Master Thesis

John Bosco Uroko

October 25, 2019, Utrecht

UROKO John Bosco

Development of a generalized "brain-click" classifier

Master thesis

Institute of Neural Engineering
University of Technology Graz
Stremayrgasse 16/IV, A - 8010 Graz
Supervisors: Prof. Dr. N.F. Ramsey, Prof. Dr. G. Müller-Putz

Co-Supervisor: Dr. E.J. Aarnoutse, Dr. Z. Freudenburg

Graz, October 25, 2019

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum / Date

Unterschrift / Signature

Table of content

1 Abstract

Objective: The main goal of this project is to create a neural network model that uses time signals to classify if a participant tries to generate a "brain-click" or not. The intention hereby is that the neural network achieves good performances for different kinds of locked-in patients.

Methods: In an effort to develop a generalized brain-click classifier the created neural network must be tested and evaluated continuously. But it is also important to know how to prepare the different datasets for each participant.

Results: The results show the performance of the model for each participant and why it works better for one than the other. Additionally a little insight into the neural network is given, and how it works with the given data.

Conclusion: The results show that the neural network works very good as long as certain requirements are met. The created model is even capable of adding additional classes and achieving reasonable results. The project also shows that the model could be improved by means of fine-tuning.

2 Zusammenfassung

Ziel: Das Hauptziel dieses Projekts ist es, ein neuronales Netzwerk zu erstellen, das Zeitsignale zur Klassifizierung verwendet. Dabei soll das Netzwerk erkennen, ob ein Teilnehmer versucht, "brain-clicks" zu erzeugen oder nicht. Ein weiteres Ziel ist, das neuronale Netz so generalisierbar zu gestalten, dass es für Menschen, die aus verschiedenen Gründen unter dem Locked-In-Syndrom leiden, gut funktioniert.

Methoden: In dem Bestreben, einen generalisierten Klassifikator zu entwickeln, muss das erzeugte neuronale Netzwerk kontinuierlich getestet und evaluiert werden. Außerdem ist die individuelle Aufbereitung der Datensätze für jeden Teilnehmer, von großer Wichtigkeit.

Ergebnisse: Die Ergebnisse zeigen die Leistung des Modells für jeden Teilnehmer und warum es für den einen besser funktioniert als für den anderen. Zusätzlich wird ein kleiner Einblick in die Funktion des neuronale Netzwerks geboten, und auch wie gut es die unterschiedlichen Datensätze klassifiziert.

Schlussfolgerung: Die Ergebnisse zeigen, dass das neuronale Netzwerk unter bestimmten Umständen sehr gut funktioniert. Das erstellte Modell ist sogar in der Lage, nach der Addition von weiteren Klassen trotzdem vernünftige Ergebnisse zu erzielen. Das Projekt zeigt auch, dass das Modell durch die Anpassung verschiedener Parameter zusätzlich verbessert werden könnte.

3 Introduction

3.1 Motivation

Humans suffering from a Locked-in syndrome (LIS) are aware of their environment but are not able to execute any voluntary muscle movements and therefore can not communicate with other people. The reason is that people in this state are not able to execute any voluntary muscle movements such as the vocal apparatus, which is needed for speech. A common exception, are muscles needed for eye movements. Although there is no cure for this disorder at the moment, there are ways of improving those people's life with the help of assistive technologies [2]. An often seen measure is to use an eye-tracker as long as eye movement is unimpaired.

Different conditions can lead to LIS, e.g. Amyotrophic lateral sclerosis (ALS), brainstem strokes, traumatic brain injuries, etc. But regardless of the cause, assistive technologies do not always work. For such circumstances, Brain-Computer Interfaces (BCI) are a good alternative. A BCI is a method used to bypass the neural communication pathway by recording brain signals with e.g., electrodes and using those to control devices, prosthesis, etc. People with impaired neural pathways benefit from those technologies.

There are different methods for controlling portable BCI-devices in real-time, which are for example Electroencephalography (EEG)[3], Elektrocorticogramm (ECoG)[1] or the Utah intracortical electrode array (UEA) [4]. These methods have different characteristics as seen in Table 1.

Method	Invasiveness	Anatomical level
EEG	non-invasiv	Scalp
ECoG	minimal invasiv	Subdural
UEA	invasiv	Cortex

Table 1: BCI methods with there level of invasiveness and anatomical level

3.2 Utrecht Neuroprosthesis - State of the art

3.2.1 A fully implanted BCI in a Locked-In patient with ALS

The Utrecht NeuroProsthese (UNP) group at the University Medical Center (UMC) Utrecht has accomplished controlling a communication device with the help of BCI [1]. For that purpose, ECoG electrodes were implanted in an ALS patient. To use the system, the operator needs to attempt a hand movement to execute a "brain-click". This click is then used to choose a letter seen on a display (Fig.1-D).
For the existing system [1] two electrode strips were implanted over the sensorimotor cortex (M1) and the dorsolateral prefrontal cortex (DLPFC) (Fig.1-A). The measured signals are transferred to an antenna with the help of a transmitter implanted under the operator's chest (Fig.1-B).
The transmitter either transfers the 200Hz time-signal or the power-signals for the beta and gamma frequency bands in a 5Hz rate. The signals source is the M1-electrode-pair.

To distinguish clicks, from non-clicks different approaches are possible. In the past, the linear discriminant analysis (LDA) method was the gold standard when it came to classifying data. Also the currently used method uses different frequency bands which are then classified with the help of LDA. But with the uprising of machine learning (ML), the favourised method is changing gradually. This project therefore shows an ML-model that does not need frequency-bands but the raw time signal.

The main goal of this project is to create a machine learning model that improves the overall performance over the LDA model. Although the current system works very well, it is still far from perfect. Another goal is to create a generalized model that can be used for a second patient suffering from a brain-stem stroke.

3.2.2 BCI and Machine learning

There have been different approaches when it comes to controlling a BCI with different machine learning methods [5]. The method that is still used very often is feature extraction and selection. Therefore the signal is filtered (temporal and spatial), the frequency features are extracted and then the most contributing frequency bands are selected (Common Spatial Patterns [6]). The selected features are then used for classification.
It has been often shown that BCIs in combination with machine learning achieve better performances. But since there are different methods in machine learning and different features that can be used, there are also different approaches for trying to improve the performance.
Further one must keep in mind that there are different kinds of brain signals which need to be approached differently (e.g., SSVEP, P300,...).

- One possibility, for example, is the calculation of the frequency spectrogram of each signal and using those as input signals. As the input is a two-dimensional signal, Deep convolutional neural networks are often used [7].

- Another way of classification is to extract feature bands and use the extracted information for classification. Therefore different machine learning methods are often used. Such as neural networks, support vector machines, nearest neighbours or the Linear Discriminant Analysis (LDA) [8][9][10].

- A way of avoiding preprocessing is to use the raw time signal as input signal. First steps have been made for simple tasks and different machine learning methods have been applied to them (multi layer perceptron, decision tree, support vector machine, Nearest Neighbour Classifier) [11] [12].

To conclude, with the knowledge based on the UNP study and the use of an appropriate machine learning model, the aim of this project is to improve classification.

3.2.3 Future possibilities

To make the system even more versatile more methods were taken into consideration during this project. One possibility of improving the system is the automatic correction of falsely made clicks. This could speed up spelling in case many false clicks are made.
Another way of improving the system is to add a second click. The second brain click would be executed with another type of brain signal. This system could act as a left and right click of a computer mouse.

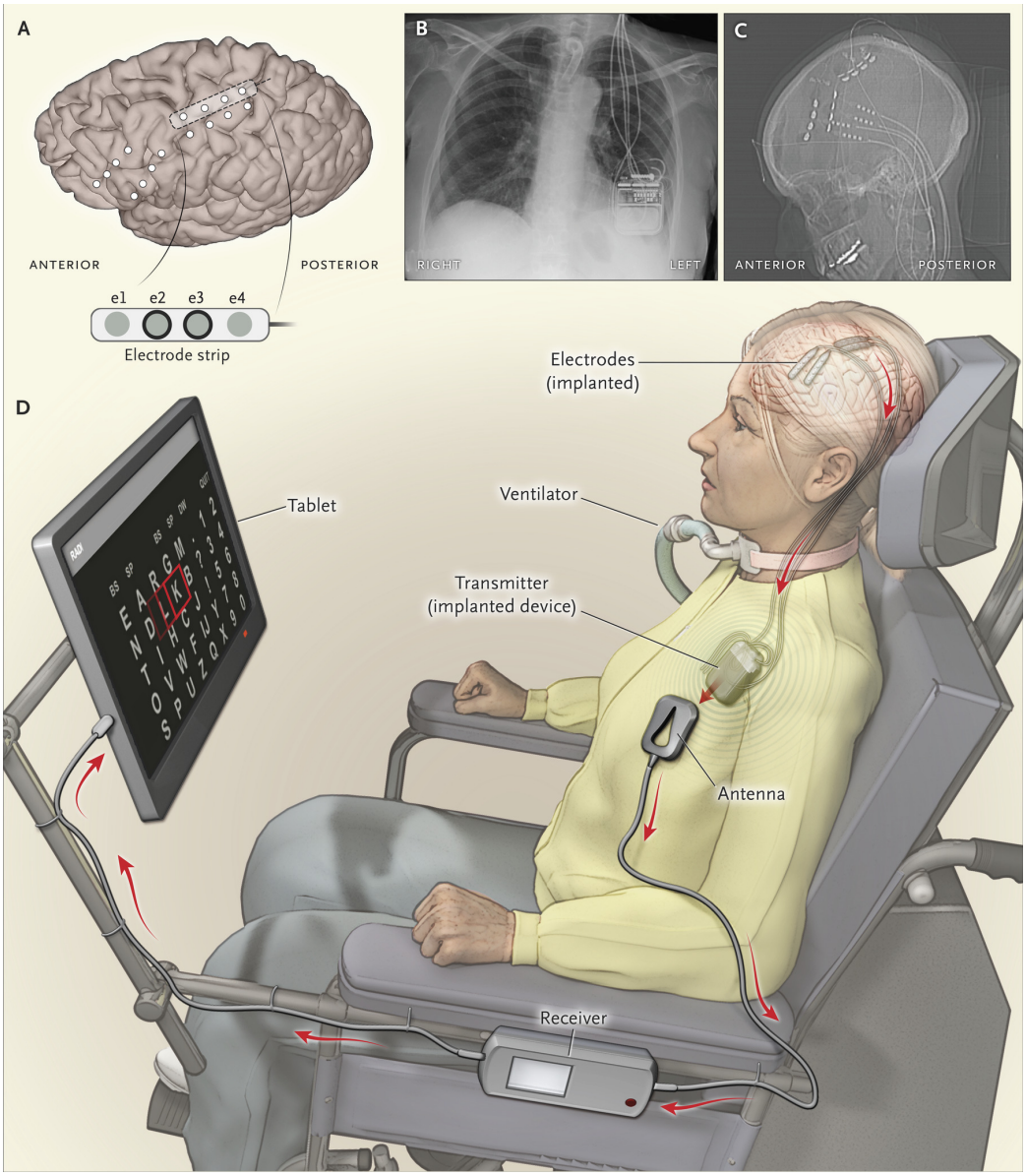


Figure 1: UNP-Setup for BCI communication [1]

4 Methods

The following topics describe the participants and which tasks were executed during signal recording. Further, it describes the data used for training the machine learning model. And at last, it describes the evaluation methods used for the machine learning model.

4.1 Participants

For testing the created models data of two participants were analyzed. Both are in a locked in state and can not communicate with other people without assistive technology. As described in chapter 3, participants were implanted with ECoG electrodes which were used for data acquisition. The first participant (UNP1) was diagnosed with ALS in 2008 and joined the study in 2015. She is severely paralyzed and anarthric as a result of her disease and she was 58 years old at the time of informed consent. During pre- and post-surgical neurological evaluation her sensibility was intact. The second participant (UNP4) suffered a brain-stem stroke in 2004 and joined the study in 2017. Since then she has been severely paralyzed and anarthric as a result of her disease and she was 39 years old at the time of informed consent[13].

4.2 Tasks

To train the machine learning model, data is used for training and testing it. Therefore different tasks were set up to extract trials.

4.2.1 Whack-A-Mole

The Whack-A-Mole (WAM) task as seen in Figure 2 is built up like the classic whack-a-mole game. The game board contains a grid of holes, whereas one of the holes contains a mole. The aim of the operator is to "hit" the mole with the help of a brain click.

At the beginning of the task, a selection box moves from top to bottom of the screen. As soon as a click is detected the box highlights and the selection box moves from left to right on the chosen row. If another click is detected, the chosen grid cell is highlighted. In case a row or cell without a mole was clicked the task restarts and the selection box jumps back to the top of the screen. The task also restarts after choosing the cell with a mole.
The period in which a cell/row can be selected is 4 sec, but can be defined manually.

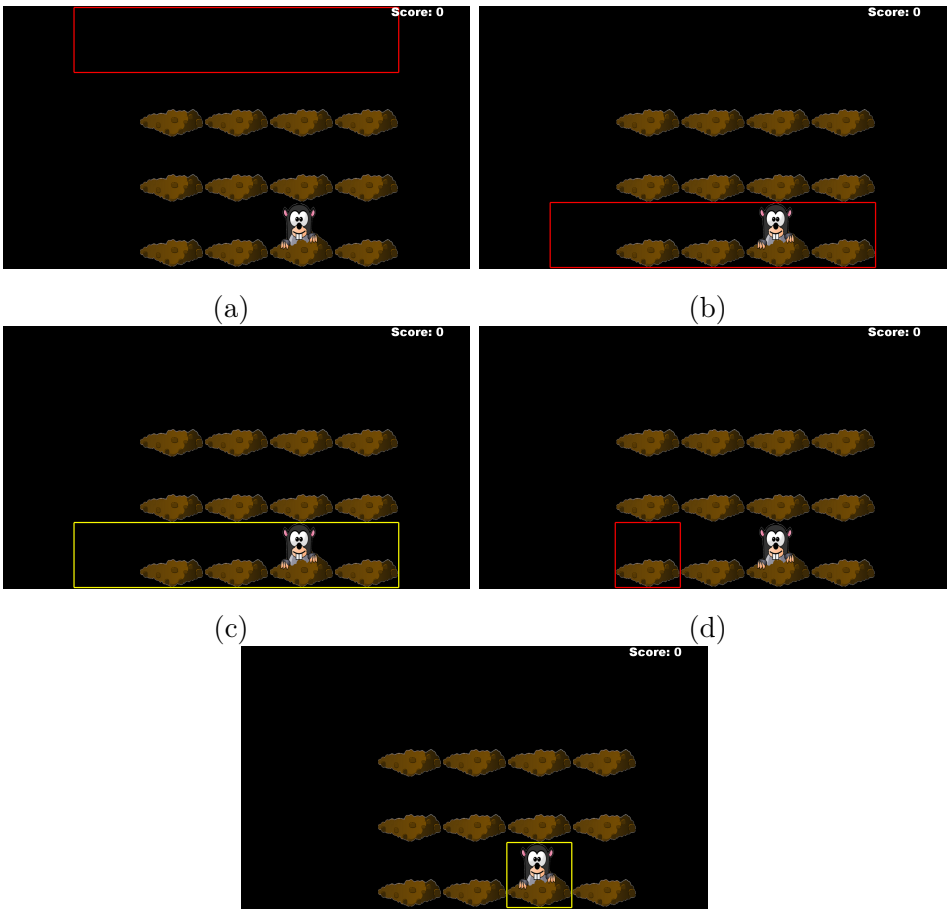


Figure 2: **WAM task:** (a) At the beginning the selection box stays above the grids of holes. (b) As soon as the task starts, the selection box starts moving downwards. This should be done until it reaches the row with the mole. (c) As soon as brain-click is detected the box highlights in yellow. (d) Then the box turns to a square and starts moving horizontally. (e) As soon as the square surrounds the mole another brain-click needs to be executed.

