatial dimensions of each feature map to a single value by computing the average. This reduction helps in focusing on the most essential information within each feature map while discarding less relevant spatial details. This can be especially beneficial in capturing the global context of the input, making the model more invariant to spatial translations. By taking the average across the entire feature map, GAP introduces a degree of translation invariance. This means that the model becomes less sensitive to the precise location of a feature within the spatial dimensions. Each value in the output of the GAP corresponds to the average activation of a feature map, providing a form of attention to different parts of the input. This can be valuable for understanding which features contribute most to the final decision. Finally, a fully connected layer performs classification based on learned features. With 512 input features, it produces 2 output features using softmax activation for probability distribution generation.

Besides the differences in kernel size, such as (1, 51) instead of (1, 64) for the first convolutional layer, which align with the requirement of spatial convolution to be (1xN), there are additional changes incorporated into the model. These changes include the introduction of dropout layers after each convolution, as opposed to having only one initial dropout layer, and adjustments in padding. These modifications were made based on experimental trials, particularly with Optuna, resulting in improved model performance.

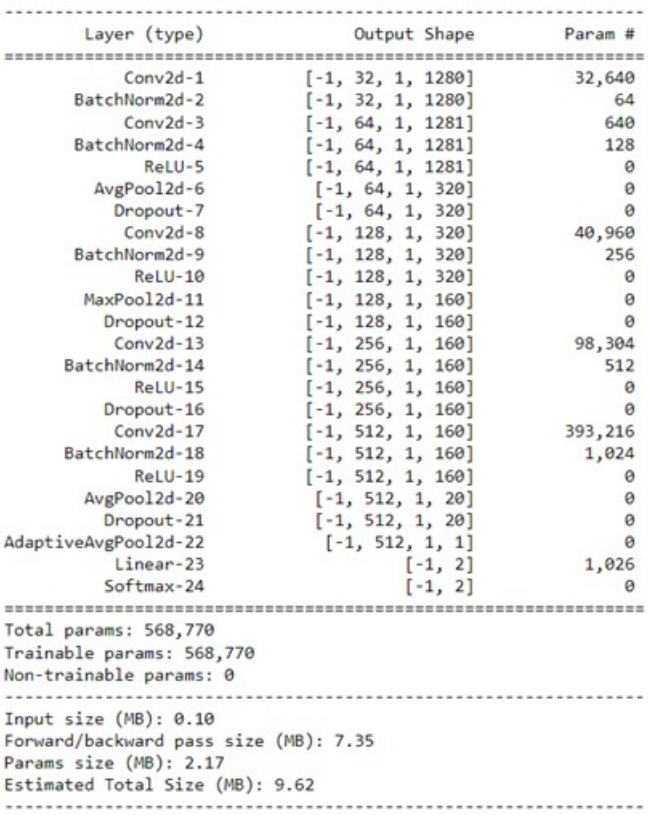
With this updated architecture and the previously discussed input shape of (20, 1280), the total number of parameters for the model amounts to 568,770. Full details about the network architecture and the number of parameters can be found in Figure 3.7.

DeepConvNet

DeepConvNet (Deep Convolutional Network) is a variation of CNN specifically designed for processing sequential data and which has emerged as a popular choice for seizure prediction tasks due to its effectiveness in extracting relevant features from EEG signals. The exact origin of DeepConvNet is difficult to pinpoint as it's not a single, specific architecture. It's a general term encompassing various CNN configurations with multiple convolutional layers stacked on top of each other. DeepConvNets emerged as a natural progression, allowing for deeper architectures to learn even more complex patterns in data.

Why DeepConvNet?

EEG signals are inherently complex, containing both spatial (across different brain regions) and temporal (over time) information. DeepConvNets excel at capturing these intricacies through their convolutional layers. By stacking multiple convolutional layers with



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 1, 1280]	32,640
BatchNorm2d-2	[-1, 32, 1, 1280]	64
Conv2d-3	[-1, 64, 1, 1281]	640
BatchNorm2d-4	[-1, 64, 1, 1281]	128
ReLU-5	[-1, 64, 1, 1281]	0
AvgPool2d-6	[-1, 64, 1, 320]	0
Dropout-7	[-1, 64, 1, 320]	0
Conv2d-8	[-1, 128, 1, 320]	40,960
BatchNorm2d-9	[-1, 128, 1, 320]	256
ReLU-10	[-1, 128, 1, 320]	0
MaxPool2d-11	[-1, 128, 1, 160]	0
Dropout-12	[-1, 128, 1, 160]	0
Conv2d-13	[-1, 256, 1, 160]	98,304
BatchNorm2d-14	[-1, 256, 1, 160]	512
ReLU-15	[-1, 256, 1, 160]	0
Dropout-16	[-1, 256, 1, 160]	0
Conv2d-17	[-1, 512, 1, 160]	393,216
BatchNorm2d-18	[-1, 512, 1, 160]	1,024
ReLU-19	[-1, 512, 1, 160]	0
AvgPool2d-20	[-1, 512, 1, 20]	0
Dropout-21	[-1, 512, 1, 20]	0
AdaptiveAvgPool2d-22	[-1, 512, 1, 1]	0
Linear-23	[-1, 2]	1,026
Softmax-24	[-1, 2]	0

Total params: 568,770
Trainable params: 568,770
Non-trainable params: 0

Input size (MB): 0.10
Forward/backward pass size (MB): 7.35
Params size (MB): 2.17
Estimated Total Size (MB): 9.62

Figure 3.6: EEGNet Model Architecture Summary.

varying filter sizes and strides, DeepConvNet can learn patterns at different scales, from localized activity to broader trends across multiple brain regions. DeepConvNet has shown competitive performance compared to traditional machine learning models and other deep learning architectures for seizure prediction. Studies have reported seizure prediction accuracy ranging from 75% to 90% depending on the specific DeepConvNet architecture, dataset characteristics, and evaluation metrics used [48].

DeepConvNet Architecture

There isn't a single, universally accepted DeepConvNet architecture, but it typically involves multiple convolutional layers stacked on top of each other, but the exact configuration (number of layers, kernel sizes, stride lengths, etc.) can vary depending on the specific application and dataset. As a reference to build a DeepConvNet it is possible to take the model proposed by Schirrmeister et al. in 2017 [49] and its general architecture, which is also applied in this project, is represented at Figure 3.8. The model begins with a series of convolutional layers. The initial convolutional layer processes input data with 20 channels using 25 filters, each with a kernel size of (1, 5). Subsequent layers progressively increase the number of output channels, reaching 50, 100, and finally 200 output channels. All convolutional layers share a common configuration, employing a kernel size of (1, 5), a stride

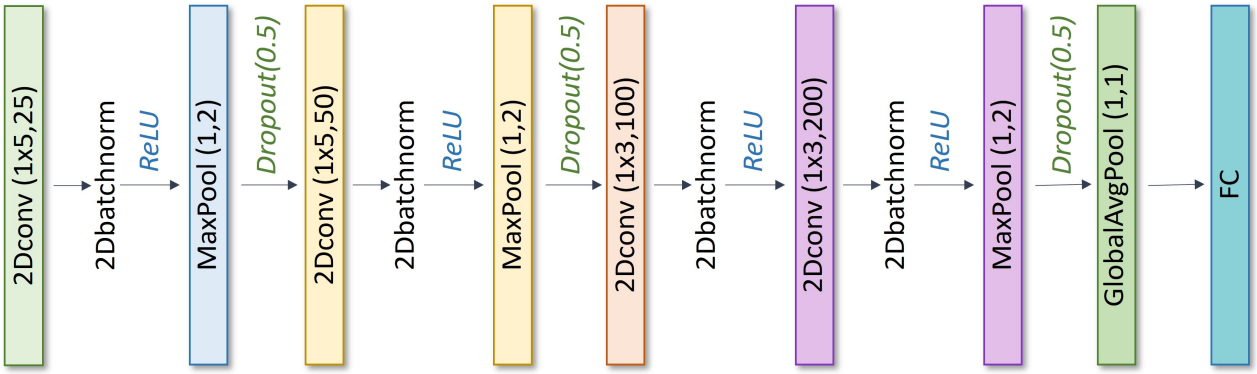


Figure 3.7: DeepConvNet architecture developed for this project.

of (1, 1), and padding of (0, 0). Following each convolutional operation, the Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity.

After each convolutional layer, batch normalization is utilized to standardize and stabilize the activations. Additionally, a max-pooling layer is employed with a kernel size of (1, 2) and a stride of (1, 2) to downsample the feature maps. Dropout layers with a dropout probability of 50% are inserted following each max-pooling layer. This strategy helps prevent overfitting by randomly deactivating a fraction of input units during training.

