

# Functional Magnetic Resonance Imaging

*Exploring data driven analysis of brain activations  
in rest and effort mode using deep learning.*

Eivind Erslund Kolbeinshavn

Master's thesis in Software Engineering at  
Department of Computing, Mathematics and  
Physics,  
Western Norway University of Applied Sciences

Department of Informatics,  
University of Bergen

June 2019



Western Norway  
University of  
Applied Sciences



# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Motivation . . . . .	6
1.2	Related Work . . . . .	6
1.3	Target Audience . . . . .	7
1.4	Thesis Structure . . . . .	8
<b>2</b>	<b>Research Approach</b>	<b>8</b>
2.1	Project plan . . . . .	8
2.2	Research Questions . . . . .	9
2.3	Research Methodology . . . . .	9
2.4	Research Validation . . . . .	9
2.5	Project Conclusion . . . . .	9
<b>3</b>	<b>Background</b>	<b>10</b>
3.1	Functional Magnetic Resonance Imaging . . . . .	10
3.2	Effort Mode and Default Mode Network . . . . .	11
3.3	Machine learning . . . . .	11
<b>4</b>	<b>Data Acquisition</b>	<b>12</b>
4.1	Data Collection . . . . .	13
4.2	Mental Rotation . . . . .	14
4.3	Working Memory . . . . .	14
4.4	Mental Arithmetic . . . . .	15
<b>5</b>	<b>Pre-Processing</b>	<b>15</b>
5.1	Why Pre-process? . . . . .	15
5.2	Data Management . . . . .	16
5.3	Realign and Unwarp . . . . .	16
5.4	Normalize . . . . .	17
5.5	Smooth . . . . .	18
<b>6</b>	<b>Analysis using Neural Networks</b>	<b>19</b>
6.1	Artificial Neural Network . . . . .	19
6.2	Convolutional Neural Network . . . . .	20
6.3	Learning . . . . .	21
<b>7</b>	<b>Design and Implementation</b>	<b>23</b>
7.1	Design . . . . .	24
7.2	Implementation . . . . .	24
7.3	Network Model . . . . .	31

<b>8</b>	<b>Results</b>	<b>32</b>
8.1	Results from Early Iterations . . . . .	32
8.2	Increasing the Accuracy . . . . .	35
8.3	Final Results . . . . .	41
8.4	Answering the Research Questions . . . . .	49
<b>9</b>	<b>Conclusion</b>	<b>50</b>
9.1	Further Work . . . . .	50
9.2	Conclusion . . . . .	51
9.3	Classifier . . . . .	51

## Abstract

Machine learning is a growing topic in computer science. The growing phenomenon that is machine learning keeps showing its usefulness in a large amount of scenarios, from advertising to research. Terabytes of data are stored every day on our computers and on computers in use by large businesses, research groups, hospitals, etc. These data serves many purposes from medical data used in diagnosis, or consumer data from online shopping to ensure that the consumer demand is fulfilled. However, with the large amount of data that are produced, the applications of the data can stretch far beyond that. Looking at data from one patient can be used to diagnose the patient, looking at several patient data set can be used to find patterns in the collections of patients and help prevent a disease in the future. However, working with large amount of data are no simple task, and extracting the important features in the data can be just as difficult. This is where the concept of machine learning plays a large part today. In the field of neural imaging, the applications that machine learning provide regarding finding patterns and correlations can be very useful. A fMRI project can create substantial amounts of data and finding patterns in these data efficiently in new ways can lead the way for findings granting better understanding on how the brain, and we, work.

## Acknowledgment

I want to thank my supervisor Jon Eivind Vatne for giving me the opportunity to work on this thesis. He provided guidance to me through this thesis and the writing process and this thesis would not be possible without him.

I also want to extend thanks to my supervisors at Haukeland University hospital/Mohn Medical Imaging and Visualization Center (MMIV), Eli Renate Grüner and Lars Erslund for letting me part of this project. They introduced me to the project and guided me through everything related to fMRI. I was allowed to take part in every part of the fMRI-project, from data acquisition to pre-processing of the data. I was provided with a environment to work in and was allowed to join the research team at Haukeland on their seminars. For this I am very thankful, and again, this project would not be possible without them.

Lastly, I want to thank some of the people helping me through different steps of the project. I want to thank Frank Riemer for helping me with the pre-processing of the data. And I want to thank Sindre Kolbeinshavn for the design of some of the figures used in this thesis.

# 1 Introduction

This section will introduce the structure of this thesis and the motivation behind it. Work related to this thesis will be discussed and who this thesis might be of value for.

## 1.1 Motivation

The motivations for this project started with the suggestion of the existence of a generalized non-specific task-dependent network that showed correlation with the task-independent network, discussed in the article "On the existence of a generalized non-specific task-dependent network"[1]. The observation of this network was made while Kenneth Hugdahl, one of the authors of this article, when he was preparing for a lecture looking at different brain activation patterns related to different cognitive tasks. An observation like this can have a massive impact on the field of neural imaging and how brain activation of cognitive tasks are interpreted. From this article, it is clear that the ability to observe hidden patterns in neural imaging can be important. It was suggested to make a deep learning classifier to observe if a machine learning approach would be able to find patterns in fMRI BOLD-images related to cognitive tasks.

## 1.2 Related Work

This project is based on the observations done in the article about the Effort Mode Network(EMN) from researchers at Haukeland University Hospital.[1] This article, and other articles related to medical imaging and classifications of brain activation network are therefore all valuable and related to this project.

Machine learning is something that is widely used today and finding something that machine learning has not been used for can be a challenge. However, machine learning, or rather deep learning used to predict network related to brain activations from an fMRI-project is not that common. In this thesis, the machine learning algorithm that will be addressed is the three-dimensional convolutional network for classification, therefore articles related to this will be most useful.

Several article exist discussing approach of implementing a machine-learning algorithm to classify medical images. In addition, several pre-trained deep

learning models exists for classification of medical images or other three-dimensional images. This include deep learning networks platforms like NiftyNet[3] and deep learning networks like DeepMedic[4].

However, just a few articles discuss the usages of deep learning on resting-state functional magnetic resonance imaging, especially using three-dimensional neural network on blood-oxygenation-level-dependent (BOLD) fMRI images.

An article related to the usages of three-dimensional convolutional neural network on fMRI BOLD-images from 2018 states: "To the best of our knowledge, this is the first study that employs a single blood-oxygenation-level-dependent (BOLD) fMRI volume as the input of the 3D-CNN for task classification"[5]. This article describes an approach to predict sensorimotor tasks using 3D-CNN, which is very related to this project. The article as mentioned, uses BOLD-images of the sensorimotor tasks as input and uses the input in a 3D-CNN and compares it to another deep neural network pre-trained on the same data set. It concludes that the 3D-CNN outperforms the other feedforward deep neural network.

The 3D-CNN proposed in this thesis will have a similar approach to implementation done in the article; "3D convolutional neural network for feature extraction and classification of fMRI volumes"[5], but will tackle a different data set related to brain activation of cognitive tasks, and will use a more complex network structure to better represent the data. In addition, the article uses a modified LeNet-5 network, while the network proposed in this thesis will be constructed from the ground up using Keras with TensorFlow.

### **1.3 Target Audience**

This thesis is mostly target at people in the field of computer science and people working with medical imaging or other complex images, mostly three-dimensional. The thesis focuses on using deep learning to classify functional magnetic resonance images and will therefore have more importance for people working with classifying brain activations or action recognition in other fields. This does not limit the thesis to be relevant to people working in other fields of interest since machine learning, deep learning, image classification, and working with medical imaging are all topics of relevance and topics that will most likely continue to have more importance in the time to come.

## 1.4 Thesis Structure

Reading this first section will give a brief introduction to the topic and how it will be approached. In Section 2 the research approach in this thesis is discussed. Section 3 gives insight in how functional magnetic imagining works in addition to some of the concepts related to fMRI that will be used in this thesis. Machine learning concepts are introduced in this section as well. In Section 4 the data acquisition is explained in addition to the brain activations of different cognitive tasks used in this project and how they were collected. Pre-processing is addressed in Section 5, explaining each step of the process. In Section 6 concepts related to artificial neural networks that are used in this project are presented. Section 7 will explain the implementation and design choices done along the way. It will also introduce problems that accrued and how some they were fixed during the implementation. The results will be discussed in Section 8 from early iterations of the project to the final results. The last section will conclude the project and discuss changes that could be done in the future.

## 2 Research Approach

### 2.1 Project plan

The project I have started working on is a project lead by the fMRI research group at Haukeland. The purpose of project is to explore data driven approaches to analyze the acquired neural imaging data and target the methods most sensitive and most reliable to pick up the changes between the EMN and DMN states of the neural activations.

First, I will be working closely with the fMRI research group at Haukeland University Hospital on the mentioned project. Understanding the analysis is important for developing a software that can optimize the analysis. I will then start working on how this can be done differently, looking on which methods that can be used to improve the analysis using software engineering. Lastly I will create a software that can be used in this analysis and afterwards improve it to be used in similar analysis. The expected result of the project will be: Better understanding of the two networks EMN and DMN. Optimization of the analysis process related to these networks. Software to be utilized in conjunction with a similar analysis later.



## **2.2 Research Questions**

The research questions for this thesis will be: How do EMN and DMN interplay in each individual participant? Here the focus will be on each cognitive tasks in conjunction with the DMN and how separable they are.

How can the experiment be optimized, i.e. shortened in time without the loss of accuracy- to develop an easy-to-include experiment for future clinical settings or multicentre studies. Here the focus will be on implementing a machine learning approach to the fMRI analysis. It was decided in the beginning of this project to use a three-dimensional convolutional neural network (3D-CNN) as the model for this implementation.

## **2.3 Research Methodology**

The method of approach to answer the research questions is a quantitative one. Through a series of fMRI scans conducted by Haukeland University Hospital as a part of a project to get a better understanding of the connection between EMN and DMN, data was collected from several volunteers, including myself. The data will be analyzed using a machine learning model. The important aspect here is to get an overview of the different cognitive neural activations in relations to the default mode network, so that it is possible in later projects to improve on the methods used in the analysis. The main goal of this thesis is to use the information gained from the analysis to make/improve software such that these experiments can be shortened in time without the loss of accuracy.

