Bachelor Thesis

Deep EEG: A Biologically Inspired Signal Processing Chain for Brain Data

Justus Friedrich Hübotter

International Studies Biomimicry B.Sc.
Hochschule Bremen
August 8th, 2017

First Supervisor:
Prof. Dr. Jan-Henning Dirks
Professor for biological structures and biomimicry at Hochschule Bremen

Second Supervisor:
Prof. Dr. Frank Kirchner
Director of Robotics Innovation Center (DFKI, Bremen)

Academic Advisors:
Dr. Mario Michael Krell
International Computer Science Institute (UC, Berkeley)

Dr. Elsa Andrea Kirchner
Robotics Innovation Center (DFKI, Bremen)

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

HSB

"If a technological feat is possible, man will do it.
Almost as if it's wired into the core of our being."

– Major Motoko Kusanagi
Ghost in the Shell (1995)

Statutory Declaration

I hereby declare that I have written the presented Bachelor Thesis independently, without any help whatsoever, and that I have not used any sources or resources other than the ones I have indicated. All passages, which are taken literally or meaningfully from publications, have been indicated as such with the respective sources.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig, ohne fremde Hilfe angefertigt habe und dass ich keine anderen als die von mir angegebenen Quellen und Hilfsmittel genutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich unter Angabe der Quellen als solche kenntlich gemacht.

Place, Date Justus F. Hübotter

Abstract

This bachelor thesis proposes a new, artificial neural network and deep learning–based processing chain for electroencephalographic data classification and compares it to a classical, SVM–based processing chain (baseline). This represents a strong interdisciplinary approach, using a *biologically inspired* deep learning method on a *neurocientific* problem, while practical implementation requires knowledge in *computer science*. Spatial and temporal filtering is achieved through convolutional layers, as they have brought breakthroughs in various fields of machine learning and pattern recognition in the recent past. For a proof of concept, P300 data is used for training and testing the new node within the pySPACE framework. Different preprocessing cases of EEG data are compared to see whether data reduction is helpful or if it might be obstructive when trying to generate subject independent features. Results of the project show the successful implementation of an optimized, standard processing chain with high classification accuracy for current and future EEG–aided projects to improve human–machine interaction. While performance exceeded the baseline in all tested cases significantly, the true potential of this technology lies in subject transfer. Also, it is shown that classification performance is superior when the data contains high frequencies. However, further studies with more data are necessary to comprehend automatically generated features and interpret their meaning in a neuroscientific relation.

Keywords: Brain–Computer Interface, Electroencephalographic Data Classification, Deep Learning, Subject Transfer

Zusammenfassung

Diese Bachelorarbeit stellt eine neue Verarbeitungskette zum Klassifizieren von EEG Daten vor, basierend auf künstlichen neuronalen Netzen und Deep Learning und vergleicht diese mit einer klassischen, SVM basierten Methode. Dies stellt einen hochgradig interdisziplinären Ansatz dar, welcher ein *biologisch inspiriertes* Verfahren auf ein *neurowissenschaftliches* Problem anwendet, während die Implementierung auf Kenntnissen der *Informatik* basiert. Räumliche und zeitliche Filter werden durch Convolutional Layer ersetzt, welche in naher Vergangenheit Durchbrüche in verschiedenen Feldern von maschinellen Lernverfahren und Mustererkennung erzielen konnten. Als Machbarkeitsnachweis verwendet diese Studie P300 Daten zum Trainieren und Testen eines neuetwickelten Knotens für das pySPACE Framework. Verschiedene Fälle der Vorverarbeitung für EEG Daten werden miteinander verglichen, um festzustellen, ob Datenreduzierung hilfreich ist oder das Generieren von subjekt–unabhängigen Merkmalen erschwert. Die Ergebnisse des Projektes zeigen die erfolgreiche Implementierung einer optimierten Prozesskette mit hoher Klassifikationsgenauigkeit für momentane und zukünftige Projekte zum Verbessern von EEG basierten Mensch–Maschine Schnittstellen. Obwohl die Ergebnisse die Baseline in allen getesteten Fällen signifikant übertreffen, liegt das wahre Potential dieser Technologie im Subjekt Transfer. Außerdem konnte gezeigt werden, dass die Klassifikation signifikant besser erfolgt, wenn die EEG Daten hohe Freqenzen beinhalten. Trotzdem müssen weitere Studien zu diesem Thema mit mehr Daten durchgeführt werden, um automatisch generierte Merkmale nachzuvollziehen und ihre Bedeutung in einen neurowissenschaftlichen Kontext zu bringen.

Contents Page

1 Introduction

Recent advancements in technology and neuroscience allow a more widespread usage of brain signals to interact with the environment. Big enterprises such as Facebook or Neuralink have declared establishing Brain–Computer Interfaces (BCIs) into everyday life as one of their major goals for the upcoming decade. A possible way to realize such interaction is given through reading and real time interpretation of brain activity in the form of electroencephalographic (EEG) data (Kirchner, 2014). In this context, BCIs can be seen as pattern recognition systems for neuronal activity.

At the German Research Center for Artificial Intelligence (DFKI), there are various projects working with this technology for different applications. To identify certain events during online, single–trial brain signal decoding and react accordingly, strong real–time classification is indispensable. For example, a developed motion supporting exoskeleton needs to know planned movements before being executed (Kirchner et al., 2013a, 2016b).

It is imaginable, that in the future the connection between the human brain and computer technology opens up an extremely broad field of applications, which cannot even be completely grasped just yet.

1.1 Vision & Motivation

The vision behind this project is that one day BCIs can be used with minimal effort to effectively support industrial, medical as well as everyday tasks. To achieve this goal, brain activity *reading* and *processing* must be improved. Therefore, it must be the long term goal of the respective research groups to create BCIs, ideally fulfilling the following vague requirements. The future BCI shall be:

- Easy to use, without expert knowledge of neuro– or computer science,
- Applicable with minimized training and setup time,
- Robust to errors and
- Fast, granting real–time interpretation of relevant brain activities

For improving brain signal reading, there are various projects currently working on novel data acquisition techniques, such as minimal invasive, long–term brain implants (Minev et al., 2015) or dry electrode EEG (Lopez-Gordo et al., 2014). However, this thesis is focused on improving the processing and classification of obtained data.

When it comes to data processing, key to minimized training may be creating subject independent features. It is imaginable, that future hardhats or headsets with embedded dry electrodes are used for brain activity monitoring or to control assisting robots in simple or complex industrial scenarios. Evoked brain signals appear somewhat different for each person (Luck, 2014). Also, depiction changes over time, as electrode positioning and contact quality may vary inner–session and from one day to another. Using only sparse information to describe found features in measured brainwaves may lead to unrecognizability and false classification, especially when applied to data of other subjects. However, the fundamental brain activities for each possible user remain the same (Luck, 2014). So instead of learning a set of strongly simplified key features for every eligible person, a very general dataset of desired patterns in EEG data may help to reduce training time for each individual (Lemm et al., 2005). Such a dataset could be automatically updated from monitored user data and thereby be constantly growing. In an ideal case, it would make training obsolete after all.

This theoretic scenario brings up some questions. Can such subject independent features be found? And where would one look for them? A possible solution to this problem could be deep learning. These relatively new state of the art algorithms for classification problems are able to cope with great amounts of data and learn relevant features automatically, even if they sometimes are too abstract to grasp by human beings.

1.2 Project Objective

In this bachelor thesis a new, artificial neural network (ANN) and deep learning (DL) based processing chain is created and compared with a classical processing chain (baseline) for EEG data classification. This represents a strong interdis-

ciplinary approach, using a *biologically inspired* DL method on a *neurocientific* problem, while practical implementation requires knowledge in *computer science*. Within this project the following key questions are ought to be answered:

1. Is the DL approach appropriate for P300 EEG data classification?
2. How much of the classical preprocessing and data reduction is necessary?
3. Can achieved results outperform existing processing pipelines on the same data?
4. Can subject independent features be found?

In order to find respective answers, different setups for an ANN using state of the art methods for spatial and temporal filtering will be tested. Taking different examples from literature into consideration, evaluated factors include:

- Three different sampling rates,
- Three different frequency ranges,
- Spatial & temporal filtering with convolutional layers,
- Quantity of fully connected layers & included nodes and
- State of the art regularization methods

Result of the thesis should be an optimized, DL based, standard processing chain with high classification accuracy for current and future EEG–aided projects to improve human–machine interaction. The architecture shall be inspired by the mentioned classical chain, which allows easy comparison to the baseline. This project is important to pursue the overall goal of creating subject independent BCIs, which allow application in various fields with minimized training. The proposed processing chain shall be intuitive to use, even without major knowledge of DL, neuroscience or complicated prior manual preprocessing.

The following Section introduces several state of the art components which are important for this project. Section 3 explains the used dataset and methods in detail. In Section 4, achieved results are displayed, evaluated and discussed. The conclusion refers back to the above–stated research questions and sums up where the project has lead, but also gives a short outlook on what is to be expected in the future.

2 State of the Art

As this project combines expertise from multiple fields of research, it involves various methods and techniques of which their connection to one another may not appear obvious at first sight. This Section will highlight some of the key components of this project and their relation to one another. It is important to understand that although some of the introduced concepts are based on ideas from the early last century, it was not until now that technology has caught up to a point, where combining them reveals the true potential lying within.

2.1 Brain–Computer Interfaces

A Brain–Computer Interface (BCI) is a communication pathway between the central nervous system of living being and another external device. This term was commonly used after Vidal (1973) published a review on the applicability of electrically measured brain activity as control input for computer devices. He suggested that *"such a feat is potentially around the corner"* and tried to answer the question whether BCIs might be used for *"controlling [...] prosthetic devices or spaceships"* (quoted from Vidal (1973)). Although the paper was written more than four decades ago, the topic remains prevailing until today.

BCIs have two main components: One to *obtain* brain signals and the other to *interpret* them. Kaur and Singh (2017) describe various ways to encounter the given problem of establishing communication between a biological structure and a technical device. A key factor to differentiate between BCIs is whether *invasive* or *non–invasive* methods are used to read out signals from the brain.

(Partial) invasive methods, such as electrocortiogram or direct signal derivation from single braincells grant a comparatively better signal strength and spatial assignability than it holds for non–invasive methods. However, they have the obvious and great disadvantage, that one or multiple sensors have to be placed underneath the skull through a surgical opening. This can be achieved momentarily, while the skull remains open, or long–term with respective intracortical implants. Nevertheless, a foreign object will be in direct contact with the brain, potentially influencing its functionality or bringing up a risk for other

complications such as inflammation, repulsion or damaging brain tissue. Secure electrode arrays for long–term recording or stimulating neuronal cell activity, which can be implanted minimal invasive, are currently subject of research but have yet to persist in vivo (Gilmour et al., 2016). Also, these procedures bring great cost and effort of preparation to the entire process and might in many cases not withstand a risk–effectiveness consideration (Kaur and Singh, 2017).

Non–invasive methods also mostly rely on electrical measures of brain activity. This approach records signals less effectively, because the currents have to pass the skull and other tissue before reaching the measuring electrodes. Theoretically, the needed equipment is well tested, movable, increasingly low–cost as well as easy to use. The sparse signal–to–noise–ratio (SNR) and the sensitivity to artifacts caused by external electrical contaminations are the greatest drawbacks for this technology (Kaur and Singh, 2017). Consequently, experiments are sometimes performed stationary within a shielded cabin. Ideally, a small set of dry electrodes would be used for everyday applications of a BCI such as the *EMOTIV EPOC*, granting minimized setup preparation time. Unfortunately, it has shown to be unreliable in other research tasks and needs further improvements to be considered a suitable state of the art technology (Duvinage et al., 2013). This project is using brain signals recorded with the non–invasive method of electroencephalography, briefly explained in the next subsection.

When signals have been received, further processing is necessary to correctly identify and interpret relevant aspects. For this matter a computational unit *filters* the data stream and uses at least one algorithm for signal *classification*. Both key elements are objects of investigation in this project and will therefore be explained in subsequent Sections.

Obtained information can be used to monitor relevant brain activity when performing specific tasks (Kirchner, 2014), or predefined commands can be sent to a device of which control is desired. There are various imaginable fields of *mind controlled* applications, such as medical rehabilitation (Kirchner et al., 2016b), industrial and research–based robotic teleoperation (Muelling et al., 2015), or the gaming industry (Kaplan et al., 2013). This technology is particularly interesting for people that have no other option to communicate due to their inability of muscle control (Locked–in syndrome) (Kaur and Singh, 2017).

2.2 Electroencephalography

The idea that thoughts, emotions, behavior and character traits find their origin in the human brain is quite old. Until the 20th century however, this theory was hard to prove, as there was simply no way to measure and quantify such abstract occurrences. A great leap was taken by Berger (1929) with his revolutionary implementation of a device, able to sense electrical currents emerging from the surface of the neocortex. This method is today well known and widely spread under the name electroencephalography (EEG).

Despite various recent inventions, such as the functional magnetic resonance imaging (fMRI), EEG obtains a significant position in research and diagnostics. Hobson (2009) names four convenient main characteristics which justify this: As mentioned before, this technique can be used non–invasive on the outside of the scalp. It has an outstanding temporal resolution at the millisecond range. Also, needed equipment is easy to use and considerably inexpensive, compared to fMRI. But what seems to be the most important unique feature, is the ability of the respective subject to be able to move freely throughout the process of measuring its brainwaves.

Hobson (2009) further explains that the peripheral neurons of the neocortex are generally arranged in a layered structure, while their main axes remain parallel towards one another. Each single cell induces an ionic current at its surface. The neurons palisade like macro–structure creates an integration of these nearby currents. When a neural population is activated in a synchronized matter, the superposition of evoked extracellular currents can be measured with an electrode of perpendicular orientation on the scalp. More abstractly speaking, the brain can be seen as volume conductor with local differences in electrical potentials. These different potentials can be measured in reference to either a specific location or the overall mean value. Figure 1 shows this functional principle of electroencephalography in a schematic matter.

After the potential differences have been recorded by the electrodes, an amplifier increases the original signal strength of few μV for each channel and transforms it from analog to digital, so it can be further processed by a computer.

However, the assignability or spatial resolution of the incoming signals are considerably weak. This is due to irregularities like the sulci of the cortex, the tissue and bone layers between signal source and receiver and the big gap between the sensing electrodes compared to the neurons distance to one another (Hobson, 2009).

In order to get comparable results between different subjects of investigation, electrode placement is normalized. The number of used electrodes can vary from only a few up to 256 at the moment. Figure 2 shows the 64 used electrode positions in this project. This represents an extended version of the widely used *10–20 system* (Jasper, 1958).

EEG data consists of overlaying sine–waves of diverse amplitudes, frequencies and phase differences (Zschocke and Hansen, 2011). Specific frequency ranges have been summarized to frequency bands of which the borders between them vary in literature. Table 1 shows the division of Zschocke and Hansen (2011). The composition of measured EEG bands have been linked to different conditions of brain activity. Most of the relevant portions of ERPs for neuroscience consist of frequencies between 0.01 and 30 Hz (Luck, 2014).

Table 1: Frequency bands in EEG after Zschocke and Hansen (2011).

Name	**Frequency range [Hz]**	**Example association**
sub–δ	< 0.5	Deep sleep
δ	0.5 – 4	Continuous–attention tasks
θ	4 – 8	Drowsiness, idling, repression
α	8 – 13	Perpetuation of vigilance
β	13 – 30	Active thinking, focus, high alert
γ	> 30	Cross–modal sensory processing

The mathematical law of the *Nyquist Theorem* states, that any frequency of half the sampling rate or less can be fully encoded within discrete samples. It also states, that occurring frequencies of at least twice the sampling rate will appear as low frequency artifacts within digitized data. However, in practice the sampling rate should be three times the upper cutoff frequency (Luck, 2014). This becomes important when considering different down sampling rates and bandpass filters to process EEG data before training classification algorithms.

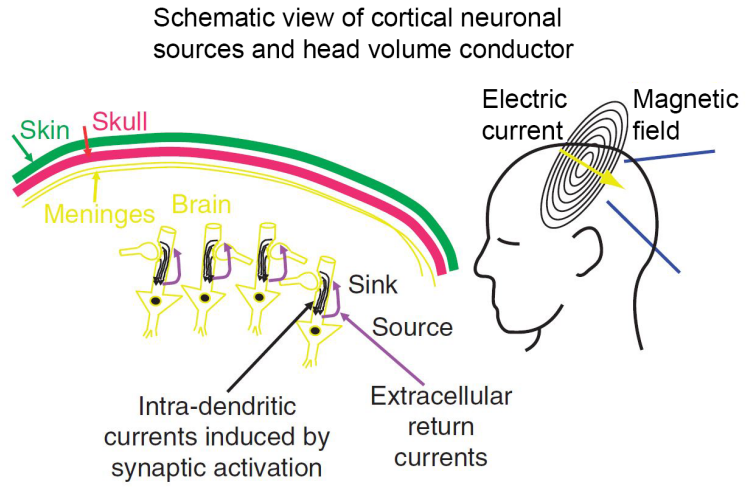

Figure 1: Functional principle of electroencephalography, measuring emerging electrical currents at the surface of the skull. Graphic changed from Hobson (2009).