At the end of the architecture, a global average pooling layer aggregates spatial information across the entire feature map, generating a single feature vector for each channel. Subsequently, a fully connected (linear) layer processes the extracted features to predict the output classes. With 200 input features, this layer produces 2 output features using softmax activation to generate a probability distribution over the output classes.

Compared to EEGNet, this architecture is less complex, with a total of 134,902 trainable parameters. Consequently, it is expected that this model will require less time to train. Full details about the network architecture and the number of parameters can be found in Figure 3.9.

ResNet

Residual Networks (ResNets) have garnered significant attention in seizure predicting field due to their ability to tackle challenging tasks involving complex data patterns. However, the application of ResNets to raw EEG signals for seizure prediction presents a unique challenge.

According to results from the study by He et al. [50] ResNets revolutionized the field of deep learning. Traditional CNNs often suffer from the vanishing gradient problem, where gradients become too small during backpropagation, hindering the training of deeper networks. ResNets elegantly address this issue by introducing residual connections. These connections allow the network to learn from the sum of its inputs and the outputs of previous layers,

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 25, 1, 1276]	2,500
BatchNorm2d-2	[-1, 25, 1, 1276]	50
ReLU-3	[-1, 25, 1, 1276]	0
MaxPool2d-4	[-1, 25, 1, 638]	0
Dropout-5	[-1, 25, 1, 638]	0
Conv2d-6	[-1, 50, 1, 634]	6,250
BatchNorm2d-7	[-1, 50, 1, 634]	100
ReLU-8	[-1, 50, 1, 634]	0
MaxPool2d-9	[-1, 50, 1, 317]	0
Dropout-10	[-1, 50, 1, 317]	0
Conv2d-11	[-1, 100, 1, 313]	25,000
BatchNorm2d-12	[-1, 100, 1, 313]	200
ReLU-13	[-1, 100, 1, 313]	0
MaxPool2d-14	[-1, 100, 1, 156]	0
Dropout-15	[-1, 100, 1, 156]	0
Conv2d-16	[-1, 200, 1, 152]	100,000
BatchNorm2d-17	[-1, 200, 1, 152]	400
ReLU-18	[-1, 200, 1, 152]	0
MaxPool2d-19	[-1, 200, 1, 76]	0
Dropout-20	[-1, 200, 1, 76]	0
AdaptiveAvgPool2d-21	[-1, 200, 1, 1]	0
Linear-22	[-1, 2]	402
Softmax-23	[-1, 2]	0

Total params: 134,902
Trainable params: 134,902
Non-trainable params: 0

Input size (MB): 0.10
Forward/backward pass size (MB): 3.82
Params size (MB): 0.51
Estimated Total Size (MB): 4.44

Figure 3.8: DeepConvNet Model Architecture Summary.

facilitating better gradient flow and enabling the training of significantly deeper architectures.

However, the main issue of the ResNet application on EEG signal is that ResNet is typically designed for image analysis. Several studies [51, 52] have utilized ResNets for seizure prediction, but they often transform EEG signals into image-like representations (e.g., spectrograms) to leverage pre-trained ResNet models. While effective, this approach introduces an additional processing step, which is also computationally and memory demanding.

A promising alternative approach, as demonstrated by Diyuan Lu et al. [53], involves applying ResNets directly to raw EEG signals. This study achieved an impressive accuracy of 91.8% in seizure prediction, highlighting the potential of ResNets when tailored for this specific application.

ResNet68 architecture

ResNets achieve their impressive depth by stacking multiple residual blocks one after another. The specific configuration of these blocks defines the overall architecture. The most common variation is bottleneck block, that utilizes 1x1 convolutional layers to reduce the dimensionality of feature maps before and after the main 3x3 convolutions. This reduces computational cost while maintaining feature representation. The fundamental building block of a ResNet is the residual block. This ingenious design allows the network to learn from the

sum of its inputs (X) and the outputs (F(X)) of a processing pathway within the block. A residual block includes: batch normalization, applied before and after the processing pathway to stabilize the learning process, ReLU, multiple convolutional layers stacked together to extract features at different scales and shortcut connections, which directly adds the original input (X) to the output (F(X)) of the processing pathway. Figure 3.10 allows to understand better the complex ResNet architecture. ResNet variants are primarily distinguished by two

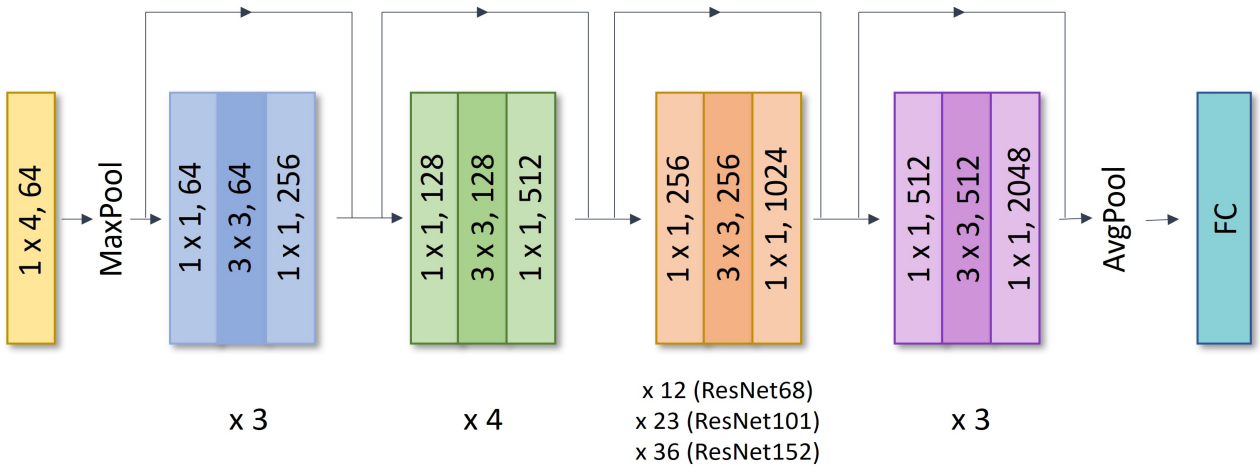


Figure 3.9: General ResNet architecture.

factors: depth and width. Depth is characterized by the total number of stacked residual blocks. Deeper networks can potentially learn more complex patterns but require more training data and computational resources. ResNet variants are often denoted as "ResNet-X," where X represents the total number of layers after the first convolutional layer and the final fully connected layer. For instance, ResNet-68 has 66 residual blocks (excluding the initial and final layers). Another distinguishing feature id width, which represents the number of channels (feature maps) within each residual block. Wider networks can potentially capture more intricate features but always come at the cost of increased memory usage and computational demands.

For the project a balanced variant between computational demands and feature learning capability is chosen, which is ResNet68. The input to ResNet68 is a multi-channel EEG signal, in this case with 20 channels. The initial convolutional layer, processes this input, employing a kernel size of (1, 4) to extract features along the time axis and padding of (0, 1) to preserve spatial dimensions. This layer outputs feature maps with 64 channels.

Following the first convolutional layer, batch normalization is applied, followed by a ReLU activation function (relu). A max-pooling layer (maxpool) with a kernel size of (1, 2) and stride of (1, 2) is then utilized to down-sample the feature maps along the time axis, facilitating spatial abstraction and increasing the receptive field.

The subsequent layers (layer1, layer2, layer3, layer4) consist of multiple residual blocks

stacked together. These layers progressively increase the number of channels and down-sample the feature maps to capture hierarchical features. Specifically, layer1 consists of 3 blocks, layer2 consists of 4 blocks, layer3 consists of 12 blocks, and layer4 consists of 3 blocks. The number of blocks in each layer is defined by the layers argument passed to the ResNet constructor. Each residual block within a layer consists of three convolutional layers with kernel sizes of (1, 1), (3, 3), and (1, 1), respectively. These layers are followed by batch normalization and ReLU activation functions (relu). The first convolutional layer may incorporate a down-sampling operation if the stride is not equal to 1 or the number of input channels is not equal to the number of output channels multiplied by the expansion factor. This operation uses a 1x1 convolutional layer and batch normalization.

The expansion factor, is set to 4, indicating that the output channels of the last convolutional layer are four times the number of output channels specified for the block. The output of each residual block is obtained by adding the input to the output of the last convolutional layer after applying the appropriate down-sampling operation, if necessary. This residual connection facilitates gradient flow and aids in training deeper networks.

Finally, the global average pooling layer aggregates spatial information across the entire feature map, resulting in a single feature vector for each channel. This vector is then passed through a fully connected layer to produce the final output logits, which are subsequently used for classification.

In this project, all model parameters were trained from scratch, without utilizing any pre-trained models. As a result, the total number of trainable parameters amounts to 30,246,722.

Chapter 4

Experimental Design

This chapter focuses on how the deep learning models are trained to predict seizures. There will be an explanation of what is used to guide the model's learning and how its performance is measured on the seizure prediction task.