## **2.4 Research Validation**

The thesis will be validated by the performance of the working software. The criteria for the performance will be how well it can pick up and differentiate the different cognitive tasks, how optimized this process is, and how this can be used in other similar analysis. The thesis will also be validated by looking at the result of the analysis of these networks. The metric for validation will be the accuracy of which the classifier implemented with machine learning, is able to predict the different neural networks.

## **2.5 Project Conclusion**

The conclusion of the thesis will consider the working software and its significance to the analysis of the networks. I will look at both the strengths and

weaknesses of the software and discuss how the software could be improved, if possible. I will also consider and discuss how it can be applied to other analyses and then discuss what changes should be made to make this possible. The process of the project is also a valid subject of discussion, where I will look at the different approaches that was used to finish the software and which of these approaches that worked, and which approaches that did not produces the wanted results, and why.

### 3 Background

This chapter will introduce the information needed to get an understanding of the work done in this thesis. A simple introduction to fMRI and the Blood oxygenation level dependent (BOLD) imaging will first be presented. In later sub-sections, concepts related to machine learning will be introduced. Most of the information about machine learning will be from the book "Machine Learning, and algorithmic approach"[6].

#### 3.1 Functional Magnetic Resonance Imaging

Functional magnetic resonance imaging (fMRI) is a measurement technique that is used to measure changes in activity (function) in the brain, using an MRI scanner to retrieve data from the tissue and visualize these data with images. This is different from regular MRI where the focus is to study the anatomy of the brain (or other parts of the body). Functional magnetic resonance imaging is primarily done by looking at the blood flow in different parts of the brain during the execution of different sensory or cognitive tasks[2].

It is then possible to study all the parts of the brain that are active during these different tasks and find connections between them, creating a network. Researching the different networks of the human brain can be very important for getting a better understanding of how the brain works and then use this information to prevent, treat, and understand many of the illnesses that are related to the brain.

When area of the brain is activated to solve a task, the neural activity increases the regional cerebral blood flow (rCBF) to compensate for the increase in metabolic activity. Oxyhemoglobin, oxygen-loaded hemoglobin, is diamagnetic and therefore will give an increase in the signal strength that is used retrieve information. This in turn makes the parts of the brain that are activated during the task that the person in the MRI-scanner performed simple to differentiate from the other parts of the brain using contrast[2]. A

key concept in MRI and fMRI is the ability to look at contrast to highlight the areas of the brain or other areas of the body of interest. In fMRI, we want to use a contrast that can highlight the metabolic activity in certain parts of the brain during the fMRI scans. Because of this, the contrast that nearly all fMRI studies rely on is the blood-oxygenation-level-dependent (BOLD) contrast[2]. In this project, BOLD-contrast neural image is what will be used to classify different cognitive neural activations.

### **3.2 Effort Mode and Default Mode Network**

As mentioned, researching the different neural networks of the brain is important getting a better understanding of the brain in the field of medicine. One of these networks is the Default Mode Network (DMN). DMN is the network that is active when the brain is in a resting state or in other words, a task independent network. Recently the fMRI research group at Haukeland University Hospital, suggested the existence of a similar network, the Extrinsic Mode Network (EMN), or Effort Mode Network, that was active when the brain solved cognitive tasks. It is found that these two networks are connected in that if one of the networks is up-regulated, the other is down-regulated and vice versa[1].

### **3.3 Machine learning**

Machine learning is about making an algorithm adapt its actions to data so that the actions can be more accurate. Learning is a key concept, and the objective of a machine learning algorithm is to learn from experience. Another key concept is the ability for the machine learning model to follow instructions so that it can generate a better performance based on previous instructions[8]. The learning part of machine learning can be divided into different types like supervised learning, unsupervised learning, or reinforced learning.

In supervised learning, the machine learning algorithm is provided with a set of training data and a set of labels/targets with the correct responses. The algorithm tries to create a generalized solution to fit all possible data points using the training data. Supervised learning is the most commonly used learning method and is used in several machine learning algorithms like, decision trees, support vector machines, neural networks, etc. Supervised learning is the learning method that will be used in this project.

Supervised learning can then be grouped into two approaches to deal with different problems, regression and classification. To solve regression problems

a mathematical function is suggested to best fit each data point in a data set. New unlabelled data points can then be predicted using the created mathematical functions. Classification problems consists of taking a data point from a data set and predicting to which class the data point belongs. The classification problem is discrete meaning that each data point belongs only to one class/label. Classification is the method that will be used in this project. The training data will be a set of neural images and the classes will be the different cognitive tasks which are related to each image. This will be explained in detailed in Section 4 about data acquisition and management.

Most machine learning approaches follows the same guidelines during development of a machine learning model:

First data must be collected and prepared. The data is the input in the machine learning algorithm and what the algorithm uses to predict new data points or labels. It is therefore important to ensure that the quality of the data is acceptable. Often professionals in the field from where the data is collected takes part in ensuring that the data will be useful and that data that would be damaging to the learning process is discarded.

Secondly, the important features in data must be specified. There are many approaches for feature extracting based on the data and what features that are deemed as important in the data and in the machine learning project.

Next, an algorithm has to be chosen. Several different machine learning algorithm exists that each is useful in different scenarios. Parameters that can be changed before training in the chosen algorithm is called hyper parameters. Tuning of these parameters is important so that the chosen machine learning model can best represent the input data.

Training the machine learning model is the next part of the process. The data set is given as input to the algorithm and the algorithm tries to represent these data. Lastly, the model must be evaluated and tested for the accuracy on the data [6].

## 4 Data Acquisition

Data collection and preparation is an important and time consuming part of a fMRI-analysis, and of a machine learning process. Large amounts of data was collected in 2016, 2017, and 2018 for this fMRI-analysis. In this section, the data acquisition process is presented with an explanation of the cognitive tasks that the participants were instructed to perform.

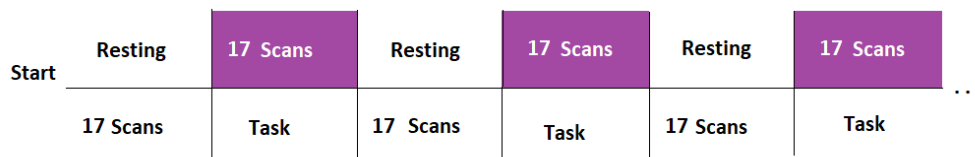
## 4.1 Data Collection

The data for this project was collected at Haukeland University Hospital using one of several MRI-scanners available. The participants are placed in the MRI-scanner containing a large electromagnet that produces a magnetic field with the power of 3.0 Tesla in the scanner. The magnetic field aligns the hydrogen atoms in the participants body, or in this case, in the brain. Radio waves are used to change the alignment of the hydrogen atoms, when the scanner stops sending radio waves the hydrogen atoms returns to their align state in the electromagnetic field. A signal is produced from this process where the hydrogen atoms realign and the scanner interpreters this signal to create an image. The oxygen-loaded hemoglobin, oxyhemoglobin, will produce a different signal and this is why it is possible to differentiate the different working part in the brain[2],[7].

Fifty-four healthy participants, including myself, was each scanned for about an hour and were instructed to do different task throughout the scanning process. There were three different tasks that the participants were asked to solve: mental rotation (MR), working memory (WM), and mental arithmetic (MA). The tasks were represented to the participants through a set of goggles that could endure the magnetic field while providing the task from a computer outside the room where the scanner where located. The tasks had binary answers and were answered by clicking on a button that was provided to the participants. Starting from a resting state, each participant would do these tasks in interval for 17 scans then rest for 17 scans as showed in Figure 1 and 2

Task name:	Onset time:	Duration
MR1	0	17
WM1	34	17
MA1	68	17
WM2	102	17
MA2	136	17
MR2	170	17
MA3	204	17
MR3	238	17
WM3	272	17

Figure 1: Table showing the different task and when they were performed.



**Tasks:**

- Mental Rotation:** Comparing 3D-objects using mental rotation.
- Working Memory:** Modified Stroop test to better test working memory (short-term-memory)
- Mental Arithmetic:** Simple addition with the goal of finding all numbers adding up to the sum of eleven.

Figure 2: Showing the scanning process.

## 4.2 Mental Rotation

In the mental rotation task the participants was asked to determine if two three-dimensional object were the same object just rotated differently. Several images with two three-dimensional objects were showed in quick succession and when the participant thought that the two object were identical, they would push the button.

## 4.3 Working Memory

The working memory task was a modified Stroop test. Different words describing colour was presented to participants. The words also had different colours and the point of the task was to remember the colour of the word, not the colour the word described. Adding to this, the participants were asked to click the button when a colour appeared that had appeared four words earlier.

## 4.4 Mental Arithmetic

Lastly, the mental arithmetic task presented the participants with two numbers. The task was to click the button every time these two numbers added up to eleven. When the scanning was finished for each participant the data was systematically stored and were controlled by radiologist before the data could be used for research.

## 5 Pre-Processing

Pre-processing is a crucial part before using the neural images in an analysis. Pre-processing is important to improve the detection of the areas of interest. This can have a large impact on the analysis, as stated in the article "Impact of functional MRI data pre-processing pipeline on default-mode network detectability in patients with disorders of consciousness", where the usage and importance of pre-processing on resting state fMRI is discussed.[16]

As mentioned in Section 4 the data preparation can be time consuming. A large amount of the time in this project went to pre-processing of the data and getting a understanding of the different tools and neural images that was used.

In this section, we will take a look at the pre-processing process in this project, and different pre-processing steps will be explained.

### 5.1 Why Pre-process?

Before the fMRI-data collected for this thesis can be used in an analysis, it needs to be pre-processed. Slices from the fMRI data acquisition can end up with different variations like noise, distortion, or other variations. There are several different reasons for this, either technical, where the variation source is from the scanner, or physiological, where its stems from the subject. By pre-processing the data, it is possible to correct these variations that can be quite severe, and even spoil the entire experiment if not corrected. The data set that were used in this project went through a pre-processing pipeline using SPM12 (Statistical Parametric Mapping) in Matlab, a software package which is designed for analysis of brain imaging.