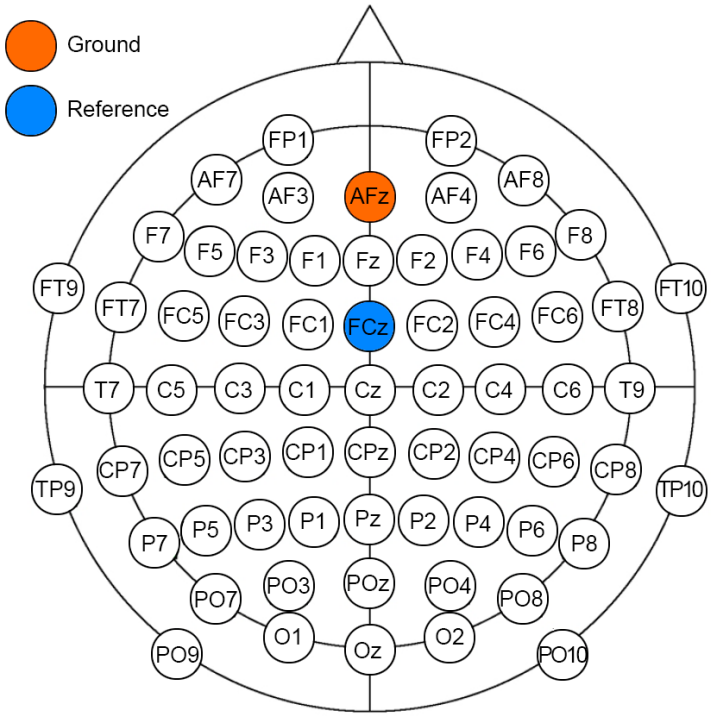

Figure 2: Setup of electrode placement for electroencephalography used in this project (actiCAP, 64 electrodes, Brain Products). The given letters indicates the brain region on which the respective electrodes are placed: **O** occipital, **P** parietal, **T** temporal, **C** central, **F** frontal, **Fp** pre–frontal.

2.2.1 Event Related Potentials

Based on the EEG from Berger (1929), Davis (1939) studied effects of acoustic stimuli within brain activity. He was able to identify reoccurring patterns linked to specific stimuli. Today we call these phenomena *event related potentials* (ERPs). As they are small in size compared to the noisy EEG data, they are best visible averaged over a number of trials (Sammut and Webb, 2011). Due to the above–stated conditions under which EEG data is sampled, ERPs cannot easily be allocated with a specific group of neurons, but the outstanding temporal resolution allows conclusions on sequential activations and occurring frequencies.

ERPs can have an external stimulus as a source, but can also be found associated with cognitive tasks (Luck, 2014). For example, lateralized readiness potentials (LRPs) are evoked when movement of a specific limb is planned. These information are valuable for BCIs that connect to medical rehabilitation or prosthetic devices (Kirchner et al., 2016b; Li et al., 2014).

2.2.2 The P300 Wave

Polich (2007) describes the P300 (or P3) wave as a positive ERP component, usually arising about 300 *ms* after a specific external stimulus of interest (e.g. auditory or visual). The surfacing signal is typically best measured in the central region of the head, e.g. electrodes Fz, Cz or Pz. He differentiates between P300, depending on their origin and occurrence. In *Oddball* experiments, stimuli alternate between frequent *standards* and infrequent *targets*, as shown in Figure 3. When no stimulus attribute change is detected, the current mental model or schema of the stimulus context is maintained and only sensory potentials are evoked. For P3a, perception of a rare or unknown stimulus is needed and is associated with context memory updating, as shown in Figure 4. For P3b, response of the subject to such a rare, task–relevant stimulus is required, associated with an attention based task and respective neuronal activity.

Although P300 is suitable for brain activity monitoring, it may not ideal for some BCI applications such as robot control, it is sufficient for this proof–of–concept study. Motor imagery or LRP monitoring appears to be better suitable for control tasks, as they directly represent user intention (Kirchner et al., 2013b).

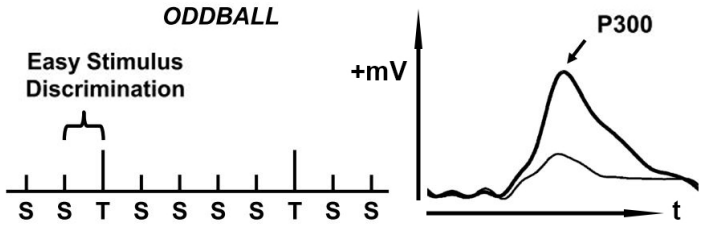

Figure 3: The oddball task presents two different stimuli in a random sequence, with one occurring less frequently (target = T) than the other (standard = S). If a perceived target stimulus requires no further action, usually P3a is evoked. However, if the stimulus is task–irrelevant and subjects are instructed to actively respond, this is associated with P3b. On the right an averaged P300 wave is shown in contrast to an average standard response. Graphic and caption changed from Polich (2007).

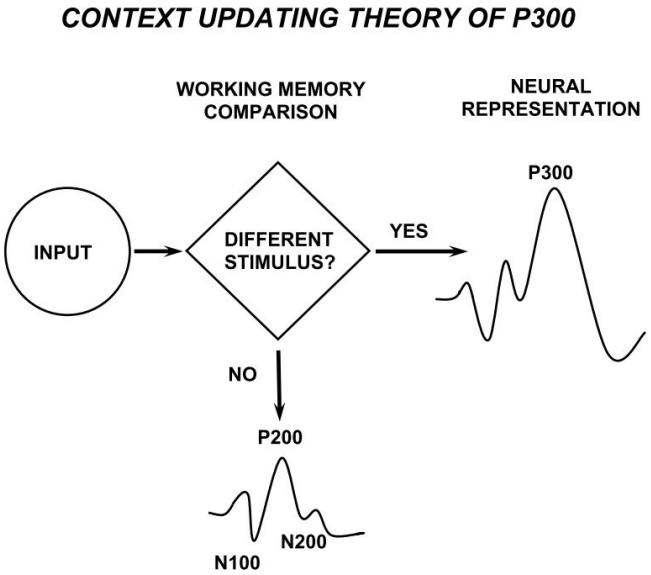

Figure 4: Schematic illustration of the P300 context–updating model. Stimuli enter the processing system and a memory comparison process is engaged that ascertains whether the current stimulus is either the same as previous stimuli or not. If the incoming stimulus is the same, the neural model of the stimulus environment is unchanged, and sensory evoked potentials (N100, P200, N200) are obtained after signal averaging. If the incoming stimulus is not the same and the subject allocates attentional resources to the target, the neural representation of the stimulus environment is changed or updated, such that a P300 potential is generated in addition to the sensory evoked potentials. In this case however, subjects are looking for a predefined rare stimulus instead of stimulus change. Graphic and caption taken from (Polich, 2007).

2.3 Deep Learning

McCulloch and Pitts (1943) tried to explain how neurons in the brain might work and thus modeled a simple neural network using electrical circuits. It was not until 17 years later, when Widrow et al. (1960) used this model and first computer technology to developed the Multiple ADAptive LINear Elements (MADALINE), a simple *artificial neural network*. It was used for binary pattern recognition and is applied for eliminating echoes on phone lines in modified forms until today.

In their recent reviews, Schmidhuber (2015) as well as LeCun et al. (2015) describe the development of current artificial neural networks based on our basic understanding of the human brain and the layered structure of the neocortex. The plasticity of single cells, their connection to other cells and entire areas of the brain can be abstracted as the in– or decrease of their unique connection intensity. In simplified terms, the strength of this bond is dependent on the quantity of its usage, first described by Hebb (1949).

This phenomenon has been model for todays architectures of ANNs, which adapt neuronal connections through *backpropagation* and an assisting *error function*. This procedure is not further explained in this thesis as no changes are made from the standard procedure which is closer elucidated by Hünniger (2013) or Shamwell et al. (2016). The main ANN structure consists of an input layer, a number of hidden layers and an output layer, each containing multiple artificial neurons or nodes as shown in Figure 5. Each connection of individual neurons has a weight value, which is updated after each training step of the network. Every neuron sums up all connected outputs from the previous layer, multiplied with the respective weight as shown in Figure 6. Depending on the neurons or layer's specific *activation function*, output is generated and forwarded to the next layer. While this general *feed–forward* structure remains, size, shape and function of different layers vary strongly for each use case and developer. The following subsections introduce some state of the art techniques or layers relevant to this project.

Bengio et al. (2007) proposed a deep feed–forward network for unsupervised machine learning. This original model with multiple fully connected hidden

layers, allowed a deeper abstraction of the learned training data and its features, later coining the term *deep learning*. A great advantage of DL is the automated feature generation within the abstraction of training data in the ANN layers. This generally allows to work with data without major knowledge about relevant features beforehand.

A prevailing issue with this technology is the incomprehensibility of what happens within the *black box*, making this technology unavailable for strongly regulated services for now. Backtransformation and visualization of learned features within convolutional layers through heatmaps, indicating each data point's relevance for the process is desired. This allows to regain information about significance of individual learned features within a DL network and thereby draw conclusions on the underlying (e.g. neurological) processes. Various methods have been introduced, trying to track and explain individual network decisions (Haufe et al., 2014; Krell and Straube, 2015; Sturm et al., 2016).

Since the introduction of DL, it was able to bring breakthroughs in various fields of machine learning, such as processing images, video, speech, audio and the respective recognition (Längkvist et al., 2014; LeCun et al., 2015; Schmidhuber, 2015). Just alike videos and speech, EEG data has a relevant time dependency (temporal dimension) used to create multidimensional features. Likewise, over the last years, various architectures for DL approaches have been generated, to solve the key problems of spatial and temporal filtering on EEG data (Schirrmeister et al., 2017). High accuracy reported within literature for this pursue emphasizes the state-of-the-art position for this technology.

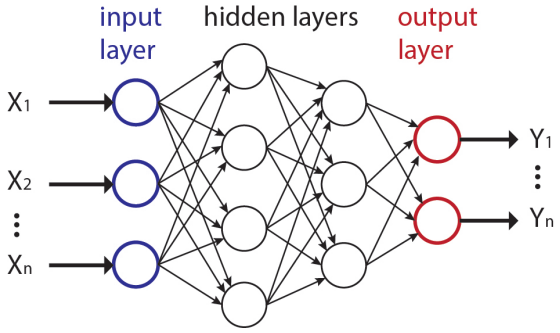

Figure 5: General layered structure of artificial neural networks. A network can contain any number of input and output neurons as well as hidden layers. Every connection between neurons displayed by a thin arrow is associated with a weight value.

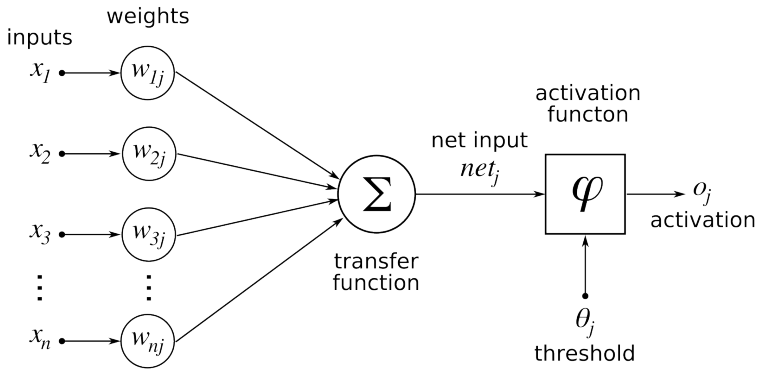

Figure 6: Artificial neuron model from Hünniger (2013). Each input x_i is multiplied with the respective neuronal connection weight w_{ij} before being summed up. If the net input net_j is greater than the threshold θ_j, it is forwarded to the nodes specific activation function φ to calculate the output o_j.

2.3.1 TensorFlow & Keras

When developing new DL architectures, respective research groups or enthusiasts do not have to build everything from scratch. Due to the rapidly arising interest and success of this topic, one can make use of steadily growing libraries, strongly optimized for the specific use of training ANNs.

One of these libraries is the open–source software Tensorflow developed by Google (Abadi et al., 2015). It supports various low–level Application Programming Interfaces (APIs), including one for Python. Low–level in this context means, that it is meant to perform very basic calculations such as matrix multiplications and the underlying source code is optimized for respective operations

in terms of speed. Although it is possible to develop ANNs solely within a Tensorflow API, the library only supports some of the core functions and newly developed features must be implemented by hand. This means that one would write many lines of code to implement functions, which have been defined by other people before and is therefore inconvenient and redundant.

While Tensorflow can be used for a wide range of applications, Keras is a high–level, Python–based API specialized in developing DL architectures (Chollet et al., 2015). It supports many novel functions like the ones listed in the upcoming Subsections. The functions are predefined and can be implemented using mostly one or few lines of code. When it comes to calculations, Keras uses the highly optimized libraries of other software, momentarily supporting Tensorflow or Theano. This means that Keras is dependent on this "underneath" laying software, which is expressed in the term *Tensorflow–backend*.

Graphics processing units (GPUs) are favored for DL due to their many cores compared to conventional central processing units (CPUs). While sequential code runs faster on CPUs due to higher clocking frequencies, GPUs are optimal for simple, parallel processing such as the matrix multiplications in backpropagation and feed–forward in convolutional layers.

2.3.2 Fully Connected Layers

As the name already implies, each neuron in these layers is connected to each input neuron of the prior layer. This is the basic form of ANN layers, which can be found in the first simple, analogue networks of McCulloch and Pitts (1943) as well as Widrow et al. (1960). In todays computer models, every connection between individual neurons has a weight value which is updated after each training batch, an assemblage of a specific number of examples or instances. Therefore, these layers tend to generate a great amount of data which slows down the training process. This is not considered efficient for many applications and although various activation functions can be used, functionality is limited. Because of these drawbacks, network sparsity with improved functionality is desired.

2.3.3 Convolutional & Pooling Layers

Hubel and Wiesel (1962) studied neurons in the visual cortex, when they found that some individual cells only responded in the presence of edges with specific orientation within a restricted region of the visual field. These representations are called *simple features* and the cells *simple cells* respectively. They also found that these neurons are organized in a columnar architecture. In order for represent more *complex features* than straight edges, multiple arrays of such columns are connected hierarchically, each combing inputs from previous layers. Within only few layers, abstract features and geometrics such as e.g. faces or other patterns can be represented. This was not only a great breakthrough in neuroscience, but also model for convolutional neural networks or *ConvNets*.

Before backpropagation had established as the core function for learning in ANNs, Fukushima (1980) created a computer–based, self–organizing neural network for pattern recognition. This so called *Neocognitron* was based on the findings of Hubel and Wiesel (1962) and thereby highly biologically inspired. The fundamental scheme is shown in Figure 7. A functional unit includes a simple layer, creating features on a map, each over an individual area on a given input image. The subsequent complex layer combines created features by respective fields, partially overlapping their neighbors. The connection of two subsequent functional units is of modifiable intensity (weight). These units are chained until original data is abstracted to a unique feature on a single output neuron. An example process is described in Figure 8. This network was primarily used for identifying single letters or digits. Due to overlapping sampling from previous receptive fields, classification is unaffected by shift in position.

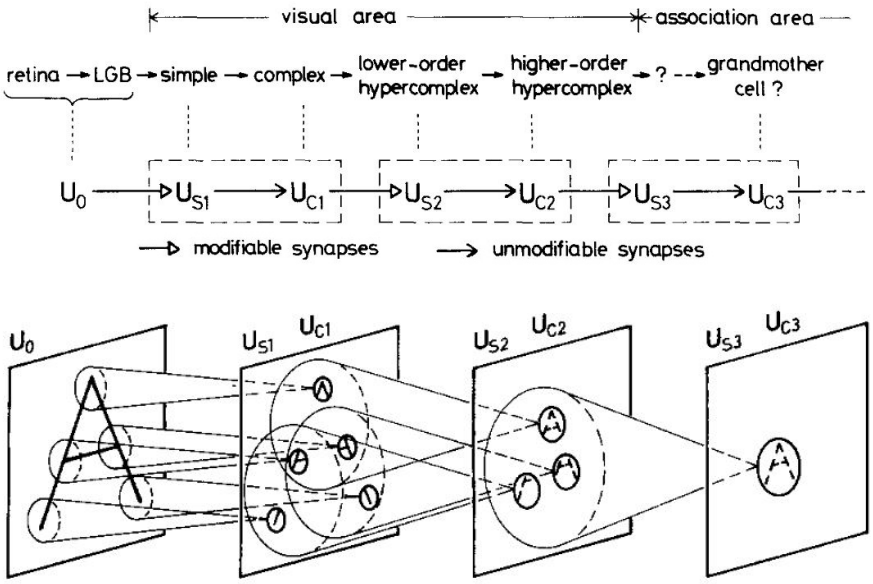

Figure 7: Correspondence between the hierarchy model by Hubel and Wiesel (1962) and the neural network of the Neocognitron: A simple and complex layer are combined to a functional unit. The sequential artificial model includes all steps from the biological model starting with perception, abstraction and finally association. Graphic and caption changed from Fukushima (1980).

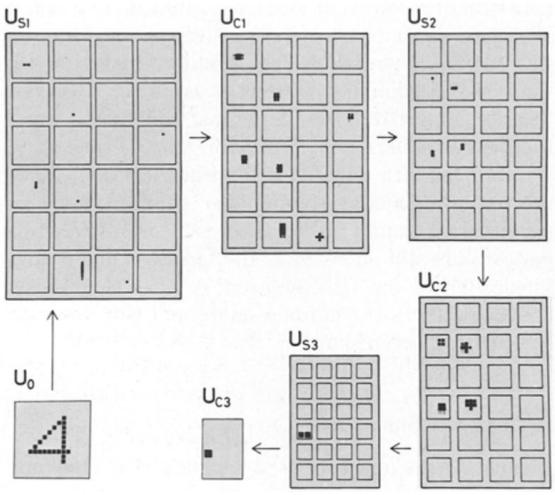

Figure 8: Neocognitron functional example: Input of a pixelized 4 (U_0) is abstracted on a map of 24 features by the following simple layer (U_{S1}). The following chain of simple and complex layers result in a final abstraction (U_{C3}), which uniquely represents given input. Graphic from Fukushima (1980).