4.2.2 Continuous Whack-A-Mole

The continuous Whack-A-Mole (cWAM) task is very similar to the WAM. The only difference is that the game board is not a grid but a continuous

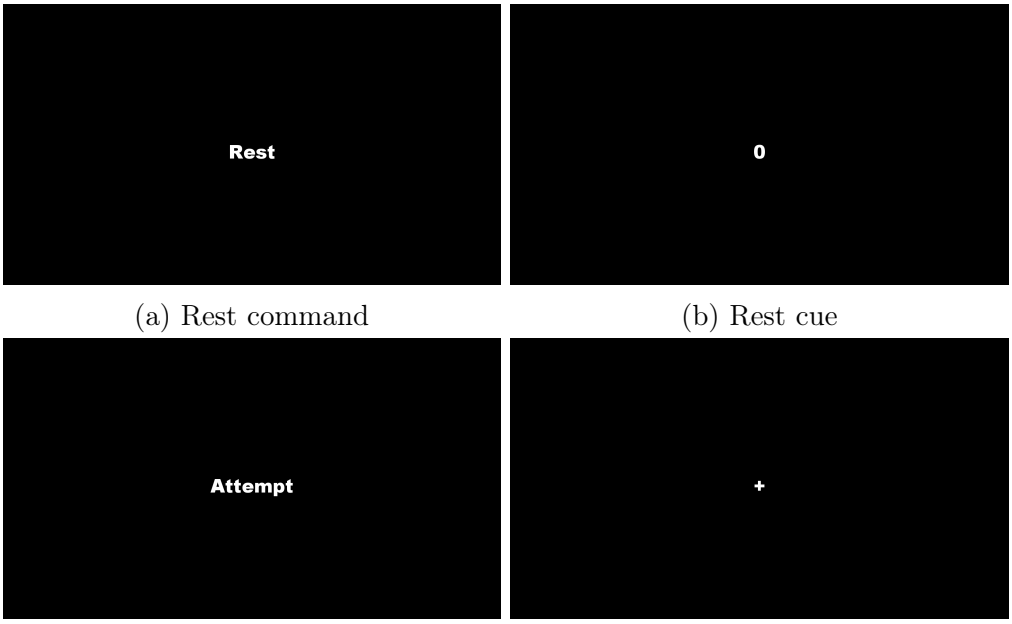
row of holes in which a mole appears from time to time. Therefore the selection box moves from left to right during the task. And as soon as the box surrounds a mole a click should be executed. The mole switches position if it is clicked or missed (moved past). On the display the selection box stands still and the mole moves from right to left. If the end of the row is reached, the task starts all over again.

Figure 3: Display of continuous Whack-A-mole task

4.2.3 Localizer Task

In Figure 4 screenshots of the Localizer task are shown. There are basically two different actions performed during the task which are "Rest"(Fig.4a) and "Attempt"(Fig.4c). During the Rest cue, the operator is told to relax and not to think of anything. During the Attempt cue, the operator needs to execute a previously defined action. In this case, the instruction was to attempt hand movement. The difference to the previous tasks (WAM, cWAM) is that movement attempt is continuously done over the whole cue period which lasts 15 sec.

This task has got several purposes. Before one can extract trials the best bipolar pair needs to be chosen. This is done with the help of this task. Another purpose is the extraction of the best features. After running this task multiple times the most promising frequency ranges are chosen for a brain-click detection. On the long run the Localizer task is used to investigate the stability of the brain signals.



(a) Rest command (b) Rest cue

(c) Attempt command (d) Attempt cue

Figure 4: Display of the standard Localizer task

4.3 Data Processing

4.3.1 Data Acquisition

The combination of implanted ECoG electrodes and amplifier/transmitter (Fig.1) is used for data acquisition. Therefore the recorded data from the electrodes is sent to the amplifier. The amplifier can then work in two different modes.

- **Power Mode**
  The power mode is used and designed for everyday usage. It transmits control-signals with a frequency of 5Hz. These control-signals are frequency bands extracted from the time-signal and sent every 0.2 sec. The advantage of this method is that signals transmitted at this pace are battery saving. This is very important because in case the battery needs to be changed, a surgery must be done. Therefore a long battery

life is essential. On the other hand, only little information is sent.

- **Time Mode**
  In Time Mode the amplifier transfers the recorded signals with a frequency of 200Hz. This mode depletes the battery of the device fast and therefore it should not be used permanently. It is mainly used for research purposes. Nevertheless, future implantable devices should be rechargeable which would make this mode the better choice.

4.3.2 Data Preparation

The data recorded in previous tasks was used to train the machine learning model. Therefore the signals needed some preparation. As already known there is a decrease in the low-frequency bands (beta[14]) and an increase in the high-frequency bands (gamma[15]) while a brain-click is executed. The rise and fall of frequency bands occur within a time range of approximately 4 sec before a click. This circumstance was used in previous models (see chapter 4.4.1) to classify whether a click occurred or not. The following analysis does not use the changes in the frequency bands directly, but the raw time signal.
Depending on the task, data preparation looked different.

- **Single execution task**
  Single execution tasks are WAM and cWAM tasks since just one hand movement attempt should be executed during the cue. To extract trials each cue is considered as its own trial. And for the initial classification and labelling of trials the frequency-band subtraction from chapter 4.4.1 is used. The trials were therefore labelled as True Positive, True Negative, False Positive
  In case a click is detected during the cue, the click is seen as a reference point and the required window is extracted depending on the click position.

In case no click is detected during the cue, the end of the cue is seen as a reference point and the required window is cut depending on the reference.

- **Multiple execution task**
  The only multiple execution task is the localizer task. During a cue, multiple attempts of a defined action are executed. Therefore another approach is chosen to extract multiple trials from one cue.

  a. One possibility is to use the frequency-band subtraction method on each cue. Each found click is then used again as a reference point and the respective windows are extracted for further use. This is the preferred way to extract cues because the click is time locked to the actual position of the click.

  b. Another way of extracting trials is to take the cue and cut windows (e.g., 4 sec windows) until the end of the cue. All the trials are then labelled as clicks or no clicks, depending on the cue.
  This method was used as first test to see how well the NN-model would perform with the localizer data. Further, the idea behind it was that the model would figure out itself what is a click or not.

  Both methods were tested in this project. But method a) was the one selected due to better results.

After extracting the trials they are labelled according to their features. The prepared dataset is then split into Training-, Validation- and Test-Datasets.

For this project the data that should be used for machine learning needs to be transformed into an appropriate format to work with it.

Signal properties:

Channels : M1 & DLPFC
Frequency : 200Hz

The labels for the existing data were "True Positive" (TP), "True Negative" (TN), "False Positive" (FP), "False Negative" (FN) and "Timing mistakes" (TM). Timing mistakes are False Positives that happened shortly after an actual cue.
To be able to work with the data the labels needed to be changed into "Click" and "No-Click". This was done as shown in table 2.
Depending on the signal of interest the trials were cut at the according position. Since the NN-model should predict clicks, time-windows before actual clicks were extracted. In further experiments the error potentials were of interest too, and therefore also post feedback data was included.

Old labels	New labels
TP	Click
TN	No-Click
FP	No-Click
FN	Click
TM	Click

Table 2: New labels corresponding to old labels

4.4 Neural Network

As machine learning model a neural network (NN) was chosen. Starting with a feed-forward neural network the model was extended by adding convolutional NN layers (CNN[16]) and recurrent NN layers (RNN). The extension of the network resulted from evaluations of different network configurations.

The network adaptation was also done to find a network with the best performance. The initial goal was the classification between click and non-click. With increasing performance, the addition of other classes was considered. For example the detection of error potentials after a falsely recognized brain-click.

4.4.1 Previous systems

Performance of the following systems were used to compare against the created NN-model in this work.

i. **Frequency-band subtraction**
   According to [1], it is already known that during motor movement attempt there is an increase in the Gamma frequency band and a decrease in the Beta frequency band (Fig.5). This circumstance was used to create a control signal, by calculating the difference between the Gamma (high-frequency band = HFB) and Beta (low-frequency band = LFB) frequency band.
   A click was therefore classified as one as soon as the result exceeded a previously defined threshold for a certain time.
   With this method, an average accuracy of 85% was achieved over 40 runs of the WAM task.

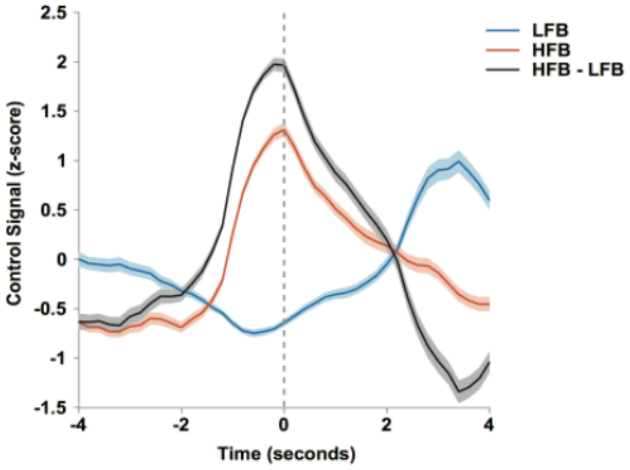


Figure 5: Frequency-band change during hand movement attempt[1]

ii. **Deep CNN**

A more sophisticated way was the use of a Deep CNN (Fig. 6). Therefore each trial in time domain is converted into a spectrogram. The spectrograms are then used for training, and testing how well the Deep CNN performs.

With this method, an average accuracy of 97% can be achieved.

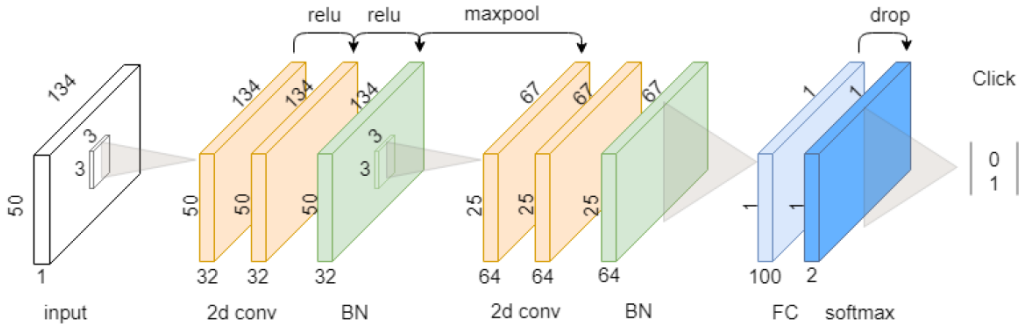


Figure 6: Deep CNN for classification of spectrogram trials [7]

Figure 7 shows an example of spectograms with different characteristics. One can clearly see that for true positive clicks the Gamma frequency increases and the Beta frequency decreases before a click. The same characteristics one can see in figure 5. False positives were possibly classified as positive clicks due to the short increase in Gamma frequency and the slight decrease in Beta. As expected the true negative does not show any significant changes. And probably the False negatives were classified as such because the difference between Gamma and Beta did not pass a certain threshold.

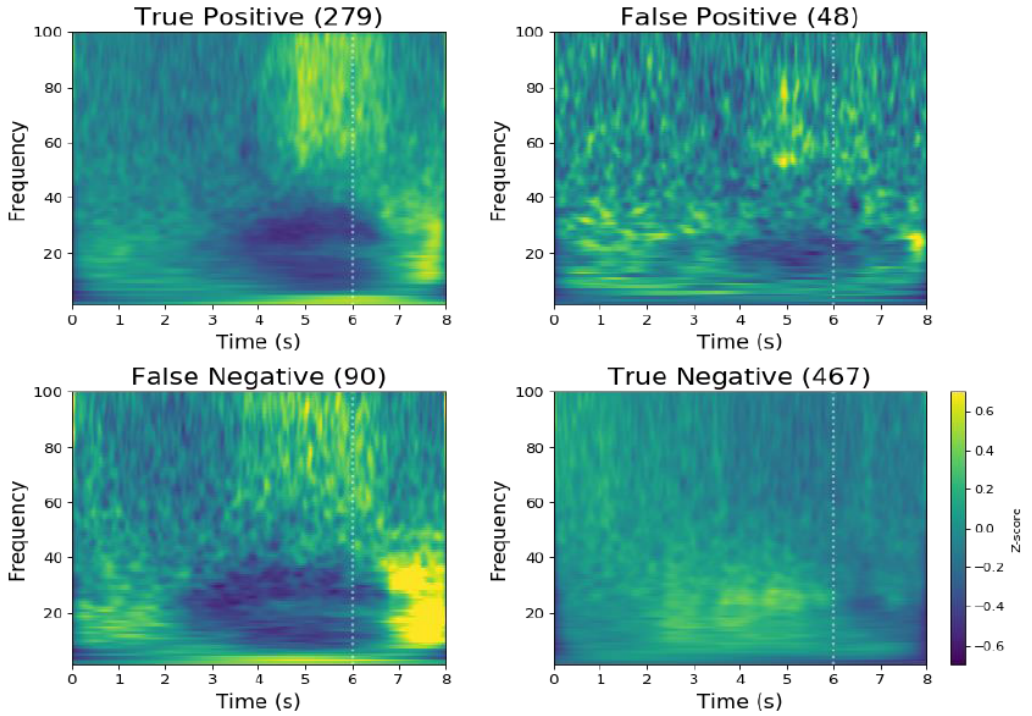


Figure 7: Spectrograms for different trials [7]. The number in brackets shows the amount of trials used for calculation of the respective spectogram.

4.4.2 NN Layers

Neural networks can have different types of properties. The most common layer types are described in the next points.

- **Dense Layer**
  Dense Layers in NN are layers with nodes in which each node is connected with each node of the previous layer(Fig. 8). It is very simple but also the least sophisticated NN layer. In the beginning, these layers were also used to create a simple Feedforward NN to calculate the accuracy. That means that if a trial is pushed into a NN-model, the model tries to find relations between each sample of the trial during training. Therefore the backpropagation method is used. It calculates the gradients of a function and tries to find a local minimum [19].

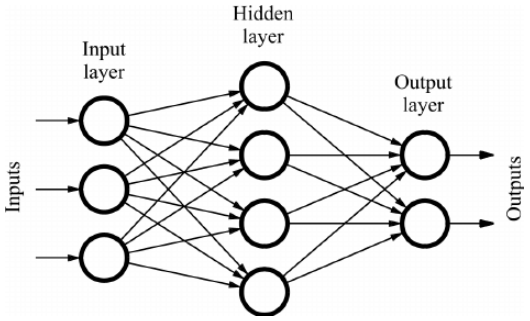


Figure 8: Example of a Dense layer (Hidden Layer) [20]

- **Convolutional Layer**
  A more promising way of looking at a trial is looking at different regions of the signal and trying to find relations. These found relations can then be used for training. One way to do this is to use a convolutional layer. A convolutional layer tries to find spatial information in signal-segments. Therefore the filter/kernel is adapted during training. A convolutional layer usually has multiple filters/kernels. An example

can be seen in Figure 9.

Since the neural network adapts the values in the kernels by itself, only the size needs to be defined. This depends on the amount of coherent information is expected in the signal. For this project, larger kernels also lead to the detection of lower frequency ranges.

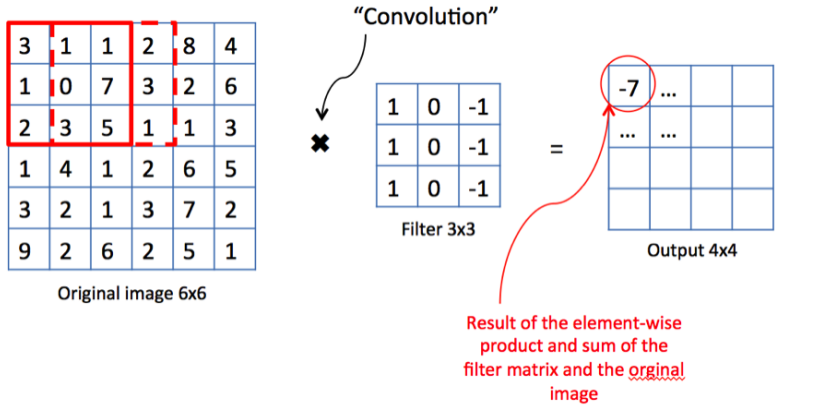


Figure 9: Example of a convolutional layer with one filter/kernel [40]

- **Recurrent Layer**

  Another way of extracting more information out of a trial is using recurrent layers. These look at the temporal information in a signal. So when making predictions of a certain sample, previous samples are taken into account for the calculations. An example of a recurrent layer can be seen in Figure 10.

  In each kernel there are three additional gates. The input gate (central gate; Fig.10) controls the extent to which a new value flows into the cell, the forget gate (left gate; Fig.10) controls the extent to which a value remains in the cell and the output gate (right gate; Fig.10) controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

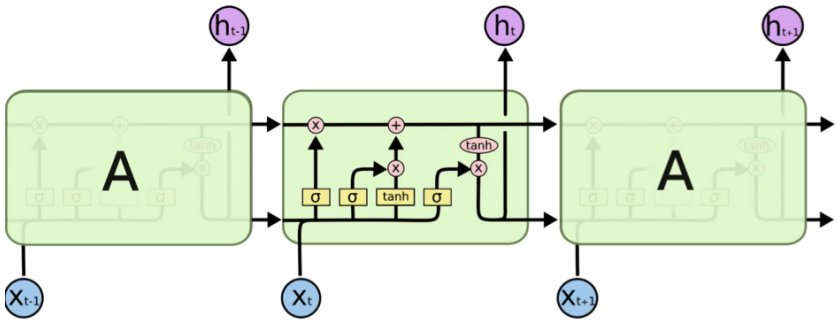


Figure 10: Example of a recurrent Layer with three kernels [41]. The blue nodes act as input and the purple as output nodes. Each path with an activation function in the kernel presents a gate.

- **Max-Pooling**

  The Max-Pooling layer is used to simplify a signal without losing too much information. Therefore the maximum of each pool is calculated and saved for further calculations.

  In Fig.11 an example of max pooling can be seen. This example uses a pool size of 2x2 and a stride of 2. The pool defines the number of values taken under consideration to calculate the maximum. The stride defines how many samples the pool gets shifted after calculating the maximum.