4.1 Training Process

Weights initialization

The first step in the training process is weights initialization. Each connection between neurons, in a deep learning model imagined as a complex network of neurons, has a weight that determines the strength of its influence. These weights are essentially the "knobs" the model adjusts during training to learn patterns in the data. However, starting with all weights set to zero, the network might get stuck in a bad spot where neurons never fire or always fire together. This is because the initial gradients might be very small or all the same, hindering the learning process.

To address this issue, weight initialization techniques, that assign initial values to these weights in a strategic manner, are employed. One popular technique is Xavier initialization, implemented in the code using nn.init.xavier-uniform, which is a function from the PyTorch library. Xavier initialization aims to set the initial weights in a range that allows gradients to flow effectively through the network during training. This helps avoid the vanishing gradient problem or the exploding gradient problem (gradients becoming too large). Xavier initialization ensures that the activation variances (how spread out the activations are) are roughly the same across different layers in the network. This promotes faster and more stable learning.

After that, as it was described before, the dataset is divided into 5 folds for cross-validation. The training loop is then repeated 5 times, once for each fold. During each iteration, the data is loaded using a DataLoader in batches. Additionally, Z-score normal-

ization is calculated on the training set and applied to the data before training begins. The training process starts by feeding the training set to the model, followed by the validation set. The model's performance is evaluated using the accuracy metric on the test set after each fold.

Adam optimizer

The model is trained using gradient descent optimization techniques, specifically the Adam optimizer, with a specified learning rate schedule. The Adam optimizer (Adaptive Moment Estimation) is a widely used and powerful algorithm for optimizing the weights of deep learning models. It addresses some of the shortcomings of its predecessors, particularly Stochastic Gradient Descent (SGD), by incorporating adaptive learning rates for each parameter. Unlike SGD's fixed learning rate, Adam estimates an individual learning rate for each parameter based on historical gradients. This allows the optimizer to adjust the learning rate dynamically for different parameters, accelerating convergence in some directions while preventing oscillations in others. Also, the optimizer incorporates momentum, similar to SGD with momentum, to accumulate past gradients and provide a smoother update direction. However, Adam utilizes decaying momentum terms to prevent vanishing gradients in situations with long chains of dependencies. Adam is less sensitive to the initial learning rate compared to SGD. It can often find good solutions even with a wider range of learning rate choices. The main hyperparameters of the Adam optimizer are:

- Learning Rate (α): The learning rate determines the step size used to update the model parameters during optimization. It is a crucial hyperparameter that controls the speed and stability of the training process. In this project, the initial learning rate value is set to 0.0001, a choice made after careful consideration of empirical observations and subsequent Optuna optimization.
- Beta1 (β_1): This hyperparameter controls the exponential decay rate for the first moment estimate (mean) of the gradients. It is typically set to 0.9 by default.
- Beta2 (β_2): Beta2 controls the exponential decay rate for the second moment estimate (uncentered variance) of the gradients. It is typically set to 0.999 by default.
- Epsilon (ϵ): Epsilon is a small constant added to the denominator to prevent division by zero and stabilize the optimization process. It is typically set to a small value such as $1e^{-8}$ by default.

Loss Function

For this project the loss function is established using criterion = nn.BCEWithLogitsLoss(). This choice aligns perfectly with the binary classification task at hand. Unlike the standard

binary cross-entropy loss which operates on class probabilities, BCEWithLogitsLoss integrates the sigmoid function directly into the loss calculation. It takes model outputs (logits) and true labels (0 or 1) as input, applies the sigmoid function to the logits, transforming them into probabilities between 0 and 1 and calculates the binary cross-entropy loss between the transformed probabilities and the true labels. This proves particularly advantageous because the model outputs logits (raw scores) before applying an activation function. By merging these steps, BCEWithLogitsLoss offers several benefits: improved numerical stability, enhanced computational efficiency, and the ability to directly optimize the logits, ultimately leading to better classification performance.

Gradient Accumulation

Another technique, called gradient accumulation is implemented in this project, which is a way to train deep learning models more efficiently [54]. Traditional training utilizes a fixed batch size, where the model processes a subset of the entire dataset at each iteration. Gradient accumulation tackles this by accumulating gradients across multiple smaller batches before performing a single weight update. In the beginning, during a forward pass, a mini-batch of data is passed through the model, resulting in predictions. A chosen loss function calculates the difference between these predictions and the true labels, yielding a single scalar loss value. The loss value is accumulated over a specified number of steps (accumulation steps, that are defined before training loop). This loop gathers gradient information from multiple forward passes without immediately updating the model weights after each pass. Once gradients are accumulated, the optimizer executes backpropagation using the combined gradient information. This single backpropagation step incorporates the aggregated gradients from all accumulated mini-batches, leading to an update of the model weights based on a larger virtual batch size.

This way, gradient accumulation can reduce the total number of optimizer steps required per epoch, leading to faster training, especially for larger models. By averaging gradients across multiple batches, accumulation can mitigate the effects of noisy gradients, potentially leading to more stable training behavior.

The optimal number of accumulation steps depends on various factors like GPU memory limitations, batch size, and model sensitivity. Large accumulation steps offer several advantages. They allow for more efficient memory usage by reducing the frequency of weight updates, which can be beneficial for models with limited GPU memory. Additionally, large accumulation steps can lead to faster training times as fewer optimization steps are performed per epoch. However, using large accumulation steps may result in less frequent weight updates, which can slow down the convergence of the training process and may lead to suboptimal performance, especially if the model encounters noisy or rapidly changing gra-

dients. On the other hand, small accumulation steps offer the advantage of more frequent weight updates, which can help the model converge faster and potentially achieve better performance. With smaller accumulation steps, the model can adapt more quickly to changes in the training data and gradients. However, using small accumulation steps may require more memory resources, as each optimization step results in an immediate update of the model weights.

For this project the accumulation steps = 2 is chosen, it means that gradients are accumulated over two mini-batches before performing a single optimization step.

Validation

Validation serves as the cornerstone of robust deep learning models. It ensures the model generalizes well beyond the data it's explicitly trained on. By evaluating performance on unseen examples, validation safeguards against overfitting.

One potent tool within the validation arsenal is the learning rate scheduler. This mechanism controls the magnitude by which the model's weights are adjusted during training. A high learning rate allows for rapid progress, but it can also lead the model to overshoot the optimal solution. Conversely, a very low learning rate can cause sluggish training and potentially hinder convergence. In this project, the scheduler is implemented like this: scheduler = ReduceLROnPlateau(optimizer, mode='min', patience=2, factor=0.1, verbose=True). Let's see more in details the meaning of each parameter:

- mode: Determines whether the scheduler should minimize or maximize the monitored metric. For instance, when monitoring the validation loss, setting `mode='min'` instructs the scheduler to decrease the learning rate when the validation loss stops decreasing.
- patience: Represents the number of epochs with no improvement in the monitored metric before the scheduler reduces the learning rate. It helps prevent premature learning rate reductions and allows the model to explore the parameter space more thoroughly.
- factor: Specifies the factor by which the learning rate will be reduced. For example, setting `factor=0.1` reduces the learning rate by a factor of 10 when triggered by the scheduler.
- verbose: Controls whether the scheduler prints updates about learning rate adjustments during training, providing insights into its behavior.

Selecting appropriate values for these parameters involves a balance between responsiveness to changes in the validation metric and stability during training. A higher patience value may allow the model to explore the parameter space more thoroughly but could delay convergence if set too high. Conversely, a lower patience value may lead to premature learning rate reductions, hindering the model's ability to find the optimal solution.

In addition to scheduling the learning rate, early stopping is another powerful technique for preventing overfitting and improving model generalization. The process involves monitoring the validation loss or metric during training and terminating the training process when the performance stops improving. In this work, the early stopping parameters are set to these values: best validation loss = float('inf'), patience = 5, current patience = 0. Variable "best validation loss" keeps track of the best validation loss observed during training. Similar to the parameter in the scheduler, "patience" specifies the number of epochs with no improvement in the validation loss before terminating training. "Current patience" tracks the number of consecutive epochs with no improvement in the validation loss.

When the model enters the evaluation mode, like in validation case (model.eval()), the dropout layers and batch normalization are disabled as they are not necessary during evaluation. If in the validation loop, the validation loss does not improve for a certain number of epochs (patience), early stopping is triggered, and training terminates. Finally, the scheduler adjusts the learning rate based on the validation loss to facilitate further optimization.

4.2 Evaluation Metrics

The evaluation of the model's performance is conducted using various metrics throughout the training process. During training, both training and validation losses are continuously monitored to gauge the model's performance. The training loss reflects the discrepancy between the model's predictions and the actual labels within the training dataset. Similarly, the validation loss measures the model's performance on a separate validation dataset, providing insights into its generalization ability.