After backpropagation had been established, the first ConvNets have been developed by Yann LeCun for image processing, somewhat based on the Neocognitron (LeCun et al., 1995). Similar to the processing of visual input in the occipital lobe of the human brain, first simple features such as lines and their orientation are recognized by specific Kernels. Later in the process, these simple features combine and form nonlinear, more *complex features*, i.e. geometric patterns or faces (LeCun et al., 2015). Convolutional layers are often found in combination with following pooling layers, described below. In terms of the work from Fukushima (1980), convolutional layers can be seen as an advancement of the simple layers. Mathematically speaking, a filter window is moved over an input matrix and for each position the dot product of input datapoints and kernel weights result in the value (feature) on an accordingly shaped output map as shown in Figure 9. Kernel weights remain the same for each position on the input matrix. This strongly reduces the needed weights per layer compared to fully connected layers but introduces the two hyperparameters filter quantity, size and stride (step size of filter for each dimension of input). While few handmade kernels from simple algorithms are sufficient for specific feature extraction, convolutional layer usually rely on greater numbers of kernels with randomly initialized weights. This often results in the network finding relevant features for classification, where they would not have been expected at first sight. ConvNets are commonly used for image processing and object recognition, containing convolutional layers within the early and midrange sequential architecture to automatically generate feature maps. Therefore, 1D, 2D and 3D convolution is natively implemented in Keras. However, these tools have been alienated to process all kinds of data with similar input shape with great success (Schmidhuber, 2015).

While convolutional layers are somewhat equivalent to simple cells in the occipital lobe and the Neocognitron, *pooling layers* can be seen as respective complex cells, combining previously generated features. They also have a distinct filter size and stride. Input features are compared and a forwarded feature is chosen after a determined regulation. Two common methods for feature reduction are average pooling, creating the mean value of respective filter input, or maxpooling, forwarding the greatest input value. The latter is shown in

Figure 10. Usually, the presence of an existing feature in a vague region of an input image is sufficient for classification, while the exact position may not be needed. Thereby, performance becomes unaffected by minor changes in input data, such as in handwritten digits or characters.

All in all, based on the research of Hubel and Wiesel (1962), the visual cortex was model for a self–organizing visual pattern recognition algorithm, initially developed by Fukushima (1980). It was later further modified by LeCun et al. (1995) and is today a standard method used in DL. In this context it has been diverted and applied to various types of data, such as EEG streams by Shamwell et al. (2016) and many more.

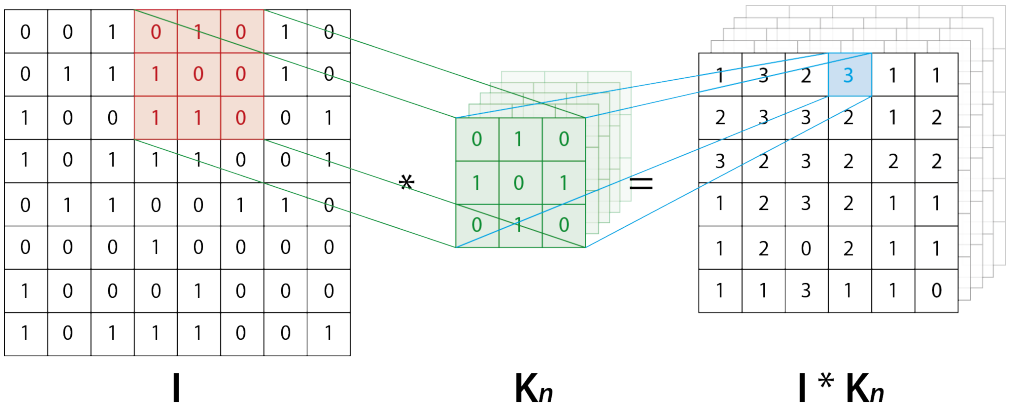

Figure 9: Convolution with a filter size of 3 x 3 and a stride of 1 x 1 results in overlapping feature generation. Multiplying input I with the kernel K_n (dot product) creates the output matrix. Because the weights in K_n are the same for each position on I, considerably less weights need to be stored compared to a fully connected layer creating the same output size. Each convolutional layer can contain any number n filters of the same size and stride, each generating an individual output feature map. Initialization is usually random.

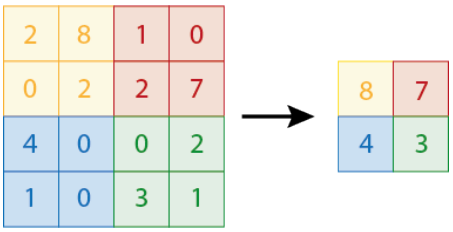

Figure 10: Maxpooling with a filter size of 2 x 2 and a stride of 2 x 2: The greatest value within the moving filter window is forwarded to the output feature map. Because filter size and stride are equal in this example, input areas do not overlap.

2.3.4 Activation Functions

As shown in Figure 6, every artificial neuron has an activation function to determine that neurons output, depending on the net input. While a biological neuron cannot have a negative output, an artificial neuron potentially can. The weights to the following layer determine whether its output operates in a, excitatory or inhibitory matter. However, different activation functions have advantages and disadvantages when used for DL. Figure 11 shows an overview on some commonly used variations. One differentiates between bounded (tanh, logistic sigmoid) and unbounded activation functions (identity, rectified linear unit). Bounded activation functions have the disadvantage of a vanishing gradient, while unbounded outputs tend to increase undesignedly high, distorting relation to other outputs. The latter can be controlled by regularization methods, such as activation constraints. Hahnloser et al. (2000) first introduced the rectified linear unit (ReLU), which has a steady gradient for positive net input and induces network sparsity for negative net input. Due to these characteristics, it is today one of the most commonly used activation functions for ConvNets.

The output layer of a network for classification usually works with softmax function, converting its input to a probability between 0 and 1 for affiliation of each class, while the sum of all probabilities remains 1.

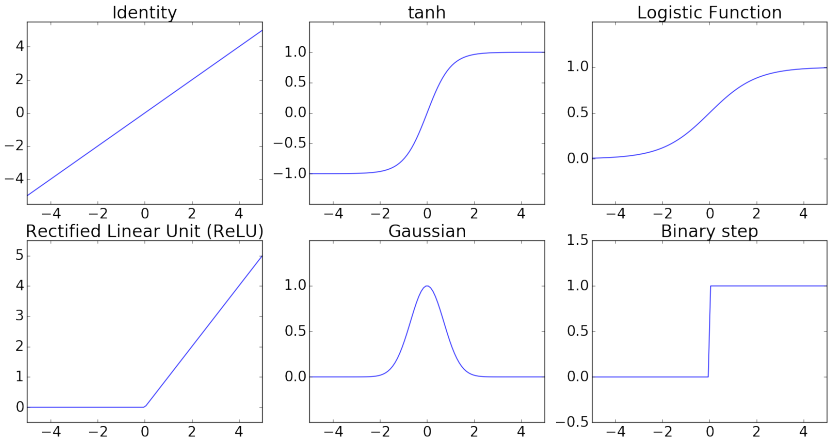

Figure 11: Commonly used activation functions for artificial neurons. One differs between bounded (tanh, log,...) and unbounded (ReLU, identity,...) output.

2.3.5 Network Regularization Methods

In the past years, several complementary functions have been added to the state of the art methodology of DL to increase classification performance, prevent overfitting or speed up training. The following paragraphs introduce and briefly explain some procedures relevant to this project. This list is by no means complete as uncountable variations exist and changes are introduced rapidly.

Early stopping As training proceeds, the network learns a general model of given input data and training & validation error decrease. At some point, the general model has been learned and the network starts to learn noise or very specific examples of input data. At this point, validation loss usually starts to increase again. Early stopping observes this (or any other) metric and stops the training process. This might happen instantly or after some patience. In general, early stopping is often used to prevent overfitting and automatically tunes the hyperparameter of training epochs.

Learning rate reduction Instead of stopping training completely when validation loss stops decreasing, the learning rate can be reduced by any factor x to keep optimizing the learned model without overfitting.

Local response normalization In neurobiology, the concept of *lateral inhibition* is well observed: An excited neuron subdues its neighbors. This contrast amplification increases the overall sensory perception. In visual processing as well as for the sensation of touch this promotes edge detection and alertness to changes rather than to constant stimuli.

Especially when working with the ReLU activation function, creating theoretically unbounded output, some kind of response normalization is advised. Therefore, Krizhevsky et al. (2012) proposed local response normalization (LRN) to dampen activations which are equally large in any local neighborhood. This creates an artificial inhibition and reinforces neurons with relatively large responses. However, this method is regarded outdated and is commonly replaced with batch normalization.

Batch normalization When the input distribution of an ANN changes, this is called *covariate shift*. When it happens on internal nodes of an ANN, its called *internal covariate shift*. A problem arises, because true data $P(Y|X)$ is represented by a model $P(Y|X, \theta)$ with θ being the parameters of the model, which are at best an approximation of $P(Y|X)$. This leads to a shift in training data distribution compared to the true (global) distribution. Normalizing input data as part of preprocessing (setting each features mean to 0 and variance to 1) can improve classification accuracy. However, after just one layer output feature maps may not be normalized anymore and this effect amplifies throughout an ANN. Feed–forward and backpropagation are unsuitable tools to prevent internal covariate shift. For shallow architectures resulting effects are minimal, but for truly deep networks this issue is advised to be addressed.

Ioffe and Szegedy (2015) came up with a specialized algorithm to cope with this problem. Extensively simplified speaking, batch normalization layers normalize featuremaps of a training batch between other (nonlinear) layers. For this matter, a new trainable scaling parameter β_n is introduced, while dimensionality n corresponds to the respective output size. This minimizes the effect of small weight changes in initial layers on later layers and therefore generally allows the usage of higher learning rates.

Dropouts Dropouts were initially proposed by Srivastava et al. (2014) and are usually applied to the fully connected layers in the end of a ConvNet. The dropout rate between 0 and 1 corresponds to the probability of every individual neuron in respective layer to be deactivated for a single training example. This reduces the general chance of overfitting and the formation of a *grandmother neuron*, representing a single class. The entire model performance is less dependent on specific neurons and thereby more robust. However, training time is increased.

L1 & L2 Regularization These methods penalize high weight values of an ANN, thus they are also described as *weight decay* (Krogh and Hertz, 1992). While L2 penalizes the square value of each weight, L1 uses the absolute value. L2 keeps weights low and L1 reduces them to zero if no strong gradient counteracts. While the introduced sparsity makes networks simplified, easier to interpret and may increase classification performance, L1 creates quite a number of *dead neurons* which is in many cases not desired.

Kernel constraints This method simply constraints the weights of any layer to be either in a specific range, e.g. force non–negativity, or to have a certain mean value. Similar to the L1 regularization, a forced mean weight of 0 will create a relatively large number of dead neurons.

2.3.6 Alternative to ConvNets

Another often used variant of ANNs for time series processing are recurrent nets with long short–term memory (LSTM). While backpropagation is based on the idea of a differentiable network, recurrent nets are theoretically infinitely deep. Initially introduced by Hochreiter and Schmidhuber (1997), LSTM solves the problem of dissolving error rates within deeper layers. In simplified terms, this is achieved through omission of activation functions and storing given input values over a specific (long or short) period of time. This method is successfully used for classifying, predicting and processing time series data and is widely spread in todays speech recognition software. Due to these characteristics, it has also been applied on EEG data (Längkvist et al., 2014). However, as this represents a fundamentally different approach in contrast to ConvNets, it is not considered in this work, but is worth mentioning as it represents a potential alternative in the search for subject independent features.

2.4 EEG Data Classification

In real world applications, the main challenge of EEG–based BCIs is to successfully identify target EPRs within noisy data streams. Due to their variable appearance and small amplitudes, it is not easy to differentiate them from the more often occurring standard waves, as no response averaging can be made. In its simplest form this represents an unbalanced, binary classification scenario.

When EEG data is recorded in experimental setups, markers are added to the raw EEG signals, allowing supervised machine learning algorithms to be trained on the respective data. For this purpose, the data stream is usually segmented into windows of a set length, each beginning at a previously marked position. Besides this partitioning, there are various techniques used to preprocess EEG data. Usually, dataset size reduction for faster processing with minimal information loss is desired. Commonly used methods include down sampling and bandpass filtering to the desired frequency range, standardizing windows for a better SNR, using fewer channels or creating new pseudo–channels, abstracting wave segments through simple features and normalizing these once again.

After preprocessing, there is a great amount of single or multiple classification algorithms that can be trained on specific datasets to later eventually correctly recognize unknown test data. Every algorithm has advantages and disadvantages, so there is no single *best option* and each use case should be carefully considered. Prior approaches in this field used for example Support Vector Machines (SVMs), Restricted Boltzmann Machines, regularized Linear Discriminant Analysis or combinations of various other standard machine learning algorithms (Lotte et al., 2007). Classification performance can be measured by a vast variety of metrics, also having individual focuses of importance.

With this multiplicity of options and tunable hyperparameters for a seemingly easy initial problem, comes a complicated comparability of procedures. Therefore, it is important to have a clearly defined baseline and metric to decide whether a newly developed or applied algorithm is better than another or not. Unfortunately, published papers do not always share their explicit processing chain or interim results, making comparison and replicability even harder.

2.4.1 pySPACE

Handling great amounts of EEG stream data can be challenging. Also, trying to effectively compare the steadily growing number of possible machine learning tools for preprocessing and classification may get complicated easily. Therefore, Krell et al. (2013) introduced and set up an automated and Python–based signal processing and classification environment (pySPACE). This software is specialized in processing multi–sensor windowed time series data, such as ERPs in EEGs. It is highly modular and contains a growing number of algorithms of which some are shown in Figure 12. When multiple CPUs or GPUs are available, parallelization is automated. Configuration is possible through files in the YAML format, making understanding of Python redundant for users. Different approaches can be defined as node–chains, optimized, compared and then saved as a standard reference flow. Next to widely used libraries, like for example the Scikit–learn tools, some handy extensions such as a cross–validation splitter, grid search and result visualization scripts are contained. Furthermore, self–developed or preexisting algorithms can easily be added, as done within this project. All in all, this framework is a strong asset to research groups performing experiments with EEG or similar time series data.

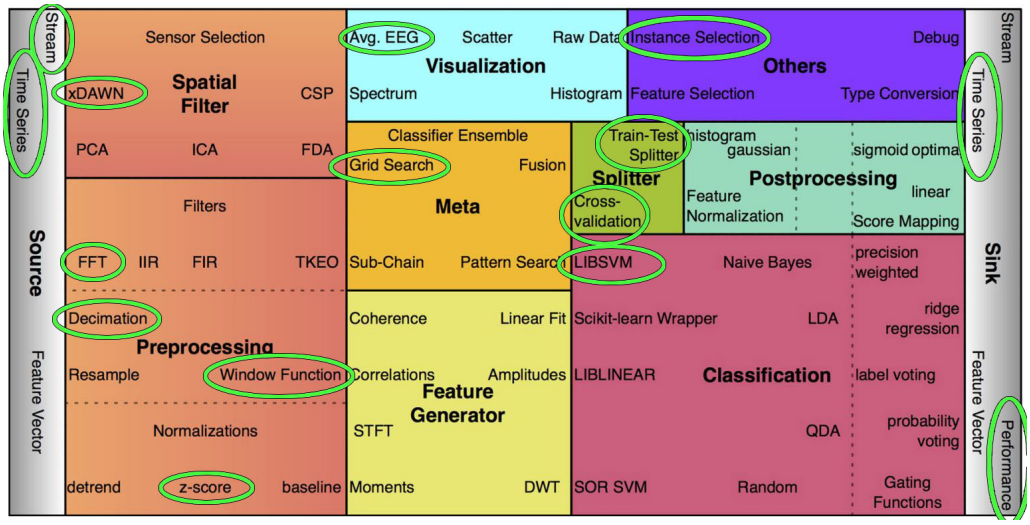

Figure 12: pySPACE overview showing some of the processing nodes of respective categories. Some of the used nodes have been highlighted in green. Graphic changed from Krell et al. (2013).

2.4.2 SVM Baseline

Kirchner et al. (2016a) developed a data processing chain for classification of P300, consisting of eight subsequent steps, as shown in Figure 13 A. The original continuous raw EEG stream is obtained with a $5\,kHz$ sampling rate on 64 channels. The data is processed by a DC removal filter to center the signals around zero. Prior experiments have shown that the major part of the P300 ERP is represented in the δ–band. Therefore, the sampling rate is decimated to $25\,Hz$ and a cutoff frequency of $4\,Hz$ is applied as an anti–alias filter. Continuous data is segmented into $1\,s$ windows. For spatial filtering the xDAWN filter (Rivet et al., 2009) is used to reduce dimensionality, remove artifacts and improve the SNR. Eight most relevant pseudo–channels remain for further processing. For feature generation, straight lines are fitted to $400\,ms$ long segments of every window with $120\,ms$ intervals (see Figure 13 B). Obtained features are normalized by setting their mean value to 0 and respective standard derivation to 1. Subsequently, a SVM is trained and used for binary classification. At last, a threshold optimization is performed to increase future classification accuracy. This procedure shall be the baseline for the new signal processing and classification chain proposed in this project.

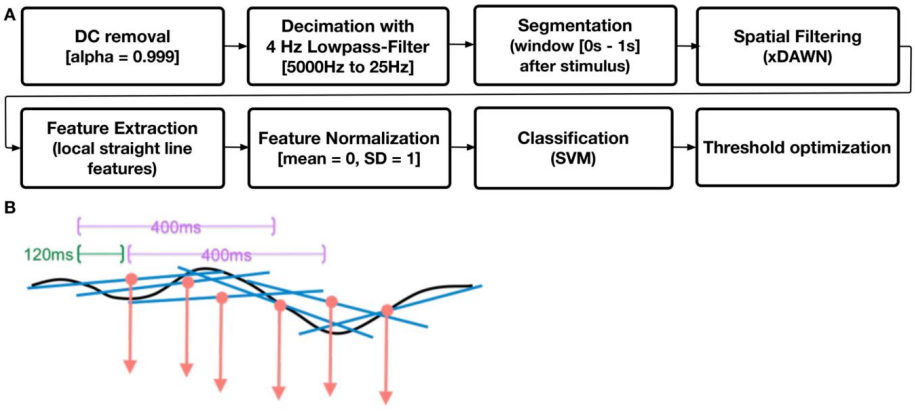

Figure 13: **A** P300 SVM baseline reference flow based on strong data reduction and SVM classification. **B** The entire information within the $1\,s$ window is broken down to six local straight line features, more specifically to respective slopes. Following classifier can retain information on vague curve development but has no knowledge of contained frequency spectrum. This is reportedly sufficient when ERP representation remains similar. Graphic from Kirchner et al. (2016a).