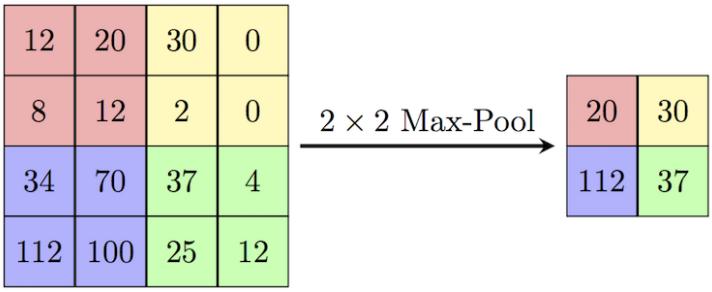


Figure 11: Representation of the Max-pooling method [38]. Each colored tile in the left matrix is a pool with a size of 2x2.

- **Dropout**

  A dropout layer is used for generalization purposes. During training of the NN it randomly selects nodes to exclude for each epoch. For this reason, the output does not depend too much on single nodes, but the dependency gets more distributed between the nodes. A schematic example can be seen in Figure 12.

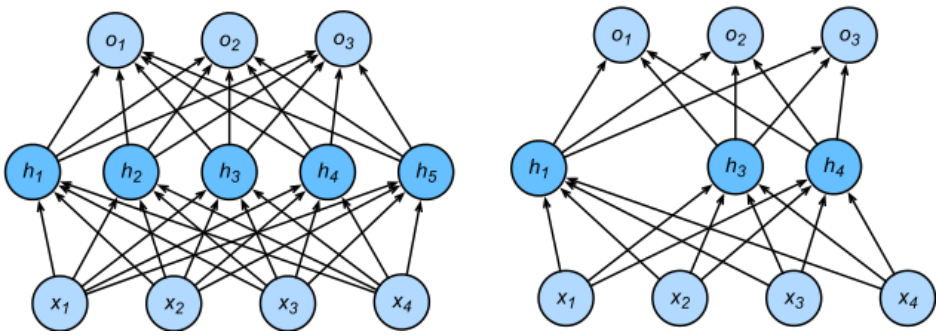


Figure 12: Representation of the Dropout method [26]. After each epoch, random selected nodes are neglected for calculations. The amount of nodes neglected, depends on the defined dropout rate (between 0-1).

4.4.3 Evaluations

For each change in the model's properties, evaluations were made, to check if a positive effect took place or not. The evaluations made, were:

- **Accuracy**

  Accuracy calculations are done to compare how good NN-models performs. Therefore the trained NN-model is tested on an unseen Test-Dataset and the amount of correctly classified trials is compared to all trials. It is calculated as follows:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where:

acc	=	accuracy
TP	=	True Positive
TN	=	True Negative
FP	=	False Positive
FN	=	False Negative

- **Predict over time**

  Although the model is only trained over trials it is also interesting how it performs over a given time signal. The sliding window method [17] is performed to see predictions at certain time points. This was especially done to see if click predictions outside cues were low enough. As shown in Figure 13, the window has a fixed length and moves by a given step(duration) from left to right.

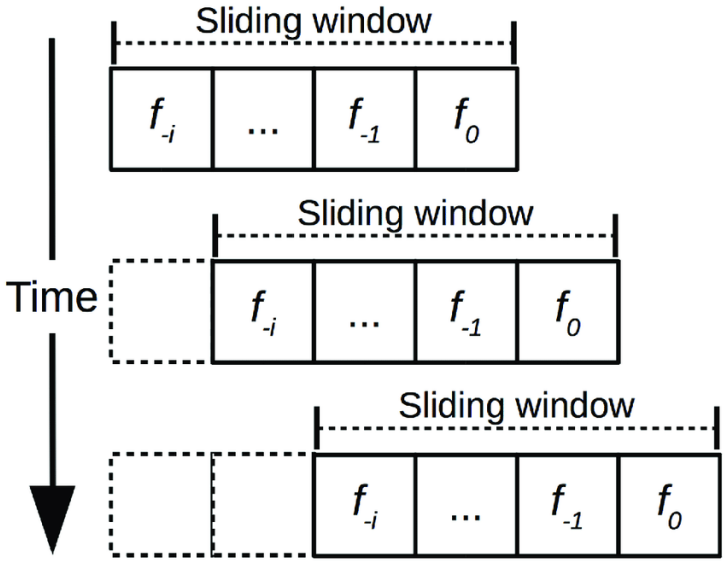


Figure 13: Schematic representation of a sliding window[18]

- **Confusion matrix**

  To better understand the accuracy values of the NN, and where misclassifications took place, a confusion matrix is used. A big amount of FPs or FNs could therefore tell where the model has problems classifying trials.

  An example is that the NN-model has problems converging during training. So the NN mistakes a class with another, due to high similarities in the signals.

- **Model-type validation**

  To compare different NN-models with each other a model-type evaluation needs to be carried out. Therefore each model compared gets trained a reasonable amount of times. Afterwards accuracy-distributions are calculated. And in the end, all distributions are compared with each other to find the best model.

- **Decode nodes**

  Another way of evaluating a model is to try to decode what the NN encodes in its layers and nodes. After training, each node in the NN learned a certain part of the signal (e.g., frequency band, time-locked feature), and uses this information to classify a new unseen trial. For this purpose, a gradient ascent method [23] is used. Therefore, a random input is sent through the NN, and the gradient of each node is trying to be maximized by changing the input. This is done until the input does not change anymore. In the end, each node represents the encoded feature structure learned by the network. An example can be seen in Figure 14.

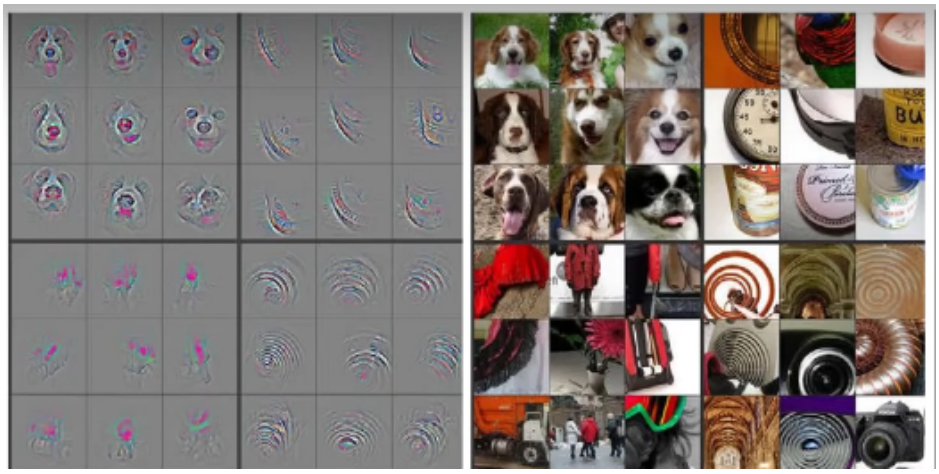


Figure 14: Examples of decoded filters and their original pictures[42]. This example shows four different nodes that were trained on different data (dogs, round edges, color red, curves). The left side showing the relevant features per model. And the right side showing the input data.

- **Size of dataset**
  When working with further patients it is good to know how much data is needed to train and test the model properly. Out of that reason the model is evaluated with different amounts of training data. This is again done a reasonable amount of times. Then the accuracy distribution is calculated and compared with each other.

4.4.4 Training methods

Neural Networks belong to the supervised machine learning methods [21]. That means the NN needs to be trained on labeled data before it can classify unseen data. This is also called the Training-Dataset.
To rate how well a created model performs, another dataset is needed. After training the NN with the Training-Dataset the NN tries to classify the unseen data. This is also called Test-Dataset.
Another question that needs to be answered is how long the NN needs to

be trained. If training is too short the NN-model will not converge and the classification accuracies for Training- and Test-Data are low. If training takes too long, the NN's performance is also low, due to overfitting. The Training-Dataset's accuracy would be high, but the Test-Dataset's accuracy would be low.

During this project different approaches were used, which are listed below:

- **Early Stopping** (Default method)
  The goal is to find the "sweet spot". This can be accomplished with the early stopping method [22] (Fig.15). Therefore a Validation-Dataset must be added. During training of the NN, both errors for Training- and Validation-Dataset are watched. As soon as the Validation-Error increases, the training is aborted. The created model is then appropriately fit.

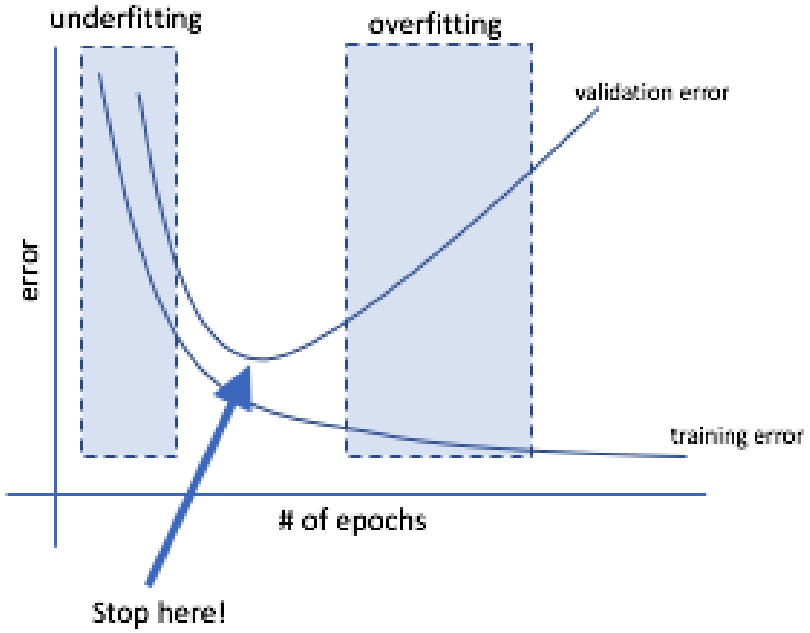


Figure 15: Error-curves during training of a neural network[39]

- **Adapted early stopping**
  For NN-models that have difficulties converging the early stopping method can be adapted. Therefore a patience parameter can be defined [24]. If the patience parameter is greater than zero, early stopping is not applied until the Validation-Error consecutively increased the given amount of times. An example can be seen in Figure 16.

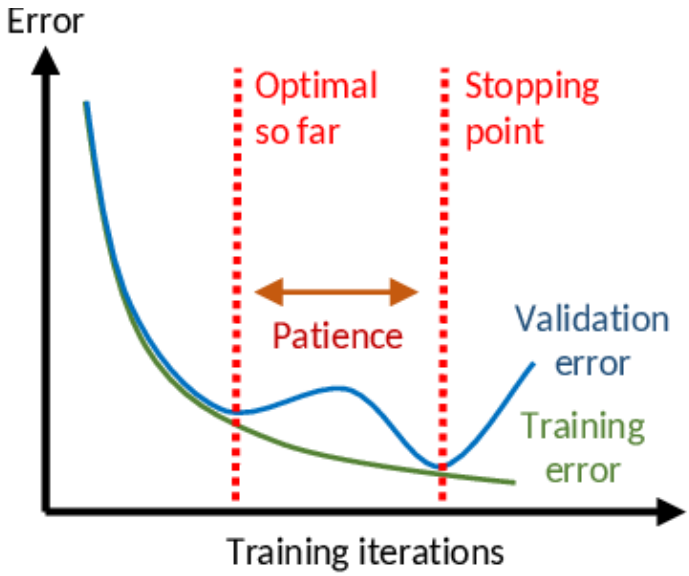


Figure 16: Error-curves during training of a neural network[25]

- **Define epoch amount**
  In case early stopping is not an option one can define the number of epochs the NN-model should use for training. Therefore it is important to know in which range the epochs should be. This can be found out with the epoch evaluation method described in chapter 4.4.3. The advantage of this method is that one can avoid the Validation-Dataset which can be added to the Training-Dataset.

4.4.5 Fine-Tuning

As soon as everything possible has been done to improve the model's performance, the NN can be fine-tuned. Each layer type has different ways of fine tuning.

- **Dense layer**
  - Adapt amount of nodes (also named kernels or filters for other layer types)
  - Define activation function (e.g., Rectified Linear Unit, Sigmoid, Tanh,...) [27][28] (For this project the default Keras activation functions were used)
  - Weight initialization [28]
- **Convolutional layer**
  The properties defined in the Dense layer can also be adapted in the convolutional layer. In addition to several other parameters
  - Kernel size: Defines the size of the convolution window moving on the input
  - Stride: Defines how much the convolution window moves after each calculation (For this project it was set to 2)
- **Recurrent Layer**
  The properties defined in the Dense layer can also be adapted in the recurrent layer.
- **Pooling layer**
  - Adaptation of pooling method (e.g., Maximum, Average)
  - Pool size: Defines the window size over which pooling is done

  - Stride: Defines how much the pooling window moves after each calculation

- **Dropout layer**
  The dropout rate can be defined between 0 and 1. (For this project it was set to 0.5)

Note: *The list of fine tuning methods just shows which properties were changed during this project. There are many more properties that can be adapted.*[24]

4.5 Approach

Since there were already two approaches of classifying clicks described in chapter 4.4.1 another approach was taken into consideration.
The idea of using a machine learning method was kept. But instead of calculating the spectogram for each trial the raw time-signal is used. And therefore, a neural network was set up and adapted over time. Additionally, to the M1 channel, DLPFC was added, since information in that region which could improve accuracy was expected as well. Another benefit expected, when adding DLPFC was the future introduction of a second click in the system. And the final goal was to test the same system on people that have fewer trials and weaker signals

5 Results

The upcoming topic describes different types of results. Firstly, the neural network is described and how it evolved over time. Additionally, different choice decisions are going to be described. For model selection, mainly data from UNP1 was used, since it has been used for validation in previous experiments [1] [7] as well. Another part of the results is the application of the NN-model on UNP4.

5.1 UNP1

UNP1 was the first participant in this study [1]. Therefore the highest amount of recorded signals were with this user. Further, model validation was mainly done with data of this participant.
The data used comes from the WAM task (chapter 4.2.1). The amount of trials is 1467 and the duration of each trial is 8 sec (6 sec before click; 2 sec after click).

5.1.1 Building neural network

- **First NN-model**
  In the beginning, a normal feed forward NN was built with two hidden layers. One having 250 nodes and the other 80 nodes as you can see in Figure 17. To avoid overfitting early stopping was used.

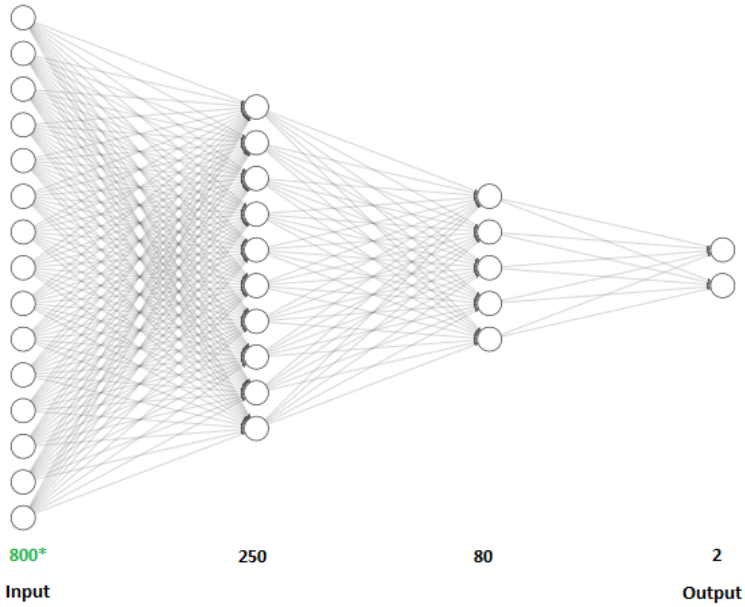


Figure 17: 1st model: Feed forward neural network with two hidden layers

The M1 channel was chosen and an appropriate trial size needed to be evaluated. The first trial sizes were chosen similarly to the described trials in [7]. Table 3 shows the results after the trial size evaluation.

Trial-length	Mean accuracy
4sec	74.8
2sec	81.5
1.6sec	**82.7**

Table 3: Mean accuracies for different trial sizes

In the end the trial size of 1.6 sec is chosen since it had the highest accuracy. Additionally, the accuracy is almost as good as the frequency subtraction method in chapter 4.4.1.

- **Add DLPFC channel**

  After finding the best trial size for the M1 channel, the DLPFC channel was added. According to [29], the window needed for DLPFC, to identify motor clicks is larger than for the M1 channel. It says that at least $2 - 4$ sec are needed.

  To add DLPFC trials to the input the signal is concatenated with its corresponding M1 trial. Using both channels and training the Feed-forward network (Fig. 17) with it did not improve the accuracy. Also, different trial lengths for the DLPFC channel were tried. But this did not lead to improvements either.

  After observing those results, the network was solely trained and tested on DLPFC trials. Only using the DLPFC channel is even worse which shows that the feed forward NN is not capable of learning features using imagined motor tasks, from the DLPFC channel. The accuracies are in the range of 55-60%. The best results are achieved with trial lengths of 4 sec

- **Add NN-layers**

  The next step was the addition of a dropout layer between output and hidden layers to improve generalization of the NN. Further CNN- & RNN-layers to create a more sophisticated NN-model were added. Figure 18 shows the more sophisticated NN.

