Sapienza
Università di Roma

EEG signal denoising and arifact removal using machine learning techniques

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Applied Computer Science and Artificial Intelligence

Martina Doku
ID number 1938629

Advisor
Prof Danilo Avola

Academic Year 2022/23

Thesis not yet defended

EEG signal denoising and arifact removal using machine learning techniques
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: doku.1938629@studenti.uniroma1.it

dedication

Abstract

This thesis aims to investigate the use of machine learning techniques for EEG signal denoising. The first part of the thesis is dedicated to the introduction of the problem, the statement of the research questions and an overview of the basic concepts. The second part is dedicated to history and the state of the art of signal analysis, and more specifically of the EEG signal analysis and denoising methods. The third part is dedicated to the machine learning techniques. The fourth part is dedicated to the experimental results and the last part is dedicated to the conclusions and future works.

Contents

Chapter 1

Introduction

The chapter is built as follows: in the first section there is a brief introduction of the problem, the reasons that led to the choice of the topic and the outline of the thesis. In the second section there is the statement of the research questions. In the third section there is a brief description of the basic concepts.

1.1 Problem statement

What is EEG in the first place and why is it important? Electroencephalography (EEG) is a non-invasive technique used to measure the electrical activity of the brain. EEG signals are still among the less explored ones in the field of signal processing, despite their widespread use in clinical practice and research. In recent years, there has been a growing interest in developing denoising methods for EEG data to improve their quality and reliability.

The main objective of this thesis is to investigate and compare different EEG denoising methods, and to evaluate their performance in terms of signal quality, artifact removal, and preservation of underlying brain activity. Specifically, we will focus on the most recent machine learning techniques, advanced models like the 'transformers'. We will also investigate the potential of combining different denoising methods to improve the performance of EEG denoising.

Overall, this thesis aims to provide a better understanding of the strengths and limitations of different EEG denoising methods, and to help researchers and clinicians make informed decisions when selecting the most appropriate denoising method for their EEG data analysis. By improving the quality of EEG signals, we can enhance our understanding of brain function and ultimately contribute to the development of more effective diagnostic and therapeutic tools for neurological disorders.

1.2 Research questions

The research questions are the following:

- How can we remove artifacts from EEG signals?

- How can we exploit the most recent machine learning techniques for EEG denoising?
- What are the strengths and limitations of these new methods?
- What are the performance of these new methods?

1.3 Basic concepts

In this section we will introduce the basic concepts that will be used in the thesis. The first part is dedicated to the EEG signal, the second part is dedicated to the EEG denoising.

1.3.1 EEG

When talking about EEG,in this thesis, we are referring to the electroencephalogram, in particular we are interested in the EEG waves. Electroencephalogram (EEG) waves are the patterns of electrical activity that are recorded by EEG measurements. These waves have different frequencies and amplitudes, and they reveal different states of brain activity. We divide the EEG waves in 5 main categories depending on their frquency:Alpha, Beta, Theta, Delta and Gamma waves.

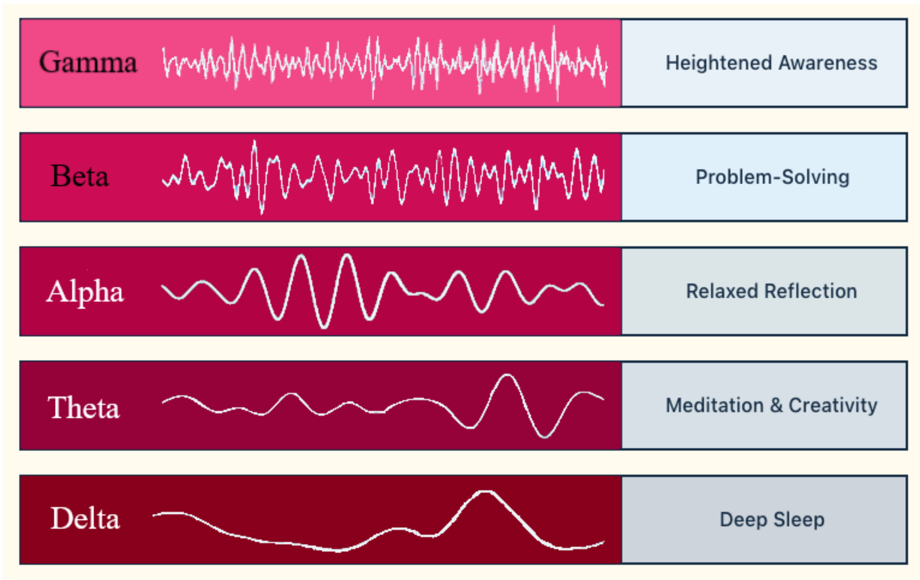

Figure 1.1. Types of EEG waves

Delta waves are typically observed during deep sleep and have a frequency of less than 4 Hz. Theta waves are associated to light sleep or drowsiness and have a frequency of 4-7 Hz. Alpha waves correspond to the relaxed and awake state, witha frequency of 8-13 Hz. Beta waves, on the other hand, are associated with active cognitive processing and have a higher frequency of 14-30 Hz. Gamma waves have a frequency of 30-100 Hz and are associated with higher cognitive functions such as attention and memory.

It's important to remember that EEG waves are not distinct entities but represent a continuous spectrum of activity that can be influenced by various factors such as task demands, attention, and emotion. Interpreting EEG waves requires expertise and context since different patterns of EEG activity may reflect different states of brain activity depending on the individual and the experimental conditions. Furthermore, research has shown that EEG waves can be useful in clinical diagnosis and prognosis, as well as in the assessment of cognitive function and brain injury. Therefore, understanding the various EEG waves and their characteristics can provide valuable insights into brain function and activity.