2.4.3 ConvNets applied to EEG Data

The potential advantages of deep learning applied to EEG data are well summarized by Schirrmeister et al. (2017). The research group considers various studies focused on movement related data and cognitive tasks as models for a novel DL architecture. Taking multiple examples from literature into examination, results show better classification performance for deeper networks. While not requiring hand crafted features, DL enables end–to–end learning on raw data. A great advantage is also, that this generally allows to analyze learned features and weights as well as activation (output) for a specific input. With the help of novel visualization techniques, mapping such ConvNet characteristics back to the surface of the skull, frequency bands or temporal windows, this grants to draw conclusions about fundamental neuronal activities. Such knowledge is hard or impossible to acquire when working on heavily reduced data. In the conclusion it is stated that "*ConvNets are not only a novel, promising tool in the EEG decoding toolbox, but combined with innovative visualization techniques, they may also open up new windows for EEG–based brain mapping*" (quoted from Schirrmeister et al. (2017)).

A major problem with BCI technology is subject transfer, as EEG data varies strongly between tested individuals and even the moment of data capturing. One would need a great amount of data from multiple subjects, performing a similar task (paradigm) to avoid overfitting the particular ANN. In order to use DL for BCIs with minimized training time, subject independent features are proposed. This has been matter of recent studies, for example discussed by Shamwell et al. (2016). Not only did they outperform a classical xDAWN–based classical processing chain, using visualization techniques, they are able to show where subject independent features are assumed to be.

2.4.4 ConvNet Baseline

Shamwell et al. (2016) use a ConvNet for EEG data classification to detect P300. The EEG data originates from 18 subjects performing a rapid serial visual presentation image triage task, presumably evoking a P3a. It is sampled at 1024 Hz on 256 scalp electrodes and reduced to 64 electrodes most closely according to the standard 10–10 system. The dataset is sub–sampled to 256 Hz and bandpass filtered between 0.5 and 50 Hz. Therefore, it contains information from all frequency bands except sub–δ. The data stream is segmented into windows of 1 s. The resulting single training examples have a dimension of 256 x 64 datapoints. Although this is a 2D matrix representation, neighboring datapoints of the two dimensions do not posses the same relationship as they do in classical images. Therefore, applied filters for both dimensions are first handled separately and only later combined to create *spatio–temporal features.*

Key structure of the algorithm is a multilayer artificial neural network. It consists of four convolutional, two pooling and three fully connected layers as shown in Table 2 and Figure 14. The first two convolutional layers (T1 & T2) and the intermediate pooling layer create 128 feature maps and only reduce the temporal output dimension from 256 to 11. This represents a temporal filter with automated feature generation. Subsequently the convolution of layer S1 and the following second pooling layer create 512 feature maps by reducing the spatial dimension from 64 datapoints to 1. Afterwards, the final temporal convolution processes the data to 256 feature maps of only one datapoint each. After every convolution, LRN is applied as a regularization method. In the end of the network two fully connected layers with 500 nodes each and a dropout rate of 50 % are implemented before the output layer with two neurons, one for each class. Both fully connected and all convolutional layers each have a ReLU activation function. The output layer has a softmax activation function to map network output to a classification class probability distribution and determines the binary cross–entropy loss value.

The order of placing the temporal filter before the spatial filter is chosen because the temporal dimension reportedly contains more relevant information about the presence of an ERP within EEG data windows. While for traditional

ConvNets for image processing filter size and stride are chosen equally great, in the applied filters size is greater than stride which creates strong overlapping. This has also been reported to be helpful reducing the error rate and overfitting by Krizhevsky et al. (2012).

In order to find subject independent features, the network is trained on half the dataset (9 subjects) and tested on the other half. The number of standard training examples is reduced to the number of targets to create class balance, which reportedly improves stochastic gradient decent optimization. Learning rate decay is used when validation loss flattens and early stopping when it stops decreasing to avoid overfitting.

Table 2: Baseline ANN model from Shamwell et al. (2016). After each convolutional layer, LRN is applied.

Layer	**Filter size**	**Stride**	**Output size**
Convolution T1	64 x 1	4 x 1	128 @ 49 x 64
Pooling	3 x 1	2 x 1	128 @ 24 x 64
Convolution T2	4 x 1	2 x 1	128 @ 11 x 64
Convolution S1	1 x 64	1 x 1	512 @ 11 x 1
Pooling	3 x 1	2 x 1	512 @ 5 x 1
Convolution T3	5 x 1	1 x 1	256 @ 1 x 1
Fully Connected	Dropout 50 %		500
Fully Connected	Dropout 50 %		500
Fully Connected			2

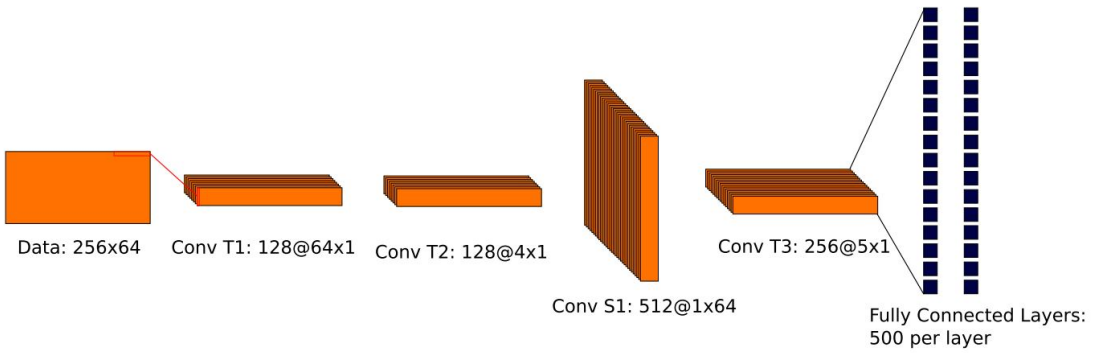

Figure 14: Baseline ANN model based on convolutional layers for spatial and temporal filtering. Graphic taken from Shamwell et al. (2016).

3 Methods

3.1 The Brio Dataset

The P300 data used for training and testing of the ANNs as well as comparison to the SVM baseline has been generated in 2009 and is provided by the German Research Center for Artificial Intelligence (DFKI). Six subjects each completed five experimental sets a day on two separate days in a *Virtual Labyrinth Oddball* setup as shown in Figure 15 (Kirchner, 2014). The five sets per day are combined into one *session* for this project.

In the shown Oddball task, subjects are instructed to actively respond only to the target stimuli and otherwise to refrain from responding, evoking a P3b. However, subjects are also performing a primary task (dual task scenario), which induces additional neuronal activation measured by the EEG. Kirchner et al. (2016a) filter only a band between 0 and 4 Hz in order to separate the P300 as good as possible. However, by doing so, even relevant information for ERP recognition in higher frequency bands are ignored.The original ratio of task–relevant target to task–irrelevant standard stimuli is approximately $1:6$. Because subjects sometimes miss targets, this ratio becomes further unbalanced. The exact numbers of usable examples for each session are listed in Table 4.

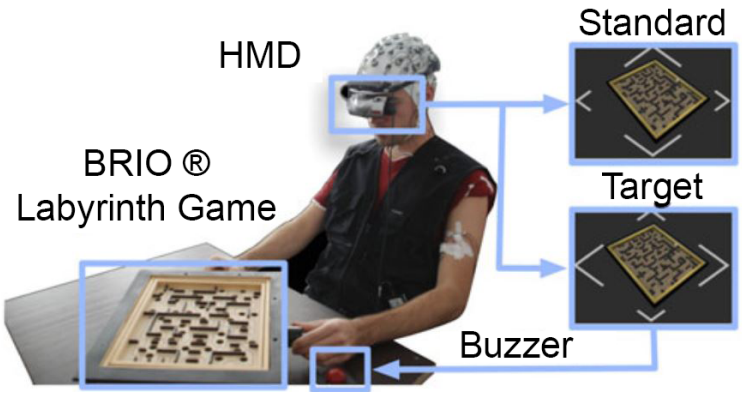

Figure 15: While working on a main task (BRIO Labyrinth Game) the subjects are shown visual stimuli over a head mounted display (HMD). Subjects are instructed to use the buzzer only when displayed corners on the left and right side appeared bigger than on the upper and lower corner (target stimulus). EEG data is recorded on a 64 channel actiCAP and markers added for the respectively shown stimulus. Graphic changed from Woehrle et al. (2015).

3.2 EEG Data Preparation

Due to the ideal case of a BCI performing classification in real-time, preprocessing should be optimized, taking as little time as possible but enabling fast classification. While raw EEG can be used for DL, a common and fast approach to process EEG data is the bandpass filter to increase the SNR and classification performance. Furthermore, the sampling rate can be varied to minimize the data size per example and further processing time. Both characteristics of EEG data, sampling rate and contained frequency bands, are somewhat dependent on another as described in subsection 2.2.

Different variations of sampling rate and bandpass filter are subject of investigation in this study. This is examined to determine if higher frequency bands contain valuable information to increase ERP classification accuracy, which might also be helpful to create subject independent features. Schirrmeister et al. (2017) list various suitable ConvNet based studies to identify ERPs with low cutoff frequencies reaching from $0 - 8\,Hz$ and high cutoff from $15 - 200\,Hz$.

First, two channels are removed from the data, as they do not contain EEG information. Before dimensionality reduction, time series data is segmented in windows of $1\,s$ length each, starting with a marker of either standard or target stimulus. Afterwards, each window is standardized, setting respective mean value to 0 and the standard deviation to 1. Because of the functional principles of DL, no further feature generation or manual selection is necessary. Table 3 shows all nine different cases tested in this project. While case 1 represents raw input data, case 6 is equivalent to the baseline from Shamwell et al. (2016) and case 9 refers to Kirchner et al. (2016a). Each case fulfills the guideline proposed by Luck (2014) which states, that the sampling rate should at least be three times as high as the upper cutoff filter frequency to minimize induced artifacts.

Figures 16, 17 and 18 show standard and target average ERPs from subject 3 session 2 electrode Pz for cases 1, 6 and 9 respectively. In given examples the difference between target and standard are well visible. The courses of curves are similar for the first $300\,ms$ until the P300 peaks clearly exceed the standard waves. Compared to the description of Polich (2007) however, P300 appears considerably late and flat (Kirchner, 2014).

Table 3: All different preprocessing cases. While cases 3, 7 and 9 include only sub–δ & δ band, other cases also include higher frequency ranges.

Case #	Sampling rate [Hz]	Bandpass filter [Hz]
1	1000	none
2	1000	0.5 – 50
3	1000	0 – 4
4	250	none
5	250	0 – 50
6	250	0.5 – 50
7	250	0 – 4
8	25	none
9	25	0 – 4

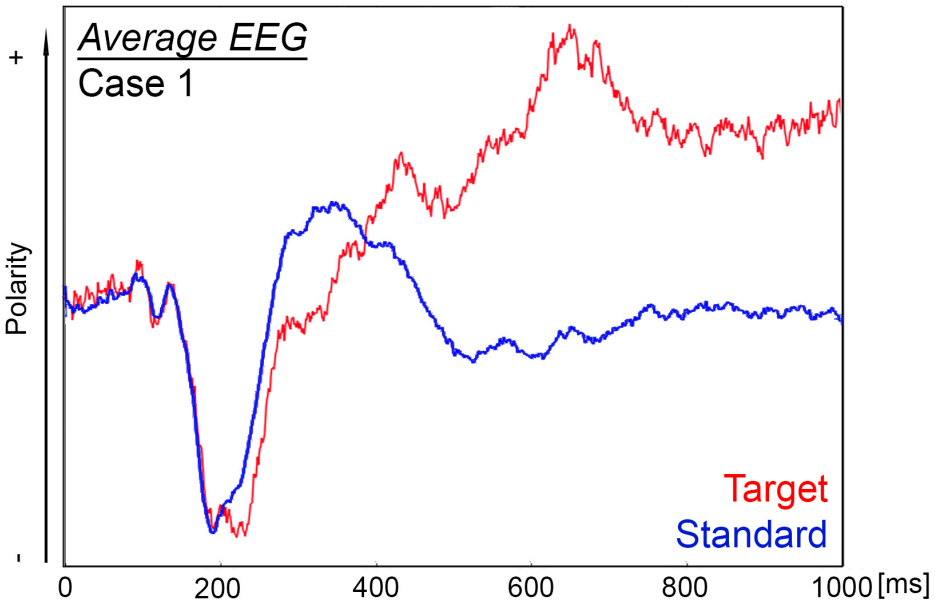

Figure 16: Average graphs for preprocessing case 1 (raw data) for both standard (blue) and target (red) from subject 3 session 2 electrode Pz. Sampling frequency: 1000 *Hz*. Bandpass filter: None. Data is standardized and curves aligned to display resemblance, scaling remains accurate.

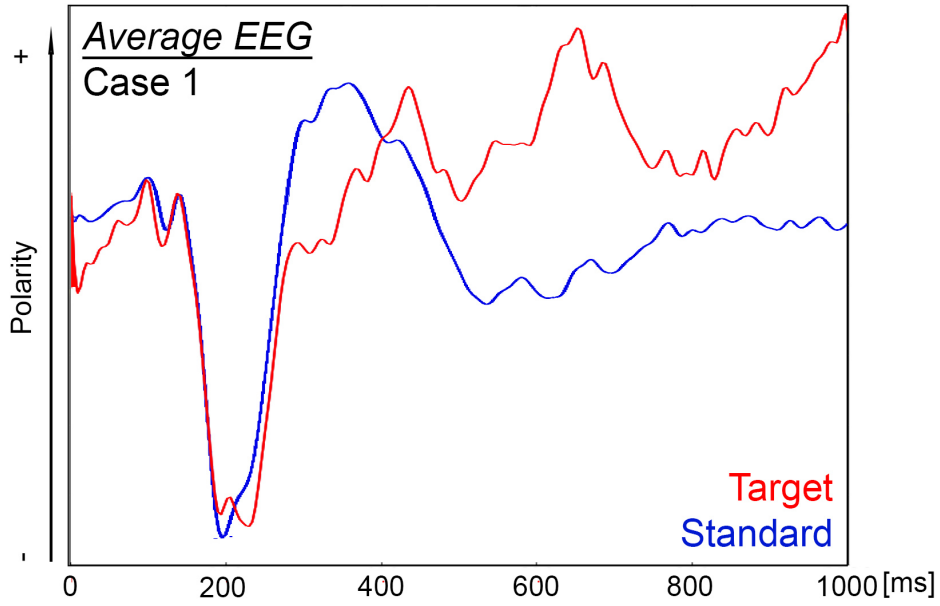

Figure 17: Average graphs for preprocessing case 6 (baseline Shamwell et al. (2016)) for both standard (blue) and target (red) from subject 3 session 2 electrode Pz. Sampling frequency: 250 *Hz*. Bandpass filter: 0.5 – 50 *Hz*. Data is standardized and curves aligned to display resemblance, scaling remains accurate.

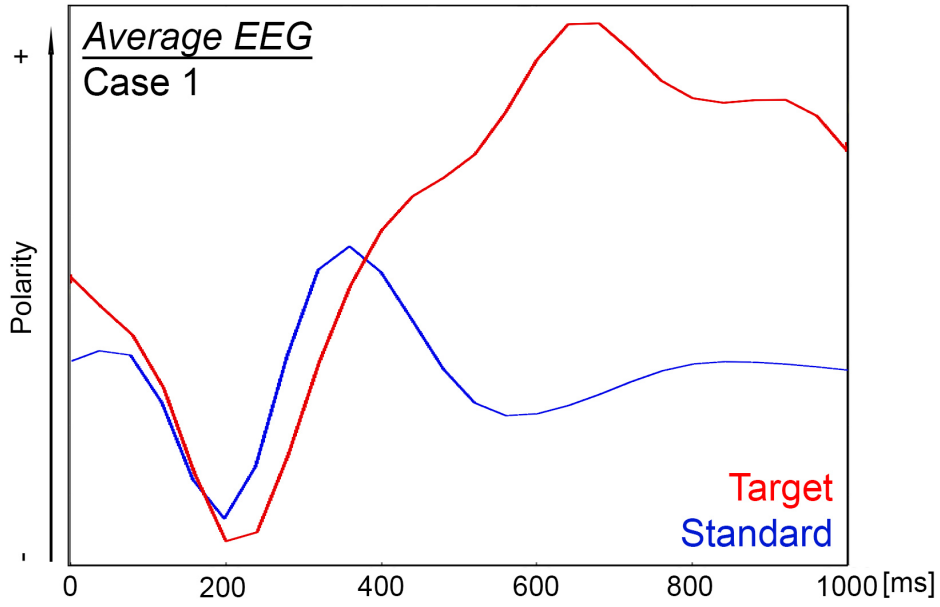

Figure 18: Average graphs for preprocessing case 9 (baseline Kirchner et al. (2016a)) for both standard (blue) and target (red) from subject 3 session 2 electrode Pz. Sampling frequency: 25 *Hz*. Bandpass filter: 0–4 *Hz*. Data is standardized and curves aligned to display resemblance, scaling remains accurate.