## 5.2 Data Management

Before the data could be pre-processed, it needed to be unzipped and extracted from IMA file format. This was done using MRICConverter, a program design for this purpose.

Next, the 306 fMRI-BOLD image files from each subject were extracted and placed into individual folders. This was done for all the scans from 2017 and 2018. The scan from 2016 had 285 images instead of 306, different from the ones from 2017 and 2018 and were therefore not included in the pre-processing or the analysis. This is because the scanning-structure could not be found from the 2016 scans, and it is therefore impossible to label the task that the subjects were performing during the scan.

After all the images from each individual scan were organized in different folders, the data were ready to be pre-processed. In this project, the fMRI-BOLD-images goes through three steps of pre-processing; Realign and Unwarp, Normalize (estimate and write), and Smooth, which all are functions in SPM12. In the pre-processing of these images most of the parameters for the pre-processing are set to default and will not be changed since these are the standards in each pre-processing step.

## 5.3 Realign and Unwarp

First the Realign and Unwarp function is used to match images. Since movement of the subject might occur in the scanning processing, it is important to remove the differences caused by this movement between scans during pre-processing so that the data can be properly used in later analysis. This is done by the Realign function. It does this using a six parameters rigid body transformation, moving the images in X, Y, or Z direction and/or rotating the images over the X, Y, or Z-axis, and a least squares approach to minimize the difference between two scans. Next it uses the Unwarp function. fMRI-BOLD scans are sensitive to magnetic inhomogeneity resulting in deformation in images around air-filled cavities in the head. These deformations might also occur out of movement in the subject. Only using the realign function is not sufficient in this case because of changes in position, shape and volume as a function of time in the image. After the Realign and Unwarp step, it is created a new set of 306 images with the prefix "u", and also a mean image of the subjects brain[15].



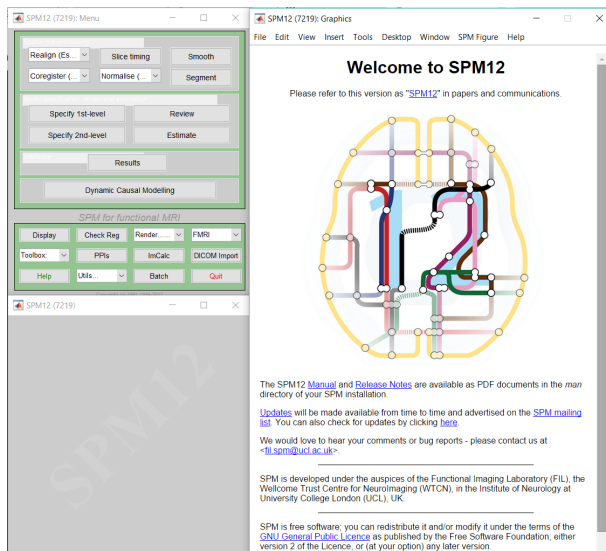


Figure 3: SPM12 GUI start screen

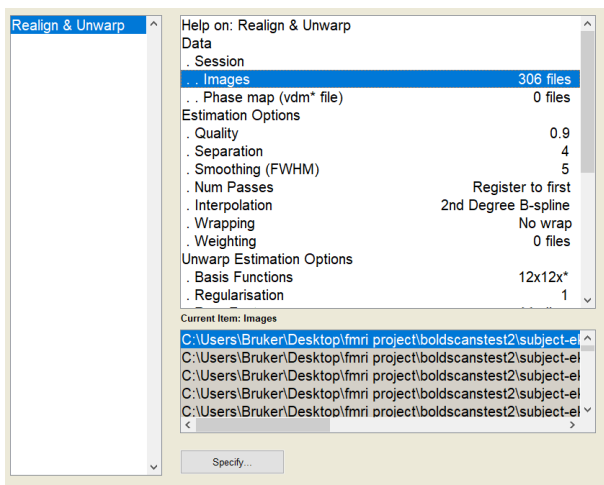


Figure 4: Realign and unwarp for SPM12

## 5.4 Normalize

The second function is the Normalize function that is used on the images with the prefix "u". The normalize function computes the warp which is applied to the images to make them into a standardized space. The mean-image from the previous step is used as the "image to align", which creates a set of warps, and the rest of the images are warped accordingly. After this

pre-processing step 306 new images are created adding the prefix "w" to the images, making a set of 306 images with the prefix "wu"[15].

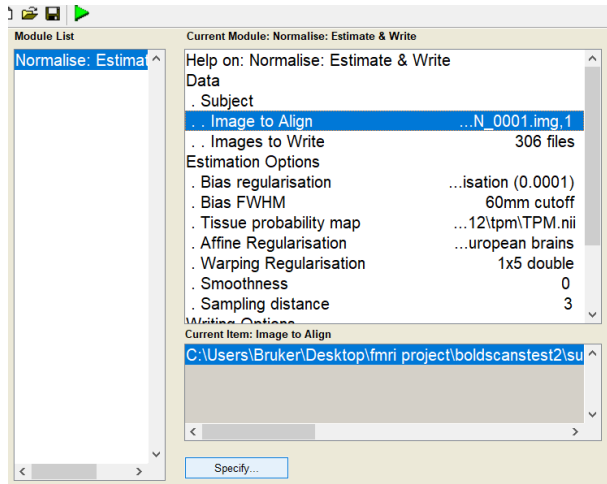


Figure 5: Normalization using SPM12

## 5.5 Smooth

The last function is the Smooth function that will be used on the images with the prefix "wu". This step is used to suppress noise in the images. It is done by applying a Gaussian smoothing kernel to the images. After this last process is done a last set of 306 images are created with the prefix "swu", and the images are ready to be used in an analysis[15].

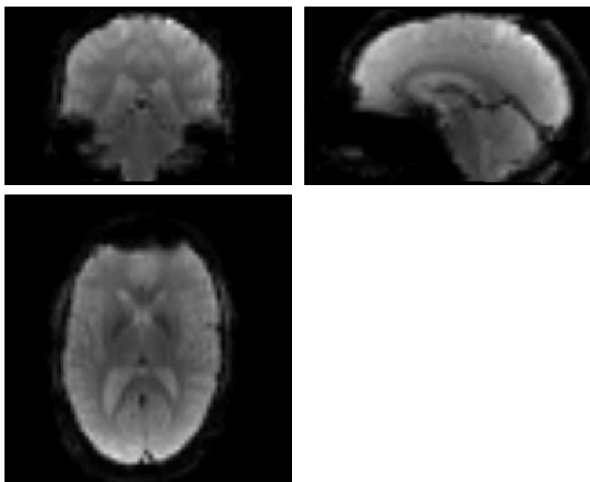


Figure 6: Images before pre-processing

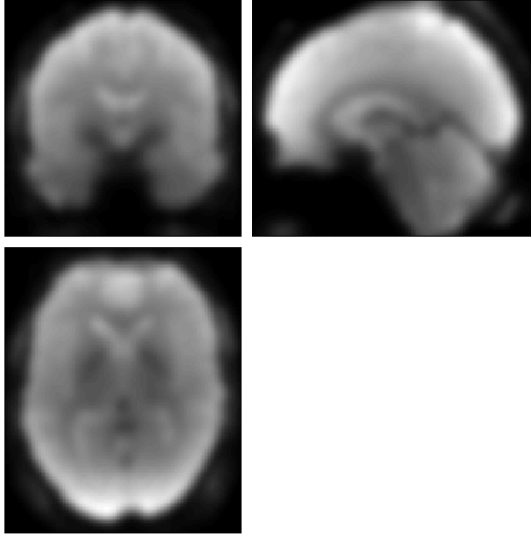


Figure 7: Images after pre-processing

Most of the visual changes that are applied to the image is done during the smooth operation which makes the images look a little more blurry. Since these images are fMRI time-series it is difficult to show exactly how these pre-processing steps effects the images.

## 6 Analysis using Neural Networks

### 6.1 Artificial Neural Network

Artificial neural networks (ANN) in computer science are models used in machine learning, drawing inspiration from biological neural networks. An artificial neural network model is composed of layers with each layer containing a set of neurons. The first layer in ANN contains the features in the data and different combinations of these features are sent to neurons in the next layer. The neurons contains an activation function and a weight. The data from previous layers are multiplied by the weight in the neuron and used in the activation function that might activate the neuron. If activated, output from the neurons continues to the next layer in combination with other activated neurons, until the last layer where the final output in the neural network predicts the label[18].

ANN differs from several other machine-learning approaches since it learns features from the data. The learning start with forward-propagation where

the learning data is sent through the network and a label is predicted. A loss-function is used to calculate the error of the prediction. Several different loss functions can be used, for example; mean-squared, categorical-cross-entropy, Kullback Leibler divergence depending on the use case of the neural network. Each neuron that took part in predicting the output have their weight (their importance to the prediction) changed accordingly through back-propagation. This is done by using gradient decent to minimize the error from the loss function. One full cycle of the input through the network is called an epoch.

The reason for using a neural network in this project is the models ability to find patterns and learn features. The fMRI BOLD scans from a participant solving a mental rotation task might have some similarities with the scans from the mental arithmetic task and an ANN will probably find the features that differentiate the two. Looking at EMN it might also find the connections between all the task-activated brain networks and it might be easier to look at regulation between EMN and DMN.

## 6.2 Convolutional Neural Network

In this project, the focus will be on a convolutional neural network(CNN). CNN is a class of neural network that are used for image predictions/labelling. CNN is similar to a regular ANN, but it uses convolution operation using a kernel on the input in the hidden layers. The kernel is a matrix that slides over the image and that tries to represent each value that the kernel is currently covering in the image, with one value, multiplying each pixel/voxel value with the values in the kernel and then adding these together, this process is called convolution. Each neuron apply a kernel on the image promoting different features for each different kernel[19].

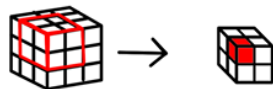


Figure 8: 3D convolutional layer used on a three-dimensional image

Pooling, often Max Pooling or Average Pooling are also often used as a

layer in CNN models. Max pooling takes the highest value from the image where the kernel currently covers and represents that area of the image with that value, while average pooling takes the average value from all values and uses that to represent the area of the image[19].