1.3.2 EEG denoising

EEG signals can be influenced by various factor that alter the real waves originated from neural activities, those factors are defined as artifacts. The artifacts can be classified as[1]:

- intrinsic artifacts: artifacts that depend on physiological sources, such as ocular artifacts (EOG) that come from eye movement and blinking, muscle artifacts (EMG) and cardiac artifacts (ECG)
- extrinsic artifacts: artifacts generated from external electromagnetic such as power line noise sources.

Denoising of EEG data is an essential task to be able to work on data and to extract meaningful information from it. The denoising process is complex and it does lead to different level of quality of the data depending on the method used, the quality of the data and the type of artifact.

There are several challenges[2] related both to single methods characteristic and general artifact removal. For example, some methods are computationally expensive and require a lot of time to be applied, some methods are not able to remove all kind of artifacts, some methods require a lot of data to be applied. On a general level, there is the problem of the lack of a standard method to evaluate the quality of the denoised data and the EEG applications are not yet fuly commercial, so there hasn't been a sufficient investment in hardware and software to make the denoising process easier.

However the main goal of latest studies is to find a method that can denoise from all kind of artifacts and that can be used in a flexible and fast way, to accomodate the needs of all the different EEG applications.

Chapter 2

Literature review

In this chapter we will present the main methods used for the denoising of EEG data. In the first part we will focus on the methods used for the removal of artifacts from signals in general, in the second part we will present the ones specifically deveolopped for the denoising of EEG data and in the third part we will explore the latest, machine learning related, methods.

2.1 Signal analysis

In this section we will present the basic techniques used for the analysis of signals that constituted a base for the ones developed for the analysis of EEG data.

2.1.1 Fourier transform

The first method used for the removal of artifacts from signals is the Fourier transform[3]. The Fourier transform is a mathematical tool used used to decompose a signal into sine and cosine functions, that are used as basis functions for the original signal. It is defined as:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{2.1}$$

where $f(t)$ is the signal and ω is the frequency. The Fourier transform of a signal is a complex number, which can be decomposed into its real and imaginary parts.

2.1.2 Fast Fourier transform

The Fourier transform is a very useful tool for the analysis of signals, but it is computationally expensive. The Fast Fourier transform (FFT)[4] is an algorithm that is used to calculate the Fourier transform of a signal in a shorter time: it reduces the time nedded to Nlog (N), obtaining a speed-up of a factor of N/log (N). The FFT is used in many different fields of science such as signal processing, image processing, etc...

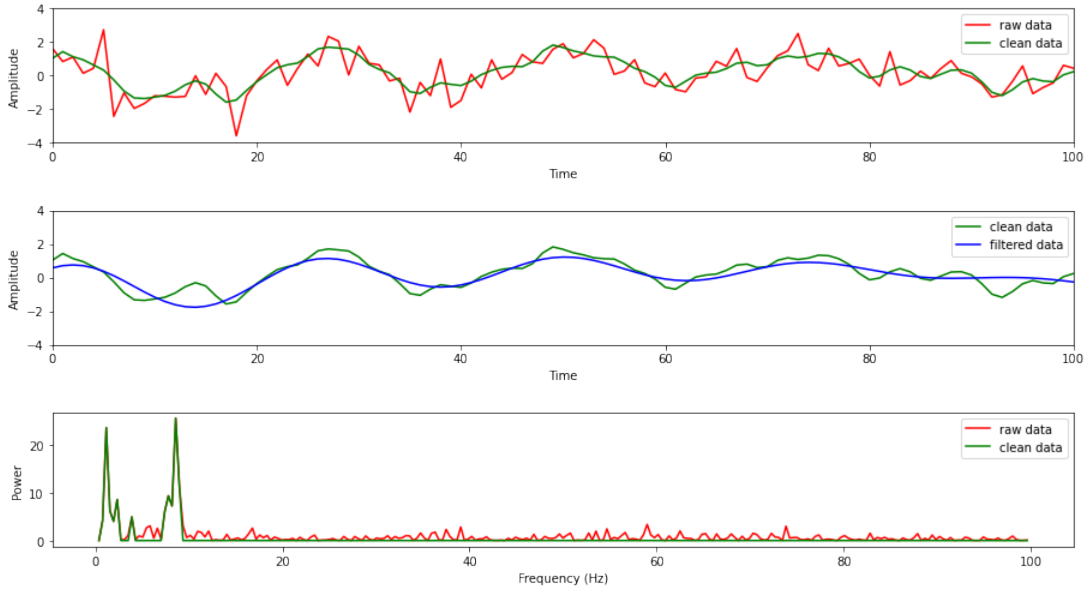

Figure 2.1. Result of the FFT of a signal

2.1.3 Short Time Fourier Transform

The Short Time Fourier Transform (STFT)[5][6] is a method used to calculate the Fourier transform of a signal. It is used with non stationary signals to find the frequency components of a signal over time. The STFT equation is defined as:

$$S(\tau) = s(t) \cdot h(t - \tau) \tag{2.2}$$

where $s(t)$ is the signal, $h(t)$ is the window function and τ is the time.
The STFT can be used to find the frequency components of a signal:

$$S(\omega) = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} s(\tau) \cdot h(t - \tau) e^{-i\omega t} d\tau \tag{2.3}$$

where $s(\tau)$ is the signal, $h(t)$ is the window function and ω is the frequency.