3.3 Data Augmentation

Size of the training datasets is crucial to avoid both, *over–* and *underfitting*. A potential problem for DL applied to EEG data is that the dataset is too small compared to the number of tunable model parameters. In general, it can be said that the bigger a dataset, the more abstraction of relevant features can be made, decreasing the risk of overfitting to specific or unique examples. Also, unbalanced class distribution can influence network behavior during training. An overrepresented class will be learned more intensively and will therefore be classified correctly more often than underrepresented classes.

There are some common ways to counter the described issue. The error–function can be expanded with factors (weights) to put emphasis on certain classes during training (cost–sensitive training) (Khan et al., 2015). However, it can be quite challenging to find suitable weights, especially when multiple classes of varying size are involved. Instead, class distribution can be made even by withholding random training examples from the overrepresented classes, until every class has the same size as the smallest class. Unfortunately, this method does not make use of all available training data and contained information.

In order to maximize usable examples, Krell and Kim (2017) propose data augmentation through spatial and temporal distortions of EEG signals. This can either increase the entire dataset or be applied only to underrepresented classes in order to improve the respective class-ratio. Temporal augmentation can be achieved through a small positive or negative window offset on the same data. Spatial distortions occur naturally in EEG data, as electrodes slightly shift position during and between sessions. Rotational augmentation can be achieved through retrospectively induced artificial electrode shift along the main rotational axes of the head, shown in Figure 19 (Krell and Kim, 2017). Therefore, normalized input data is interpolated, based on a radial basis function from SciPy. Later, new electrode positions are determined by multiplying interpolated input data with the standard rotation matrix of respective axis. This is done for each point of time and associated amplitude.

Data augmentation has enhanced network performance in multiple cases (Krell and Kim, 2017; Krizhevsky et al., 2012). Thus, this project uses rotational

augmentation of target EEG signals and afterwards standard examples are reduced to match the number of targets by pySPACE on a random basis. EEG data is augmented by the respective pySPACE node, by rotation around the y– and z–axis for -22, 0 & 22 degree. These hyperparameters were chosen, relying on the results from Krell and Kim (2017). Table 4 shows the impact of data augmentation on the number of usable examples. In this case, augmentation increased used training data by factor 5.

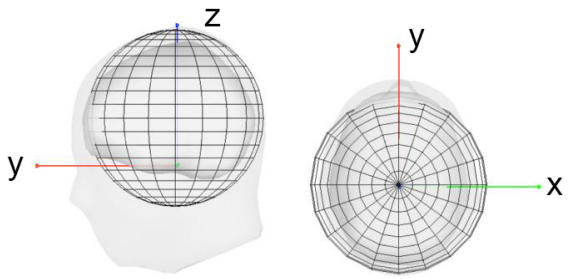

Figure 19: Axes for rotational data augmentation. In this project, augmented targets only are generated, originating from rotation around y– & z–axes by 22° in positive and negative direction. Graphic changed from Krell and Kim (2017).

Table 4: Class–size of standards and targets before and after data augmentation used for training. To balance classes for training, used examples equal twice the size of the underrepresented class (targets). Due to great difference in class distribution, only targets needed to be augmented to increase the total amount of used examples by factor 5.

	Before Augmentation				After Augmentation		
Dataset	**Std.**	**Tar.**	**Total**	**Used**	**Tar.**	**Total**	**Used**
Subj. 1 Sess. 1	3938	382	4320	764 ≙ 17.7%	1910	5848	3820 ≙ 65.3%
Subj. 1 Sess. 2	4021	361	4382	722 ≙ 16.5%	1805	5826	3610 ≙ 62.0%
Subj. 2 Sess. 1	3842	460	4302	920 ≙ 21.4%	2300	6142	4600 ≙ 74.9%
Subj. 2 Sess. 2	3763	507	4270	1014 ≙ 23.7%	2535	6298	5070 ≙ 80.5%
Subj. 3 Sess. 1	3728	501	4229	1002 ≙ 23.7%	2505	6233	5010 ≙ 80.4%
Subj. 3 Sess. 2	3700	536	4236	1072 ≙ 25.3%	2680	6380	5360 ≙ 84.0%
Subj. 4 Sess. 1	3650	545	4195	1090 ≙ 26.0%	2725	6375	5450 ≙ 85.5%
Subj. 4 Sess. 2	3594	567	4161	1134 ≙ 27.3%	2835	6429	5670 ≙ 88.2%
Subj. 5 Sess. 1	3851	496	4347	992 ≙ 22.8%	2480	6331	4960 ≙ 78.3%
Subj. 5 Sess. 2	3738	510	4248	1020 ≙ 24.0%	2550	6288	5100 ≙ 81.1%
Subj. 6 Sess. 1	3777	479	4256	958 ≙ 22.5%	2395	6172	4790 ≙ 77.6%
Subj. 6 Sess. 2	3684	552	4236	1104 ≙ 26.0%	2760	6444	5520 ≙ 85.7%

3.4 Network Architecture

While the main structure of tested ANNs is inherited from Shamwell et al. (2016), described in Section 2.4.4, some changes are applied. Batch normalization layers are tested in this project to replace LRN, which is not used. Also, the number of dense layers and included neurons are parameterized. The spatial filter dimension is set variable to respective input, 62 channels in this project. Furthermore, three different sizes of the initial temporal filter are tested. Finally, the stride and dimension of the initial temporal filter and subsequent pooling filter are adjusted. As this project considers three different sampling rates, these four parameters are subject to scaling. This process is automated by introducing the respective factor as a variable with dependency on the temporal input dimension. Functionality of the developed network is thereby theoretically unaffected by the input sampling rate or channel quantity. The number of trainable parameters varies, so saved models can only be applied to input of the same shape that they have been trained on. The final number of trainable parameters for each case is dependent on introduced hyperparameters in this Section and is therefore only determined after optimization (see Section 4.1).

3.5 Training & Testing

In this project, Keras v. 2.0.5 and Tensorflow v. 1.1.0 are used to implement tested functions in the DL architecture. All experiments in terms of training and testing the created DL networks are performed on a GPU–cluster, provided by the DFKI. Employed key hardware components are six *nVidia TITAN X* GPUs. In all cases, 15 % of each training set is separated as a validation set by Keras. During training, loss is calculated as *binary crossentropy* and optimization is handled by the *Adam* algorithm, introduced by Kingma and Ba (2014).

For a *session transfer* scenario the sets of a single day are consolidated to a session, leaving twelve individual datasets for training. After training, data of the same subject recorded on another day was used as the test set.

In a *subject transfer* scenario both sessions of all subjects but one are combined for training, generating six datasets. To check if subject independent features are learned, testing is carried out on both sessions of the remaining subject.

3.6 Hyperparameter Optimization

Along the hyperparameters introduced in Section 3.4, key hyperparameters for learning rate, batch size and regularization methods are optimized (see Section 2.3.5). Table 5 lists all twelve incurred variables and tested values. Due to limited time and computation power, the number of considered values is relatively small for each parameter. Also, due to the high number of possible combinations emerging from this twelve–dimensional optimization problem, hyperparameters were optimized in separate sub–sets. Therefore the assumption is made, that found local optima remain valid when all hyperparameters are considered. Optimization is carried out in a session transfer scenario with preprocessing case 6. Results of this Section are evaluated based on the validation accuracy.

Table 5: Tested values for hyperparameters. Considering only few options for every parameter and some constraints results in a great number of possible combinations. Therefore, optimization was carried out in sub–sets, separated by lines.

Hyperparameter	**Tested values**					
Temporal filter size	50	35	20			
Dense layer number	1	2	3	4	5	6
Dense layer size	250	500	750	1000	1250	
Learning rate	1e-02	1e-03	1e-04	1e-05	1e-06	
Early Stop patience	5	10	20	30	50	
Learning rate reduction	0.1	0.2	0.5	0.8	none	
Learning rate patience	4	6	8			
Batch norm. momentum	0.97	0.99	none			
Batch size	32	64	128	256		
Dropout	0.0	0.5				
L2 regularization	0.0	0.0001	0.001	0.01	0.1	
Kernel constraints	0	0.01	0.05	0.1	0.5	1
Possible total combinations:					7,722,000	

3.7 Evaluation

Sammut and Webb (2011) provides a broad spectrum of evaluation methods for machine learning. Classification results can be displayed in a confusion matrix as shown in Table 6.

Table 6: Confusion matrix for binary classification performance analysis.

	Condition positive	**Condition negative**
Prediction positive	True positives (TP)	False positives (FP)
Prediction negative	False negatives (FN)	True negatives (TN)

Based on these four absolute numbers, various metrics can be calculated to evaluate classification performance. However, not all metrics are suitable for each problem. First the true positive rate (TPR) and true negative rate (TNR) can be calculated.

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$TNR = \frac{TN}{TN + FP} \tag{2}$$

In this case of two unbalanced classes, balanced accuracy (BACC) is proposed by Straube and Krell (2014), due to its insensitivity to changes in class distribution.

$$BACC = \frac{TPR + TNR}{2} \tag{3}$$

Plotting the TPR over FPR ($= 1 - TNR$) in dependency of a classifier parameter such as the decision boundary results in a receiver operating characteristic (ROC) curve. An ideal classifier would reach a TPR close to 1 while the FPR remains close to 0. This generates a steep incline of the ROC curve, thus increasing the area under the curve (AUC). Since threshold optimization is not part of this project, as it can easily be automated, ROC curves are not included in the results. AUC however, is a commonly used metric to compare classifier performance and is also insensitive to class distribution. Figure 20 shows an example of a ROC curve to clarify this quite abstract metric. For both metrics, BACC and AUC a value of 0.5 represents a guessing classifier and 1 means perfect classification.

To check results of different preprocessing cases and classifiers, an analysis of variance (ANOVA) test for repeated measures is applied to the BACC of achieved results of session transfer (Girden, 1992). For subject transfer result analysis, a Wilcoxen sign-rank test is performed. Because the statistic design, hypothesis definition and results interpretation are conducted with the help of Dr. Su–Kyoung Kim, results are shown in the appendix of this thesis.

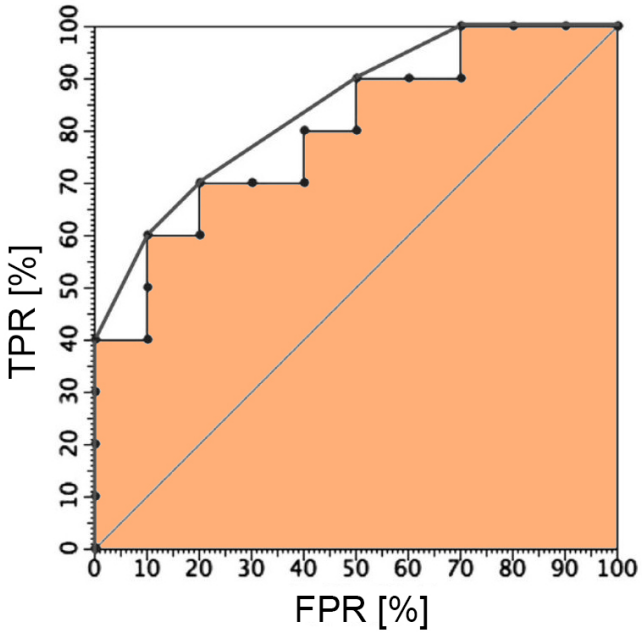

Figure 20: ROC curve example: Grey line represents a guessing classifier with an AUC (orange) of 0.5. The convex line intersects those points on the ROC curve which are optimal under some specific class distribution. The slope of each segment gives the class ratio under which the two end points of the segment yield equal accuracy. All points under the convex hull are nonoptimal. Figure and caption partially changed from Sammut and Webb (2011).

4 Results & Discussion

4.1 Hyperparameters

The following paragraphs show some of the emerged results for hyperparameter optimization. Due to the great number of tested variations and combinations not all findings can be displayed. Medians or surrounding boxplots often suggest only minor differences in balanced validation accuracy. In some cases a visualization over all sessions is unclear, so even improvements in median or mean value relatively small to overall variance are regarded. Furthermore, recorded balanced validation accuracies vary depending on observed parameter. These absolute values cannot be compared as the network may have been adjusted according to previous findings.

Although results for individual regularization methods are promising, empirical testing has shown that the previously stated assumption concerning combination of local optima is wrong. As optimization is performed relying on the balanced validation accuracy, considered regularizations sometimes proved to be obstructive for classification on the test set. This is believed to be due to the fact, that the validation set is part of the same data (day of recording) the training set emerges from. The data is too similar and therefore facilitates undetected overfitting to trained sessions characteristics. However, detected overfitting for session transfer is minimal, so further regularization hyperparameters are hard to tune and respective methods appear obsolete. For this reason, shown results may contain a value for apparent optimum as well as a value chosen for further processing.

Temporal filter size Results for comparison of different initial temporal filter sizes are shown in Figure 21 and the optimal tested value is 50, retained throughout further experiments. Although this parameter is subject to scaling, the initial temporal filter covers a 200 ms window on input data in all cases. Also overlap is considerably great with 180 ms or 160 ms for 250 & 1000 Hz or 25 Hz sampling rate respectively.

Dense layer quantity & size Comparison of different quantity and size of dense layers at the end of the network is shown in Figure 22. Compared to Shamwell et al. (2016), the number of neurons per layer was reduced from 500 to 250. This seems reasonable, hence this project uses smaller datasets in general. Although three layers seem to create minor improvements, two layers were used throughout further experiments.

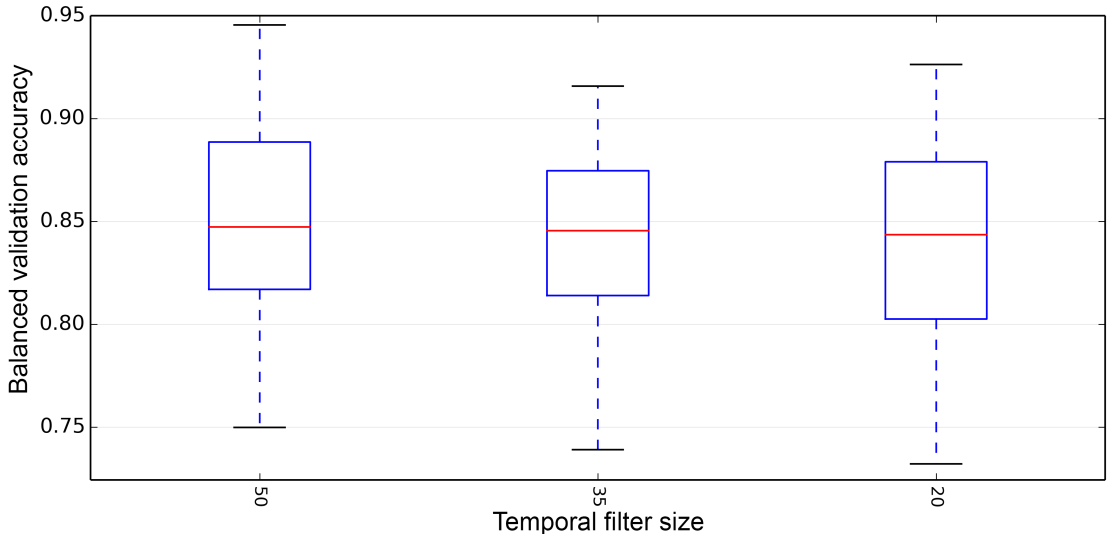

Figure 21: Initial temporal filter size of 50 showed best balanced validation accuracy. However, this parameter is subject to scaling for different input dimensions.

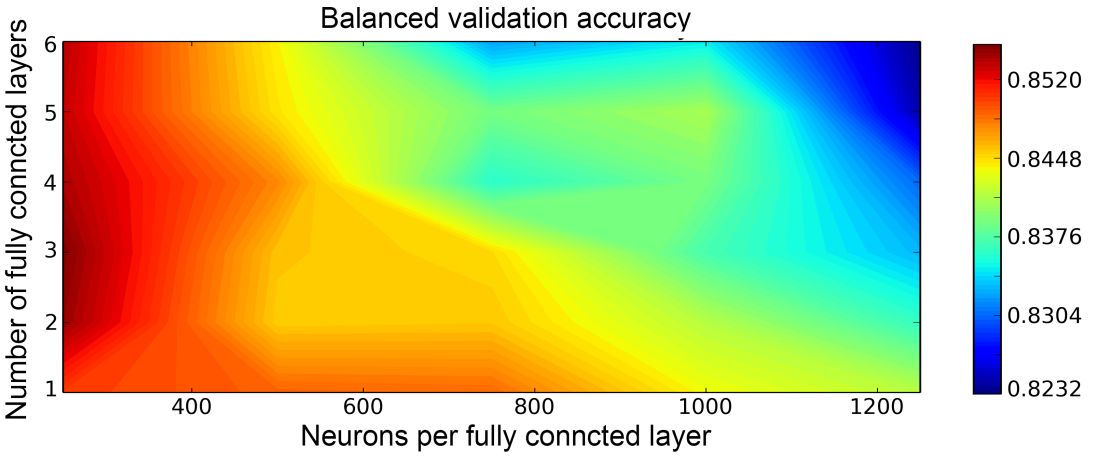

Figure 22: Dense layer quantity & size comparison. For this comparably small dataset, less neurons per layer seem preferable. Chosen values: 2 layers with 250 neurons each.

Learning rate Comparison of considered learning rates shows that 1e-04 is best suitable for this use case, as shown in Figure 25. Empirical testing with higher stop patience could improve validation accuracy for smaller learning rates, but was not able to exceed results emerging from a learning rate of 1e-04. To prevent excessive overfitting, training is stopped early when validation loss stopped improving for 30 epochs.

Figure 23 shows curves for training and validation loss over epochs. While training loss reached approximately 0 after 19 epochs, the flat course of validation loss after this point indicates that no significant overfitting to training data is acquired. To verify this, models are saved at the point of training loss becoming approx. 0 and after early stopping is triggered by Keras. Tests on the training set show, that the latter model achieved better balanced accuracies. Figure 24 shows that standards and targets are learned equally well during training.