  To be able to use M1 and DLPFC channels at the same time, the trial length was changed back to 2 sec. Although the accuracy was a little lower using 2 sec windows in the Feed-forward NN it did not have an effect on the new NN-model. After training the CNN with 2 sec windows of the M1 channel the accuracies increased to up to ~95%.

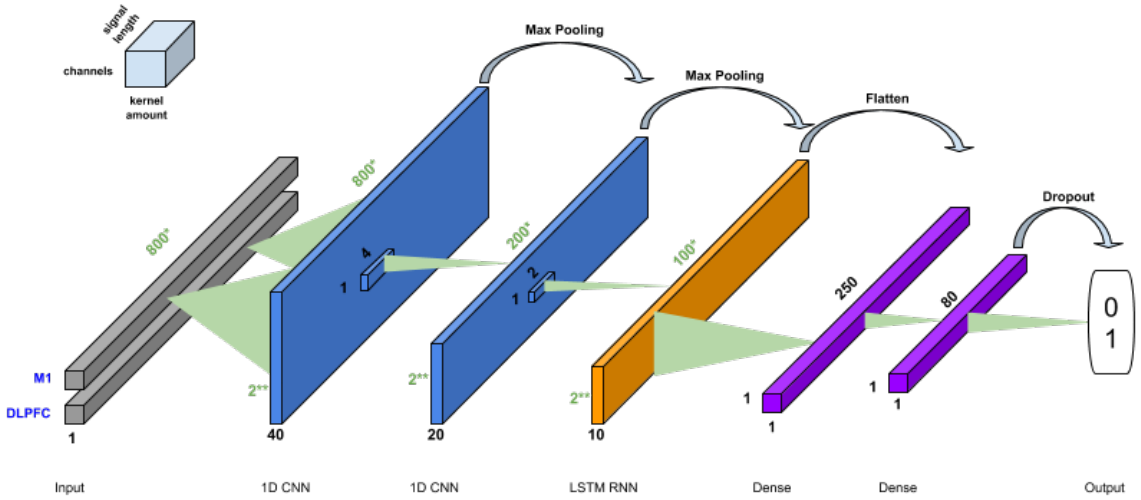


Figure 18: CNN after adding layers to the basic Feed-forward NN

5.1.2 Model validation

To make sure that the found model was the best, a model validation was performed. The results can be seen in Figures 19-23.
The default NN-model to compare to was the "Conv1D40-Conv1D20-LSTM10-Dense250-Dense80" (Fig. 18). The validation is performed to figure out if simplifications or improvements can be achieved. Each model is trained and tested 30 times and the accuracy distribution calculated.

Note: The x-labels of the following distribution figures show how each model is set up. Each layer is defined in this form: Layer-Type + Amount of Nodes (e.g., Conv1D + 60). The label structure can be seen here:

Layer1
Layer2
Layer3
:

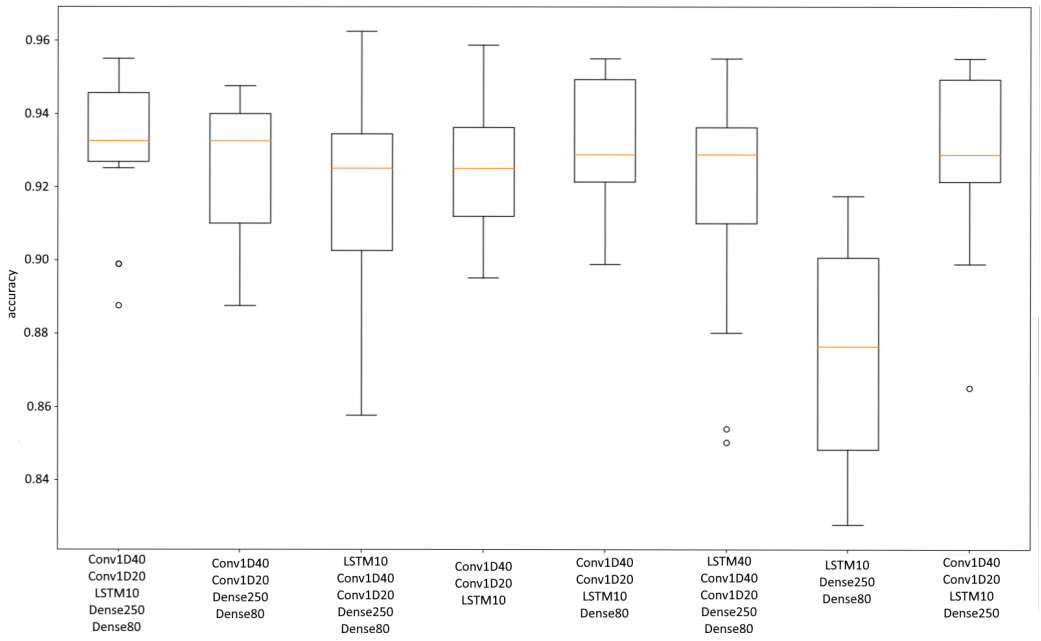


Figure 19: Accuracy distributions for first round of model validations

After the first round of validations the most promising models (Fig. 19; models 1,4 and 5) are kept and used again in the second round (Fig. 20) in addition to other NN-models.

The promising models were chosen under the premise of a high accuracy and a low standard deviation. This evaluation method was also used for all further network selection steps.

Further models were added to the second round of validation to evaluate if a different model architecture could improve performance or even simplify the model.

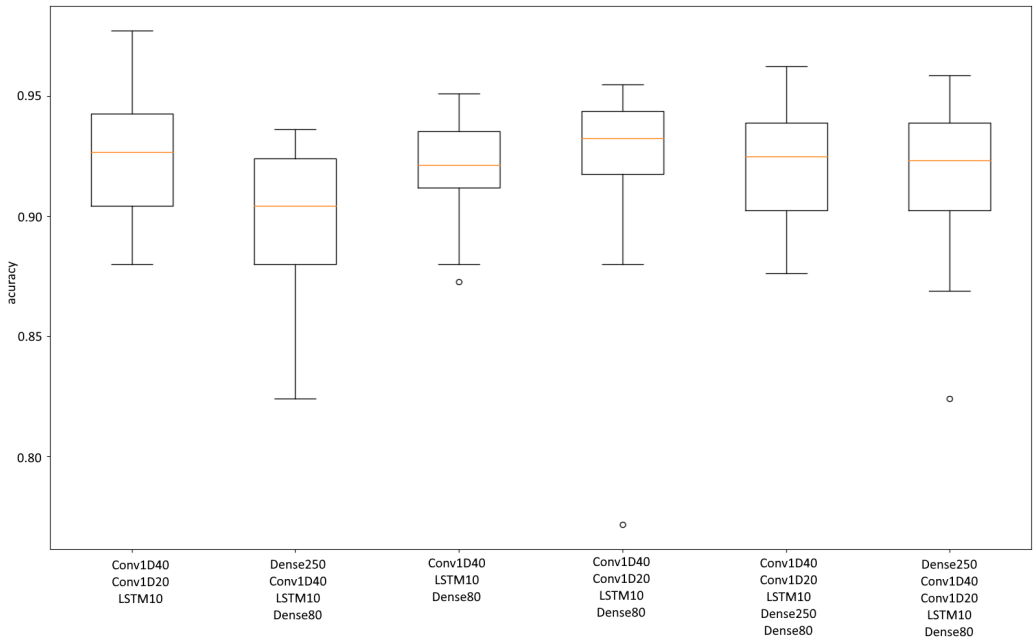


Figure 20: Accuracy distributions for second round of model validations

After the second round of validations, the three most promising NN-models (Fig. 20; models 1,3 and 4) are evaluated to find the best. Since the addition of the DLPFC channel is desired, the trial size is increased to 4 sec for both channels. First, the remaining models are trained on M1 and DLPFC channel solely (Fig. 21 & 22). And as second step the NN-models are trained on both channels at the same time (Fig. 23).

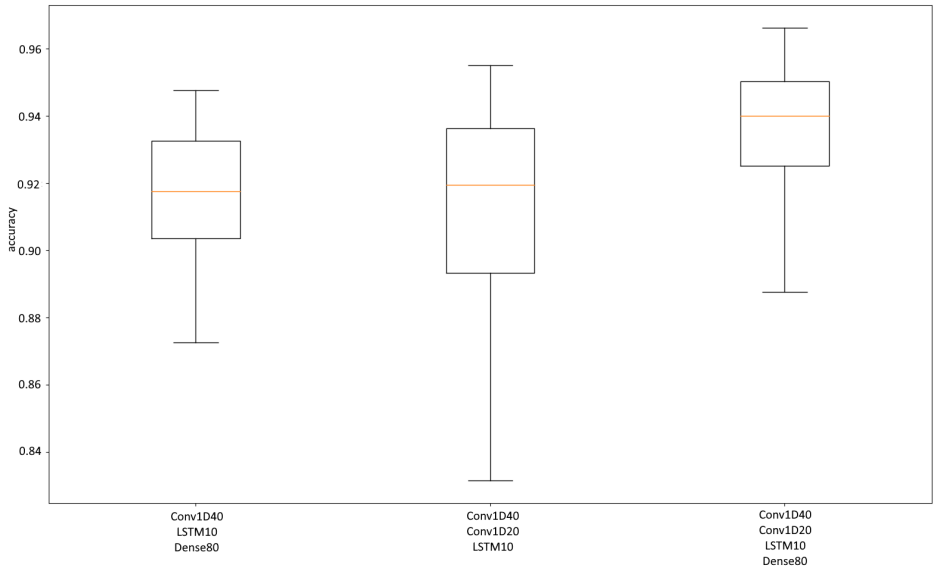


Figure 21: Accuracy distributions for best NN-models using M1 channel

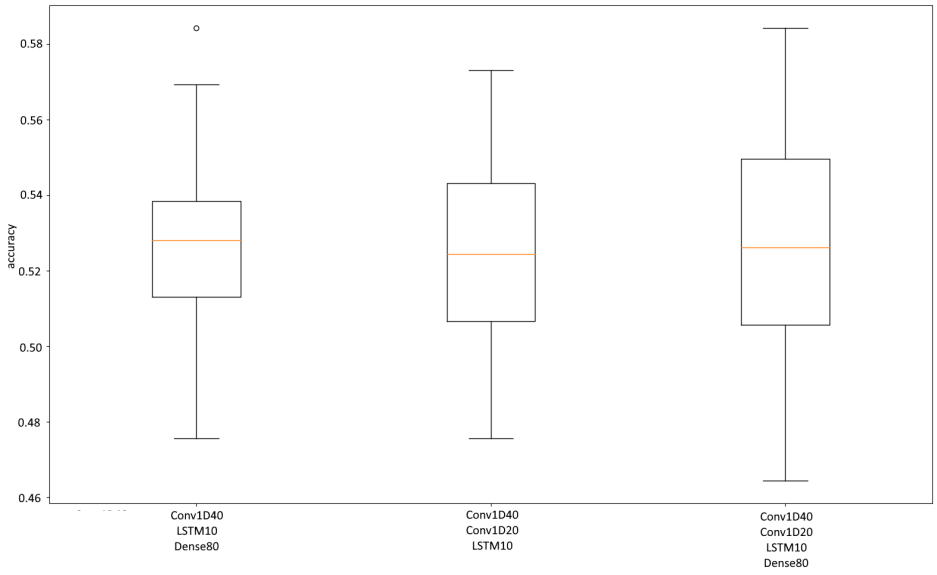


Figure 22: Accuracy distributions for best NN-models using DLPFC channel

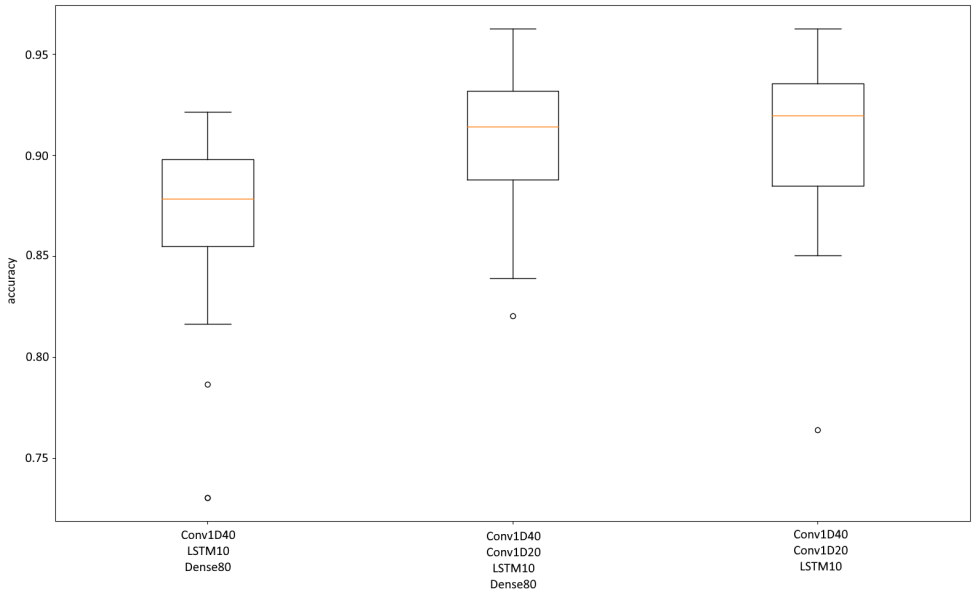


Figure 23: Accuracy distributions for best NN-models using all channels

- As seen in Figure 22 the accuracy of the models are just slightly over chance.
- Figures 21 & 23 show that "Conv1D40-Conv1D20-LSTM10-Dense80" is the best model to choose after validation.
- A simplification of the model could be achieved by removing the "Dense250" layer (compared with NN in Fig. 18).

5.1.3 NN-model properties

The chosen NN-model seen in Figure 24 achieves accuracies of 93% when using both M1&DLPFC channels and trial lengths of 4 sec. When using the model with the M1 channel and a trial length of 2 sec, then accuracies of 95% can be achieved as for the model in Figure 18. Figure 25 shows the classification results for the newly selected model (Fig. 24) in form of a confusion matrix.

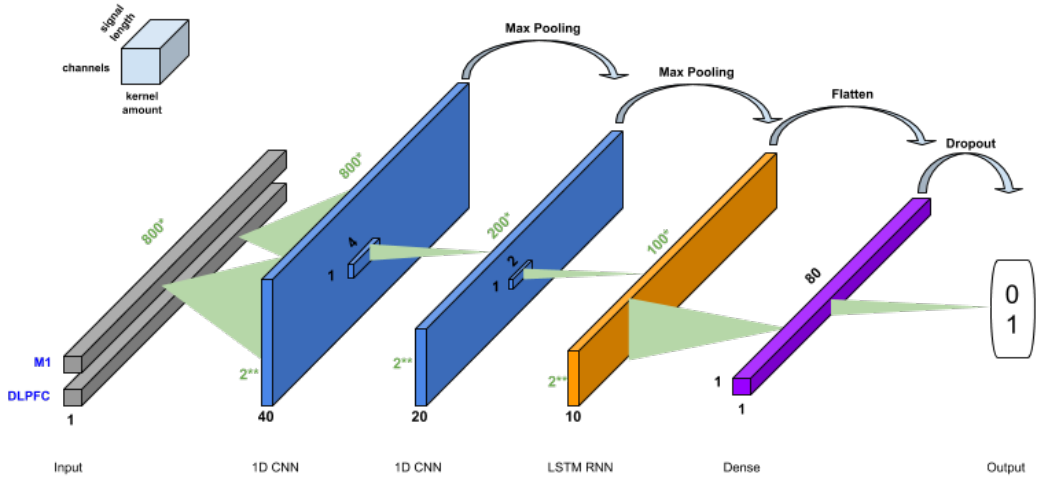


Figure 24: CNN after evaluation and simplification

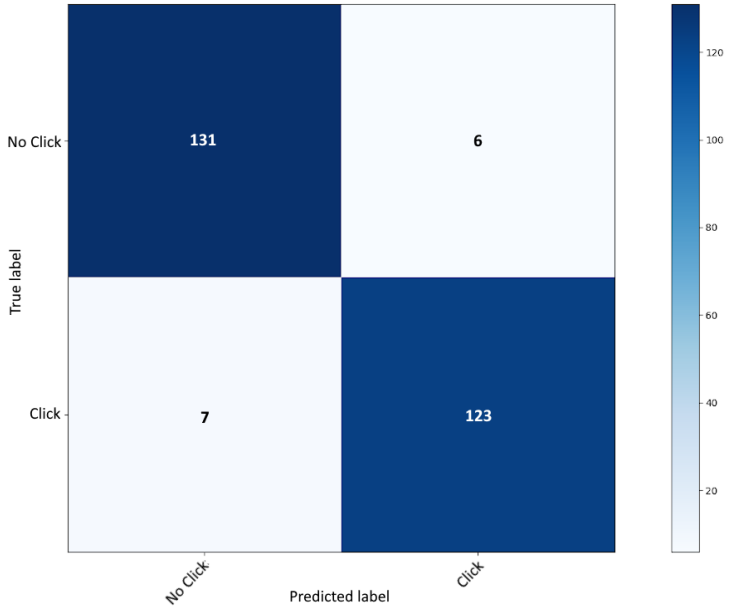


Figure 25: Confusion matrix after testing the new CNN (Fig. 24). The model has a very low amount missclassified trials. And the amount of FPs and FNs is similar which shows that there is no bias for any class.