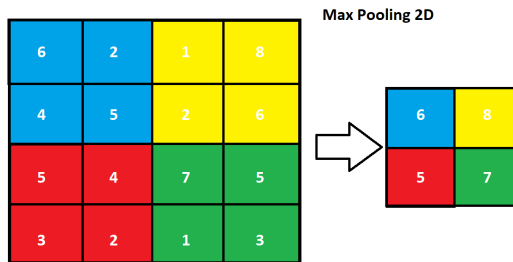


Figure 9: Showing max pooling in 2D.

The same processes then follows as in a regular ANN: some neurons activates and send their outputs to new neurons, a label is predicted, and a loss function is used to calculate the error for back-propagation where the weights of the kernels in the neurons are changed. Normally CNN is used for two-dimensional images, but it is possible to modify it to work on three-dimensional images that is needed for the images in this project. The models are very similar, but the version modified for three dimensions applies a cube, as its kernel on the three-dimensional image.

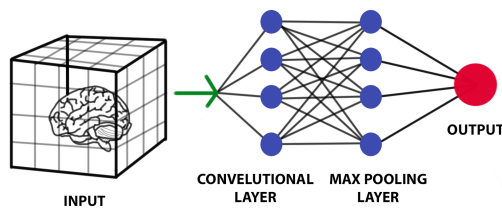


Figure 10: Simple representation of a 3D-CNN using fMRI image

### 6.3 Learning

As mentioned, the learning of the network is done using an algorithm called back-propagation. Back-propagation uses the activation function and the

derivative of the activation function in all the nodes that fires during the training in conjunction with the loss function. Back-propagation can be divided into two parts; forward-pass/forward-propagation and backward-pass/back-propagation. During the forward-pass, the data is sent to the neurons through the network and are used in the activation function in these neurons. The data from previous layers are multiplied by the weight of the neuron and the product is used in the activation function in that neuron. The output of the activation function is then sent to the next layer. The derivatives of these function is also stored in the node, but only the data from the (non-derivative) function is sent forward through the network. A prediction is made at the end and the loss/error of the prediction is calculated. The next step is the backward-pass were the goal is to minimize the loss. This is done using an optimizer. Optimizers are algorithms used to find the global minima of the loss/error. An example of one such optimizer is gradient descent/ steepest descent. The goal as mentioned is to change the weight of the neurons to better predict the label of an input. For this we need to find the derivative of the error in respect to the weight[6].

$$\frac{\partial E}{\partial w} \tag{1}$$

However, since the loss functions does not contain the weights we cannot find the derivative directly, therefore the chain rule in derivatives must be applied[6], showed in Figure 13 and in equation below.

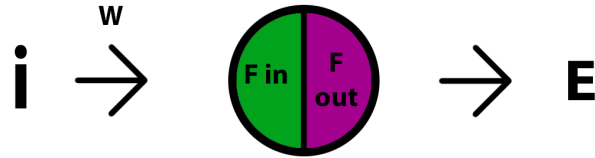


Figure 11: The input "i" is multiplied with the weight "w" that produces "F-in". F-in is sent to the activation function that produces "F-out" that is used to calculate the error

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial F_{out}} * \frac{\partial F_{out}}{\partial F_{in}} * \frac{\partial F_{in}}{\partial w} \quad (2)$$

At last the weight can be updated by subtracting the value of the previous weight with the value of  $E/w$  multiplied by a given learning rate. The learning rate determines how much the error effects the new weight.

$$W_{new} = (W - \frac{\partial E}{\partial w} * lr) \quad (3)$$

## 7 Design and Implementation

In this section, design choices will be discussed briefly. The focus in this section is on the implementation of the 3D-CNN model and choices done during development. After reading this section one should be able to replicate the implementation done in this project.

## 7.1 Design

Early on, it was decided that a 3D-CNN would be used in this project. The goal for the 3D-CNN classifier is to classify complex data with high accuracy. The design of the model should therefore represent this. In the beginning of the design phase, it was decided that the network should be a complex model to represent the complex data, in other words, a model with a large amount of layers and neurons.

## 7.2 Implementation

The implementation of neural network was done using Python (3.6) in main conjunction with the frameworks; Keras and TensorFlow. TensorFlow is an open-source library developed by Google that can be used as an interface for many machine-learning algorithms.[9] Keras is an API for high-level neural networks for Python development that can be used with TensorFlow among other libraries that are used for deep learning. Keras supports several neural networks and especially convolutional neural network that is used in this project. It is a user friendly API with focus on modularity and easy extensibility, which was determining factors for choosing this API for the project since these features are desirable when testing and building a neural network over a larger period[10].

Other packages that was used were "numpy" for array objects and computing, "sklearn" for label encoding and splitting data, "glob" for managing directories, "pandas" for reading xlsx-documents, "os" for path manipulation, "nibabel" for reading the 3D-images, and "timeit" for managing time usage.

To get TensorFlow working with GPUs the NVIDIA CUDA toolkit which provides a set of tools for high performance GPU-accelerated applications, and the library NVIDIA cuDNN (NVIDIA CUDA Deep Neural Network library) which is a library for working with deep neural network using GPUs, must be installed.[12], [13].

Not all version of CUDA and cuDNN are compatible with all version of TensorFlow. In this project TensorFlow-gpu 1.12.0 was used with CUDA 9 and cuDNN 7. For development of this network, the IDEs Pycharm Community Edition 2017.2.4 and Visual Code has been used. Anaconda was also used to set up the environment for the program to run in. The IDEs were then configured to work with the environment created by Anaconda.



```
conda create --name tf_gpu tensorflow-gpu
```

Figure 12: Creating a conda environment using TensorFlow GPU

TensorFlow comes with a lot of configuration that can be specified for your network model. In this project a TensorFlow session is specified in the start of the script to allow the GPUs to dynamically locate its memory to training[11]. Setting up a TensorFlow session were also useful in the beginning of the project to look up which devices TensorFlow finds and uses during training[11]. When the session is configured, it can be set as a Keras-session and the model will be trained using these configuration.

```
)# Creates the TensorFlow config
config = tf.ConfigProto()

# Dynamically grows GPU memory
config.gpu_options.allow_growth = True

# Log the devices (GPU, CPU) and their placement
config.log_device_placement = True

# Creates a TensorFlow session with the created config
sess = tf.Session(config=config)

# Set the default keras session to be the configured TensorFlow session
set_session(sess)
```

Figure 13: Creating a session using TensorFlow and Keras

The program starts by specifying a path to the folder containing each subject folder with the fMRI BOLD-images for each subject. The path to the onset-times for each of the different tasks are also specified at this point and the label data from the xlsx-document is saved, creating a 306x1 sized numpy array. Next a for-loop iterates over each folder for each subject and read the images with the prefix; swu since these are the images that have been pre-process. The image size is 79x95x79 and are stored in a numpy array. After each iteration of the for-loop the array containing the image data are added together vertically creating a numpy array with the dimension of 79x95x79 x N , where N is the number of subject times each image where the number of images per subject was 306. When testing the network the number of subjects has been different amounts from four to fifteen, but the final number of subjects that were used were thirty, making a 79x95x79x9180 numpy

array. The label array is then repeated for each iteration of the for-loop to be the same size as the image data, making the new array a 9180x1 sized array. Lastly, the data is split into test and training data sets for both the image data and the labels. When splitting the data it is possible to specify a random state in the split-function. The random state makes sure that the data is split in the exact same way each time when using the same random state between runs. This is important when trying to replicate previous runs in the early stages of building the network. It is then easier to see changes done to the network in the result after training.

60% of the data was split into the training set and 40% to the test set. 33% of the training set was also split into a validation set specified in the compile-method of the code and used during training. There is a large amount of data in this project so it could be argued that the training set should be smaller. But because the data is very complex, which were represented in the training result, a larger set of example to train on was important to learn the important features in the data.

Working with 3-dimensional image data can be very memory intensive. Using a laptop I was only able to read one subject data set before running out of memory. Even when working with the computers at MMIV (Mohn Medical Imaging Visualization Centre) at Haraldsplass (Bergen)[14], it was not possible to run all the data at once on an Alienware computer with several CPUs and GPUs, with early iterations of the program. This was fixed by changing the data types to address the memory issue. The data type that numpy-array used as default was a 64-bit float array. This was first changed to a 32-bit array and then later to a 16-bit float array. The same images from these arrays was compared to each other to see if the images was changed in any way when converting the data. Since there were no difference between the images it was possible to use the 16-bit float array without losing any information in the images. It was then possible to run all the thirty subject data sets.

Creating the model is a very simple process when using the Keras-framework. There are a choice between using a Sequential model or a Functional model when creating a deep learning model in Keras. The sequential API allows constructing a deep learning model layer-by layer. The functional API is more flexible and allows connections between layers not just between the previous and the next layer. For this project, a sequential model was chosen. The model is divided into twenty-one layers including the input and output layers. The model uses 3D convolutional layers, Max pooling layers,

flatten layers, dense layers and dropout layers. Each of the convolutional layers and the dense layers uses the ReLU (Rectified Linear Unit) activation function. ReLU is often seen as the default activation function because it makes the model that uses it easier to train and will often increase the overall performance[17]. ReLU is a piece-wise linear function that determines if the neurons activates or not. It is decided by checking if the input is larger than zero, using the formula below.