During the training period of potential overfitting, learning rate decay is tested to improve validation accuracy and loss. For a learning rate of 1e-04, no decay is most suitable as shown in Figure 26. However, for smaller learning rates this method seems useful, as shown in Figure 27. Shamwell et al. (2016) also report learning rate decay as helpful, but they applied training over a distinctly larger number of epochs, hence using a comparably small learning rate to begin with. Throughout this project, learning rate decay is not further considered.

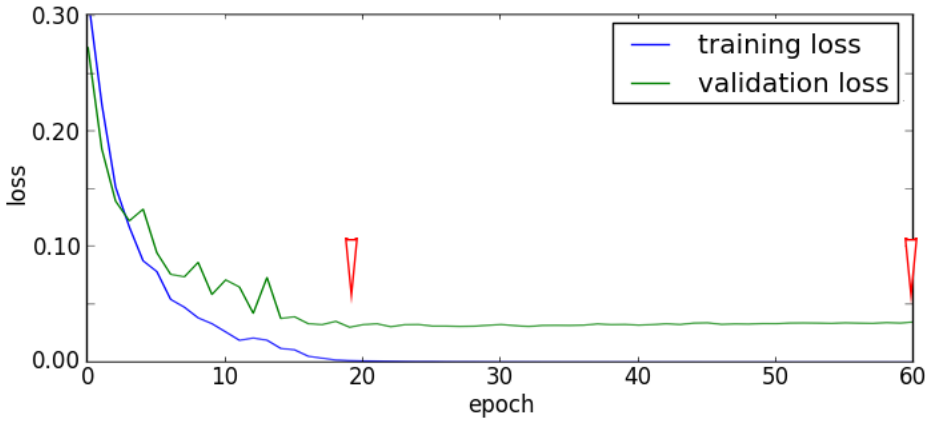

Figure 23: Example for training and validation loss for session transfer. The flat course of validation loss after training loss reached approx. 0 is a clear sign for minimal overfitting. This is verified by comparing classification results on test data of the two models saved at red indicators.

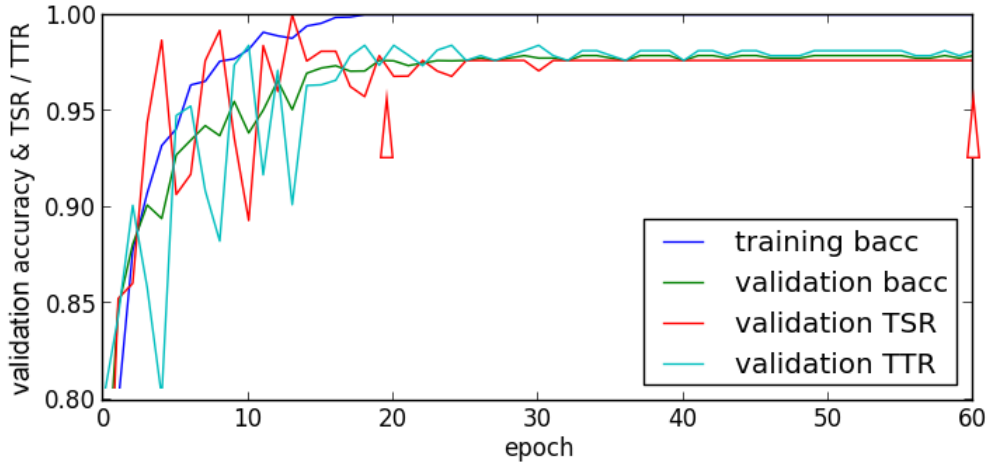

Figure 24: Classification accuracy on training and validation set as well as true target rate (*TTR*≙*TPR*) and true standard rate (*TSR*≙*TNR*) on validation set. The similar course and final value of TTR and TSR implies that both classes have been learned equally well. Example graphic from subject 3.

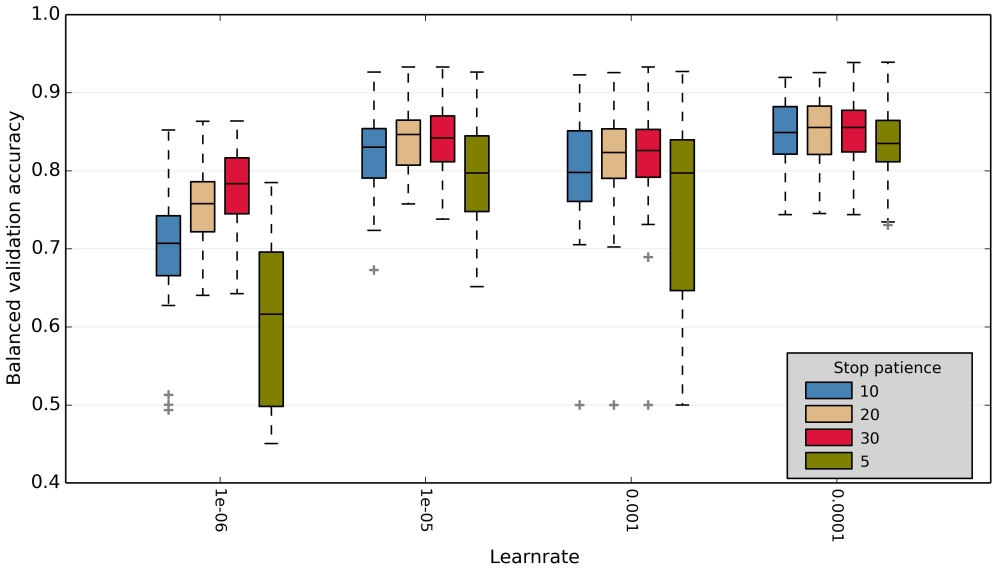

Figure 25: Validation accuracy for different learning rates & early stop patience values. Chosen: Learning rate: 1e-04 & stop patience: 30.

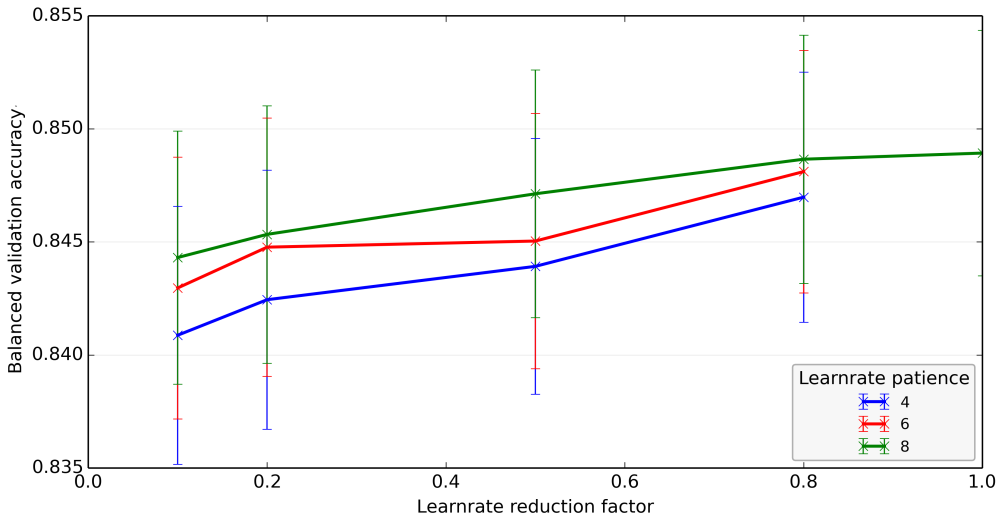

Figure 26: Results for learning rate reduction & reduction patience. Factor of 1.0 equals no learning rate reduction besides induced through Adam (Kingma and Ba, 2014), which generated best validation accuracy.

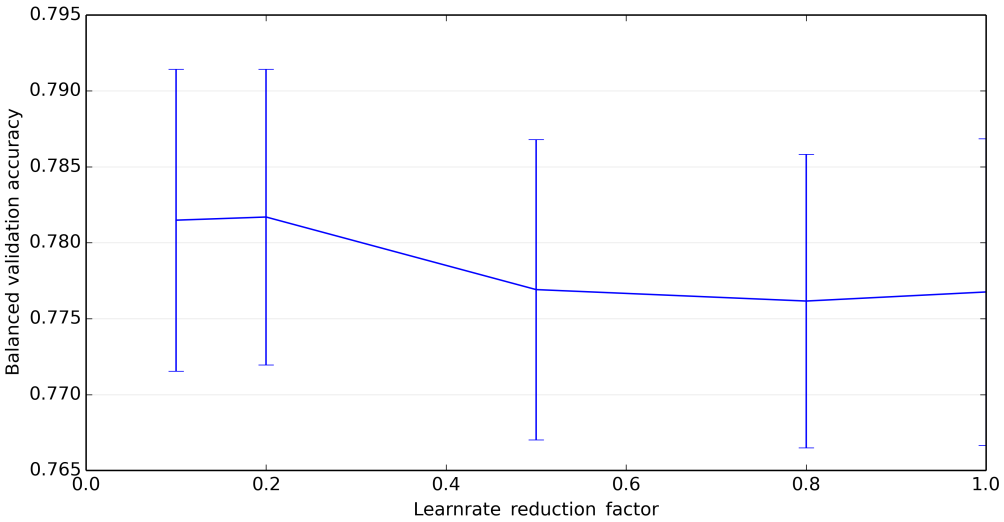

Figure 27: Learning rate reduction for smaller learning rate of 1e-06. Even though relatively short stop patience of 30 is active, learning rate reduction of 0.2 and reduction patience of 8 was able to increase performance by 0.5 %, but could not surpass performance from a learning rate of 1e-04. Effect might have been even more clear with longer patience. Great standard deviation comes from subject differences.

Batch size & normalization Figure 28 shows the validation accuracies for networks with and without batch normalization and variable batch size. The results imply that batch normalization was able to improve performance, especially for larger batch sizes. This cannot be verified for classification on respective test sets. Also, batch normalization approximately doubled needed training time. Therefore no batch normalization layers are included in the network architecture for further experiments and a small batch size of 32 examples is chosen.

Because the network consists of only few layers compared to classical deep nets for large scale image processing and data is standardized during preprocessing, batch normalization might promote overfitting. As discussed earlier, validation and training set are quite similar, hence validation accuracy increases while test performance suffers from using this method.

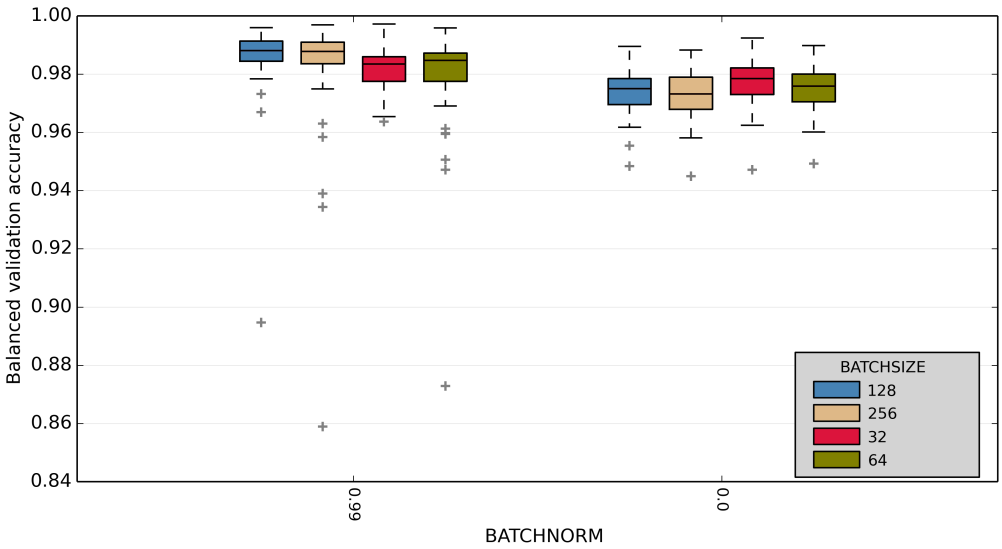

Figure 28: Validation set results comparing different batch sizes & normalization. Best results are found using the regularization method and bigger batches. However, this could not be transfered to results on the test sets. Therefore, no batch normalization and a batch size of 32 is chosen for further proceeding.

Dropout rate A dropout rate of 50 % as well as no dropout are options considered in this project. Validation results, shown in Figure 29, are insufficient to clarify whether this regularization method is helpful or not. BACC on the test set did not improve consistently over all session or subject transfer cases. Therefore, dropout is not further used throughout the project.

(Shamwell et al., 2016) use a dropout rate of 50 % for both fully connected ReLU layers. However, they use a greater number of subjects for training and 500 nodes per layer instead of 250.

L2 regularization Out of all tested values for the L2 penalization weight, 1e-03 is the only option with somehow comparable validation accuracy scores to results achieved in the absence of this regularization method. Figure 30 shows the validation accuracy for both cases. For clarification, this weight decay was also evaluated on a test set. Because no relevant improvement is induced through this method, it is not further considered in this project.

In order to better understand the effect of L2 on this use case, a more detailed weight analysis is needed. However, this is not part of this project but is a relevant factor for a possible following study to examine created features and their impact on decision making in detail.

Kernel constraints Instead of weight decay, this regularization method forces all weights to have a certain mean value. On the validation set results for a kernel constraint of 0.05 are promising. However, BACC on the test declined when the constraint is applied. Therefore, kernel constraint is not further considered in this project. It might also be that weights have a mean around 0.05 even without the regularization.

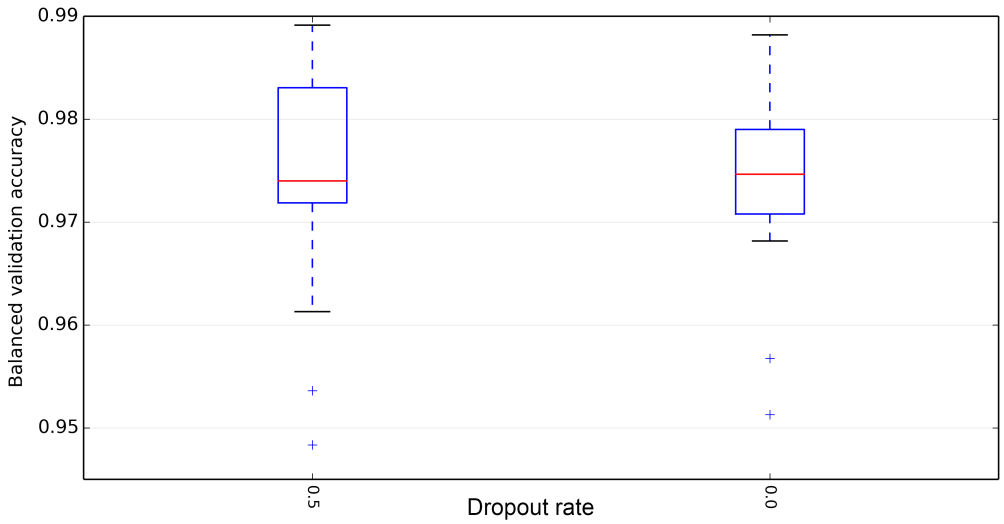

Figure 29: The validation results for a dropout rate of 50 % show no relevant improvement compared to no dropout.

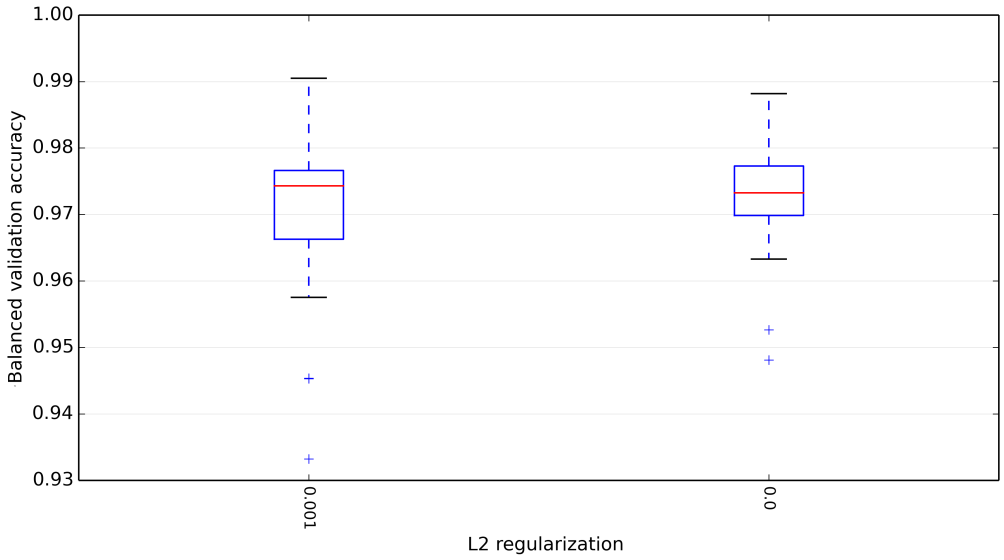

Figure 30: The validation results for L2 regularization show no relevant improvement compared to no regularization.