5.1.4 Sliding window

To see how good the model performs in "real-life" situations it can be tested on an actual run. The result of a random run can be seen in Figure 26. As one can see the prediction for a click increases before and during a cue. But also just before a click was executed with the help of the currently used system (frequency-band subtraction method). On the other hand, the predictions for a click between the cues were low.
The prediction increases one can see at the beginning of the figure are explained by the course of a run. At the beginning of each run the participant is told to execute three consecutive motor movement attempts to see if the system works.

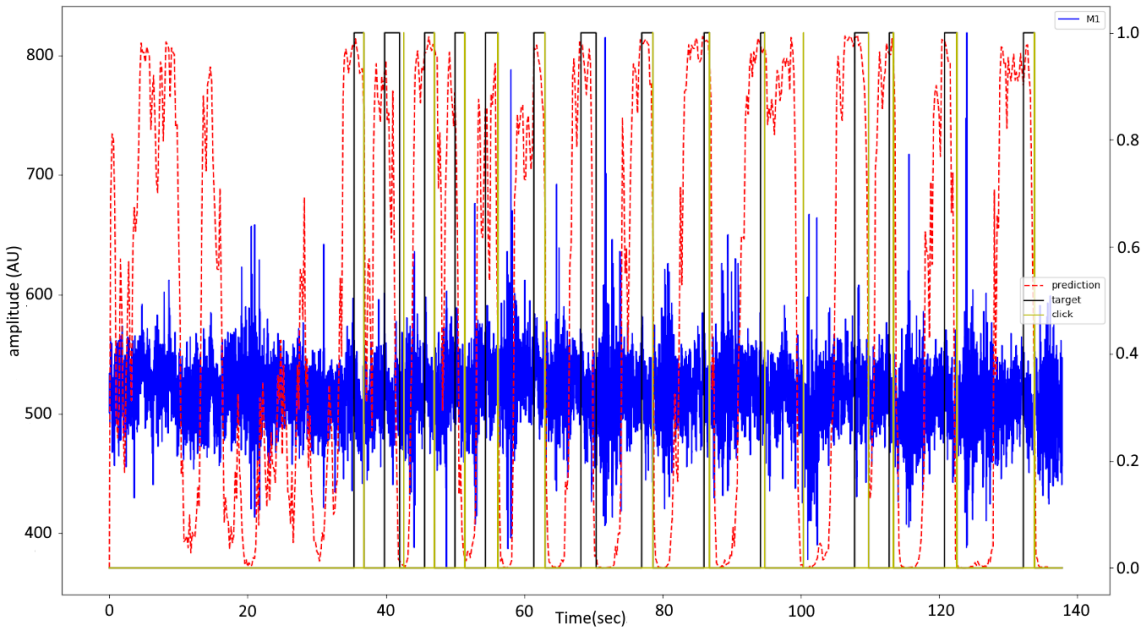


Figure 26: Click-Prediction over a random run

Prediction	...	Prediction of trial being a click
Target	...	Cue in which a click should be executed
Click	...	Timepoints of actual clicks
M1	...	Channel that is trained/tested on (in this case M1)

5.1.5 Decode nodes of NN

Following figures (Fig. 27-30) depict what the chosen NN-model (Fig. 24) decodes in its nodes after training. Each node depicts the time signal (blue) above the frequency spectrum of the encoded time domain signal (green).
When comparing the frequencies for Non-Click and Click (Fig.30) one can see that the Beta-Band is a large feature that is being used to distinguish the two classes. Further, one can see that the Beta-Band decreases during activation (motor movement attempt).
The gamma band feature is harder to observe because of it's lower amplitude. But when looking at some frequency spectra in Figure 27 (e.g., Row:4-Col:4, R:4-C:12, R:5-C:2) or Figure 28 (e.g., Row:5-Col:6) one can see peaks between 60-80Hz which are all in the gamma frequency range.
Another observation made was that some nodes do not seem to have a real impact on the signal. Their decoded time signals have little amplitudes and a frequency spectrum that seems to be noisy (e.g., bottom left in Fig. 28). Further, there seem to be nodes that just hold time information (e.g., top left in Fig. 27).

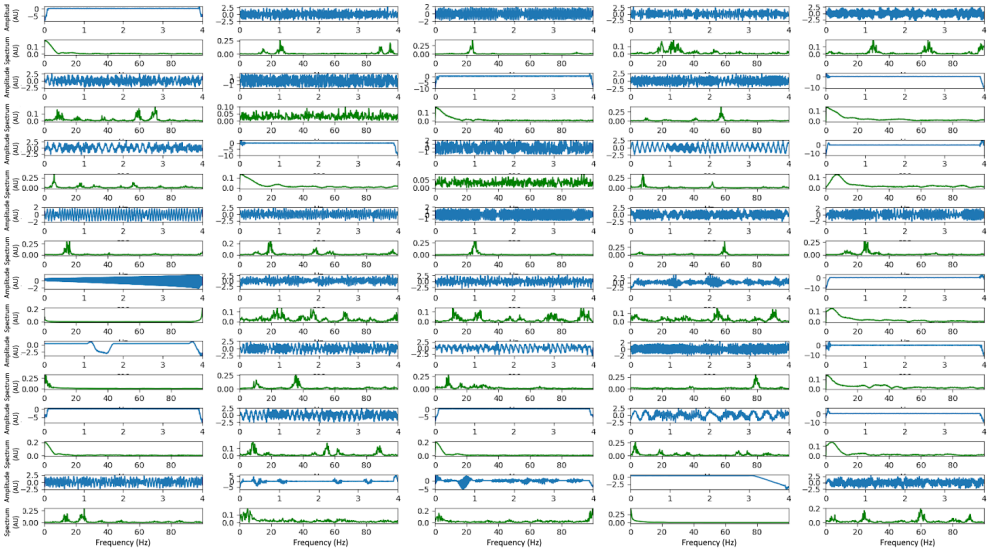


Figure 27: Decoded time and frequency information for convolutional layer with 40 nodes

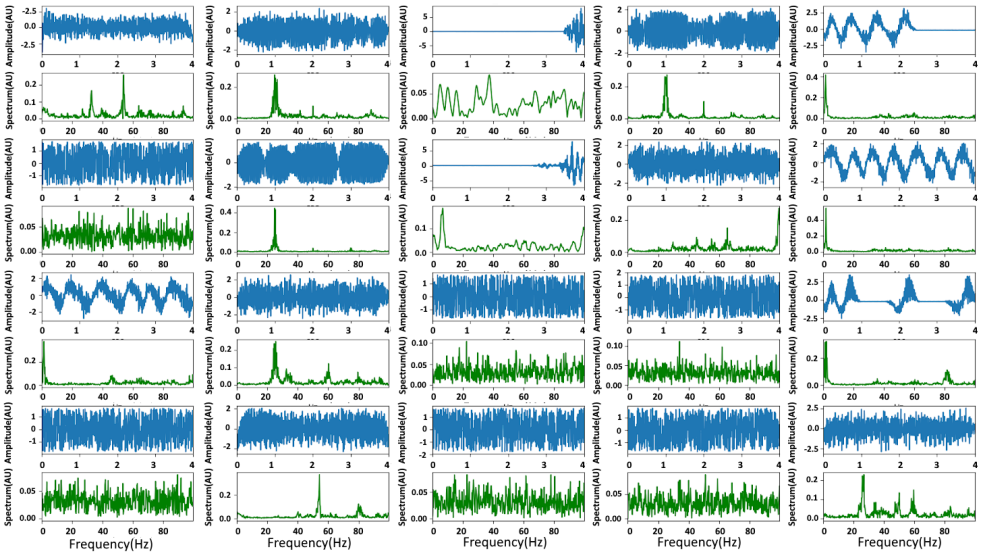


Figure 28: Decoded time and frequency information for convolutional layer with 20 nodes

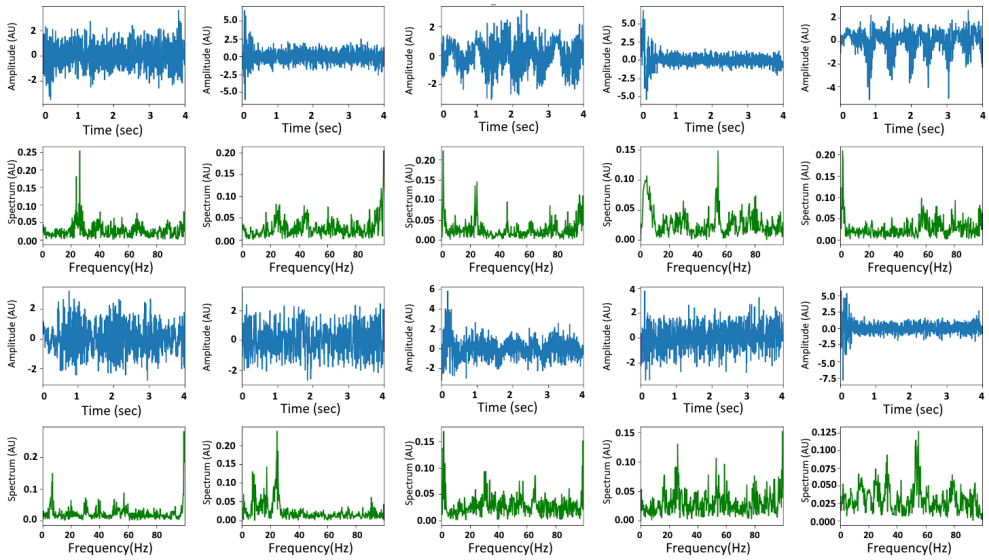


Figure 29: Decoded time and frequency information for recurrent layer with 10 nodes

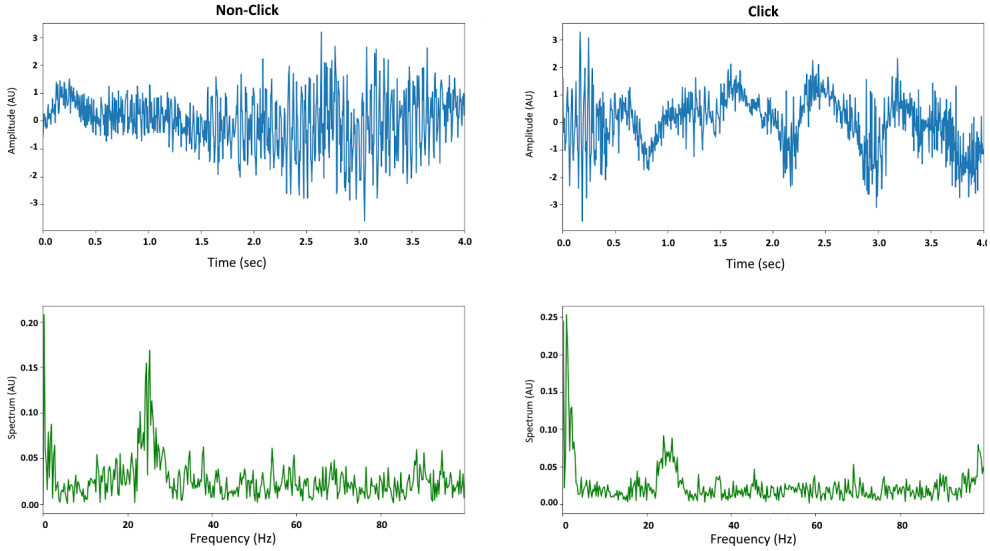


Figure 30: Decoded time and frequency information for output layer

5.2 UNP4

After finding an appropriate NN-model for UNP1 it was tested on UNP4's data. Taking trials from WAM (chapter 4.2.1) and cWAM (chapter 4.2.2) task under account there were 230 trials available. The datasets are freely configurable.

5.2.1 Minimum required data

Considering that the amount of data is much smaller than for UNP1 (chapter 5.1) the minimum amount of training data was evaluated as seen in Figure 31. For evaluation UNP1-data was used. Section: "Size of dataset" in chapter 4.4.3 describes how the evaluation was accomplished. The Training-Dataset Size increased by 10% of the total amount, for each validation step, starting at 10% of the total Training-Dataset.
The figure shows that at least 280 trials are needed for the Training-Dataset. Thus, the UNP4 dataset is likely too small.

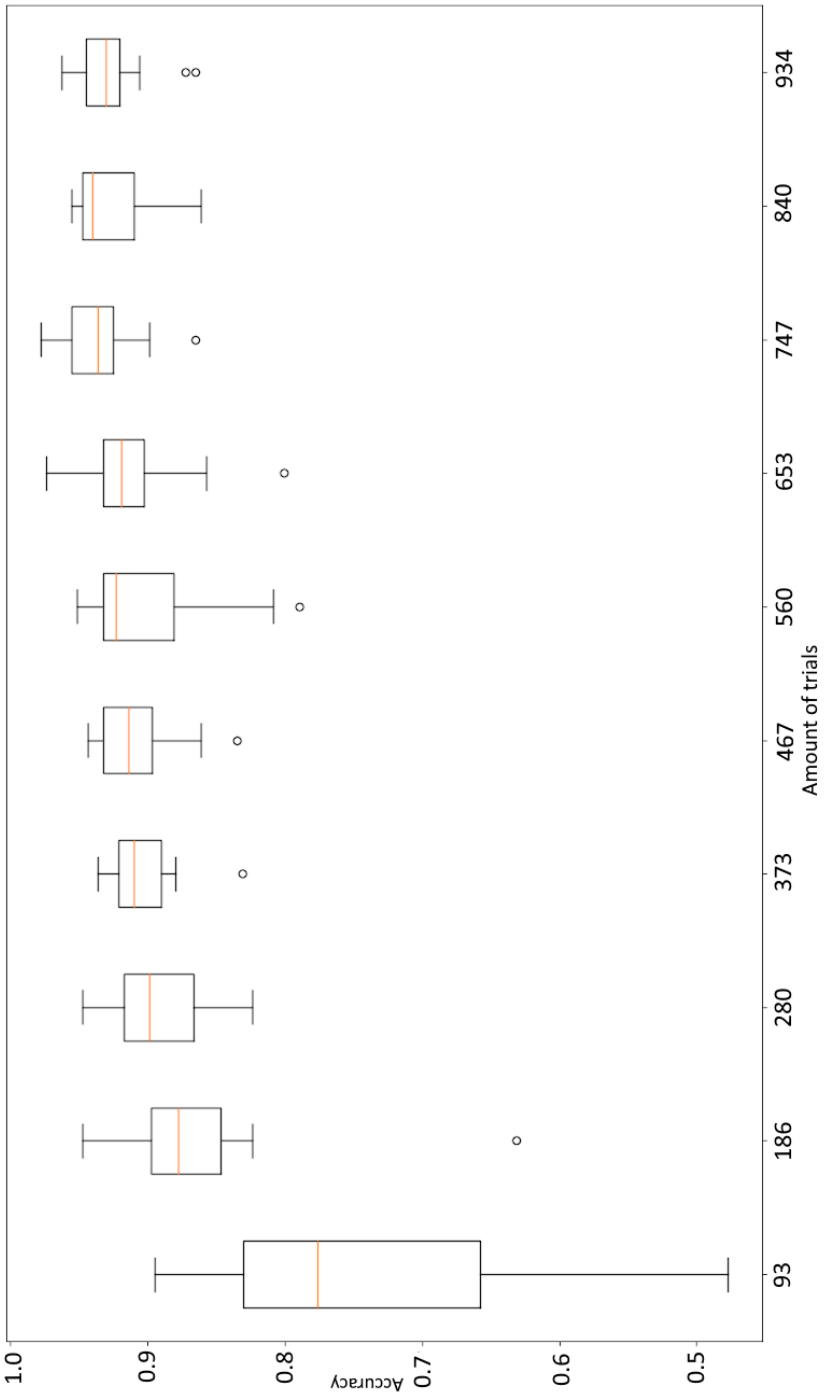


Figure 31: Accuracy distributions for different sizes of Training-Datasets

5.2.2 Train NN-model for UNP4

Due to the little amount of data different approaches were used as seen in table 4. The reason for the different approaches was that the limited amount of data complicated convergence for the NN-model.

Datasets			Results
Training	Validation	Test	
UNP1	UNP1	UNP4	Does not work
UNP1	UNP4[1]	UNP4[1]	NN-model does not converge
UNP4[2]	UNP4[2]	UNP4[2]	NN-model does not converge

[1] The dataset is split equally(50% Validation, 50% Test)

[2] Dataset: 60% Training, 20% Validation, 20% Test

Table 4: Different training approaches for UNP4 dataset

5.2.3 Adapt NN for UNP4

One approach of improving the NN's performance is to finetune the model as described in chapter 4.4.5.