Moreover, accuracy metrics are computed for both training and validation datasets, indicating the proportion of correct predictions made by the model. These accuracy values offer a clear assessment of the model's performance in terms of classification accuracy. An illustration of the training and validation performances regarding losses and accuracies can be seen in Figures 4.1 and 4.2. After training, the test accuracy is computed using the test set, providing a final evaluation of the model's performance. This evaluation ensures that the model's performance is assessed on unseen data, offering insights into its real-world applicability. The confusion matrix is generated to visualize the model's classification performance across different classes. It provides a detailed breakdown of true positive, true negative, false positive, and false negative predictions, offering insights into the model's classification errors.

The confusion matrices and AUC scores are produced for each of the 10 patients separately after training the model. This approach allows us to observe not just the overall averaged performance but also the variations of accuracies, specificities, and sensitivities for each patient. Since the model's performance is closely linked to the unique nature of the EEG signals from each patient, analyzing individual patient data is essential to understand the

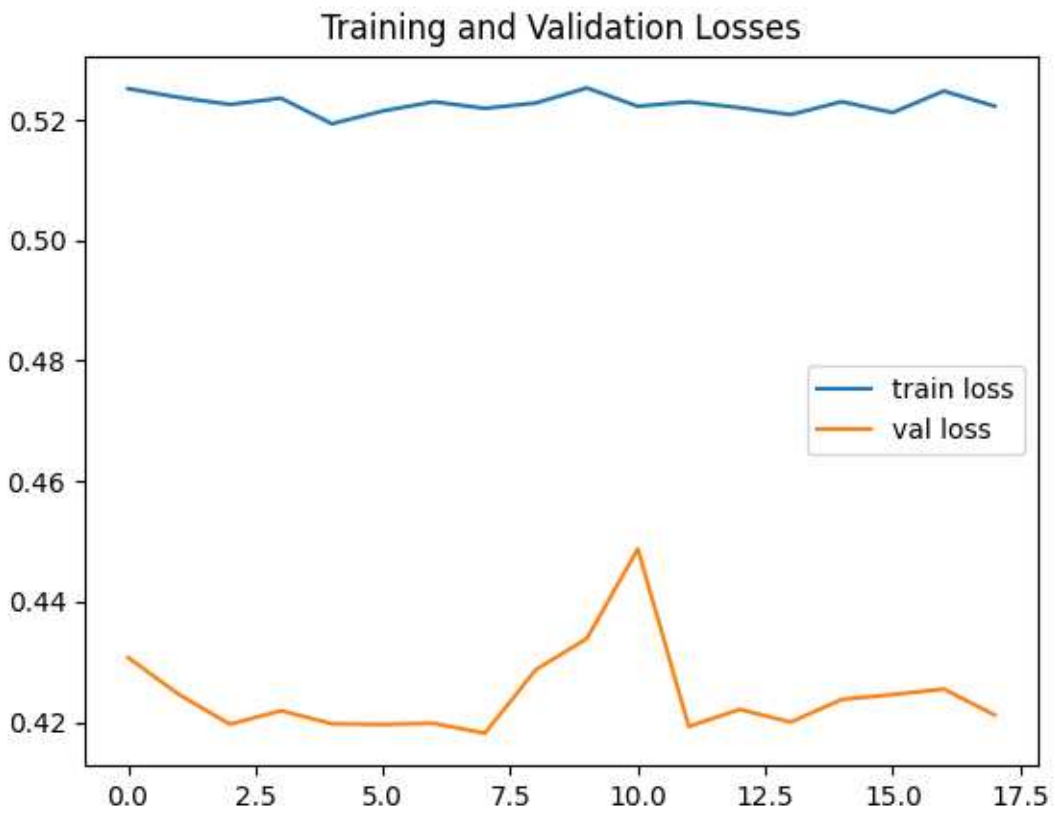


Figure 4.1: The training and validation losses of ResNet68 before early stopping are depicted. It's evident that the validation loss ceases to change and exhibit any further improvement. Continuing to train the model beyond this point would likely result in overfitting.

challenges specific to this task and to tailor interventions accordingly.

Another evaluation representation produced for each patient after training is a classification report. It provides a detailed summary of the model's performance across different classes, including precision, recall, F1-score, and support for each class. Precision and recall provide complementary perspectives on the model's performance. Precision measures the proportion of positive predictions that are truly correct (positive predictive value), while recall measures the proportion of actual positive cases that the model correctly identifies (sensitivity). Analyzing both can reveal potential imbalances in the model's predictions. F1-Score combines precision and recall into a single measure, offering a balanced view of the model's performance, especially when dealing with imbalanced class distributions. Additionally, it includes overall metrics such as accuracy, specificity, sensitivity, and the AUC score.

The Receiver Operating Characteristic (ROC) curve is plotted, along with the Area Under the Curve (AUC) score, to evaluate the model's performance across various thresholds. The

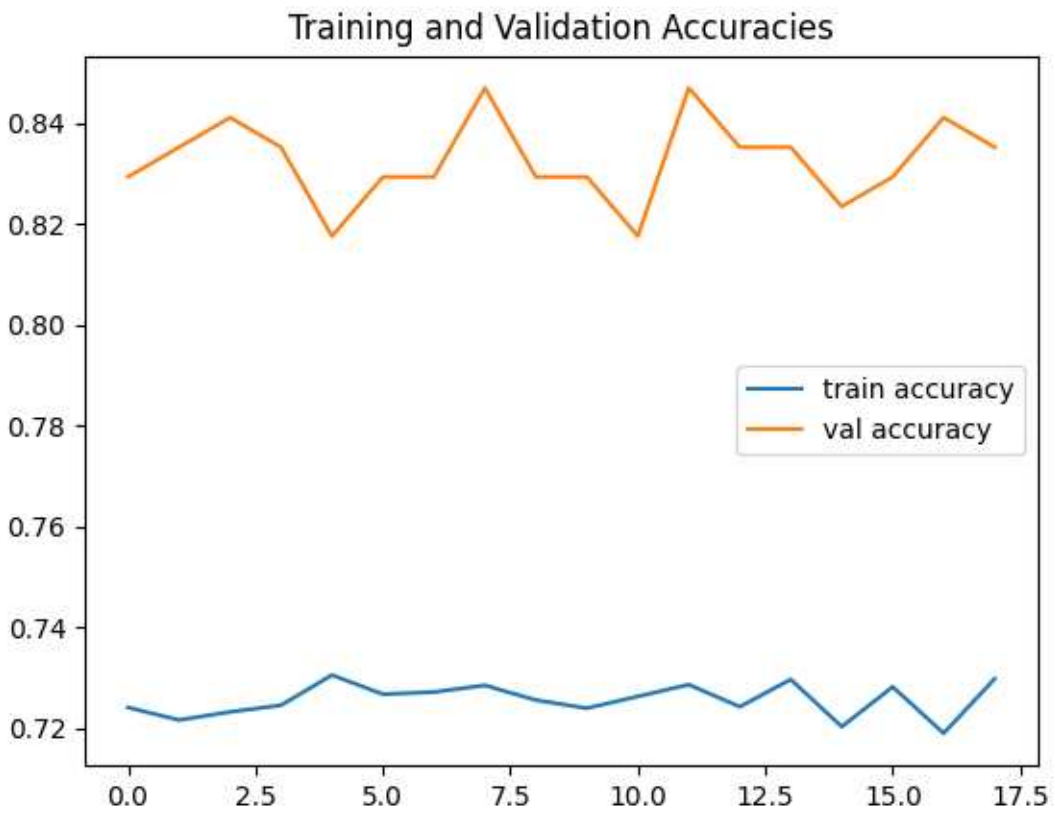


Figure 4.2: The training and validation accuracies of ResNet68 before early stopping are depicted. The trend of validation accuracy consistently remains slightly higher than the training accuracy trend, with approximately a 10% difference, and both values show minimal change over time, it suggests that the model is likely not overfitting to the training data. The lack of significant changes over time may also suggest that the model has reached a plateau in terms of learning and may not be able to further improve its performance without modifications to the architecture, hyperparameters, or the dataset itself.

ROC curve illustrates the trade-off between true positive rate and false positive rate, allowing for a comprehensive assessment of the model's discriminatory power.

Chapter 5

Results

This chapter presents the final results of the performance of three deep learning models. Their performance will be assessed using accuracy, sensitivity, and specificity metrics. The analysis will compare these metrics across different patient to understand variations in model efficacy. Additionally, a benchmark comparison will be conducted between the deep learning models and the machine learning results presented in the reference paper by Shafiezadeh et al.[8].