2.1.4 Wigner-Ville distribution

The Wigner-Ville distribution (WVD)[7] is a method used to provide the description of a signal in the time-frequency domain with higher resolution. It is described by the following equation:

$$W(\tau, \omega) = \int_{-\infty}^{\infty} s(t - \tau) \cdot s^*(t - \tau) e^{-2\pi\omega\tau} d\tau \tag{2.4}$$

where $s(t)$ is the signal, τ is the time and ω is the frequency. The main problem of the WVD is that it introduces the so called cross terms. In fact, for a signal

$$x(t) = x_1(t) + x_2(t) \tag{2.5}$$

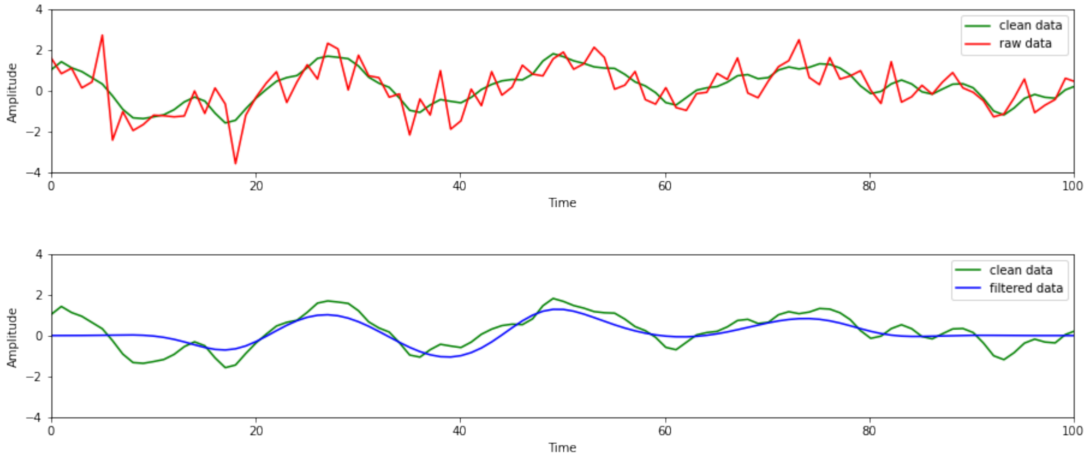

Figure 2.2. Result of the STFT of a signal

the corresponding WVD is:

$$WVD(t,f) = WVDx_1(t,f) + WVDx_2(t,f) + 2Re[WVDx_1x_2(t,f)] \tag{2.6}$$

where $2Re[WVDx_1x_2(t,f)]$ is the cross term which is removable at the expense of a loss of resolution.

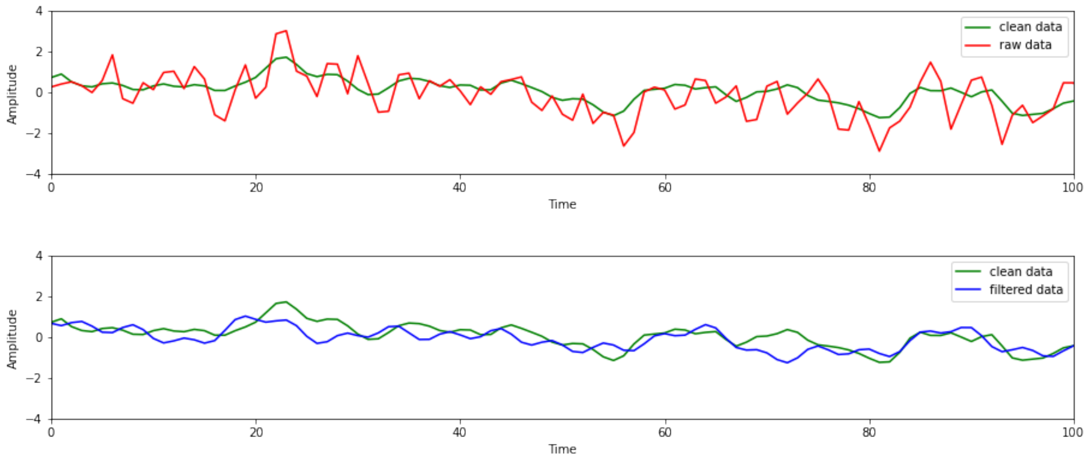

Figure 2.3. Result of the WVD of a signal

2.1.5 Wavelet transform

The wavelet transform[8] is a method used to find the frequency components of a signal over time. It is used with non stationary signals and it provides a better time-frequency resolution than the STFT and the WVD since it gives a better simultaneous time-frequency localization.
It expresses the signal as a linear combination of mother wavelets. The wavelet

transform is defined as:

$$W(\tau,\omega) = \int_{-\infty}^{\infty} s(t-\tau)\cdot\psi(t-\tau)e^{-2\pi\omega\tau}d\tau \tag{2.7}$$

where $s(t)$ is the signal, τ is the time, ω is the frequency and $\psi(t)$ is the wavelet function.
A possible choice for the wavelet function is the Morlet wavelet:

$$\psi(t) = \frac{1}{\sqrt{2c\pi}}\cdot e^{-\frac{t^2}{2c^2}}\cdot e^{i\omega_0 t} \tag{2.8}$$

where ω_0 is the central frequency of the wavelet and c is the width of the wavelet.

2.1.6 Matching Pursuit

The Matching Pursuit (MP)[10] is a greedy algorithm used to approximate a signal with a linear combination of basis functions called time-frequency atoms, selected from a dictionary of functions.
A general family of time-frequency atoms can be generated by scaling, translating and modulating a single window function $g(t)$ as follows:

$$g_\gamma(t) = \frac{1}{\sqrt{s}}\cdot g(\frac{t-u}{s})\cdot e^{i\xi t} \tag{2.9}$$

where γ is the tuple (s, u, ξ) s is the scale, u is the translation and ξ is the frequency modulation.
The algorithm aims to find a set of atoms $\gamma_1, \gamma_2, \ldots, \gamma_N$ that best approximate the signal $f(t)$, i.e.:

$$f(t) \approx \sum_{i=1}^{N} a_i\cdot g_\gamma(t) \tag{2.10}$$

where a_i are the expansion coefficients.
The vector f can be decomposed into

$$f(t) =< f, g_\gamma > g_\gamma + Rf(t) \tag{2.11}$$

where R_f is the residual vector after approximating f in the direction of g_γ.
MP iteratively decomposes the residue R_f by projecting it on a vector of D that matches R_f at best.