$$g(x) = \max(0, x) \quad (4)$$

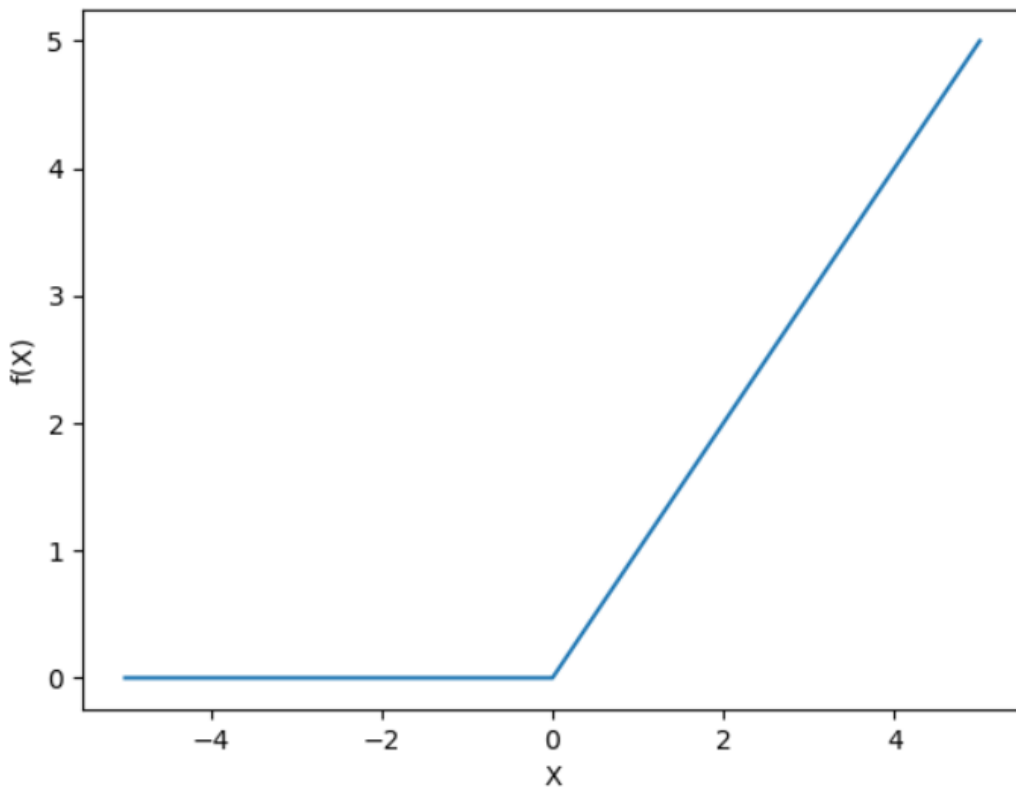


Figure 14: Graph of the ReLU activation function

During back-propagation, the derivative of this function is required for the weights to be updated. For ReLU the derivative of the function is 1.0 when x is larger than zero and 0.0 when x is less or equal to zero.

The convolutional layers uses different kernel sizes with some larger kernels in the beginning of the model and smaller kernels later in the model.

Meaning that each image will have a generic feature extraction in the first layers of the model and the convolutions sent through the network gets applied a more local feature extraction. The kernel size is also specified to reduce the dimensionality through the model so it ends up with the dimension (1x1x1xN). For kernel stride the default setting of (1,1,1) was used for a more detailed feature extraction of each image. As mentioned in Section 6, max pooling layers was used to represent the important voxels in the images during training. These were used in the beginning for finding the high value voxels early in the training, implemented in the second and fourth layer. A flatten layer was implemented before the dense layers to change the dimensionality to one-dimensional data that can be used in the dense layers. There are three dense layers, where each neuron takes all the input data from the previous layer creating a dense connection between layers. Two dropout layers were added at the end of the model to reduce overfitting. The final layer is a dense layer with four neurons with the softmax activation function. The softmax activation function is used in multiclass labelling and since the image can have four different labels this is used in the final layer with only four neurons to predict the label at the end of training.

$$\textit{Softmax}(x)_i = \frac{\exp x_i}{\sum \exp x_i} \quad (5)$$

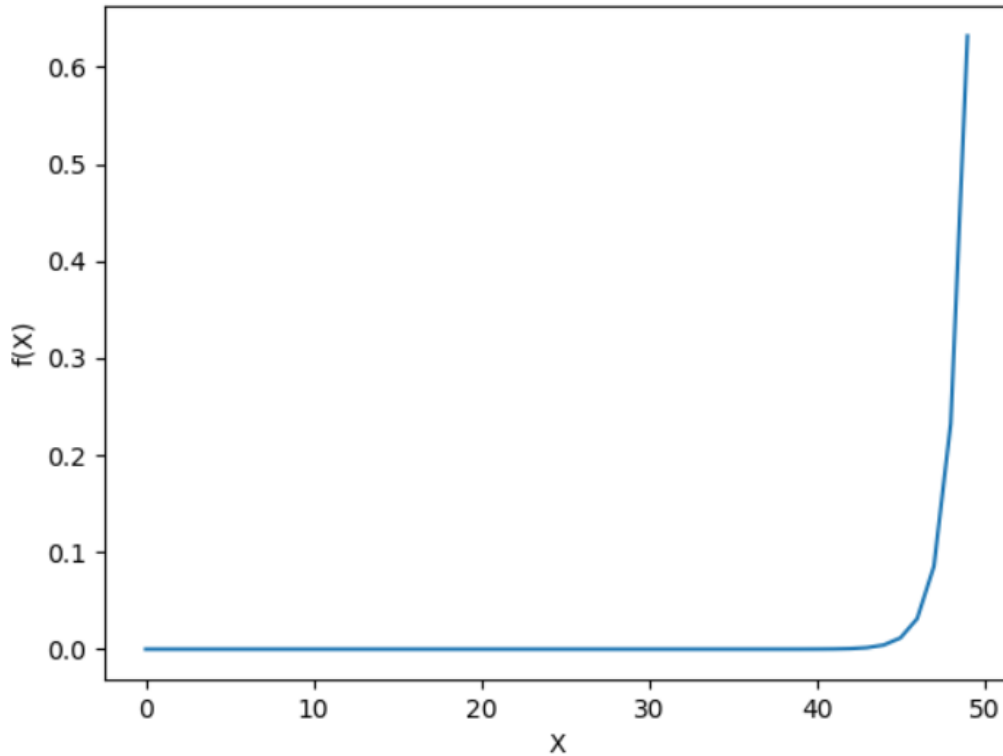


Figure 15: Graph of the softmax function with  $x$  between 0 and 50

A lot of time was put into tuning the parameters of the model. Parameters like kernel size, number of layers, and number of neurons were changed constantly during the development and testing of the model. Each change that were made to the network was done to increase the accuracy of the prediction of the classifier after training. In the beginning, the model had eleven layers and small kernel sizes of  $3 \times 3 \times 3$ , which did not produce the wanted results. The accuracy of the predictions at this point was around 50% or lower. The first thing that was changed for the kernel size was its dimensions so it could fit the image better. The images has a dimension of  $79 \times 95 \times 79$  and when applying a  $3 \times 3 \times 3$  convolutional filter the dimension of the image through the layers will end up a dimension of  $1 \times 6 \times 1 \times N$ . This was changed so that some of the filters had a kernel size of  $3 \times 4 \times 3$  to end up with the desired dimension.

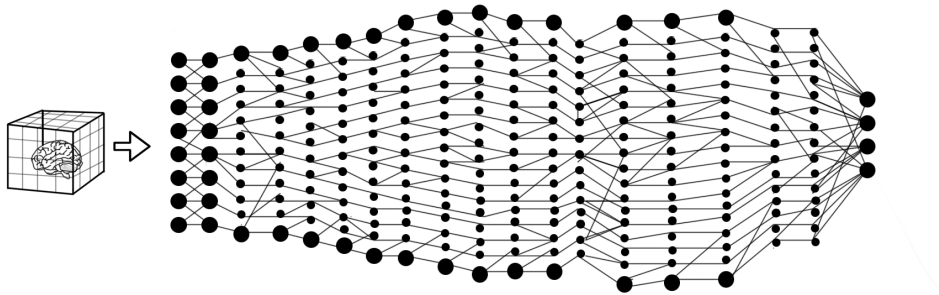


Figure 16: A representation of the implementation of the network

The last step in building the network is to compile the model. First, the optimizer is created. The optimizer that was chosen for this model is the optimizer RMSprop with a learning rate of 0.0001. The optimizer also contains a parameter called clipnorm which in this case is set to 1.0 and is used for gradient clipping. Next a loss function must be specified, which in this case ended up being the Kullback Leibler divergence loss function. Finally the optimizer with the loss function is compiled with the model and a metric for evaluation is chosen which in this case was chosen to be accuracy.

This last part of building the network went through a large amount of changes before the final draft of the model. Several different optimizers, learning rates, and loss functions were tested in combination with different combinations of the network structure. RMSprop ended up being the most reliable optimizer after some testing using the optimizer Adam and Stochastic Gradient Descent (SGD). Different values of learning rates were also tested from 0.1 to 0.0001. The smallest learning rate were chosen because of the complexity of the data. With a larger learning rate, the model might not find the features that are important. The decision of loss functions stood between Categorical Cross-Entropy, Sparse Categorical Cross-Entropy, and Kullback Leibler Divergence Loss, which all are used in multi-class classification. Kullback Leibler Divergence were chosen since it performed best during training during early iterations of the model. The clipnorm parameter specified in the

optimizer was crucial for making the artificial neural network model work. A problem occurred during training making the model stop learning after number of epochs. It was revealed that this was related to weights overflowing and to solve this issue gradient clipping was introduced using the clipnorm parameter specified in the optimizer. Gradient clipping is a method used to stop the gradient for taking large steps during updates that would result in the updated weights to either overflow or underflow. This is discussed in section 8.2 about the results from this project.