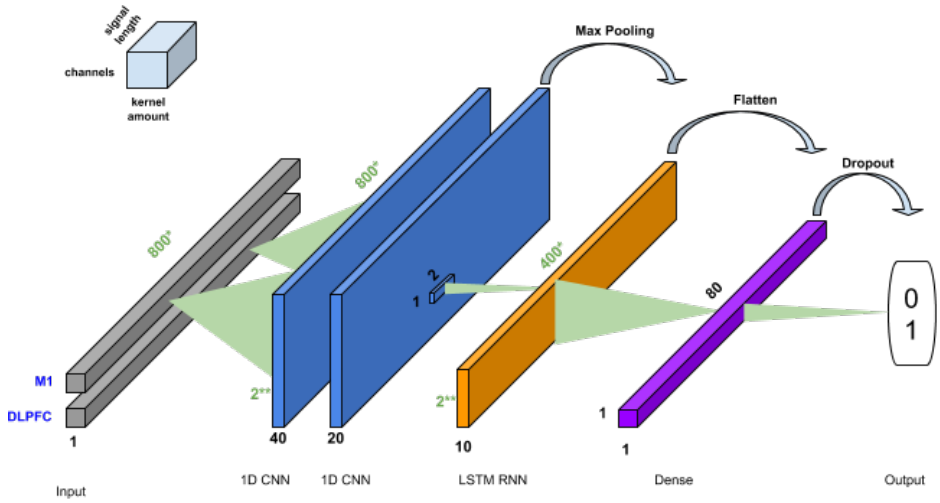


Figure 32: CNN without one of the Max-Pooling layers

5.2.4 Increase Training-Dataset

To increase the amount of Training-data there were different possibilities which are shown in this chapter.

i. **Add Validation-Data to Training-Data**
   A possibility of increasing data is not to use early stopping and therefore add the Validation-Dataset to the Training-Dataset.
   It is known that UNP4 takes longer producing brain-clicks than UNP1 ($\sim$4-6sec)[29]. Therefore trial lengths in the range of 2 sec, 4 sec and 6sec were tested. The distribution of Clicks and Non-Clicks could differ because the Test-Dataset was randomly set before each run

   In general, the NN-model had problems converging for UNP4 due to the little amount of data and the imbalanced distribution of classes (Non-Clicks:$\sim$70% & Clicks:$\sim$30%). That means that the model was able to achieve good accuracies ($\sim$70%), but when looking at the confusion matrices one could see that the model did not distinguish the classes at all. The model just classified all trials as Non-Clicks. So it was decided that the confusion matrices were used to evaluate the performance of the NN-model.

   Due to the conversion problems, just the best results were depicted in Figures 33-36. Figure 33 shows the results for 6 sec trials trained with the M1 channel for 25 epochs. Compared to Figure 34 which had the same properties except for a smaller trial length of 4 sec, the shorter trial size performed slightly better.

   Also, the addition of DLPFC was investigated as seen in the results in Figure 35. But this decreased accuracy again compared to the results of UNP4 solely using the M1 channel shown in Figure 34.

At last the epoch amount was adapted. It seemed that 18 epochs were a good value to work with as seen in Figure 36. To define this specific amount, the confusion matrices were used for evaluation. If the amount of epochs was low the accuracy stayed at an average of 70%, due to the distribution of Clicks and Non-Clicks. That mean that all trials were defined as Non-Clicks and therefore reached a classification accuracy of ∼70%. For that reason the amount of epochs was increased until the NN started converging and the accuracy reached its peak.

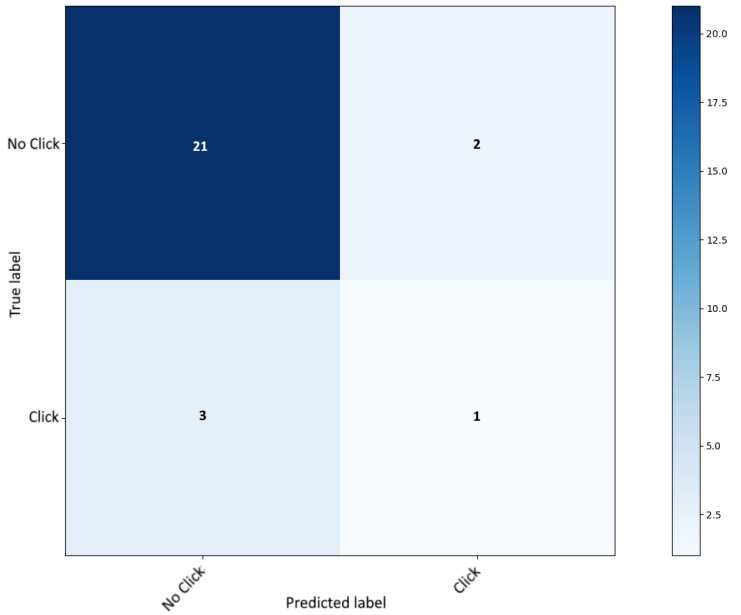


Figure 33: UNP4-data; 6 sec trials; channel M1; 25 epochs

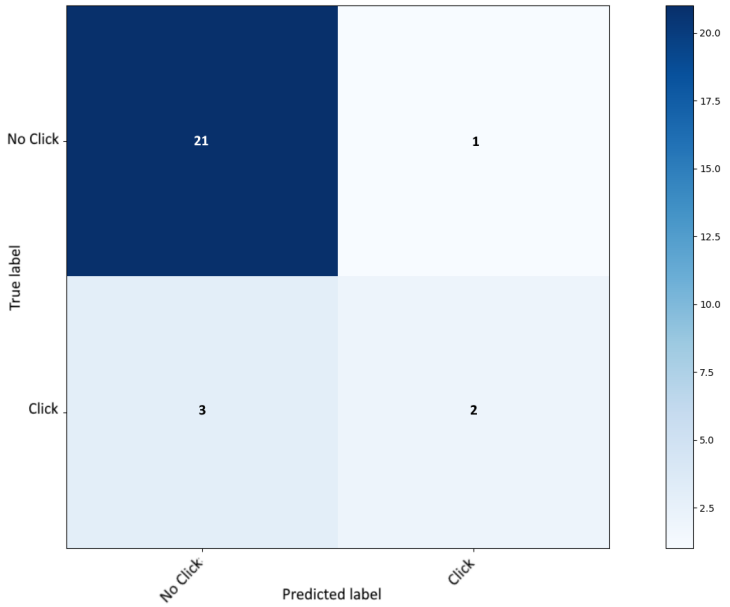


Figure 34: UNP4-data; 4 sec trials; channel M1; 25 epochs

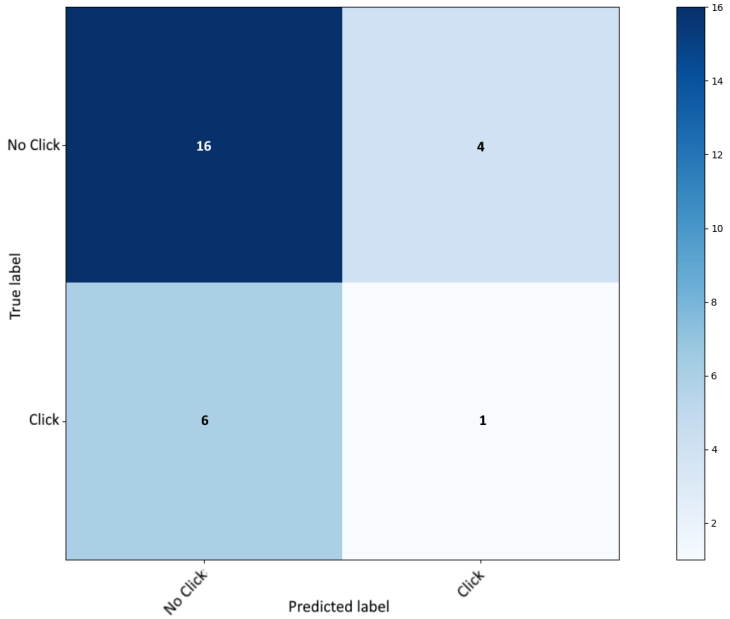


Figure 35: UNP4-data; 4 sec trials; channels M1&DLPFC; 25 epochs

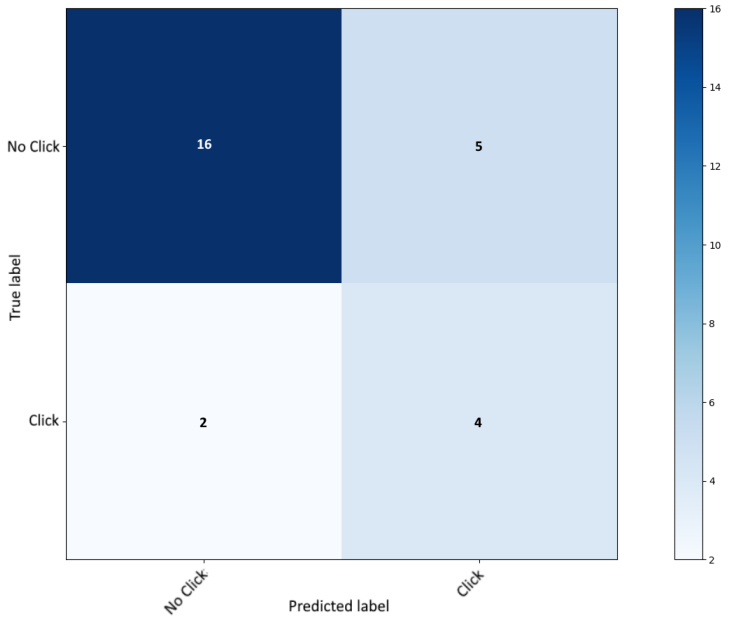


Figure 36: UNP4-data; 4 sec trials; channels M1&DLPFC; 18 epochs

ii. **Use Localizer data for training**

As described in chapter 4.2.3, Localizer trials were different to the WAM and cWAM task. But using them increased the amount of training data significantly. For that reason early stopping was used again to improve generalization. The training result can be seen in Figure 37.

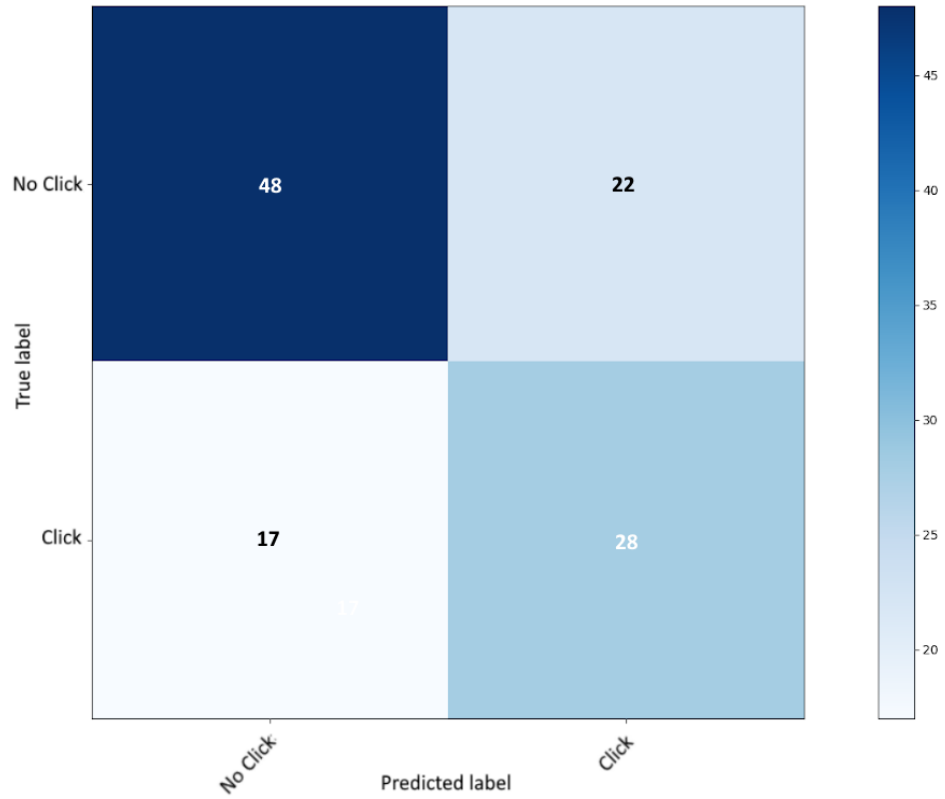


Figure 37: UNP4-data; 4 sec trials; channels M1; trained with Localizer data

From this point on the Training-Dataset consisted of trials from the Localizer task. And the Test-Dataset contained trials from the WAM and cWAM task. The NN-model also converged more often than when training only on WAM and cWAM data. In Figure 38 the accuracy distribution for 40 runs is depicted.

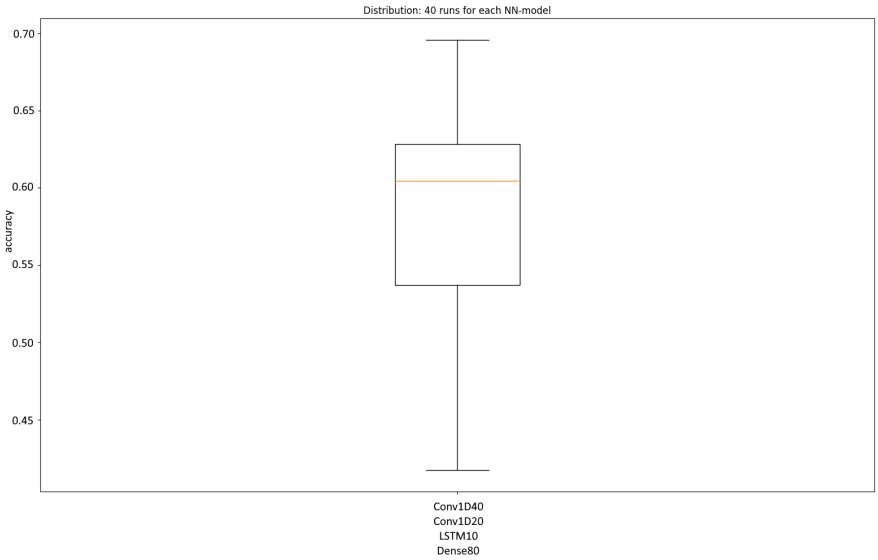


Figure 38: Accuracy distribution for UNP4 using the M1 channel

	NN-model	Frequency subtraction
accuracy	60%	67%

Table 5: Performance-comparison of UNP4 for different models. The NN-model was trained on the data of the localizer task and tested on WAM and cWAM data.

5.2.5 UNP4 signal properties

Due to the performance, only slightly above chance of the NN-model on UNP4's data, the Training- and Test-dataset was analyzed as seen in Figures 39&40. This was mainly done to see if the signal properties differ from each other.

In figure 39 (Training-Dataset) one can see that the amplitudes in 10Hz range are larger for TNs and FPs, and lower for TPs and FNs, which makes sense. Since an amplitude decrease is expected during activation of the motor attempt. Although, in figure 40 (Test-Dataset) the amplitudes in 10Hz range

are very similar.