5.1 EEGNet Results

The training of the EEGNet model on a dataset comprising 10 patients, divided into 5 folds, utilizing GPU acceleration, required approximately 3 hours to complete. This training exhibited the best performance metrics in terms of an average accuracy of 65.78% and an average specificity of 82.94%. Among all patients, Patient 5 displayed the highest accuracy at 71.08%, accompanied by a notable specificity of 86.62% and a commendable sensitivity of 68.52%, which stands as the highest sensitivity achieved by any patient in this model. Patient 6 also showcased a noteworthy accuracy of 70.40%, while Patient 8 achieved an accuracy of 68.09%. Notably, all patients demonstrated high specificity, with a slight variation ranging from 71.99% for Patient 9 to 89.34% for Patient 6, highlighting the EEGNet model's proficiency in accurately classifying interictal samples. However, only for p7 the specificity of EEGNet is higher than that of ML model for Patient 7. Moreover, the variations in accuracy (the lowest accuracy is 61.81% for patient 7), sensitivity, and specificity between patients were relatively low, indicating a consistent performance of the model across different patient datasets. This uniformity suggests that the model's predictive capabilities remain robust regardless of individual patient characteristics. For simplicity, the results of the ML model, which are taken as a reference, are illustrated again in Table 5.1. All the results of EEGNet model are available for reference in Table 5.2. High accuracy and specificity of EEGNet

Patients Number	Gender	Total Seizures	ACC (%)	SEN (%)	SPE (%)
p1	m	10	82.90	68.99	100
p2	f	3	85.75	66.61	100
p3	m	2	79.49	49.91	100
p4	f	6	85.92	83.40	88.82
p5	f	3	84.20	62.08	100
p6	f	4	80.36	69.19	100
p7	f	5	76.95	79.99	72.39
p8	m	3	86.23	66.68	100
p9	m	2	81.77	49.97	100
p10	f	2	73.24	49.80	100
		Average	**81.68**	**64.66**	**96.12**

Table 5.1: RCV results with XGBoost for the Eugenio Medea IRCCS Hospital data set. Table adapted from [8].

Patients Number	Gender	Total Seizures	ACC (%)	SEN (%)	SPE (%)
p1	m	10	65.62	54.77	86.45
p2	f	3	64.91	50.63	79.77
p3	m	2	63.01	42.70	81.98
p4	f	6	63.68	55.95	72.56
p5	f	3	**71.08**	**68.52**	86.72
p6	f	4	70.40	65.61	**89.34**
p7	f	5	61.81	43.65	87.75
p8	m	3	68.09	60.87	84.27
p9	m	2	64.83	57.51	71.99
p10	f	2	64.34	49.82	88.56
		Average	**65.78**	**55.00**	**82.94**

Table 5.2: Results of EEGNet model.

signify the model's adeptness in correctly identifying non-seizure samples, which is crucial for minimizing false alarms. However, the model's sensitivity, currently at 55%, suggests that it may not capture every preictal (seizure) event, potentially leading to missed detections. Only for patients p5 and p9 the sensitivity of EEGNet is higher than that of ML for the same patients. Achieving a satisfactory balance between sensitivity and specificity is crucial in mitigating the risk of missed seizures while minimizing false alarms.

The training and validation loss trends after training the EEGNet model can be observed in Figures 5.1.

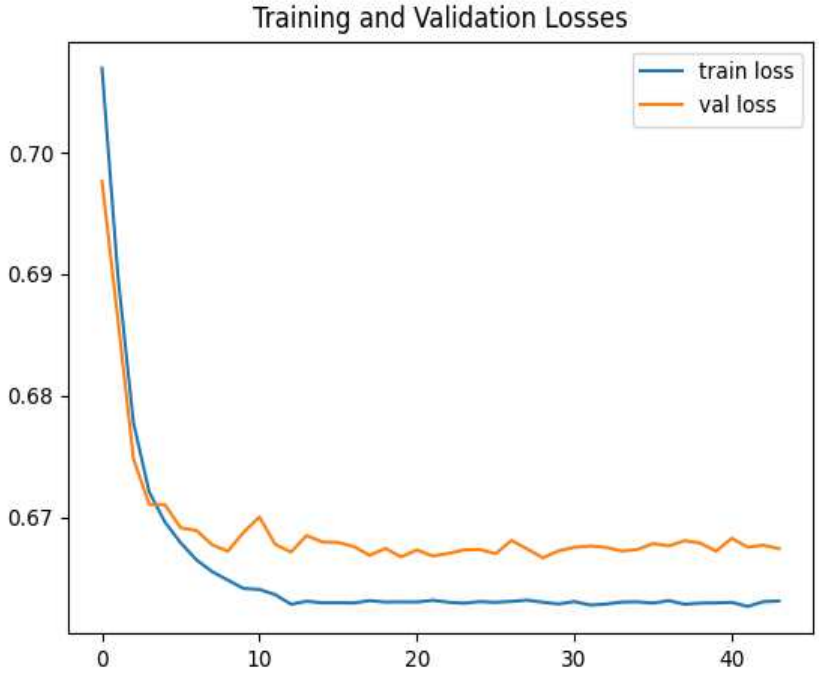


Figure 5.1: The plot of the training and validation loss for the EEGNet model. Both the training and validation losses decrease sharply within the first 5 epochs. This rapid decline indicates that the model is effectively learning from the training data and improving its performance. However, after these initial epochs, the decrease in both losses becomes less pronounced, and their values stabilize around 0.63 for the training loss and 0.66 for the validation loss. This suggests that the model has reached a point where it is no longer making significant improvements in performance on either the training or validation data. Throughout the training process, the validation loss consistently remains slightly higher than the training loss. This indicates that the model is experiencing some level of overfitting, as it performs slightly worse on unseen validation data compared to the data it was trained on.

Patients Number	Gender	Total Seizures	ACC (%)	SEN (%)	SPE (%)
p1	m	10	65.56	67.69	**63.69**
p2	f	3	57.83	74.41	41.94
p3	m	2	60.46	75.26	46.84
p4	f	6	**68.31**	86.03	47.91
p5	f	3	62.40	71.20	53.60
p6	f	4	65.44	88.69	41.85
p7	f	5	57.27	94.57	23.18
p8	m	3	62.69	91.69	32.17
p9	m	2	60.85	89.39	33.93
p10	f	2	62.32	**95.60**	28.78
		Average	**62.31**	**83.45**	**41.39**

Table 5.3: Results of DeepConvNet model.

5.2 DeepConvNet Results

The training of the DeepConvNet model required the least time compared to the other models, completing in less than two hours. DeepConvNet exhibits the best sensitivity values among all the models, with an average sensitivity of 83.45%. Remarkably, for 3 out of 10 patients, it achieves a sensitivity higher than 90% (p10=95.60%, p7=94.57%, p8=91.69%), while the lowest sensitivity recorded is 67.69%. Thus, there is a notable variance in sensitivity between patients, although it remains lower than that observed in the ML model. For patients p2, p3, p4, p5, p6, p7, p8, p9, and p10 (9 out of 10 patients), the sensitivity of DeepConvNet is higher than that of ML for the same patients, indicating a noticeable improvement. A high sensitivity indicates the model's ability to effectively predict seizures and generate alarms. However, the specificity of the DeepConvNet model is not satisfactory, being the lowest among all other networks at only 41.39%. This suggests that the model may produce false alarms at a rate higher than chance. Only two out of ten patients exhibit a specificity higher than 50% (p1 = 63.69% and p5 = 53.60%). Despite this, the accuracy of DeepConvNet remains satisfactory, indicating overall good performance, albeit not excellent. Patient 4 demonstrates the highest accuracy of 68.31%, with a very good sensitivity of 86.03% but a low specificity of 47.91%. The results are detailed in Table 5.3.

The training and validation loss trends after training the DeepConvNet model can be observed in Figure 5.2.