### 7.3 Network Model

In this subsection, the 3D-CNN model is presented visually and with a table of the network structure. The values in the table are color coded and the colored values in the table represent the arrows and layers in Figure 17

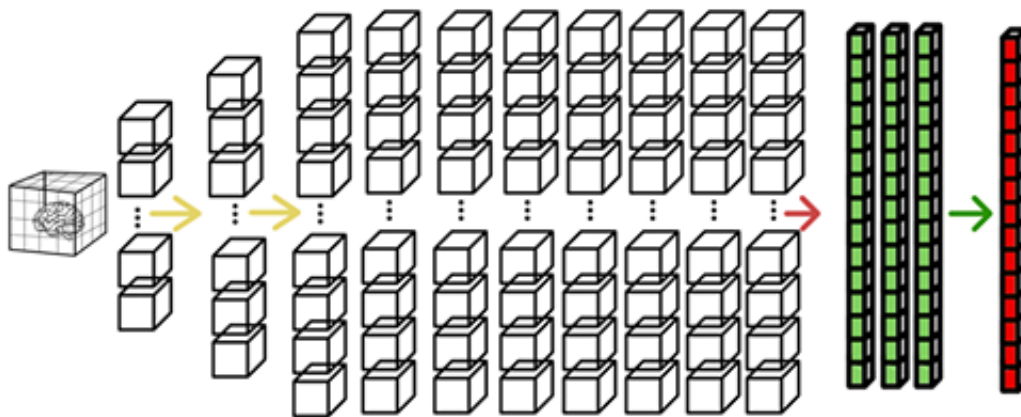


Figure 17: A representation of the final 3D-CNN-model

Layer nr:	1	2	3	4	5	6	7	8	9	
Layer type:	3D-CNN	3D-Max-pool	3D-CNN	3D-Max-pool	3D-CNN	3D-CNN	3D-CNN	3D-CNN	3D-CNN	
Kernel size:	7*8*7	3*3*3	5*6*5	2*2*2	3*4*3	3*4*3	2*2*2	2*2*2	2*2*2	
Activation f()	Relu		Relu		Relu	Relu	Relu	Relu	Relu	
Neurons:	8		16		32	64	128	254	508	
Layer nr:	10	11	12	13	14	15	16	17	18	19
Layer type:	3D-CNN	3D-CNN	3D-CNN	Flatten	Dense	Dense	Dense	Dropout	Dropout	Dense
Kernel size:	2*2*2	2*2*2	1*1*1							
Activation f()	Relu	Relu	Relu		Relu	Relu	Relu			Softmax
Neurons:	1016	508	508		1016	1016	1016			4

Figure 18: A table showing the different parameters in the 3D-CNN-model.

## 8 Results

The 3D-convolutional deep learning network classifier that was created in this project ended up predicting the cognitive neural activations and the default mode network with an accuracy of over 85%, which is a great result. In this section, the result from the project will be presented and discussed. First the result from early iterations will be discussed and problems that occurred during training. Next, methods to increase the accuracy will be presented. Lastly the final results will be presented and discussed.

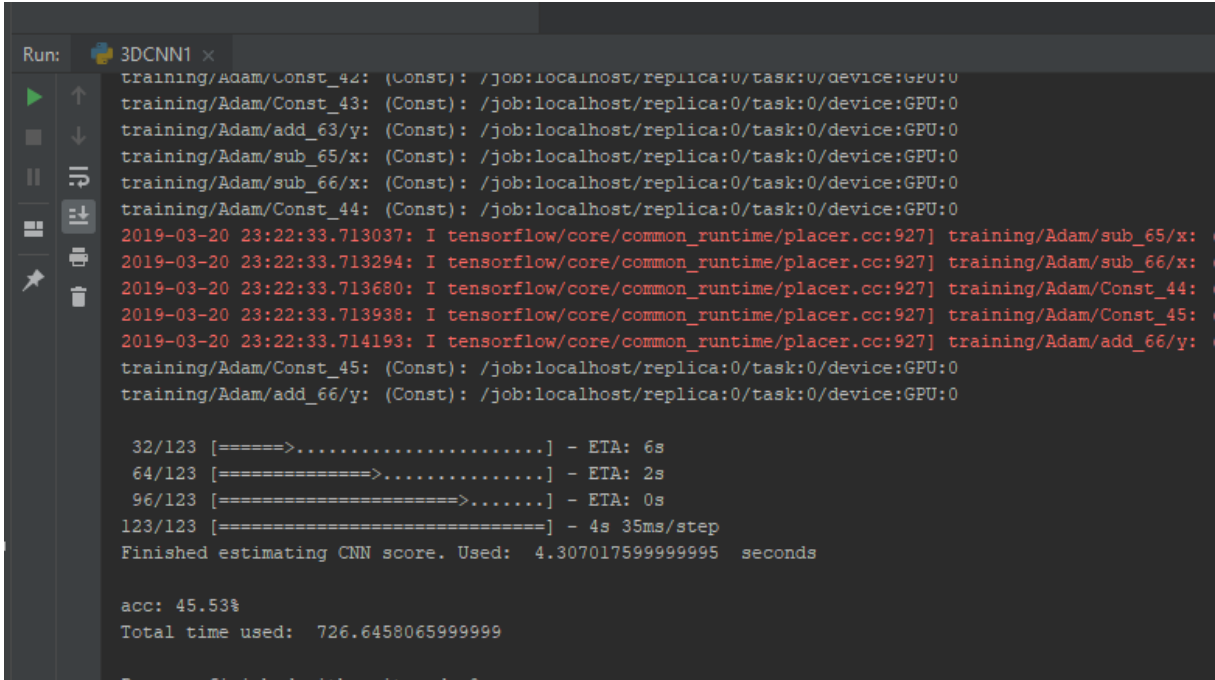
### 8.1 Results from Early Iterations

As mentioned in Section 7.2, early iterations of the deep learning network was not producing high prediction accuracies. It mostly produces accuracies between 45% and 55%. This was early in the project and during the construction phase of the network, so lower accuracies was expected. During this phase just a few data set were used to test the network. In the beginning, only three data set were used, corresponding to 918 images. Using a few data set was desirable in the beginning during the setup to minimize training and prediction time to figure out if the network was fully functioning. Also, as mentioned in Section 7.2, the data was very memory intensive and running just half of the data sets could be a problem for even some of the computers provided at MMIV.

Accuracies of about 50% are not great results. That the accuracies was in that area of prediction could be an indication of that the network was not learning anything useful from the data making the network predict randomly each time, which support the accuracy results of around 50%. Many changes



were made, most explained in Section 7.2 to make the network end up with a high accuracy of above 80%. The next subsection will go more in depth in the reason for these changes and the result that was this produced. Also in the early iteration of the neural network Tensorboard was not used, and most of the graphs and figures were created using the matplotlib-package in Python.



```
Run: 3DCNN1 x
training/Adam/Const_42: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/Const_43: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/add_63/y: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/sub_65/x: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/sub_66/x: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/Const_44: (Const): /job:localhost/replica:0/task:0/device:GPU:0
2019-03-20 23:22:33.713037: I tensorflow/core/common_runtime/placer.cc:927] training/Adam/sub_65/x:
2019-03-20 23:22:33.713294: I tensorflow/core/common_runtime/placer.cc:927] training/Adam/sub_66/x:
2019-03-20 23:22:33.713680: I tensorflow/core/common_runtime/placer.cc:927] training/Adam/Const_44:
2019-03-20 23:22:33.713938: I tensorflow/core/common_runtime/placer.cc:927] training/Adam/Const_45:
2019-03-20 23:22:33.714193: I tensorflow/core/common_runtime/placer.cc:927] training/Adam/add_66/y:
training/Adam/Const_45: (Const): /job:localhost/replica:0/task:0/device:GPU:0
training/Adam/add_66/y: (Const): /job:localhost/replica:0/task:0/device:GPU:0

32/123 [=====>.....] - ETA: 6s
64/123 [=====>.....] - ETA: 2s
96/123 [=====>.....] - ETA: 0s
123/123 [=====>.....] - 4s 35ms/step
Finished estimating CNN score. Used: 4.307017599999995 seconds

acc: 45.53%
Total time used: 726.6458065999999
```

Figure 19: The network predicting with 45% accuracy

```
Run: 3DCNN1 x
2019-03-20 22:44:10.47229: I tensorflow/core/common_runtime/placer.c
32/490 [>.....] - ETA: 32s
64/490 [==>.....] - ETA: 17s
96/490 [===>.....] - ETA: 12s
128/490 [====>.....] - ETA: 9s
160/490 [=====>.....] - ETA: 8s
192/490 [=====>.....] - ETA: 6s
224/490 [=====>.....] - ETA: 5s
256/490 [=====>.....] - ETA: 4s
288/490 [=====>.....] - ETA: 3s
320/490 [=====>.....] - ETA: 3s
352/490 [=====>.....] - ETA: 2s
384/490 [=====>.....] - ETA: 1s
416/490 [=====>.....] - ETA: 1s
448/490 [=====>...] - ETA: 0s
480/490 [=====>.] - ETA: 0s
490/490 [=====] - 9s 18ms/step
Finished estimating CNN score. Used: 8.895769999999999 seconds

acc: 51.43%
Total time used: 2e-07

Process finished with exit code 0
```

Figure 20: The network predicting with 51% accuracy

```

Run: 3DCNN1 x
18/918 [.....] - ETA: 1:26 - loss: 6.2681 - acc: 0.6111
20/918 [.....] - ETA: 1:25 - loss: 6.4472 - acc: 0.6000
22/918 [.....] - ETA: 1:25 - loss: 6.5938 - acc: 0.5909
24/918 [.....] - ETA: 1:25 - loss: 6.7159 - acc: 0.5833
26/918 [.....] - ETA: 1:25 - loss: 6.1993 - acc: 0.6154
28/918 [.....] - ETA: 1:25 - loss: 5.7565 - acc: 0.6429
30/918 [.....] - ETA: 1:24 - loss: 5.9100 - acc: 0.6333
32/918 [>.....] - ETA: 1:24 - loss: 6.0443 - acc: 0.6250
34/918 [>.....] - ETA: 1:24 - loss: 6.6369 - acc: 0.5882
36/918 [>.....] - ETA: 1:24 - loss: 7.1636 - acc: 0.5556
38/918 [>.....] - ETA: 1:24 - loss: 7.6349 - acc: 0.5263
40/918 [>.....] - ETA: 1:23 - loss: 7.6561 - acc: 0.5250
42/918 [>.....] - ETA: 1:23 - loss: 7.6753 - acc: 0.5238
44/918 [>.....] - ETA: 1:23 - loss: 7.3264 - acc: 0.5455
46/918 [>.....] - ETA: 1:23 - loss: 7.0079 - acc: 0.5652
48/918 [>.....] - ETA: 1:23 - loss: 7.0517 - acc: 0.5625
50/918 [>.....] - ETA: 1:22 - loss: 7.0920 - acc: 0.5600
52/918 [>.....] - ETA: 1:22 - loss: 6.8192 - acc: 0.5769
54/918 [>.....] - ETA: 1:22 - loss: 6.8651 - acc: 0.5741
56/918 [>.....] - ETA: 1:22 - loss: 6.9078 - acc: 0.5714
58/918 [>.....] - ETA: 1:22 - loss: 7.2254 - acc: 0.5517
60/918 [>.....] - ETA: 1:21 - loss: 7.5218 - acc: 0.5333
62/918 [=>.....] - ETA: 1:21 - loss: 7.7991 - acc: 0.5161
64/918 [=>.....] - ETA: 1:21 - loss: 8.0590 - acc: 0.5000
66/918 [=>.....] - ETA: 1:21 - loss: 8.0590 - acc: 0.5000

```

Figure 21: Showing the epoch during training in early iterations of the network

## 8.2 Increasing the Accuracy

First, I thought the reason for the low accuracy might be related to the amount of data used in training. Since only three data sets were used in early iterations of the network and 60% of the data was used in training most of the time (different split percentages was used during the setup of the network.). The fMRI BOLD-images are three-dimensional and therefore more complex than your standard two-dimensional image and might therefore need more data to predict the networks. To accomplish this the data structures used to store the data were changed to data structures with lower memory usage as mentioned in Section 7.2. It was now possible to use more data to train the network. On the computers at MMIV it was it was possible to run all the data from the project. However, the accuracy did not show any sign of improvement.