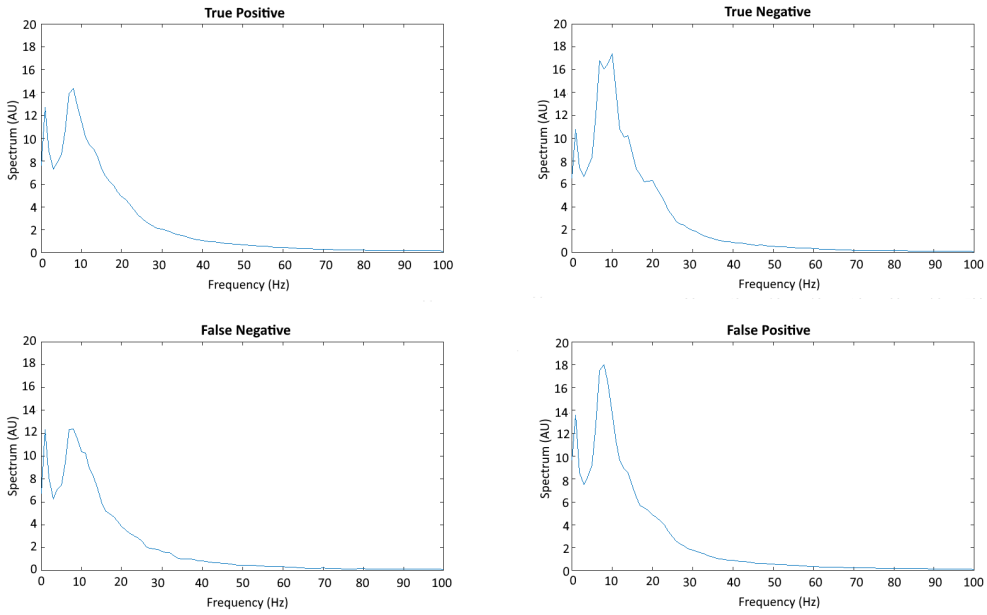


Figure 39: Frequency spectra for training-dataset (Localizer task)

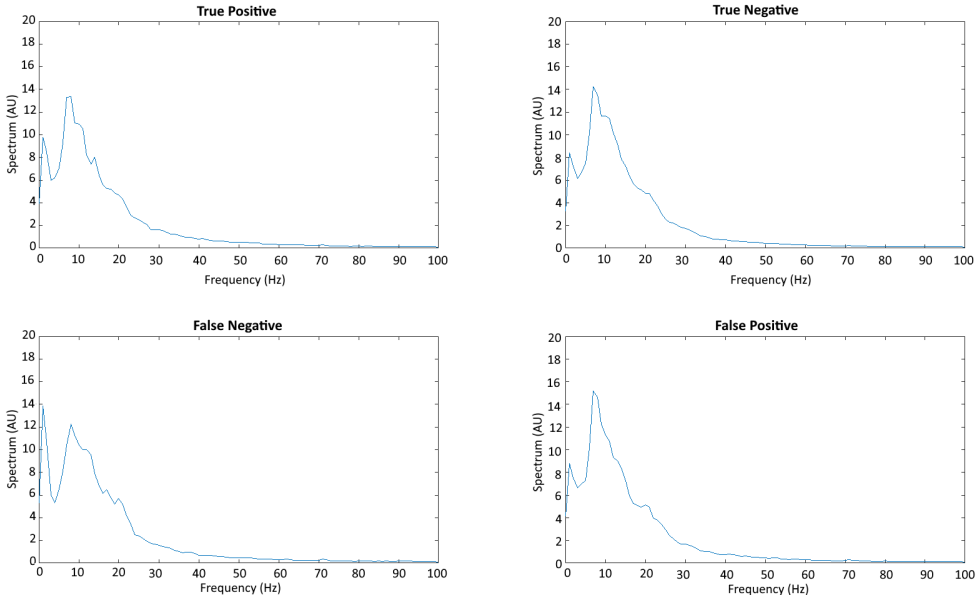


Figure 40: Frequency spectra for test-dataset (WAM/cWAM task)

5.3 Model performance

The results for the best fitting model (Fig. 32) for both participants can be seen in Figures 41, 42 and 43.
To put it all together, one can see that the model accuracy for UNP1 was able to be improved. The average performance for UNP1 using the M1 channel is 95%. But also after adding the DLPFC channel the average performance is 94%.
When looking at UNP4's results one can see that the average performance just reaches 60%, which is a little less than standard frequency subtraction method. But to achieve this accuracy the model solely trained on data of the localizer task and tested on (c)WAM data.

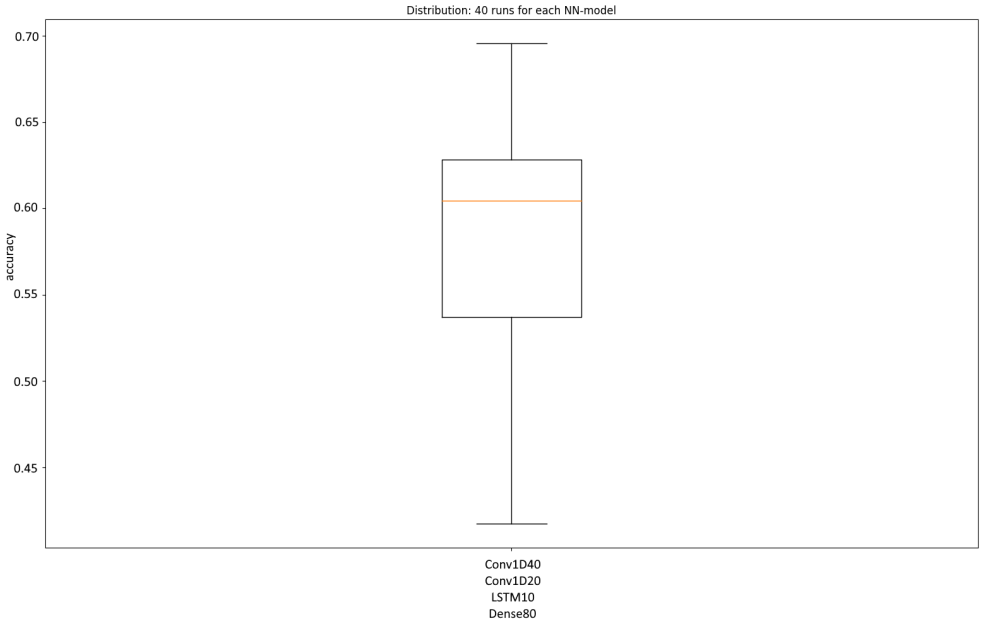


Figure 41: Accuracy distribution for UNP4 using the M1 channel

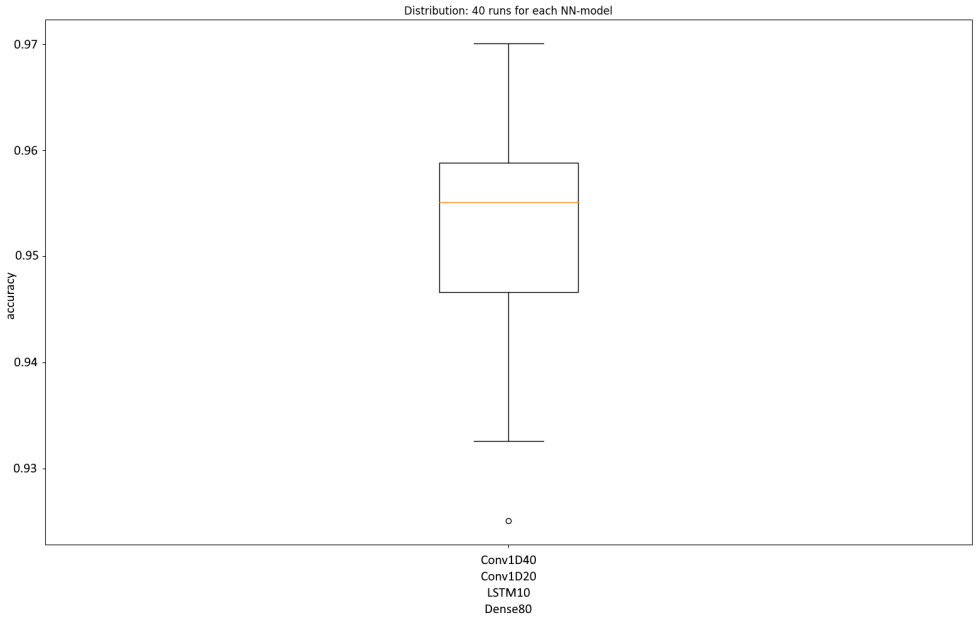


Figure 42: Best performance for UNP1 using M1 channel

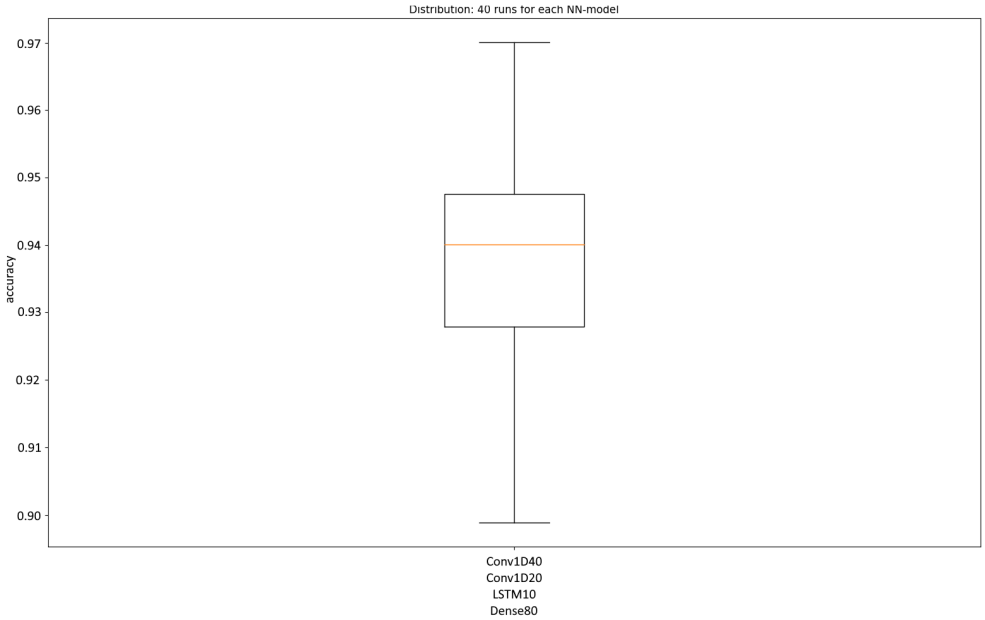


Figure 43: Best performance for UNP1 using M1&DLPFC channel

5.4 Explore model capabilities

5.4.1 Detect FP-Feedback

In order to correct mistakes made by the operator, an additional class was added which contains signals after a click. The label 'Post Trials' in Figure 44 describes the FP trials with a window of [-2;2] around a click. The model used for classification is the one depicted in Figure 32. However the output layer changed from two to three outputs for the Post Click.

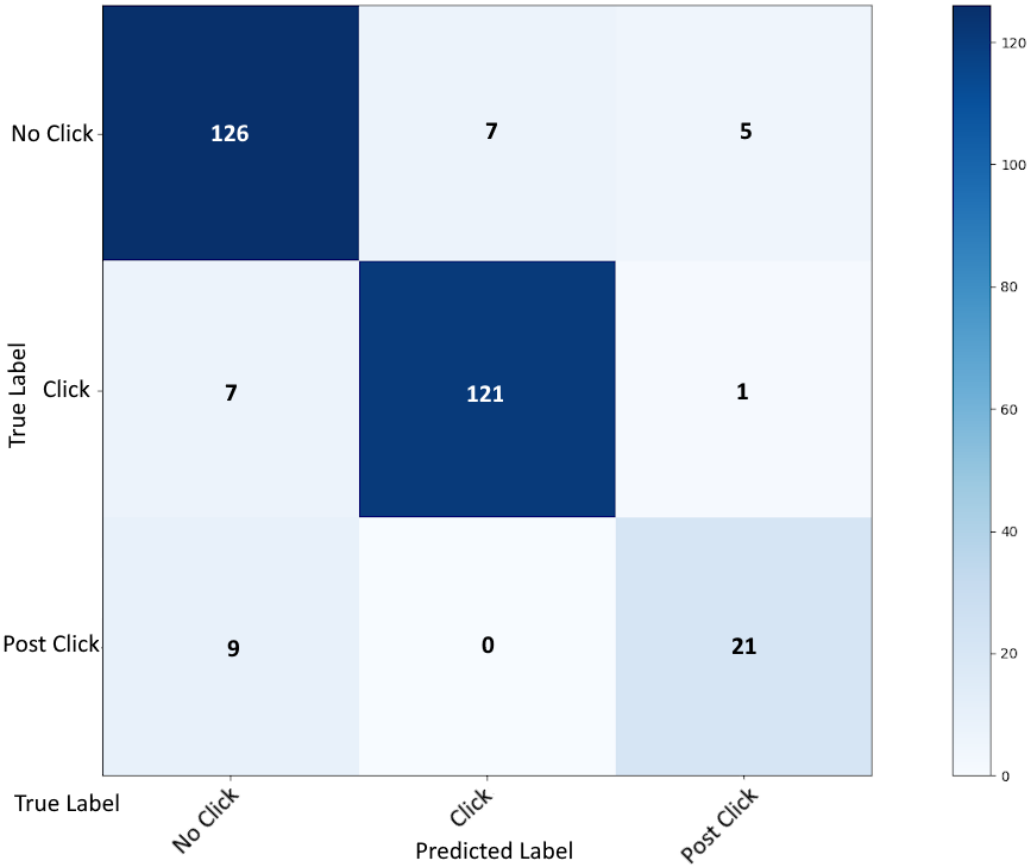


Figure 44: UNP1; M1 channel; 4 sec trials

5.4.2 Distinguish two different clicks

The NN-model was also tested on distinguishing two different clicks. The model used for classification is the one depicted in Figure 32. However the output layer changed from two to three outputs for the DLPFC Click.
The second click was executed by mentally counting backwards. The model was trained and tested on WAM data for the M1 clicks, and on localizer data for mental counting.
As seen in Figure 45 the classifier is able to classify M1 clicks very good. DLPFC clicks on the other side is often mistaken with No clicks.

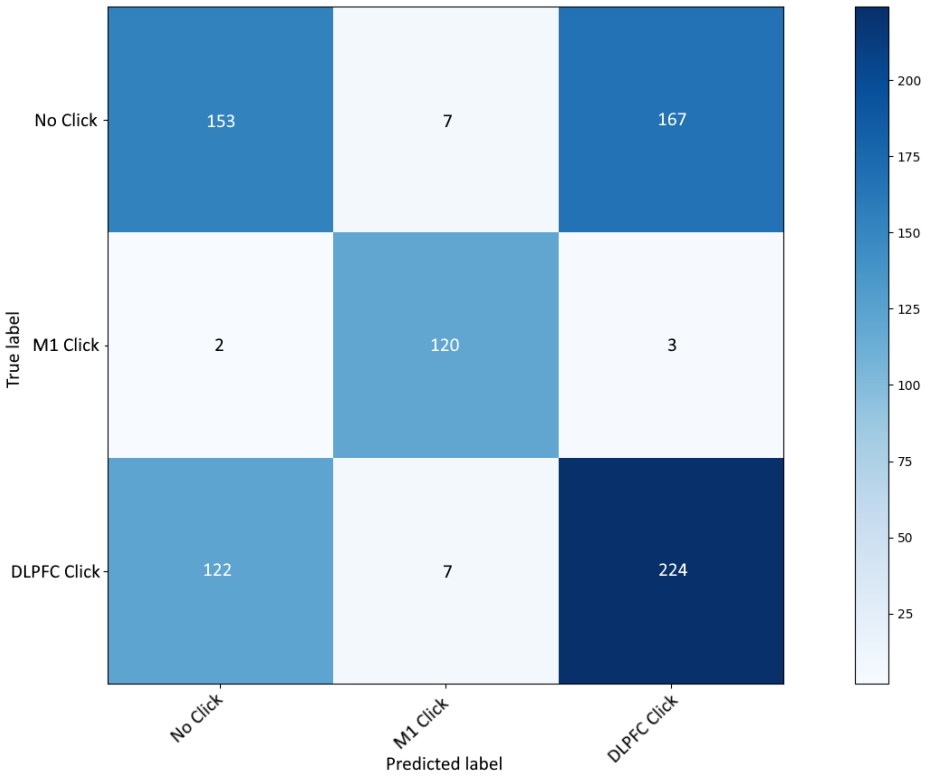


Figure 45: UNP1; M1 & DLPFC channel; 4 sec trials

* Signal length depends on length of trial (seconds) and sample rate (Fig.18,24,32)
** Value depends on how many channels are added to the Input (Fig.18, 24, 32)

6 Discussion

The generalized "brain-click" classifier is able to distinguish different types of brain patterns. But one must keep in mind that it can not work miracles. The used neural network only works well with a large dataset and if the training data is reliable.

6.1 Dataset

As already mentioned a NN-model works better the more reliable data is available. Further it also performs better, the more the signal properties differ from each class [13]. The following sections describe some issues that occurred during the project.

6.1.1 Participants

UNP1's signals are very good compared to UNP4. As a result, the NN-models performance is also worse for UNP4. This is due to several reasons.
First of all the beta and gamma bands are not as strong for UNP4 as UNP1. This is most likely due to the fact that UNP4 is suffering from a brain stem stroke, which is why her brain signals are more impaired.
Another disadvantage that results from her disease is that her signals have a lower dynamic range. That means that the brain signals amplitudes are low compared to UNP1. It also takes her longer to build up the control-signal.
UNP4's dataset is also much smaller than UNP1's [30]. This results from her shorter participation in the study. But it is also more difficult to acquire data from her because she tires very quickly. This also leads to fewer runs during training sessions.

All these facts are reasons for the bad classification results for UNP4 as seen in Figure 38.

To get a better feeling of the participants' datasets, one must understand from which tasks the trials come from.

6.1.2 Tasks

Since UNP1 has enough data for training and testing there is not any problem for the NN to distinguish between clicks or non-clicks. Therefore WAM and cWAM data were sufficient.

UNP4, on the other hand, has much less WAM & cWAM data as mentioned in chapter 6.1.1. Additionally to the afore mentioned reasons the electrodes used for bipolar measurements are not consistent. To overcome this issue the electrode pair with the highest amount of trials was used for training and testing. This decreased the amount of the already small dataset of UNP4. For this reason Localizer data was added to the dataset
However, to complicate matters more, there are two different types of tasks which are the single and multiple execution tasks (4.3.2). Furthermore, it also describes how trials are extracted from both methods.
According to [34] it is not an optimal choice to use both tasks at the same time because the signal properties can be different from each other. Nevertheless, it is better to add the Localizer data since the dataset increases a lot and classification improves as one can see in Figure 38.

One must also keep in mind that classification for UNP4 never worked that well, so far. So it is also possible that she has difficulty to perform the task as instructed to create brain-clicks. In Figure 40 one can see that the amplitudes for the frequency ranges for the different classes are very similar. But here again, the reason again could be that the dataset is just too little [30].