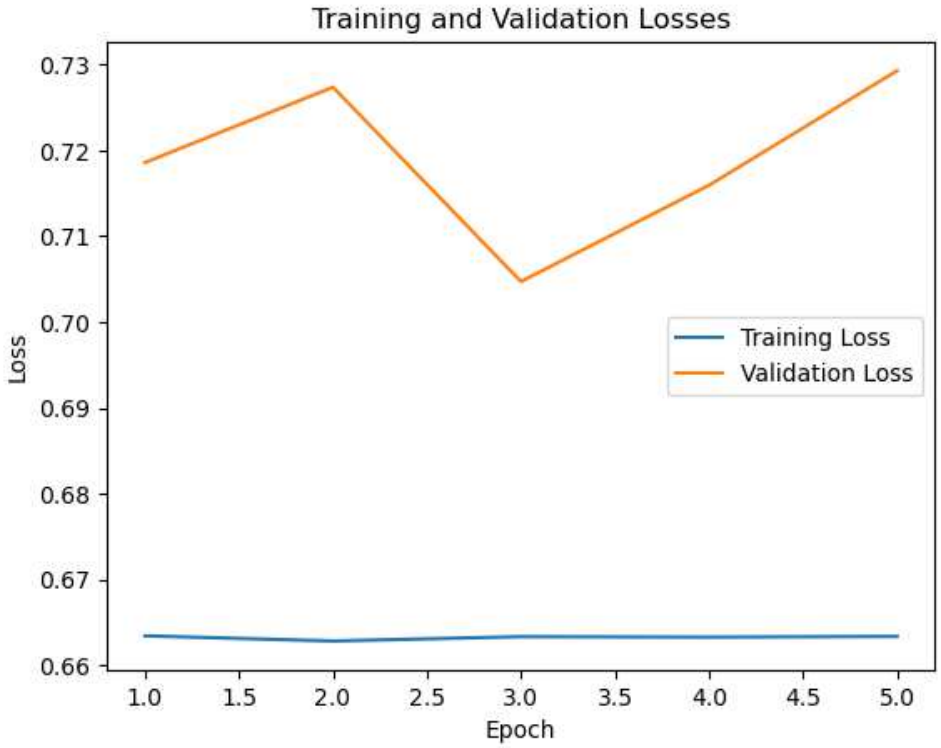


Figure 5.2: The plot of the training and validation loss for the DeepConvNet model. The training loss remains relatively stable across epochs, hovering around 0.665, suggesting that the model maintains consistent performance on the training data. However, the validation loss exhibits erratic fluctuations without showing a clear trend towards convergence. This inconsistency indicates potential overfitting to the training data, where the model performs well on the training set but struggles to generalize to unseen data, resulting in elevated validation loss.The training was stopped after a few epochs (5) due to the absence of improvement in validation loss. In fact, the validation loss begins to increase without showing any signs of improvement, as depicted on the plot.

5.3 ResNet68 Results

ResNet68, the most complex deep neural network in this study, comprises a total of 68 layers and over thirty million trainable parameters. Consequently, its training necessitated approximately 10 hours, even with GPU acceleration and the relatively small dataset of 10 patients. However, the results proved to be promising, offering potential for significant advancements in the field. The average values of accuracy, sensitivity, and specificity for ResNet68 are 63.80%, 69.71%, and 61.42%, respectively. Notably, no single metric falls below 50% for any patient, indicating the model's robust generalization ability across various EEG signals. ResNet68 demonstrates balanced performance metrics, positioning it between the EEGNet and DeepConvNet models.

The highest accuracy of 70.88% is achieved for patient 6, who also exhibits a high sensitivity of 77.55%. Patient 5 boasts the highest sensitivity (81.40%) among all patients, while

Patients Number	Gender	Total Seizures	ACC (%)	SEN (%)	SPE (%)
p1	m	10	61.81	67.50	62.86
p2	f	3	60.46	72.77	51.34
p3	m	2	60.65	58.04	58.03
p4	f	6	67.86	73.00	**70.14**
p5	f	3	67.20	81.40	60.34
p6	f	4	**70.88**	**77.55**	60.50
p7	f	5	60.73	67.75	60.43
p8	m	3	64.44	68.51	68.03
p9	m	2	61.32	69.39	62.34
p10	f	2	62.66	61.17	60.15
		Average	**63.80**	**69.71**	**61.42**

Table 5.4: Results of ResNet68.

patient 4 showcases the highest specificity (70.14%). Notably, patients 2 and 3 achieve very satisfactory sensitivity values of 72.22% and 73.00%, respectively. For patients p2, p3, p5, p6, p8, p9, p10 (7 out of 10 patients) the sensitivity of ResNet66 is higher than that of ML for the same patients.

In terms of stability, ResNet68 shows lower variance between patients compared to the ML model, even concerning the average values. This indicates that ResNet68 is the most consistent model across patients, offering reliable performance regardless of individual EEG signal characteristics. Refer to Table 5.4 for a detailed breakdown of the results obtained from ResNet68.

The training and validation loss trends after training the ResNet68 model can be observed in Figure 5.3.

All the accuracies achieved with DL models, both individual for each patient and the average ones, are lower than the accuracies of the ML model. However, the average sensitivity of DeepConvNet and ResNet68 is higher than that of the ML model, suggesting their effectiveness in identifying seizure events. On the contrary, the average specificity of the three DL models is lower than that of the ML model, indicating a higher tendency for false alarms. Figure 5.4 displays the average metrics of three DL models and an ML model.

In Figure 5.5 it is possible to observe the comparison between the ROC curves of three deep learning models.

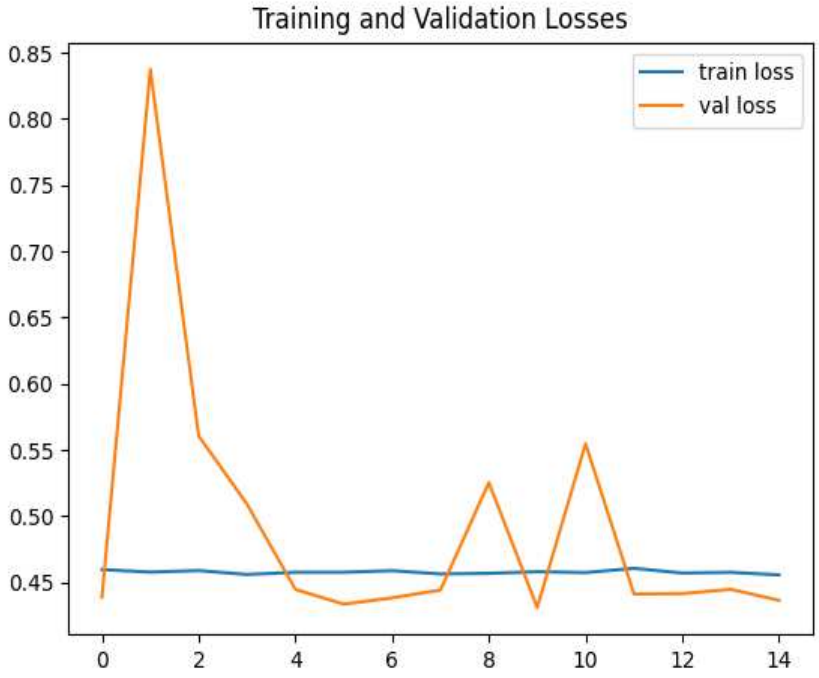


Figure 5.3: The plot of the training and validation loss for the ResNet68 model. Throughout the training process, the training loss remains relatively constant at around 0.45, indicating stable performance on the training dataset. However, the validation loss exhibits a tumultuous trend, characterized by oscillations that gradually decrease over epochs. After approximately 11 epochs, the validation loss begins to consistently fall below the training loss, maintaining this pattern without significant improvement until the early cessation of training after 14 epochs.

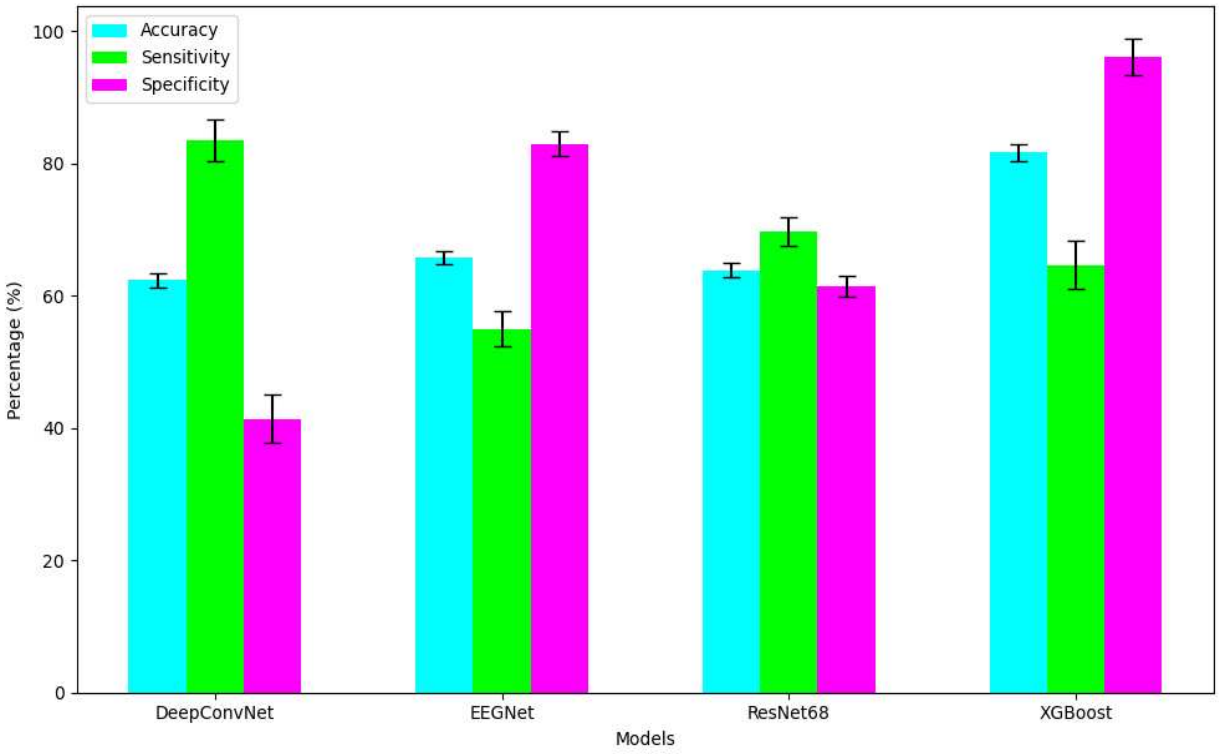


Figure 5.4: The bar chart illustrates the average performance metrics of DeepConvNet, EEGNet, ResNet68, and XGBoost, along with the standard error of the mean calculated from data obtained from 10 patients.