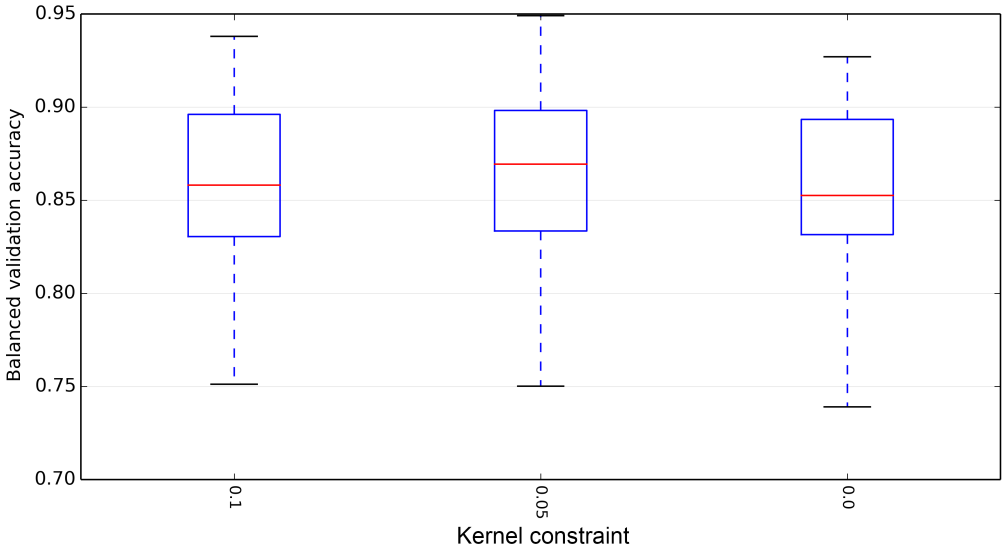

Figure 31: The validation results for a kernel constraint forcing the mean weights around 0.05 seem promising. However BACC on the test declined when the constraint is applied.

Trainable Parameters As explained in Section 3.4, stride and filter size of the initial two layers is subject to scaling. After all hyperparameters are set, the number of trainable parameters for the ConvNet is only depending on the input dimension of training data. Table 7 shows the resulting number of automatically adjusted weights for all considered sampling rates. Compared to classical ConvNets for large scale image classification, such as the network from Krizhevsky et al. (2012) (approx. 61 M parameters in 16 layers), the used architecture in this project is shallow and the maximum number of 5,105,043 trainable parameters is relatively small. However, compared to other ConvNets applied to EEG data listed by Schirrmeister et al. (2017) this network has a mediocre size.

Table 7: The number of trainable ConvNet parameters depending on the data input dimension.

Sampling rate [Hz]	Trainable ConvNet parameters
1000	4,938,003
250	4,918,803
25	5,105,043

4.2 Session Transfer

Session transfer results are verified by 10 runs for each case. Figure 32 shows the BACC for all nine cases, with and without augmented data. Data augmentation is able to increase performance in all cases. However, as the amount of training data is increased, needed learning time rises by approximately the same factor, as shown in Figure 33. In general, classification works well, regardless of considered preprocessing with mean BACCs ranging from 85.10 to 86.59. The SVM baseline achieved a mean BACC of 80.52 on the same data, but witout augmentation. Comparing results averaged over all sessions does not clarify which case is considered optimal. Therefore, needed training time, AUC and TPR are used to differentiate better.

Figure 33 shows that processing data with a 1000 Hz sampling rate results in longer training than for 250 Hz. Reducing the temporal dimension to 25 Hz does not speed up training any further. The SVM shows a mean training time of $198.86 \pm 28.82\,s$ without data augmentation, but because training is carried out on CPUs not GPUs, values are not easily comparable. Data dimension showed no effect on the average classification time whatsoever.

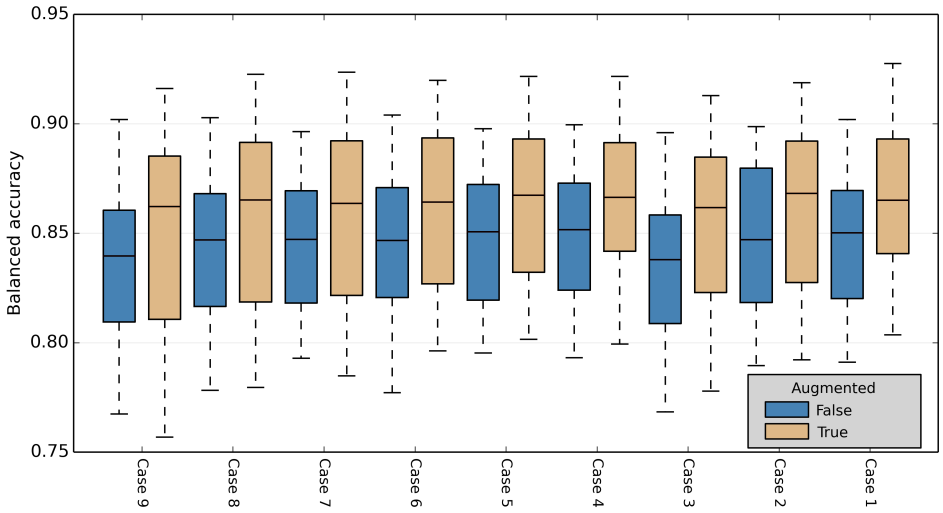

Figure 32: Total balanced accuracies for all cases with augmented (yellow) and non-augmented (blue) data. Using augmented training data improves performance. The ANN is able to classify test EEG data comparably well, independently of prior processing.

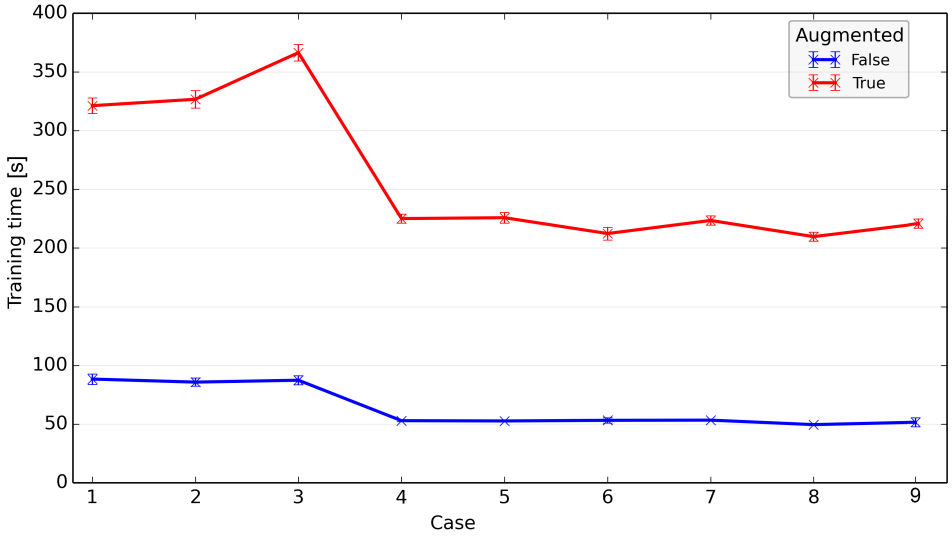

Figure 33: Needed training time increases for augmented data at approximately the same factor (5) that data is increased by. Average times do not include augmentation process itself. Also, training data with a sampling rate of 1000 *Hz* slows down training as compared to 250 or 25 *Hz*.

Figure 34 shows that different preprocessing affects the AUC of the classification algorithm and it is easier to distinguish an optimal case than for comparison of BACC in Figure 32. Case 4 appears to be most suitable for DL with highest meadian, mean, absolute low and high in measured values. A visible trend is, that for decreasing contained frequencies within data of any sampling rate, performance declines. Also, classification results of test data with a 1000 *Hz* sampling rate are worse than for 25 *Hz*. Best results are obtained with a temporal dimension of 250 *Hz* as proposed by Shamwell et al. (2016). However, applying a bandpass filter to the EEG data seems to dampen performance, although the frequency range above 50 *Hz*, not contained in cases 2 and 5 compared to 1 and 4 respectively are not considered to include valuable ERP information by Luck (2014). This decrease may be explained through artifacts induced by the filter, defiling remaining information. While in case 5 EEG data contains the sub-δ band, this is filtered out in case 6, resulting in a difference in performance. Figure 35 shows the same results for cases 1, 4, 6 and 9 split for every tested session. The variance for a specific session is much smaller than over all sessions, but differences in results emerging from different subjects explains this aspect. Case 4 proves to be best suitable for almost every tested scenario.

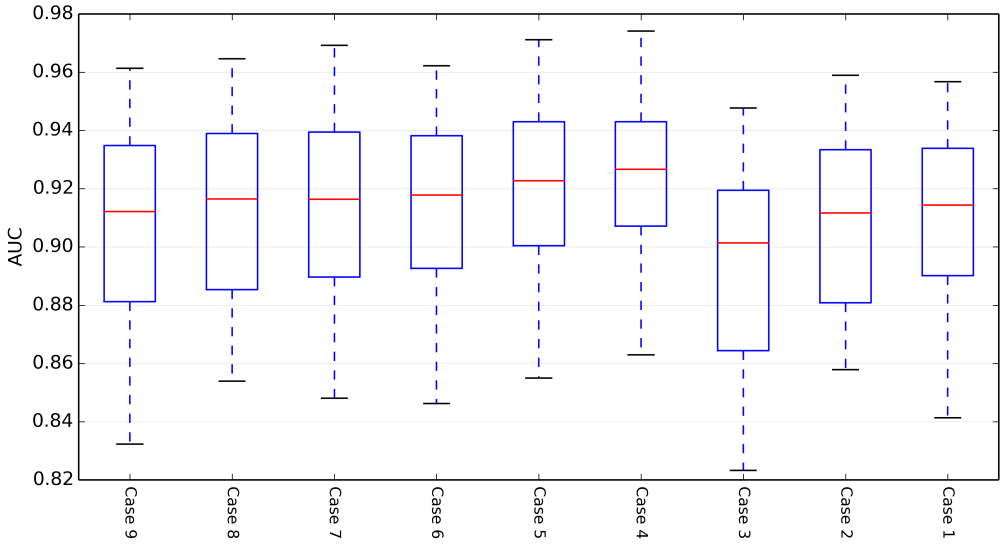

Figure 34: Case comparison & classification performance via AUC. Median values (red lines) are all greater than 0.9 and vary only within 2 %. In general, decreasing included frequency bands affects classification in a negative matter. Best results are obtained for case 4 and lowest for case 3.

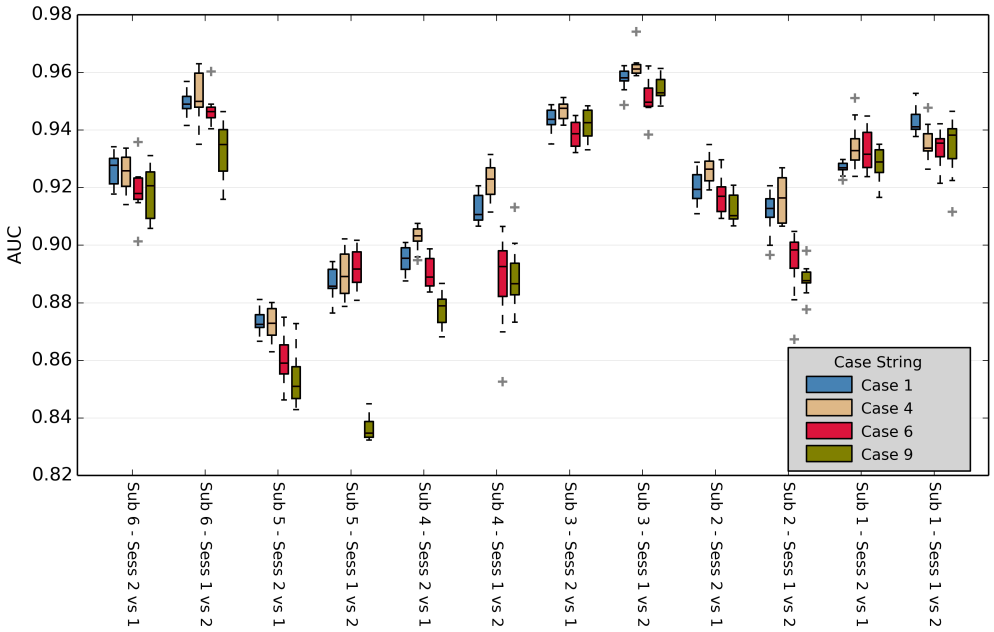

Figure 35: Classification performance for cases 1 (raw), 4 (best), 6 (baseline Shamwell et al. (2016)) and 9 (baseline Kirchner et al. (2016a)). Data of different subjects is learned and classified differentially well, leading to a great variance of results in overall case comparison.

Table 8 summarizes obtained results for session transfer and case comparison as well as SVM baseline. Case 4 emerged best AUC and BACC values, although all mean DL results are within the standard deviation of the other cases.

Although case 9 shows the worst AUC and BACC, except the SVM, it has the highest TPR. This means that P300 was correctly identified most often with greatest data reduction. However, it also has the lowest TNR, so the trained classification algorithms tend to recognize P300 even when not present. This may be helpful in some specific use cases where missing a target is more critical than a false identification.

The SVM baseline was outperformed by DL in all tested cases. It shows worst mean results and highest standard deviation for all metrics. However, the difference in BACC is only about 5 %. This baseline is a very simple version of an SVM classification and there are various more complex versions available. Also, test accuracy reportedly increases when training is performed with augmented data as well (Krell and Kim, 2017). In general DL and SVM can both be considered reasonable approaches for EEG data classification in a session transfer scenario.

Table 8: Summarized results for session transfer comparing all augmented preprocessing cases and SVM baseline. Best results are marked bold.

	AUC [%]		**BACC [%]**		**TPR [%]**		**TNR [%]**	
Case #	**mean**	**± std**	**mean**	**± std**	**mean**	**± std**	**mean**	**± std**
1	91.22	2.73	86.56	2.97	83.29	6.09	89.83	4.38
2	90.77	2.95	86.00	3.64	79.80	7.67	**92.19**	3.16
3	89.17	3.51	85.27	3.89	82.28	5.87	88.25	7.22
4	**92.37**	2.56	**86.59**	3.05	82.02	6.31	91.16	4.25
5	91.94	2.81	86.28	3.47	81.74	6.85	90.82	4.17
6	91.36	2.84	85.97	3.54	80.17	7.47	91.78	3.48
7	91.41	3.01	85.75	3.74	81.68	7.53	89.83	5.21
8	91.18	3.16	85.59	4.13	82.85	8.24	88.33	6.92
9	90.56	3.61	85.10	4.45	**83.85**	7.43	86.35	8.60
SVM	88.80	4.61	80.52	6.01	74.75	9.01	86.29	12.61

4.3 Subject Transfer

Subject transfer results are verified by 5 runs for each case. Figure 36 shows the BACC for classification on all six datasets with and without data augmentation for preprocessing case 4. In this scenario augmented training data did not increase performance. Therefore, following results are based on models that have not been trained with augmented data.

It may be that training sets include too much data with an average of 49,133 examples (9,827 without augmentation), compared to the capability of the ANN. A possible bottleneck for this is the number of nodes within the fully connected layers. Ideally each node of the last hidden layer represents a specific feature (or set of combined features) on learned data and its activation can be interpreted as the absence or presence of respective feature. When there are more key features to be learned than active nodes available, this can limit the classification performance. While the applied network only contains 250 nodes per fully connected layer, Shamwell et al. (2016) propose 500 nodes for a subject transfer scenario, but do not state the number of used training examples. However, empirical testing showed that increasing the number of nodes to 500 per layer further decreased classification performance. Therefore, the bottleneck limiting the ideal number of training examples is believed to be somewhere else within the network. Further investigations are needed to locate the relevant parameter and eventually automatically scale it to the number of available training examples, if procurable.

The results for case 1, 4 and 9 are shown in Figure 37. Again performance is similar for all cases, but case 4 achieved best mean BACC and AUC. For subject transfer results are expectedly worse than for session transfer. However, it is remarkable that balanced accuracies of nearly 90 % are obtained for the best setup (Rest vs. Sub 3). The mean BACC of case 4 is only 4.24 % below session transfer. This is a strong indication that DL is an appropriate tool to learn subject independent features. If applied to a dataset containing more subjects this could lead to a more general representation of desired ERPs, also improving performance on test data that are harder to classify correctly, such as subject 4 and 5.

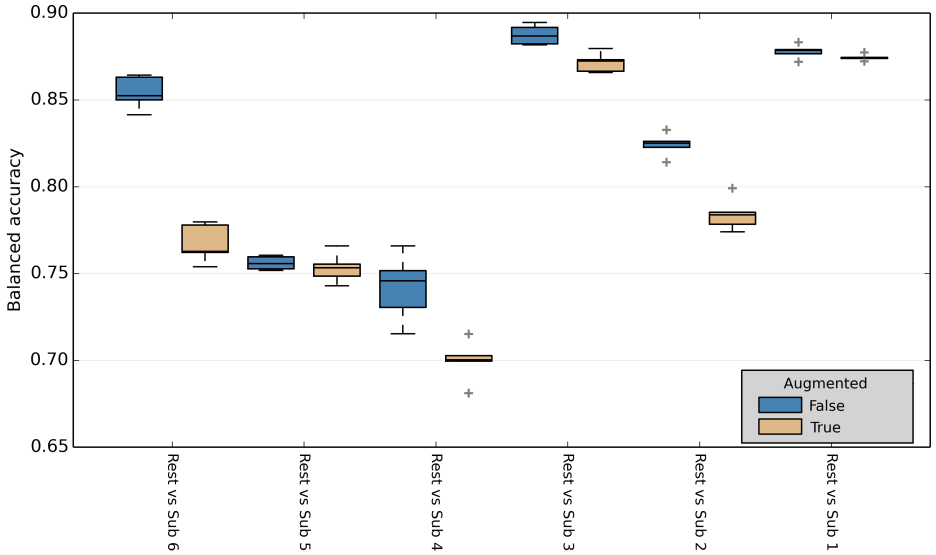

Figure 36: Data augmentation did not improve performance in a subject transfer scenario tested on case 4. It is believed that this is due to the great amount of training examples for the relatively small network.