Regardless of how large the dataset is, a classification model must be set up to distinguish the different trials as good as possible.

6.2 Neural Network

The neural network is the heart of this project. Although it is a good way of classifying data there are a lot of things that need to be taken into consideration

[24]. This makes configuration of a NN quite hard. The major topics are dealt with in the next chapters.

6.2.1 Time data

In contrast to the previous classification models (chapter 4.4.1) this model uses the time domain instead of the frequency domain for classification. The thought behind this idea is that the time domain contains all the frequency information and therefore it does not need to be extracted before classification. After decoding what was encoded in the NN-nodes (Fig. 27-30) one can see that the NN also trains on the important frequency bands. The decoded nodes are discussed in detail in chapter 6.3.1.
In the end, the time domain proved to be a good choice. An advantage of using it is that one preprocessing step can be cancelled which is the calculation of the frequency spectrum. Further, model training did not take long at all. But this of course depends on the size of the dataset, computational power and the batch size.

6.2.2 Dataset distribution

A problem that always existed during evaluation was the class distribution. Independent of the patient, the amount of Non-Click trials was always higher than the amount of Click trials. This also resulted from the tasks used for training (WAM, cWAM), which had more trials in which nothing was done. For supervised learning methods, it is preferred to have an even distribution of classes to avoid bias during classification [31].
To overcome this drawback more data or more distinguishable classes are necessary. But the quality of the data for both participants has already been mentioned in chapter 6.1. This could also be a reason why UNP1 performed better than UNP4. For this reason, the evaluation of accuracy distributions was maybe not the best measure. The confusion matrix would already give more information of the classified data. But another more informative evaluation could be the accuracy calculation (or distribution) for each class.

6.2.3 Sliding window

As one can see in figure 26 the NN-model works very good when using the M1 channel. During the cues the prediction is high and between the cues it is close to zero. Therefore the plot shows that the NN-model could be used reliably in real time situations.

6.2.4 Pooling

During this project, Max-Pooling was the first choice. But another way of pooling is Average-Pooling. In my opinion, this could still be taken into consideration since UNP4's signals have a lower dynamic range than UNP1's [32]. For UNP1 no negative changes in accuracy are to be expected with Average-Pooling. But the dynamic range for UNP4 could be improved and therefore a higher accuracy may be achieved.

6.2.5 Update neural network

An important function which is missing for the generalized brain-click classifier is an update function [33]. This update function should take the NN with already trained weights and adapt the weights. Therefore new unseen data is used to continue training with it. The advantage of updating the model instead of retraining on the complete dataset is that the training time is much less.

6.2.6 Two click classifier

In chapter 5.4.2 a first attempt of adding a second click was made. As already discussed, the M1 clicks were classified very good, but the DLPFC clicks are often mistaken with No Clicks (Fig. 45) [29].
I believe that the problem at the moment is that there is not enough data for the second click. Therefore also localizer data was used for the mental counting backwards task.
To prepare the localizer data, the method described in chapter 4.3.2, section: "Multiple execution task" - b), was used. This was done because good fitting parameters, for the frequency subtraction method, for mental counting, were not found

yet. Which means that the performance to classify DLPFC clicks could be further improved in two ways:

i. Finding good fitting parameters for the mental counting task, and therefore use method "Multiple execution task" - a).

ii. Aquire more data in time domain for the mental counting task.

6.2.7 Design

Different brain regions work at a different pace and length. But this also depends on the task. The current NN-model bounds the channels to have the same channel length (length of a trial in sec) when using M1 and DLPFC channels at the same time. It would definitely be better when those can be chosen freely. Each channel would get it is own Input and is treated differently. At some point the branches are the concatenated and used for calculating an output. Examples of neural networks with different Input amounts can be seen in Figure 46.

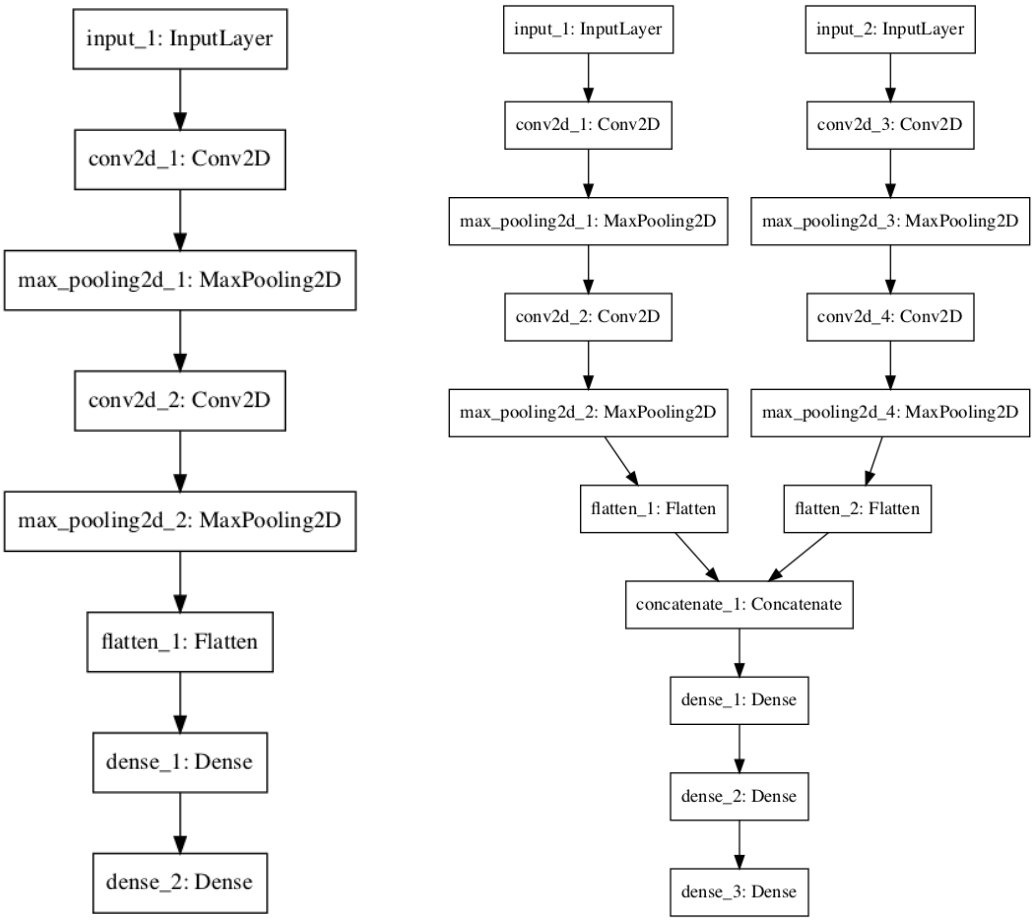


Figure 46: Example of neural networks with single and multiple inputs [43]

6.3 Better understanding of model

An important topic to improve classification is to understand what the NN does. The following chapters explain how this was attempted to be realized. They also explain how model understanding could even be improved.

6.3.1 Decode nodes

To understand what the NN-model interprets from the input signals the nodes needed to be decoded after training. This is done with the gradient ascent method (chapter 4.4.3) [23] [35].

When looking at the results for the decoded nodes one can see that patterns were able to be decoded (Fig. 27-30). Figure 30 shows the decoded output layer and the signals that were interpreted in the model after decoding. Figures 5&7 and chapter 4.4.1 show that this result is expected since the same property was used for the Frequency-band subtraction method.
The nodes in lower layers (e.g., Fig. 27, Fig. 28) also show that more frequency bands have an impact on the result of the NN. Some nodes also show that the Gamma-Band have a slight impact on the result.

At the moment the nodes are just decoded and visualized, but it is not known how much impact each node has on a specific class. To understand what is encoded in the neural network it would be beneficial if this information can be extracted as well. Further one may also see if there are more specific frequency ranges that are used for classification.

6.3.2 Recurrent Layer

Another way of improving the understanding of the neural network is the adaptation of the recurrent layer. During the whole project, standard Long-Short-Time memory (LSTM) cells were used for the recurrent layer [36]. But except for the number of cells, no fine-tuning was performed. When taking fine-tuning into consideration the performance of the NN-model could improve [24], especially if more classes are added to the neural network and if the added classes are time-relevant (e.g., error potentials [37]).

7 Conclusion

This thesis has shown that it is possible to create a neural network for the classification of ECoG-signals in time-domain, from a single bipolar ECoG. There are two main advantages using time instead of frequency domain: first of all, the elimination of computational expensive preprocessing steps. This is important since the classifier should run on a battery-driven implanted device; and secondly, the possibility of detecting time relevant signals, which makes it suitable for a wide range of applications. The generalized brain click classifier is also capable of detecting multiple classes.

However, it has been proven once again that it is important to have a large dataset and classes with distinguishable features. We could see this when comparing the performance of UNP1 and UNP4. With UNP4 having less and inferior data than UNP1.

Nevertheless, I am confident that it is possible to improve the performance of UNP4. Therefore, more data needs to be collected and the neural network parameters need to be optimized. As already mentioned in the discussion there are some possibilities of doing this.

Lastly, I think it is crucial to understand what the neural network interprets from the given data. Understanding the encoded nodes would make fine-tuning much easier and improve performance. Further, I believe that there is more useful information encoded in the brainwaves than we know at the moment. To quote Antoine de Saint-Exupe´ry: "What is essential is invisible to the eye" (Dorison et al., 2012).

References

[1] Vansteensel, MJ., Pels, EGM., Bleichner, M., Branco, M., Denison, T., Freudenburg, Z., Gosselaar, P., Leinders, S., Ottens, T., Van Den Boom, M., Van Rijen, P., Aarnoutse, EJ. and Ramsey, N. (2016). Fully Implanted Brain–Computer Interface in a Locked-In Patient with ALS. New England Journal of Medicine, 375(21), pp.2060-2066.

[2] Gruis, K., Wren, P. and Huggins, J. (2011). Amyotrophic lateral sclerosis patients' self-reported satisfaction with assistive technology. Muscle & Nerve, 43(5), pp.643-647.

[3] Mohanchandra K., Saha S., Lingaraju G.M. (2015) EEG Based Brain Computer Interface for Speech Communication: Principles and Applications. In: Hassanien A., Azar A. (eds) Brain-Computer Interfaces. Intelligent Systems Reference Library, vol 74. Springer, Cham

[4] Normann, R. and Fernandez, E. (2016). Clinical applications of penetrating neural interfaces and Utah Electrode Array technologies. Journal of Neural Engineering, 13(6), p.061003.

[5] Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, et al.. A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces: A 10-year Update. Journal of Neural Engineering, IOP Publishing, 2018, 15 (3), pp.55. ff10.1088/1741-2552/aab2f2ff. ffhal-01846433f

[6] Gao, Y. (2014). Improved Common Spatial Patterns on EEG Feature Extraction. Advanced Materials Research, 926-930, pp.1814-1817.

[7] Julia Berezutskaya, Zachary Freudenburg, Erik Aarnoutse, Mariska Vansteensel, Sacha Leinders, Elmar G. M. Pels, Nick Ramsey. 2018 Using a convolutional neural network for improved click detection in an implanted BCI setup. 7th International BCI Meeting, Pacific Grove, CA

[8] Selim, Abeer & Wahed, Manal & Kadah, Yasser. (2009). Machine Learning Methodologies in Brain-Computer Interface Systems. 1 - 9. 10.1109/CIBEC.2008.4786106.

[9] Leuthardt, E., Pei, X., Breshears, J., Gaona, C., Sharma, M., Freudenberg, Z., Barbour, D. and Schalk, G. (2012). Temporal evolution of gamma activity in human cortex during an overt and covert word repetition task. Frontiers in Human Neuroscience, 6.

[10] Branco, M., Freudenburg, Z., Aarnoutse, E., Bleichner, M., Vansteensel, M. and Ramsey, N. (2017). Decoding hand gestures from primary somatosensory cortex using high-density ECoG. NeuroImage, 147, pp.130-142.

[11] Thomas, John & Maszczyk, Tomasz & Sinha, Nishant & Kluge, Tilmann & Dauwels, Justin. (2017). Deep learning-based classification for brain-computer interfaces. 234-239. 10.1109/SMC.2017.8122608.

[12] Nurse, E., Karoly, P., Grayden, D. and Freestone, D. (2015). A Generalizable Brain-Computer Interface (BCI) Using Machine Learning for Feature Discovery. PLOS ONE, 10(6), p.e0131328.

[13] Freudenburg, Z., Branco, M., Leinders, S., Vijgh, B., Pels, EGM., Denison, T., Berg, L., Miller, K., Aarnoutse, EJ., Ramsey, N. and Vansteensel, MJ. (2019). Sensorimotor ECoG Signal Features for BCI Control: A Comparison Between People With Locked-In Syndrome and Able-Bodied Controls. Frontiers in Neuroscience, 13.

[14] Crone, N. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. I. Alpha and beta event- related desynchronization. Brain, 121(12), pp.2271-2299.

[15] Crone, N. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. II. Event-related synchronization in the gamma band. Brain, 121(12), pp.2301-2315.

[16] Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham

[17] Crone, N. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. II. Event-related synchronization in the gamma band. Brain, 121(12), pp.2301-2315.

[18] Driver behavior profiling: An investigation with different smartphone sensors and machine learning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Time-window-composed-of-nf-one-second-frames-which-group-raw-sensor-data-samples-The_fig3_315970890 [accessed 25 Jun, 2019]

[19] Hecht-Nielsen, R. (1988). Theory of the backpropagation neural network. Neural Networks, 1, p.445.

[20] Quiza, Ramon & Davim, J. (2011). Computational Methods and Optimization. 10.1007/978-1-84996-450-0.

[21] Sathya, R. and Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. International Journal of Advanced Research in Artificial Intelligence, 2(2).

[22] Prechelt L. (1998) Early Stopping - But When?. In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. (2013) Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034,

[24] Chollet, François. “Keras.” Https://Keras.Io, 2015, keras.io. Accessed Aug. 2019.

[25] Lore, Kin Gwn, "Deep Learning for Decision Making and Autonomous Complex Systems" (2016). Graduate Theses and Dissertations. 15965.

[26] Zhang, A., Lipton, Z., Li, M. and Smola, A. (2019). Dive into Deep Learning. 7th ed. p. Available at: http://www.d2l.ai.

[27] Maas, A., Hannun, A. Y., and Ng, A. Rectifier nonlinearities improve neural network acoustic models. In ICML, 2013

[28] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.

[29] Erik Aarnoutse, Anouck Schippers, Sacha Leinders, Elmar G. M. Pels, Mariska Vansteensel, Mariana Branco, Nick Ramsey, Zachary Freudenburg 2018 Thinking outside the motor cortex: Adding prefrontal cortex activity enhances decoding performance of a fully implanted motor cortex BCI. 7th International BCI Meeting, Pacific Grove, CA

[30] Pasini A., "Artificial neural networks for small dataset analysis," J. Thorac. Dis. 7(5), 953–960 (2015)

[31] Yerawar, P. and Pakle, G. (2018). A Survey of Different Techniques to Handle An Unbalanced Dataset. International Journal of Computer Sciences and Engineering, 6(12), pp.818-824.

[32] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In International Conference on MachineLearning, 2010

[33] Hassanien, A. and Gaber, T. (2017). Handbook of Research on Machine Learning Innovations and Trends. Hershey: IGI Global.

[34] Ikeda, A., Lüders, H., Burgess, R. and Shibasaki, H. (1993). Movement-related potentials associated with single and repetitive movements recorded from human supplementary motor area. Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section, 89(4), pp.269-277.

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997

[37] Kreilinger, A., Neuper, C. & Müller-Putz, G.R. Med Biol Eng Comput (2012) 50: 223. https://doi.org/10.1007/s11517-011-0858-4

[38] Computersciencewiki.org. (2019). Max-pooling / Pooling - Computer Science Wiki. [online] Available at: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling [Accessed 21 Aug. 2019].

[39] Jeremy Jordan. (2019). Deep neural networks: preventing overfitting.. [online] Available at: https://www.jeremyjordan.me/deep-neural-networks-preventing-overfitting/ [Accessed 22 Aug. 2019].

[40] Medium. (2019). Convolutional Neural Net in Tensorflow. [online] Available at: https://blog.goodaudience.com/convolutional-neural-net-in-tensorflow-e15e43129d7d [Accessed 22 Aug. 2019].

[41] Superdatascience.com. (2019). SuperDataScience. [online] Available at: https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-lstm-practical-intuition [Accessed 22 Aug. 2019].

[42] Deeplizard.com. (2019). Visualizing Convolutional Filters from a CNN. [online] Available at: https://deeplizard.com/learn/video/cNBBNAxC8l4 [Accessed 22 Aug. 2019].

[43] Brownlee, J. (2019). How to Use the Keras Functional API for Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/keras-functional-api-deep-learning/ [Accessed 29 Aug. 2019].

[44] Dorison, G., Loisillier, M., Morel, D., Zedarkcrystal, Python, I., Smith, A., Smith, O. and Saint-Exupéry, A. (2012). The little prince. Minneapolis, Minn.: Graphic Universe.