Another hurdle related to memory so far in the project was the size restriction on the network itself, since it to used up the memory on the machine. A large complex network was, in early iteration of the network, out of the question since this in turn made it difficult to process large amounts of data.

The batch size was also restricted to a small value, often either one or two because if any large value were used the system would run out of memory. Changing the data structures helped lifting some of these restrictions and made it possible to create a more complex network with larger batch sizes during training. The decisions was then made to have the network have more neurons and layers, and train for a longer time now with a larger batch size. This indeed seem to improve the accuracy.

An observation was made at this point that the network seemed to not learn from the data until several epochs in training. Starting off, it seemed to be about two-hundred epochs, with a batch size of one, before the accuracy increased. Using a large batch size, decreased the epochs needed. As above, the reason for this is most likely related to the complexity of the data. This is represented in figure 22 and 23.

The more complex neural network training on all the data was now able to produce high accuracy results. It was clear from the results and the graphs from training that the network was finally learning features from the data. During training at this stage, the epoch-accuracy had a value of almost 80%.

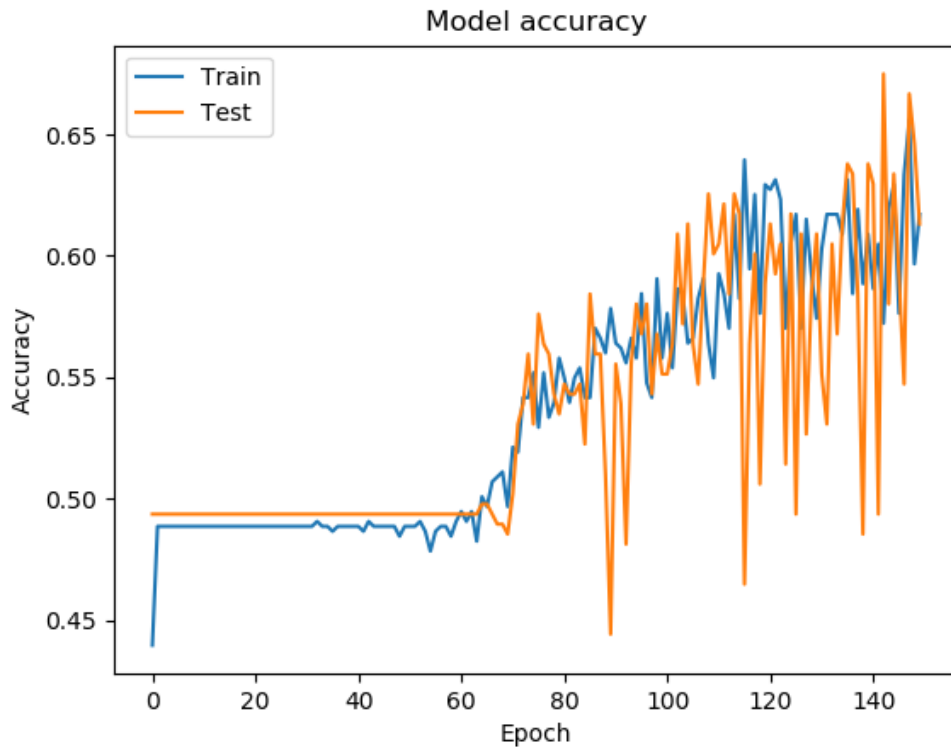


Figure 22: Graph of training during early iterations showing accuracy. The classifier learning slowly from the data

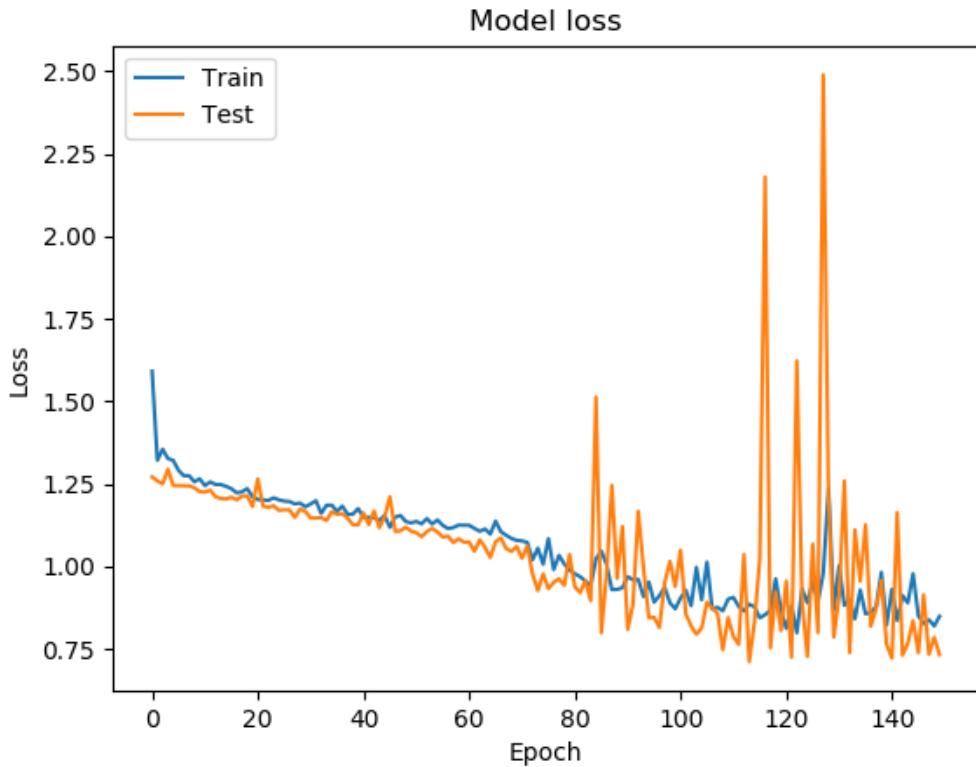


Figure 23: Graph of training during early iterations showing loss. The classifier learning slowly from the data

Getting almost 80% accuracy was a huge breakthrough in the development in the convolutional neural network and a percentage that was acceptable for the complex data. Nevertheless, one issue remained. After training for 300 epoch or more the convolutional network seemed to lose all progress during training and ending up flattening out around 50% accuracy. The network loss also increases substantially and flatten out on a high value.