2.2 EEG denoising

The denoising of EEG signals is performed using techniques developed from the field of signal processing, but more suited to the specific characteristics of EEG signals and artifacts. Here we present the most common methods.

2.2.1 Regression

The regression method[11], considers the EEG signal to be a linear combination of the artifacts and the clean EEG signal. In particular, the EEG signal is modeled as:

$$EEG = \alpha \cdot EMG + \beta \cdot EOG + \gamma \cdot ECG + \delta \cdot CEEG \tag{2.12}$$

where EEG is the registered EEG signal, EMG is the muscle artifact , EOG is the ocular artifact, ECG is the cardiac artifact and $CEEG$ is the clean EEG signal. The regression coefficients $\alpha, \beta, \gamma, \delta$ are estimated using the least squares method. The regression method is simple and fast, but it doesn't take into account the temporal correlation between the artifacts and the EEG signal, the so called cross terms. Furthermore it requires the knowledge of the artifacts signals.

2.2.2 Blind source separation

Blind source separation (BSS)[12] is a class of methods used to separate a mixture of signals into their individual components. There are several BSS algorithms, all based on an unsupervised component separation. The most common BSS algorithms are the Principal Component Analysis (PCA), the Principal Component Analysis (ICA) and the Canonical Correlation Analysis.

Principal Component Analysis

The PCA[13] is a method used to find a linear transformation of the mixture signals that maximizes the variance of the transformed signals. First, data are standardized by subtracting the mean and dividing by the standard deviation.

$$\mathbf{Z} = (\mathbf{X} - \mathbf{m})/\mathbf{s} \tag{2.13}$$

Then, the covariance matrix is computed.

$$\mathbf{C} = \frac{1}{n-1}\mathbf{Z}^T\mathbf{Z} \tag{2.14}$$

The eigenvectors of the covariance matrix are the principal components of the data while the eigenvalues are the variances of the principal components.

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{2.15}$$

The principal components found by projecting x onto those perpendicular basis vectors are uncorrelated, and their directions orthogonal. This scheme is very efficient in redundancy reduction, and can be said to maximize the amount of information spanned by a subset of dimensions of the initial vector.

Independent Component Analysis

The ICA[14] is a method used to find a linear transformation of the mixture signals that maximizes the non-Gaussianity of the transformed signals. It consists in decomposing the signal in Independent components and discarding the ones containing artifacts. In blind source separation, the original independent sources are assumed to be unknown, and we only have access to their weighted sum.[15]
The signal is modeled as:

$$x(t) = \sum_{i=1}^{N} a_i \cdot s_i(t) \tag{2.16}$$

where $x(t)$ is the mixture signal, $s_i(t)$ is the i-th source signal and a_i is the mixing coefficient.
The ICA algorithm aims to find the source signals $s_i(t)$ and the mixing coefficients a_i that maximize the non-Gaussianity of the mixture signal $x(t)$. It is based on the assumption that the sources are statistically independent.

Canonical Correlation Analysis

The CCA[16] is a method used to find a linear transformation of the mixture signals that maximizes the correlation between the transformed signals. More specifically, CCA identifies two sets of variables, X and Y, and finds linear combinations of X and Y that are maximally correlated. These linear combinations are called canonical variates. The first canonical variate is the linear combination of X and Y that has the highest correlation, and each subsequent canonical variate is the linear combination that has the highest correlation subject to being orthogonal to the previous canonical variates.

2.2.3 Empirical Mode Decomposition

The EMD[17] is a method used to decompose a signal into a set of intrinsic mode functions (IMFs). The EMD is a data-driven method that does not require any prior knowledge of the signal.
It decomposes a signal into a finite number of intrinsic mode functions (IMFs), which are functions that capture the local behavior of the signal. EMD is based on the concept of sifting, which involves iteratively extracting the local maxima and minima of the signal to generate IMFs. The basic steps of EMD are as follows:

- Given a signal $x(t)$, find all of its local maxima and minima.
- Interpolate between the local maxima and minima to create an upper and lower envelope for the signal. This step effectively eliminates the high-frequency components of the signal and captures its slowly varying behavior.
- Calculate the mean of the upper and lower envelopes to obtain a first IMF, $c_1(t)$.

- Subtract $c_1(t)$ from the original signal to obtain a new signal, $r_1(t) = x(t) - c_1(t)$.
- Repeat steps 1-4 on the residual signal $r_1(t)$ to obtain the second IMF, $c_2(t)$.
- Continue this process until a stopping criterion is met, such as the number of IMFs or the amplitude of the residual signal falling below a certain threshold.

The resulting IMFs are functions that oscillate around zero with a characteristic scale and capture the local behavior of the signal at different scales. The final residual signal is a monotonic function that represents the long-term trend of the signal.

2.2.4 Filtering techniques

Adaptive filtering

Adaptive filtering[18] is a method used to estimate the unknown input signal from the noisy output signal. It is based on the assumption that the input signal is a linear combination of the unknown input signal and the noise.
The basic steps of adaptive filtering are as follows:

- Given a noisy signal $y(t)$, estimate the unknown input signal $x(t)$.
- Initialize the filter coefficients w_0.
- For each sample $y(t)$, compute the error signal $e(t)$.
- Update the filter coefficients $w(t)$.
- Repeat steps 2-4 until a stopping criterion is met.