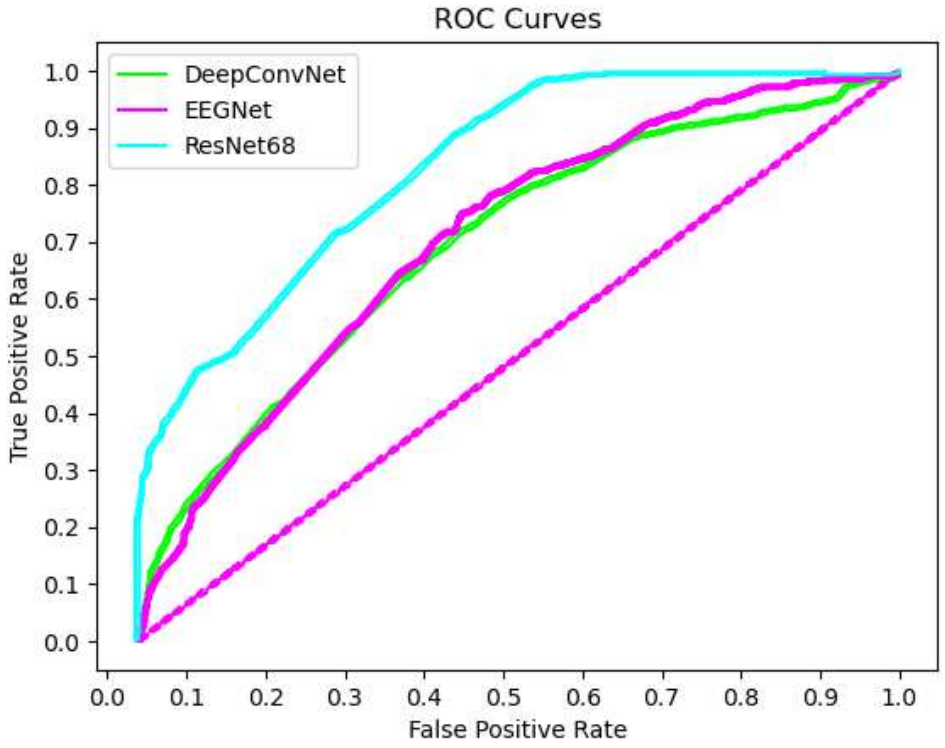


Figure 5.5: The ROC curves of three deep learning models are depicted: in green for the DeepConvNet model with an average AUC score of 0.66, sensitivity of 83.45%, and specificity of 41.39%; in magenta for the EEGNet model with an average AUC score of 0.75, sensitivity of 55.00%, and specificity of 82.94%; and in blue for the ResNet68 model with an average AUC score of 0.79, sensitivity of 69.71%, and specificity of 61.42%.

Chapter 6

Conclusions

In conclusion, the utilization of machine learning (ML) models for seizure prediction has yielded impressive results in numerous studies, as it was outlined in previous chapters. Also, deep learning (DL) models have garnered increasing interest, particularly in the field of seizure prediction. In this study, we explore this interest by applying three popular DL models to a novel dataset previously unutilized for seizure prediction. The DL models—EEGNet, DeepConvNet, and ResNet68—were trained on a dataset comprising 10 patients, with performances evaluated separately for each patient. Despite extensive training using GPU acceleration, the DL models did not demonstrate superior performance compared to the classical ML model, raising questions about their applicability in clinical settings.

Starting with EEGNet, it demonstrated acceptable average accuracy and specificity, yet its low sensitivity suggested potential limitations in capturing the preictal events. This highlights the importance of striking a delicate balance between sensitivity and specificity to minimize false alarms. One possible explanation for its subpar performance could be its inadequate complexity in capturing subtle patterns within EEG signals, which are indicative of seizure activity. Seizure events often exhibit specific patterns in EEG signals, such as rapid spikes or rhythmic activity in certain frequency bands. Ensuring good sensitivity in seizure prediction requires the model's proficiency in capturing these intricate patterns and distinguishing them from background noise or non-seizure activity. It's conceivable that the EEGNet model used in this study lacks the necessary complexity to effectively discern these subtle variations, potentially leading to inaccurate preictal event detection. Therefore, enhancing the model's complexity through architectural improvements, such as adding more layers or utilizing more sophisticated network structures, could potentially enhance its ability to capture these patterns and improve sensitivity.

Moving on to DeepConvNet, despite exhibiting the best sensitivity values among the models, its specificity fell short, indicating a high false alarm rate, indicating a need for better optimization to achieve a balance between sensitivity and specificity. Moreover, its

performance varied significantly across different patients, indicating inconsistencies in seizure prediction accuracy. Simply put, while the model performed well for some patients, it performed poorly for others.

Regarding ResNet68, despite being the most complex model and requiring extensive training time, it still requires further optimization of hyperparameters and potentially a larger dataset. Additionally, a common issue among all three models could be attributed to limitations in the representativeness of the dataset. A representative dataset should encompass a diverse range of patients, seizure types, and physiological conditions to ensure that the model learns from a comprehensive set of scenarios. If the dataset lacks diversity, such as predominantly containing similar patient profiles, the model may struggle to generalize well to unseen data or different patient populations. This lack of representativeness can impede the model's ability to learn robust features and may result in suboptimal performance in real-world scenarios.

Given that feature extraction was manual for traditional machine learning (ML) models and automated for deep learning (DL) models, it's worth noting that handcrafted features may be more interpretable compared to features learned automatically by DL models. However, DL models can directly learn hierarchical representations from raw data, which can be advantageous for capturing complex patterns but may necessitate a larger and more diverse dataset for effective generalization. DL models are expected to learn more intricate representations but are also more prone to overfitting, especially when trained on limited datasets. Therefore, the dataset that yielded positive results for ML models may not be sufficient for training DL networks.

To improve performance, several strategies can be considered. Firstly, leveraging more data could enhance model performance, which was not done in this work due to time constraints. Additionally, utilizing pre-trained EEGNet and DeepConvNet models can save time and resources. Furthermore, augmenting pre-trained models by adding more layers and increasing depth, could enhance their generalization ability and performance.

Overall, the observed weaknesses highlight the need for more sophisticated approaches in feature extraction and model architecture design. Additionally, leveraging larger and more diverse datasets, along with advanced optimization techniques, could enhance the robustness and generalization capabilities of these models, ultimately advancing their applicability in clinical settings.

Chapter 7

List of Acronyms

- Adam adaptive moment estimation
- AF adaptive filtering
- AI artificial intelligence
- AMP automatic mixed precision
- AUC area under the (ROC) curve
- BP backpropagation
- CAR common average referencing
- CNN convolutional neural network
- CPU central processing unit
- CV cross validation
- DL deep learning
- DWT discrete wavelet transformation
- EEG electroencephalogram
- FCNN fully-connected neural network
- FPR false positive rate
- GAP global average pooling
- GCN graph convolutional network

- GNN graph neural network
- GPU graphics processing unit
- HMM hidden Markov models
- ICA independent component analysis
- k-NN k-nearest neighbors
- LDA linear discriminant analysis
- LOO leave one out
- LSTM long short-term memory
- ML machine learning
- MNE minimum norm estimation
- MSE mean squared error
- NN neural network
- PCA principal component analysis
- RCV random cross validation
- ReLU rectified linear unit
- RF random forest
- RNN recurrent neural network
- ROC receiver operating characteristic
- SGD stochastic gradient descent
- SVM support vector machine
- VM virtual machine
- Tanh hyperbolic tangent

Bibliography

[1] *Epilepsy Fact Sheet*. URL: `https://www.who.int/news-room/fact-sheets/detail/epilepsy` (visited on 03/18/2024).

[2] Mark Manford. "Recent advances in epilepsy". In: *Springerlink* (2017).

[3] H B Valman. "Epilepsy". In: *British Medical Journal* (1982).