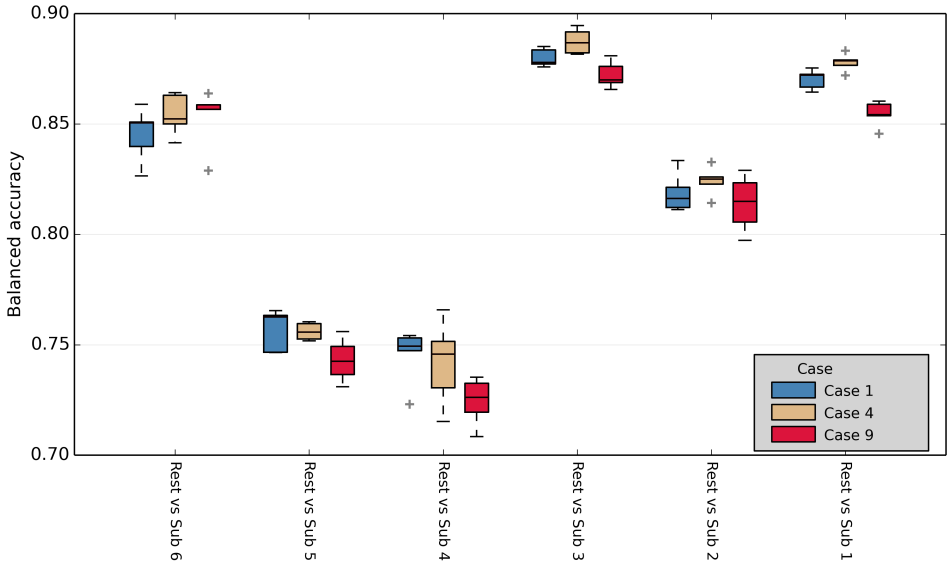

Figure 37: Subject transfer classification performance comparing cases 1, 4 & 9. Subject specific performance corresponds closely to the results of session transfer.

Table 9 summarizes the mean classification scores and standard deviation for subject transfer of the ConvNet and SVM. Results of the baseline are reduced intensively compared to session transfer. While AUC and BACC decrease for 25.33 and 20.19 % respectively, the TPR is at approximately the guessing rate of 50 % for binary classification. Shamwell et al. (2016) report an AUC of 72 % of their ConvNet for subject transfer, while training and testing occurs on data of 9 subjects each.

Table 9: Summarized results for subject transfer comparing preprocessing cases 1, 4 & 9 as well as SVM & ConvNet baseline. However, AUC from baseline is reported Shamwell et al. (2016) and is achieved on a different dataset. Best results are marked bold.

	AUC [%]		**BACC [%]**		**TPR [%]**		**TNR [%]**	
Case #	**mean**	**± std**	**mean**	**± std**	**mean**	**± std**	**mean**	**± std**
1	86.37	4.75	81.94	5.37	76.07	11.44	87.82	3.81
4	**88.31**	4.99	**82.35**	5.82	**76.14**	12.19	**88.57**	3.66
9	86.41	5.75	81.02	5.88	75.05	10.74	86.99	2.02
SVM	63.47	10.23	60.33	8.06	52,31	16,31	68,35	4,69
ConvNet	72.00	-	-	-	-	-	-	-

4.4 Conclusion

Section lists 1.2 four key questions which are ought to be answered throughout this project:

1. Is the DL approach appropriate for (DFKI) P300 EEG data classification?

Good results show that DL is an appropriate method for EEG data classification. While using biologically inspired methods from image processing, created ConvNet is comparably shallow and therefore does not rely on regularization methods to improve performance.

2. How much of the classical preprocessing and data reduction is necessary?

Classification performance is surprisingly similar for raw and heavily reduced data. Using raw data for training increases the needed training time. However, best classification is found for EEG data with a sampling–rate of 250 *Hz* containing high frequencies. This outcome is the same for session and subject transfer.

3. Can achieved results outperform existing processing pipelines on the same data?
In both cases compared baselines are outperformed significantly. Especially for subject transfer the difference in BACC are tremendous. This does not generally prove that DL is the best approach for EEG data processing, but it emphasizes the state of the art position for automated multi–dimensional data classification in general.

4. Can subject independent features be found?
ConvNet results for subject transfer are almost as good as the outcome of session transfer. This strongly implies that subject independent features have been generated. However, the differences between applied preprocessing cases are similar for both scenarios. And although compared to lower frequency ranges a significant improvement in classification is measured, it cannot be assumed that higher frequency bands contain more subject independent features than low frequencies.

Being able to give satisfactory answers to the primarily stated questions leads to the overall conclusion that this project can be considered successful.

4.5 Outlook

To be able to comprehend where subject independent features can be found, a detailed analysis of automatically generated featuremaps, respective weights and input dependent activations is necessary. Tools for this kind of analysis are becoming more and more present throughout relevant literature and should be considered for a follow–up study. Also, this ConvNet could be applied to other EEG datasets in order to identify a different type of ERP, such as LRP, and differentiate between multiple classes.

For further improvement of this or similar algorithms, more EEG data is most crucial. If at some point a very general model of desired ERPs can be created and subject transfer classification results are as good as session transfer, training of individual persons becomes obsolete for many BCI–based applications.

Steadily increasing computational power of available technologies will play a key role in the development of the approached topic. With improved hardware, complex algorithms can process more data in less time. Furthermore, cost

and size reduced design of future hardware will enable the usage of embedded brain reading on an industrial, but also personal level. With smaller chips, faster cores and stronger algorithms the ability to in a way *"read peoples minds"* may become closer than mankind has only dreamed about for centuries. It can not be completely grasped just yet where this path will lead and what opportunities, but also dangers lie within this ideas great potential.

This thesis may not have brought revolutionary knowledge or broadly unexpected results, but it supports the greater, overall idea of subject independent BCIs. Using a, in various ways, biologically inspired classification algorithm to enhance a preexisting signal processing chain has shown to be promising once again in this proof of concept study. The past century has widened our understanding of the central nervous system in a revolutionary matter. With novel technologies and steadily working research groups, insight will further extend and help to reveal greater scale functionality of the human brain. It is out of question that the abstraction from a biological phenomenon to a technical solution, called *biomimicry*, will continue to profit from this development and vice versa.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. `http://tensorflow.org/`.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Berger, H. (1929). Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570.

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

Davis, P. A. (1939). Effects of acoustic stimuli on the waking human brain. *Journal of neurophysiology*, 2(6):494–499.

Duvinage, M., Castermans, T., Petieau, M., Hoellinger, T., Cheron, G., and Dutoit, T. (2013). Performance of the emotiv epoc headset for p300-based applications. *Biomedical engineering online*, 12(1):56.

Fukushima, K. (1980). Biological Cybernetics Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybernetics*, 36(193).

Gilmour, A. D., Woolley, A. J., Poole-Warren, L. A., Thomson, C. E., and Green, R. A. (2016). A critical review of cell culture strategies for modelling intracortical brain implant material reactions. *Biomaterials*, 91:23–43.

Girden, E. R. (1992). *ANOVA: Repeated measures*. Number 84. Sage.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Haufe, S., Meinecke, F., Görgen, K., Dähne, S., Haynes, J.-D., Blankertz, B., and Bießmann, F. (2014). On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96–110.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons.

Hobson, J. A. (2009). *Encyclopedia of Neuroscience 2nd Ed.* Elsevier.

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

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Hünniger, D. (2013). *Artificial Neural Networks*. Wikibooks.org und the GNU Free Documentation License. `https://en.wikibooks.org/wiki/Artificial_Neural_Networks`.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Jasper, H. (1958). Report of the committee on methods of clinical examination in electroencephalography. *Electroencephalogr Clin Neurophysiol*, 10:370–375.

Kaplan, A. Y., Shishkin, S. L., Ganin, I. P., Basyul, I. A., and Zhigalov, A. Y. (2013). Adapting the p300-based brain–computer interface for gaming: a review. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(2):141–149.

Kaur, E. T. and Singh, B. (2017). Brain Computer Interface: A Review. *International Research Journal of Engineering and Technology*, 4(4):2395–56.

Khan, S. H., Bennamoun, M., Sohel, F., and Togneri, R. (2015). Cost sensitive learning of deep feature representations from imbalanced data. *arXiv preprint arXiv:1508.03422*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kirchner, E. A. (2014). *Embedded Brain Reading*. Phd-thesis, University of Bremen, Bremen.

Kirchner, E. A., Albiez, J. C., Seeland, A., Jordan, M., and Kirchner, F. (2013a). Towards assistive robotics for home rehabilitation. In *Biodevices*, pages 168–177.

Kirchner, E. A., Kim, S. K., Straube, S., Seeland, A., Wöhrle, H., Krell, M. M., Tabie, M., and Fahle, M. (2013b). On the applicability of brain reading for predictive human-machine interfaces in robotics. *PloS ONE*, 8(12):e81732.

Kirchner, E. A., Kim, S. K., Tabie, M., Wöhrle, H., Maurus, M., and Kirchner, F. (2016a). An Intelligent Man-Machine Interface-Multi-Robot Control Adapted for Task Engagement Based on Single-Trial Detectability of P300. *Frontiers in human neuroscience*, 10:291.

Kirchner, E. A., Will, N., Simnofske, M., Benitez, L. M. V., Bongardt, B., Krell, M. M., Kumar, S., Mallwitz, M., Seeland, A., Tabie, M., et al. (2016b). Recupera-reha: Exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation. *Technische Unterstützungssysteme, die die Menschen wirklich wollen*, page 535.

Krell, M. M. and Kim, S. K. (2017). Rotational and Temporal Data Augmentation for Electroencephalographic Data. *Journal of Machine Learning Research*, 1:1–13.

Krell, M. M. and Straube, S. (2015). Backtransformation: a new representation of data processing chains with a scalar decision function. *Advances in Data Analysis and Classification*, pages 1–25.

Krell, M. M., Straube, S., Seeland, A., Wöhrle, H., Teiwes, J., Metzen, J. H., Kirchner, E. A., and Kirchner, F. (2013). pySPACE—a signal processing and classification environment in Python. *Frontiers in Neuroinformatics*, 7:40.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42(1):11–24.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521:436–444.

Lemm, S., Blankertz, B., Curio, G., and Muller, K.-R. (2005). Spatio-spectral filters for improving the classification of single trial eeg. *IEEE transactions on biomedical engineering*, 52(9):1541–1548.

Li, J., Cui, H., Chang, C., and Hu, Y. (2014). A robotic rehabilitation arm driven by somatosensory brain–computer interface. *Int J Med Health Pharm Biomed Eng*, 8:294–297.

Lopez-Gordo, M. A., Sanchez-Morillo, D., and Valle, F. P. (2014). Dry eeg electrodes. *Sensors*, 14(7):12847–12870.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 4(2):R1–R13.

Luck, S. J. (2014). *An introduction to the event-related potential technique*. MIT press.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Minev, I. R., Musienko, P., Hirsch, A., Barraud, Q., Wenger, N., Moraud, E. M., Gandar, J., Capogrosso, M., Milekovic, T., Asboth, L., et al. (2015). Electronic dura mater for long-term multimodal neural interfaces. *Science*, 347(6218):159–163.

Muelling, K., Venkatraman, A., Valois, J.-S., Downey, J., Weiss, J., Javdani, S., Hebert, M., Schwartz, A. B., Collinger, J. L., and Bagnell, J. A. (2015). Autonomy infused teleoperation with application to bci manipulation. *arXiv preprint arXiv:1503.05451*.

Polich, J. (2007). Updating P300: An integrative theory of P3a and P3b. *Clinical Neurophysiology*, 118(10):2128–2148.

Rivet, B., Souloumiac, A., and Attina, V. (2009). xDAWN algorithm to enhance evoked potentials: application to brain–computer interface. *IEEE Transactions on*.

Sammut, C. and Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., and Ball, T. (2017). Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shamwell, J., Lee, H., and Kwon, H. (2016). Single-trial EEG RSVP classification using convolutional neural networks. *SPIE*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Straube, S. and Krell, M. M. (2014). How to evaluate an agent's behavior to infrequent events?-Reliable performance estimation insensitive to class distribution. *Frontiers in computational neuroscience*, 8:43.

Sturm, I., Lapuschkin, S., Samek, W., and Müller, K. R. (2016). Interpretable deep neural networks for single-trial EEG classification. *Journal of Neuroscience Methods*, 274.

Vidal, J. J. (1973). Toward Direct Brain-Computer Communication. *Annual Review of Biophysics and Bioengineering*, 2(1):157–180.

Widrow, B., Hoff, M. E., et al. (1960). Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York.

Woehrle, H., Krell, M. M., Straube, S., Kim, S. K., Kirchner, E. A., and Kirchner, F. (2015). An Adaptive Spatial Filter for User-Independent Single Trial Detection of Event-Related Potentials. *IEEE Transactions on Biomedical Engineering*, 62(7):1696–1705.

Zschocke, S. and Hansen, H.-C. (2011). *Klinische Elektroenzephalographie*. Springer-Verlag.

Index of Abbreviations

ANN	Artificial Neural Network
AUC	Area Under (ROC) Curve
BACC	Balanced Accuracy
BCI	Brain–Computer Interface
ConvNet	Convolutional Neural Network
CPU	Central Processing Unit
DFKI	German Research Center for Artificial Intelligence
DL	Deep Learning
EEG	Electroencephalography
ERP	Event Related Potential
fMRI	Functional Magnetic Resonance Imaging
FNR	False Negative Rate
FPR	False Positive Rate
GPU	Graphics Processing Unit
LRN	Local Response Normalization
LRP	Lateralized Readiness Potentials
LSTM	Long Short–Term Memory
pySPACE	Python–based Signal Processing and Classification Environment
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SNR	Signal to Noise Ratio
SVM	Support Vector Machine
TNR	True Negative Rate
TPR	True Positive Rate

List of Figures

List of Tables

Page

Acknowledgment

First of all, I would like to thank Dr. Mario Michael Krell for his encouragement to approach deep learning as a realistically doable topic for my thesis. Without his inspiration, this would still be a hard to grasp science fiction, discussed by supernaturally talented researchers. Thanks to Mario, it has now comprehensible reality. Besides inspirational work, he supervised my progress, answered uncountable repeated questions and spend dozens of hours on Skype, hotfixing code from California. He has proven to be extremely patient and was able to motivate me when necessary. His reviews of this work are of significant importance and indispensable.

Also, Dr. Elsa Andrea Kirchner deserves my deep appreciation for welcoming me to her research team at the Robotic Innovation Center. Although she is very busy managing multiple projects, she did not hesitate to provide me with all resources needed to accomplish this project. Her in depth understanding of neuroscience and BCI has helped to define the scope as well as the relevant aspects of my work.

Professor Dr. Frank Kirchner has earned my gratitude towards him for giving me the exceptional chance to work at the German Research Center for Artificial Intelligence and complying to evaluate my thesis. Also, Prof. Dr. Jan–Henning Dirks has shown exceptional good will by agreeing to be first supervisor in this project, although the topic mostly lies outside of his field of expertise. Without his concession this thesis would not have been possible. Both professors have put trust and faith in me which I promised to handle with respect and care.

There have been various problems, subproblems and subsubproblems to solve in order to generate results that actually make sense. Many of these encounters would have been almost impossible to overcome, if it wasn't for the help of my colleagues. Although constantly under pressure and chasing the next deadline, they were always open for my questions and took their time to help. Out of all these people, who I am going to miss working with, I would like to dedicate my special thanks to Dipl.-Biomath. Anett Seeland, Dipl.-Inf. Alexander Fabisch and Dr. Su–Kyoung Kim.

Last but not least, I am very thankful for all of my close friends and family, who supported me throughout my studies and particularly this thesis. May it be through providing simple things as food, a good laugh, cheer–up sessions or considerateness, it was, still is and will be an incredible asset to have everyone by my side whenever I need you.

Thank you very much.

Appendix

Statistical Evaluation

The following statistical evaluation was performed with major help by Dr. Su–Kyoung Kim. It can not be considered completely self–developed and is therefore listed in this appendix Section.

For session transfer, 9 different preprocessing conditions (ConvNet) and one baseline (SVM) are statistically evaluated. To this end, a one–way repeated measures ANOVA is performed with *case* (10 levels: 9 different preprocessing conditions using ConvNet and one preprocessing condition using SVM) as a subject–within factor (details for 10 cases, see Table 3). For multiple comparisons, Bonferroni correction is applied. Each condition has the sample size of 120 (6 subjects * 2 sessions * 10 validations).

There is a significant difference between 10 conditions (main effect of case [$F(9, 1071) = 99.03, p < 0.001$]). The worst classification performance is obtained when the SVM is used for P300 classification [$p < 0.001$ for SVM vs. all comparison pairs]. The best classification performance is achieved when no filter is used with sampling–rate of $250\,Hz$ (Case 4) or when the data was bandpass filtered between 0 and $50\,Hz$ with the same sampling–rate (Case 5) [Case 4 vs. Case 1, Case 4 vs. Case 2, Case 4 vs. Case 5: n.s., otherwise: $p < 0.001$, Case 5 vs. Case 1, Case 5 vs. Case 2, Case 5 vs. Case 6: *n.s.*, otherwise: $p < 0.001$]. The results indicate that the classification performance is superior when the data contains high frequencies (up to $50\,Hz$ or more than $50\,Hz$) compared to the data containing only a low frequency (up to $4\,Hz$).

For subject transfer, a Wilcoxen sign–rank test is performed to compare three different preprocessing conditions (ConvNet) and one baseline (SVM). Each condition has the sample size of 6 (6 subjects averaged over 5 validations). The worst performance is obtained when SVM is applied for classification [baseline vs. all comparison pairs: $p < 0.001$]. Again, the data containing high frequencies showed better classification performance compared to the data containing low frequencies [Case 1 vs. Case 9: $p < 0.043$]. There is no differences between the data containing high frequencies with different decimation [Case 1 vs. Case 4: *n.s.*].