The error signal is defined as the difference between the noisy signal and the estimated input signal. The filter coefficients are updated according to the following equation:

$$w(t+1) = w(t) + \mu \cdot e(t) \cdot x(t) \tag{2.17}$$

where μ is the step size.

Wiener filtering

Wiener filtering[19] is a method used to estimate the unknown input signal from the noisy output signal. It is based on the assumption that the input signal is a linear combination of the unknown input signal and the noise.
The basic steps of Wiener filtering are as follows:

- Given a noisy signal $y(t)$, estimate the unknown input signal $x(t)$.
- Compute the power spectral density (PSD) of the noise signal $n(t)$.
- Compute the PSD of the input signal $x(t)$.

- Compute the PSD of the output signal $y(t)$.
- Compute the Wiener filter coefficients $w(t)$.
- Apply the Wiener filter to the noisy signal $y(t)$ to obtain the estimated input signal $x(t)$.

The Wiener filter coefficients are computed according to the following equation:

$$w(t) = \frac{S_x(t)}{S_x(t) + S_n(t)} \tag{2.18}$$

where $S_x(t)$ is the PSD of the input signal $x(t)$ and $S_n(t)$ is the PSD of the noise signal $n(t)$.

2.3 Machine learning

The latest approaches involve the use of machine learning techniques to denoise EEG signals. Machine learning is a subfield of artificial intelligence that focuses on the development of computer programs that can learn from data. In the following chapter we will discuss the different architectures of neural networks that have been used in previous works and that will be exploited in the proposed approach.

2.3.1 Fully connected neural network

Fully Connected Neural Networks, also known as Feedforward Neural Networks or Multilayer Perceptrons, are the the most common type of neural networks. They consist of neurons[6.1.1], organized in layers and each of the neurons in a layer is connected to all the ones in the previous and subsequent layers. These connections are weighted, and each neuron computes a weighted sum of its inputs, passes this sum through an activation function, and then forwards the result to the next layer.

The architecture of a Fully Connected Neural Network consists of one or more layers of neurons, including an input layer, one or more hidden layers, and an output layer. The input layer receives the data to be processed, and each neuron in this layer represents a feature of the input. The hidden layers perform complex computations on the input data, and the output layer produces the final output of the network. The number of neurons in the input and output layers is determined by the size of the input and output data, while the number of neurons in the hidden layers is a hyperparameter that can be tuned to improve the performance of the network.

The operation of a fully connected neural network can be described mathematically using the following equations.

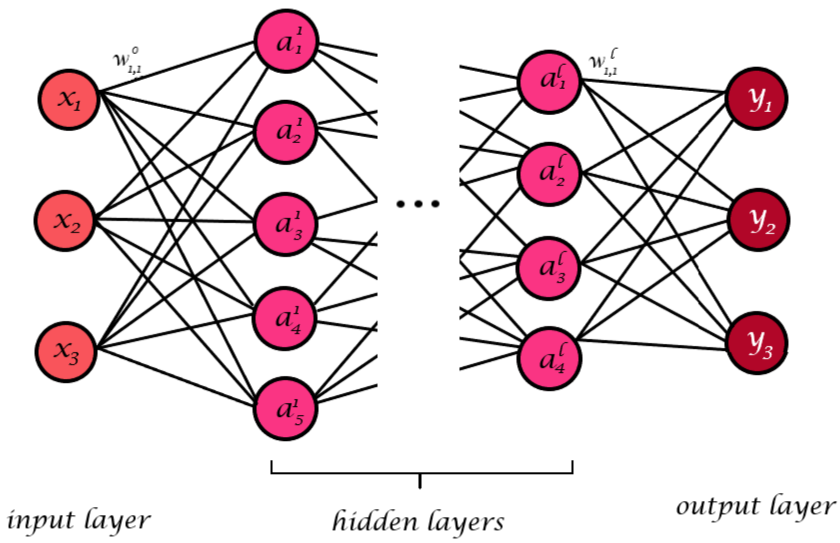

Figure 2.4. Fully connected neural network

Linear transformation

Each neuron in the hidden layers and output layer performs a linear transformation of its inputs, which is a weighted sum of the outputs of the neurons in the previous layer. The linear transformation of neuron i in layer l can be expressed as:

$$z_i^l = \sum_{j=1}^{n^{l-1}} w_{ij}^l a_j^{l-1} + b_i^l \tag{2.19}$$

where n^{l-1} is the number of neurons in the previous layer, w_{ij}^l is the weight of the connection between neuron j in layer l-1 and neuron i in layer l, a_j^{l-1} is the output of neuron j in layer l-1, b_i^l is the bias term of neuron i in layer l, and z_i^l is the pre-activation value of neuron i in layer l.

Activation function

The pre-activation values of each neuron are passed through an activation function to introduce non-linearity into the network. Commonly used activation functions include sigmoid, tanh, ReLU, and softmax. The activation function of neuron i in layer l can be expressed as:

$$a_i^l = \sigma(z_i^l) \tag{2.20}$$

where σ is the activation function.

Output computation

The output of the network is produced by the output layer, which applies a final activation function to the pre-activation values of its neurons. The activation function of the output layer depends on the task the network is designed to perform. The output of neuron i in the output layer can be expressed as:

$$\hat{y}_i = \sigma(z_i^L) \tag{2.21}$$

where L is the index of the output layer, σ is the output activation function, and $\hat{y}_i$ is the predicted value of the i-th output.

The network is trained by adjusting the weights and biases using backpropagation [6.1.3] with respect to a loss function [6.1.2].

2.3.2 Convolutional neural network

Convolutional Neural Networks (CNNs) are a type of artificial neural network architecture commonly used in deep learning for multidimensional data analysis. Unlike fully connected neural networks, CNNs take into account the spatial structure of input data, such as images, by using convolutional layers that apply filters to local regions of the input.