[4] Terence J O'Brien Roland D Thijs Rainer Surges and Josemir W Sander. "Epilepsy in adults". In: *The Lancet* (2019).

[5] U. Rajendra Acharya et al. "Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals". In: *Computers in Biology and Medicine* (2018).

[6] Christian E Elger and Christian Hoppe. "Diagnostic challenges in epilepsy: seizure under-reporting and seizure detection". In: *The Lancet Neurology* (2018).

[7] K. Malmgren et al. "Diagnosing epileptic seizures and epilepsy". In: *Läkartidningen* (2018).

[8] Sina Shafiezadeh et al. "Methodological Issues in Evaluating Machine Learning Models for EEG Seizure Prediction: Good Cross-Validation Accuracy Does Not Guarantee Generalization to New Patients". In: *Applied Sciences* (2023).

[9] Andreas Pedroni, Amirreza Bahreini, and Nicolas Langer. "Automagic: Standardized preprocessing of big EEG data". In: *NeuroImage* (2019).

[10] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. "Deep learning for electroencephalogram (EEG) classification tasks: a review". In: *Journal of Neural Engineering* (2019).

[11] Yongcheng Wu Wenqiang Yan. "A time-frequency denoising method for single-channel event-related EEG". In: *Frontiers in Neuroscience* (2022).

[12] Mark J Cook Linda Dalic. "Managing drug-resistant epilepsy: challenges and solutions". In: *Dovepress* (2016).

[13] Syed Muhammad Usman, Shehzad Khalid, and Muhammad Haseeb Aslam. "Epileptic Seizures Prediction Using Deep Learning Techniques". In: *IEEE Access, vol. 8, pp. 39998-40007, 2020* (2019).

[14] Yan Li Siuly Siuly. "Designing a robust feature extraction method based on optimum allocation and principal component analysis for epileptic EEG signal classification". In: *Elsevier* (2015).

[15] M. Iftikhar, S. A. Khan, and A. Hassan. "A Survey of Deep Learning and Traditional Approaches for EEG Signal Processing and Classification". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication* (2018).

[16] Jaiswal Abeg Kumar and Haider Banka. "Epileptic seizure detection in EEG signal using machine learning techniques". In: *Australas Phys Eng Sci Med 41, 81–94* (2019).

[17] Villamar MF Al-Bakri AF et al. "Noninvasive seizure prediction using autonomic measurements in patients with refractory epilepsy". In: *Annu Int Conf IEEE Eng Med Biol Soc.* (2018).

[18] Rekha Sahu et al. "Epileptic seizure detection: a comparative study between deep and traditional machine learning techniques". In: *J. Integr. Neurosci.* (2020).

[19] El Tahry Riëm Vanabelle Paul De Handschutter Pierre, Benjelloun Mohammed, and Boukhebouze Mohamed. "Epileptic seizure detection using EEG signals and extreme gradient boosting". In: *The Journal of Biomedical Research* (2020).

[20] Shilpa Gite Milind Natu Mrinal Bachute, Ketan Kotecha, and Ankit Vidyarthi. "Review on Epileptic Seizure Prediction: Machine Learning and Deep Learning Approaches". In: *Hindawi Computational and Mathematical Methods in Medicine* (2018).

[21] Dhahri H Awassa L Jdey I et al. "Study of Different Deep Learning Methods for Coronavirus (COVID-19) Pandemic: Taxonomy, Survey and Insights". In: *Sensors (Basel).* (2022).

[22] Abubakar Abid James Zou Mikael Huss et al. "A primer on deep learning in genomics". In: *Nat Genet.* (2019).

[23] Fishman D Jones W Alasoo K and Parts L. "Computational biology: deep learning". In: *Emerg Top Life Sci.* (2017).

[24] Caliva F Kijowski R Liu F and Pedoia V. "Deep Learning for Lesion Detection, Progression, and Prediction of Musculoskeletal Disease". In: *J Magn Reson Imaging.* (2020).

[25] Kheradpisheh SR Tavanaei A Ghodrati M, Masquelier T, and Maida A. "Deep learning in spiking neural networks". In: *Neural Netw* (2019).

[26] Khansa Rasheed et al. “Machine Learning for Predicting Epileptic Seizures Using EEG Signals: A Review”. In: *IEEE Reviews in Biomedical Engineering, vol. 14, pp. 139-155* (2021).

[27] Rubén San-Segundo et al. “Classification of epileptic EEG recordings using signal transforms and convolutional neural networks”. In: *Computers in Biology and Medicine* (2019).

[28] A. Krizhevsky G. E. Hinton N. Srivastava, I. Sutskever, and R. R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: (2012).

[29] Bart Selman Johan Bjorck Carla Gomes and Kilian Q. Weinberger. “Understanding Batch Normalization”. In: *Cornell University* (2018).

[30] Navid Ghassemi Afshin Shoeibi Marjane Khodatars et al. “Epileptic Seizures Detection Using Deep Learning Techniques: A Review”. In: *International Journal of Environmental Research and Public Health* (2021).

[31] Mustafa Sameer and Bharat Gupta. “CNN based framework for detection of epileptic seizures”. In: *Multimed Tools Appl 81, 17057–17070* (2022).

[32] Gupta M. Majumdar A. “Recurrent transform learning”. In: *Neural Netw.* (2019).

[33] Zhang Z Wei X Zhou L, Chen Z, and Zhou Y. “Early prediction of epileptic seizures using a long-term recurrent convolutional network”. In: *J Neurosci Methods* (2019).

[34] Kostas M. Tsiouris et al. “A Long Short-Term Memory deep learning network for the prediction of epileptic seizures using EEG signals”. In: *Computers in Biology and Medicine* (2018).

[35] Saroj Kumar Pandey et al. “Automated epilepsy seizure detection from EEG signal based on hybrid CNN and LSTM model”. In: *SIViP 17, 1113–1122* (2023).

[36] Shah M. Zaeemzadeh A Rahnavard N. “Norm-Preservation: Why Residual Networks Can Become Extremely Deep?” In: *IEEE Trans Pattern Anal Mach Intell.* (2021).

[37] Diyuan Lu and Jochen Triesch. “Residual Deep Convolutional Neural Network for EEG Signal Classification in Epilepsy”. In: *arXiv:1903.08100* (2019).

[38] *NumPy*. URL: `https://numpy.org/` (visited on 03/18/2024).

[39] *Pandas*. URL: `https://pandas.pydata.org/` (visited on 03/18/2024).

[40] *MNE*. URL: `https://mne.tools/stable/documentation/index.html` (visited on 03/18/2024).

[41] *PyTorch*. URL: `https://pytorch.org/` (visited on 03/18/2024).

[42] *scikit-learn*. URL: `https://scikit-learn.org/stable/` (visited on 03/18/2024).

[43] *Matplotlib*. URL: https://matplotlib.org/ (visited on 03/18/2024).

[44] Jacob White and Sarah D. Power. "k-Fold Cross-Validation Can Significantly Over-Estimate True Classification Accuracy in Common EEG-Based Passive BCI Experimental Designs: An Empirical Investigation". In: *Sensors* (2023).

[45] Rosanne O Albuquerque I Monteiro J and Falk TH. "Estimating distribution shifts for predicting cross-subject generalization in electroencephalography-based mental workload assessment". In: *Front Artif Intell.* (2022).

[46] Ao Chen Feng Yu Hao Zhang et al. "Characterizing and understanding deep neural network batching systems on GPUs". In: *BenchCouncil Transactions on Benchmarks, Standards and Evaluations* (2023).

[47] Vernon J Lawhern and et al. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". In: *Journal of Neural Engineering* (2018).

[48] Xinbin Liang et al. "Convolutional Neural Network with a Topographic Representation Module for EEG-Based Brain—Computer Interfaces". In: *Brain Sciences* (2023).

[49] Robin Tibor Schirrmeister et al. "Deep Learning With Convolutional Neural Networks for EEG Decoding and Visualization". In: *Journal of Neural Engineering* (2017).

[50] Fengxiang He, Tongliang Liu, and Dacheng Tao. "Why ResNet Works? Residuals Generalize". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[51] Dohyun Lee et al. "A ResNet LSTM hybrid model for predicting epileptic seizures using a pretrained model with supervised contrastive learning". In: *Sci Rep 14, 1319* (2024).

[52] Yating Jiang, Yao Lu, and Lingling Yang. "An epileptic seizure prediction model based on a time-wise attention simulation module and a pretrained ResNet". In: *Journal of Medical Systems* (2020).

[53] Diyuan Lu and Jochen Triesch. "Residual Deep Convolutional Neural Network for EEG Signal Classification in Epilepsy". In: *IEEE Transactions on Biomedical Engineering* (2021).

[54] *Gradient Accumulation*. URL: https://www.hopsworks.ai/dictionary/gradient-accumulation (visited on 03/26/2024).