The architecture of a CNN consists of one or more convolutional layers paired with their activation functions, followed by one or more fully connected layers. The convolutional layers perform feature extraction by applying filters to local regions of the input data, producing feature maps that highlight different aspects of the input data. The fully connected layers perform classification or regression on the feature maps produced by the convolutional layers.

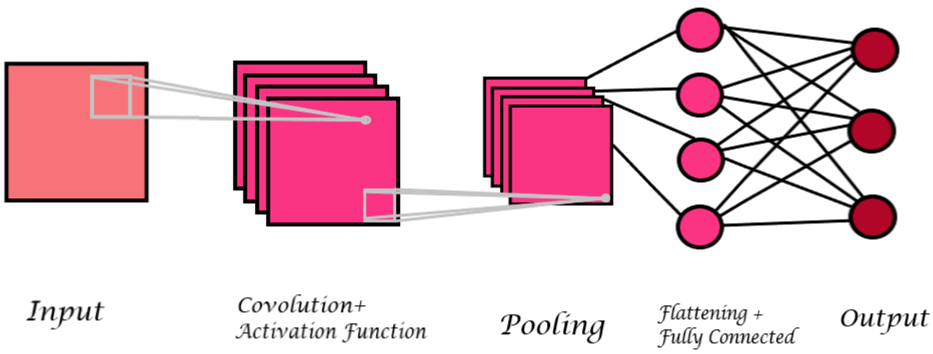

Figure 2.5. Convolutional neural network

The operation of a CNN can be described mathematically using the following equations:

Convolutional layer

In the convolutional layers, a set of filters or kernels is applied to local regions of the input data to produce feature maps. The convolution operation can be expressed as:

$$z_{i,j,k} = \sum_{l=1}^{d_{in}} \sum_{m=1}^{f} \sum_{n=1}^{f} x_{i+m-1,j+n-1,l} w_{m,n,l,k} + b_k \tag{2.22}$$

where x is the input data, w is the filter, b is the bias, d_{in} is the number of input channels, f is the size of the filter, z is the output feature map, and i, j, k are the indices of the feature map.

Pooling layer

The pooling layer downsamples the output of the convolutional layer, reducing the size of the feature maps and introducing translation invariance. Commonly used pooling operations include max pooling and average pooling that perform a max or average operation on local regions of the input data, respectively.

Flattening

The output of the pooling layer is flattened into a single vector, with dimension equal to the number of neurons of the input layer of the fully connected part of the network, to which it is passed.

Fully connected layer

The fully connected layers perform classification or regression on the feature maps produced by the convolutional and pooling layers. The operation of a fully connected layer is similar to that of a fully connected neural network.

2.3.3 Long short-term memory

Long Short-Term Memory (LSTM) Networks are a type of recurrent neural network (RNN) specifically designed to address the vanishing gradient problem[6.1.4] in traditional RNNs. LSTMs have a memory cell that is able to maintain information over long periods of time, and they use three gates - the input gate, the forget gate, and the output gate - to regulate the flow of information into and out of the memory cell. Here are the key equations that govern the behavior of an LSTM:

Input gate

The input gate determines how much of the new input should be added to the memory cell. The equation for the input gate is:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.23}$$

where i_t is the input gate activation vector, x_t is the input vector at time t, h_{t-1} is the hidden state vector from the previous time step, W_i is the weight matrix for the input gate, and b_i is the bias vector for the input gate.

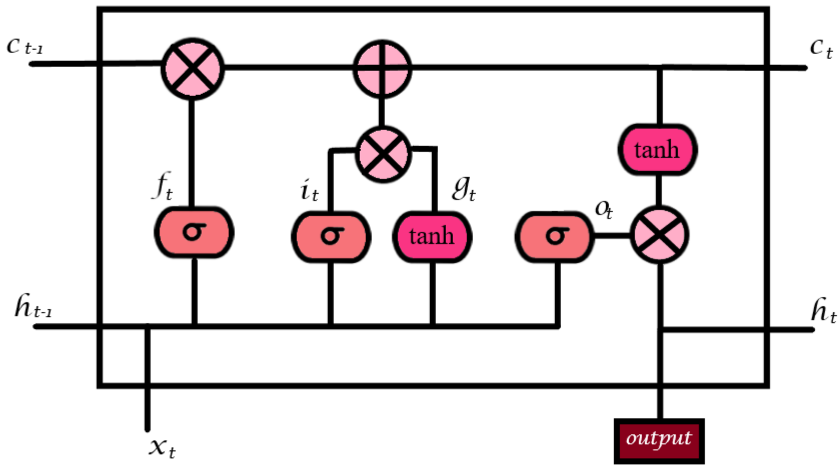

Figure 2.6. Long Short Term Memory Architecture

Forget gate

The forget gate determines how much of the previous memory cell state should be retained. The equation for the forget gate is:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.24}$$

where f_t is the forget gate activation vector, x_t is the input vector at time t, h_{t-1} is the hidden state vector from the previous time step, W_f is the weight matrix for the forget gate, and b_f is the bias vector for the forget gate.

Candidate memory cell state

The memory update combines the input with the previous memory cell state to create a new memory cell state. The equation for the memory update is:

$$g_t = \tanh(W_g[h_{t-1}, x_t] + b_g) \tag{2.25}$$

where g_t is the candidate memory cell state, x_t is the input vector at time t, h_{t-1} is the hidden state vector from the previous time step, W_g is the weight matrix for the memory update, and b_g is the bias vector for the memory update.

Output gate

The output gate determines how much of the new memory cell state should be output. The equation for the output gate is:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.26}$$

where o_t is the output gate activation vector, x_t is the input vector at time t, h_{t-1} is the hidden state vector from the previous time step, W_o is the weight matrix for the output gate, and b_o is the bias vector for the output gate.

Hidden state update

The new memory cell state is combined with the output gate to create the new hidden state. The equation for the hidden state update is:

$$h_t = o_t \tanh(c_t) \tag{2.27}$$

where h_t is the new hidden state, o_t is the output gate activation vector, c_t is the new memory cell state, and tanh is the hyperbolic tangent function.

These equations allow an LSTM network to selectively forget or remember information over time, making it well-suited for modeling sequences of data where long-term dependencies are important

2.3.4 Transformer

The Transformer is a neural network architecture designed for natural language processing tasks. It was introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017[**?**], and has since become a cornerstone of many state-of-the-art language models.

The basic architecture of the Transformer consists of an encoder and a decoder, both of which are composed of a stack of identical layers. Each layer has two sub-layers: a multi-head self-attention mechanism, followed by a position-wise fully connected feed-forward network.

The self-attention mechanism is a mechanism that allows the model to attend to different parts of the input sequence. Given an input sequence of length n, it computes a set of n output vectors by taking a weighted sum of all the input vectors, where the weights are computed by a softmax function over a set of attention scores. These scores are computed by taking the dot product between a query vector and all the key vectors, which are learned parameters of the model. The query and key vectors are obtained by projecting the input vectors into a high-dimensional space, which is learned during training.

The equations for the self-attention mechanism can be written as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q, K, and V are the query, key, and value matrices, respectively, each of which has dimensions $n \times d_k$, $n \times d_k$, and $n \times d_v$, and d_k and d_v are the dimensions of the query/key and value spaces, respectively.

The multi-head attention mechanism allows the model to attend to different aspects of the input sequence simultaneously. It does this by computing h different sets of query, key, and value vectors, each of which is obtained by projecting the input vectors into d_k, d_k, and d_v-dimensional spaces, respectively. The resulting h sets of output vectors are concatenated and linearly transformed to produce the final output.

The equations for the multi-head attention mechanism can be written as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O$$

where:

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, and W_i^V are learned projection matrices for the i-th head, each of which has dimensions $d_{model} \times d_k$, $d_{model} \times d_k$, and $d_{model} \times d_v$, respectively. W^O is a learned projection matrix that maps the concatenated output of the attention heads back to the original dimensionality of the input.

The position-wise feed-forward network is a simple two-layer neural network that operates on each position of the input sequence independently. It is applied to each output vector of the self-attention mechanism, and has the effect of transforming the vector into a higher-dimensional space, followed by a projection back into the original dimensionality.

The equations for the position-wise feed-forward network can be written as follows:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

In this equation, x is the input vector of size d_{model}, W_1 and W_2 are learned weight matrices of sizes $d_{model} \times d_{ff}$ and $d_{ff} \times d_{model}$, respectively, b_1 and b_2 are learned bias vectors of size d_{ff} and d_{model}, respectively, and ReLU is the rectified linear unit activation function. The output of the position-wise feed-forward network is also a vector of size d_{model}.

Chapter 3

Methodology

3.1 Data

The data used in this thesis are the EEG signals of the EEGdenoiseNet dataset[20]. It is comprehensive benchmark dataset that proves to be an exemplary resource for training and evaluating deep learning-based EEG denoising models. It is composed of 4514 clean EEG epochs, 3400 EOG epochs, and 5598 EMG epochs, making it an inclusive and extensive dataset that caters to the needs of researchers and practitioners alike.

The incorporation of these various types of epochs in the dataset enables to generate a considerable number of noisy EEG epochs with their corresponding ground truth data. As a result, this compilation has proven to be an invaluable resource for model training and testing, facilitating an enhanced understanding of EEG denoising.

3.2 Preprocessing

3.3 Denoising

3.4 Machine learning

Chapter 4

Experimental results

4.1 Results

4.2 Discussion

Chapter 5

Conclusions and future work

5.1 Conclusions

5.2 Future work

Chapter 6

Appendix

6.1 Neural network

6.1.1 Neuron

A neuron, also known as a node or a perceptron, is a fundamental building block of a neural network. It is a mathematical function that receives one or more inputs and produces an output based on those inputs.
A typical neuron in a neural network consists of three main components:

Inputs: These are the signals or information that the neuron receives from other neurons or from the outside world. Each input is assigned a weight, which determines how important it is to the neuron's output.

Activation Function: The activation function is a non-linear function that takes the weighted sum of the inputs and produces an output. The output of the activation function is then passed on to the next layer of neurons in the network.

Bias: The bias is an additional input that is used to adjust the output of the activation function. It can be thought of as a threshold that the weighted sum of the inputs must cross before the neuron "fires" or produces an output.

6.1.2 Loss function

A loss function is a mathematical function that measures the difference between the predicted output of the network and the actual output (or "ground truth") for a given set of input data.
The goal of a neural network is typically to minimize the difference between the predicted output and the ground truth. The loss function provides a quantitative measure of this difference, which the network can use to adjust its parameters (such as the weights and biases of the neurons) in order to improve its predictions.
There are many different types of loss functions that can be used in a neural network, depending on the specific task the network is trying to perform. Some common examples include:

- Mean Squared Error (MSE): This is a popular loss function for regression problems, where the goal is to predict a continuous output value. It measures the average squared difference between the predicted and actual values.
- Binary Cross-Entropy: This is a loss function that is commonly used for binary classification problems, where the goal is to predict a binary output (e.g., 0 or 1). It measures the difference between the predicted probability of the positive class and the true probability (which is either 0 or 1).
- Categorical Cross-Entropy: This is a loss function that is commonly used for multi-class classification problems, where the goal is to predict one of several possible output classes. It measures the difference between the predicted probability distribution over the classes and the true distribution (which is typically represented as a one-hot encoded vector).

By minimizing the loss function during training, the neural network can adjust its parameters to improve its predictions on the training data. However, it's important to also evaluate the performance of the network on a separate set of validation or test data to ensure that it is not overfitting to the training data.

6.1.3 Backpropagation

The weights and biases of the network are updated during training using an algorithm called backpropagation, which computes the gradient of the loss function with respect to the network parameters. The gradient is then used to update the weights and biases of the network using an optimization algorithm such as stochastic gradient descent.

6.1.4 Vanishing gradient problem

The vanishing gradient problem is a common problem in deep neural networks that occurs when the gradient of the loss function with respect to the weights and biases of the network is very small. This problem can be solved by using a different activation function, such as the rectified linear unit (ReLU) activation function, which does not suffer from the vanishing gradient problem.

Bibliography

[1] Jiang, X.; Bian, G.-B.; Tian, Z. Removal of Artifacts from EEG Signals: A Review. Sensors 2019, 19, 987.

[2] Wajid Mumtaz, Suleman Rasheed, Alina Irfan, Review of challenges associated with the EEG artifact removal methods, Biomedical Signal Processing and Control, Volume 68, 2021, 102741, ISSN 1746-8094

[3] Boashash, B., ed. (2003), Time–Frequency Signal Analysis and Processing: A Comprehensive Reference, Oxford: Elsevier Science, ISBN 978-0-08-044335-5.

[4] Cooley, James W. (1987). The Re-Discovery of the Fast Fourier Transform Algorithm (PDF). Microchimica Acta. Vol. III. Vienna, Austria. pp. 33–45. Archived (PDF) from the original on 2016-08-20.

[5] R. Álvarez, E. Borbor and F. Grijalva, ”Comparison of methods for signal analysis in the time-frequency domain,” 2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM), 2019, pp. 1-6, doi: 10.1109/ETCM48019.2019.9014860.

[6] W. -k. Lu and Q. Zhang, "Deconvolutive Short-Time Fourier Transform Spectrogram," in IEEE Signal Processing Letters, vol. 16, no. 7, pp. 576-579, July 2009, doi: 10.1109/LSP.2009.2020887.

[7] E. Chassande-Mottin and A. Pai, "Discrete time and frequency Wigner-Ville distribution: Moyal's formula and aliasing," in IEEE Signal Processing Letters, vol. 12, no. 7, pp. 508-511, July 2005, doi: 10.1109/LSP.2005.849493

[8] S. Zhou, B. Tang and R. Chen, "Comparison between Non-stationary Signals Fast Fourier Transform and Wavelet Analysis," 2009 International Asia Symposium on Intelligent Interaction and Affective Computing, Wuhan, China, 2009, pp. 128-129, doi: 10.1109/ASIA.2009.31.

[9] V. M. Pukhova and M. S. Stepanova, "Up-Chirplet and Down-Chirplet Transforms of Non-Stationary Signals," 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, Russia, 2019, pp. 1221-1225, doi: 10.1109/EIConRus.2019.8657179.

[10] S. G. Mallat and Zhifeng Zhang, "Matching pursuits with time-frequency dictionaries," in IEEE Transactions on Signal Processing, vol. 41, no. 12, pp. 3397-3415, Dec. 1993, doi: 10.1109/78.258082.

[11] Klados, M.A.; Papadelis, C.; Braun, C.; Bamidis, P.D. REG-ICA: A hybrid methodology combining blind source separation and regression techniques for the rejection of ocular artifacts.Biomed. Signal Process Control

[12] J. A. Nascimento, J. M. S. Nascimento, and J. M. C. Nascimento, "Blind source separation: A review," in IEEE Signal Processing Magazine, vol. 26, no. 6, pp. 22-38, Nov. 2009, doi: 10.1109/MSP.2009.937000.

[13] Casarotto, S.; Bianchi, A.M.; Cerutti, S.; Chiarenza, G.A. Principal component analysis for reduction of ocular artefacts in event-related potentials of normal and dyslexic children. Clin. Neurophysiol. 2004, 115, 609–619.

[14] Jung, T.P.; Makieg, S.; Bell, A.J.; Sejnowski, T.J. Independent component analysis of electroencephalographic and event-related potential data. Cent. Audit. Process. Neural Model. 1996, 2, 1548–1551.

[15] Vigário, R. Extraction of ocular artifacts from EEG using independent component analysis. Electroencephalogr. Clin. Neurophysiol. 1997, 103, 395–404

[16] N. Robinson, K. P. Thomas and A. P. Vinod, "Canonical correlation analysis of EEG for classification of motor imagery," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2317-2321, doi: 10.1109/SMC.2017.8122967.

[17] V. Bajaj and R. B. Pachori, "Classification of Seizure and Nonseizure EEG Signals Using Empirical Mode Decomposition," in IEEE Transactions on Information Technology in Biomedicine, vol. 16, no. 6, pp. 1135-1142, Nov. 2012, doi: 10.1109/TITB.2011.2181403.

[18] He, P., Wilson, G. Russell, C. Removal of ocular artifacts from electro-encephalogram by adaptive filtering. Med. Biol. Eng. Comput. 42, 407–412 (2004). https://doi.org/10.1007/BF02344717

[19] Somers, B., Francart, T., Bertrand, A. (2018). A generic EEG artifact removal algorithm based on the multi-channel Wiener filter. Journal of neural engineering, 15(3), 036007.

[20] EEGdenoiseNet: A Comprehensive Benchmark Dataset for EEG Denoising, https://arxiv.org/abs/2103.03894