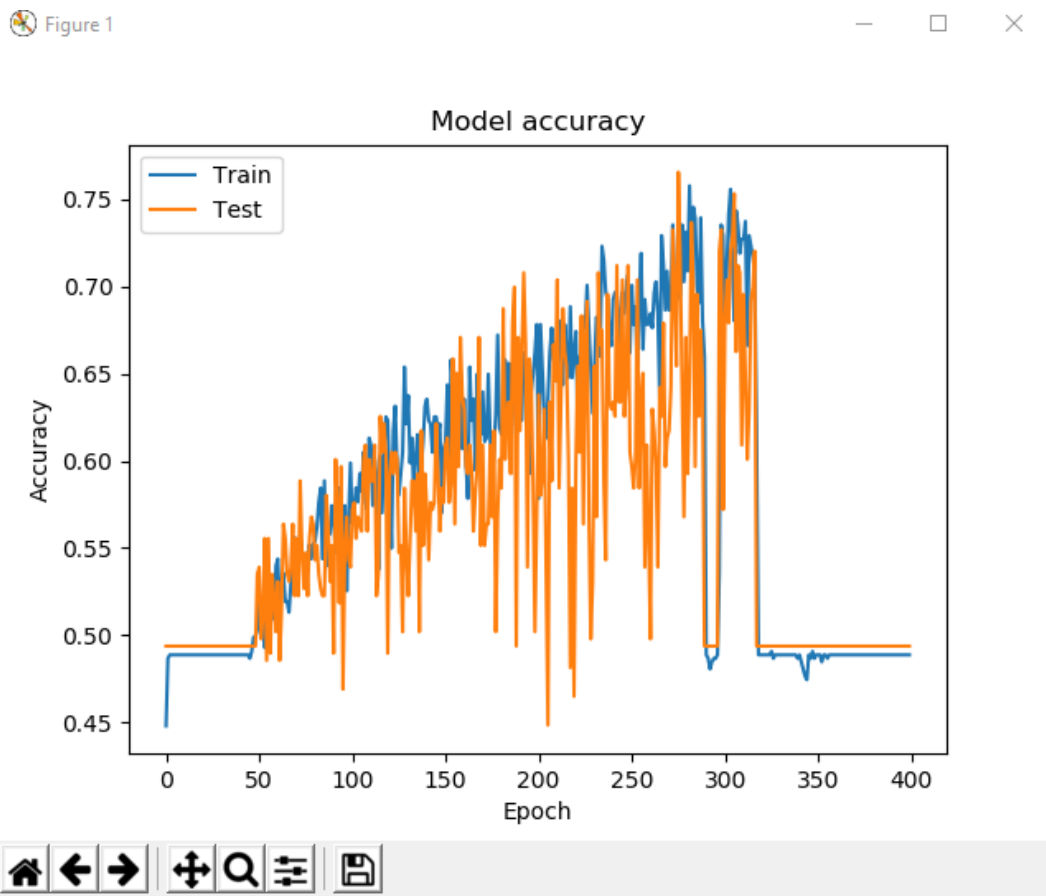


Figure 24: Graph of training accuracy with the curve flattening

Figure 1

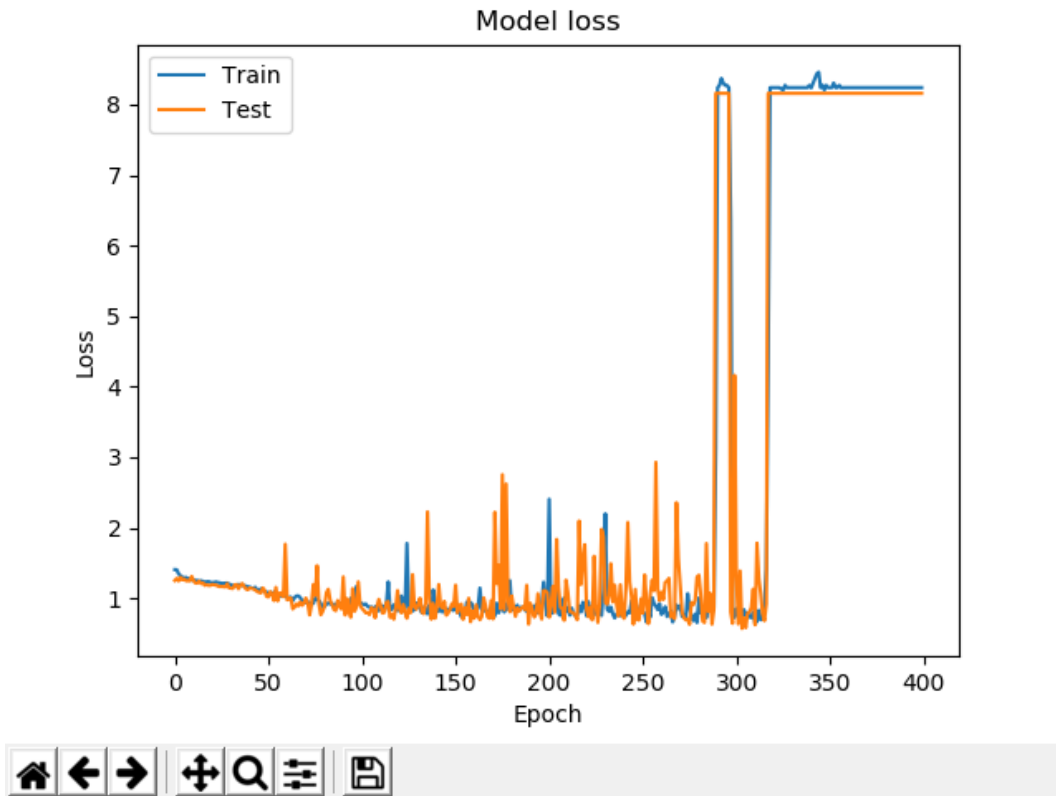


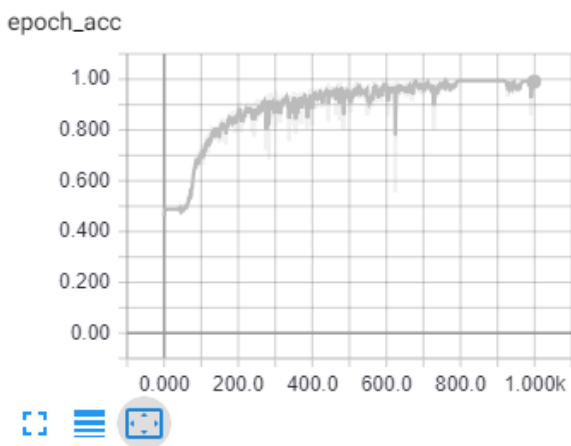
Figure 25: Graph of loss with the curve flattening

To better test this new undesirable behaviour the network model was tested with different epochs and different batch sizes to determine when the network model started failing. From the several runs of the model it seemed to be very fluctuating when the model started failing, but it was always after more than 100 epochs. It seemed the reason for this behaviour was due to gradient explosion. Gradient explosion happens when the error gradients in the network model becomes so large that the updates to the weights becomes become large and the values of the weights overflow. When the weights in the network overflows their value will become NaN and the weights are no longer useful. This seemed to fit the predicament that the model was facing since it out of nowhere lost its progress. To mitigate this the clipnorm parameter was added to the optimizer to prevent gradient explosion. The network model was now functioning perfectly and it was able to train on all the data sets without complications.



### 8.3 Final Results

With the fully functional classifier several training iterations was done to test the predictions accuracy of the network. In this final stage, the Tensorboard framework was used to get a better overview of the results in addition to a representation of a model of the final functional 3D-CNN. In the end, the deep three-dimensional neural network was able to predict the three different cognitive brain networks and the default-mode-network with over 85% accuracy consistently.



epoch\_loss

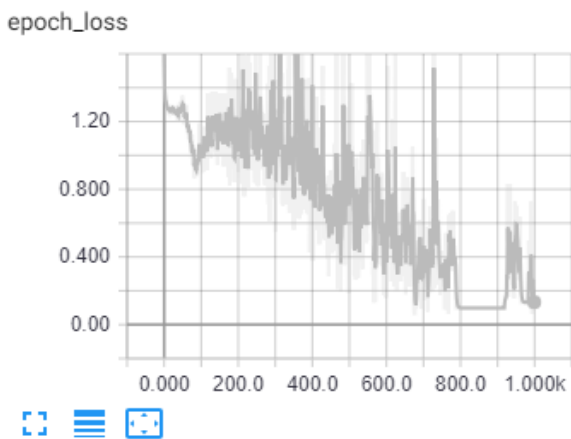
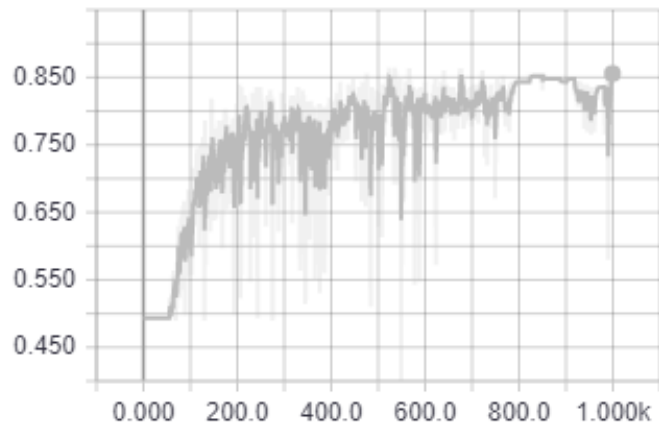


Figure 26: Graph of epoch accuracy and loss

epoch\_val\_acc

---

epoch\_val\_acc



epoch\_val\_loss

---

epoch\_val\_loss

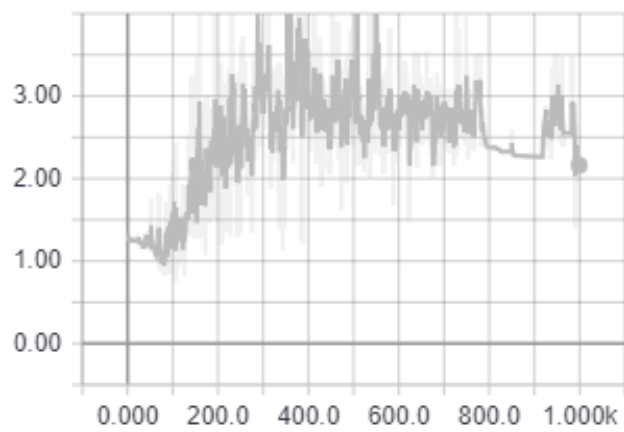


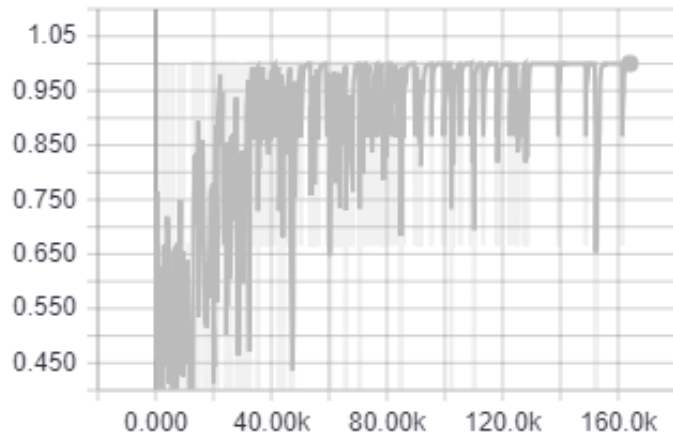
Figure 27: Graph of validation accuracy and loss

---

batch\_acc

---

batch\_acc



---

batch\_loss

---

batch\_loss

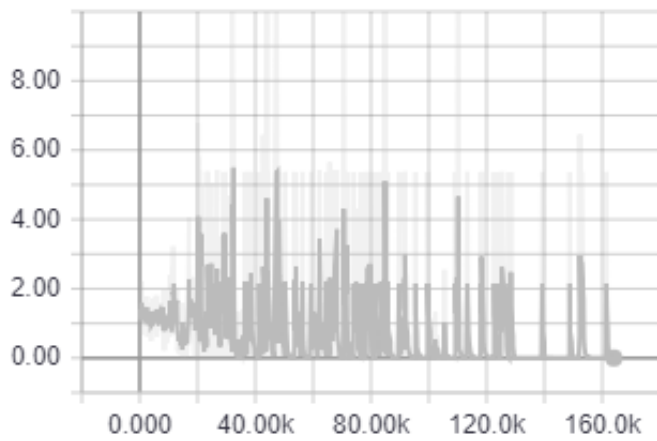


Figure 28: Graph of batch accuracy and loss

From Figure 26, 27, and 28, we can see an increase in accuracy values

compared to the accuracy in Figure 22. One of the best results from training even had over 90% accuracy. To be able to get this high of an accuracy on a complex fMRI-BOLD data set was something that was not thought of in the beginning of this project.

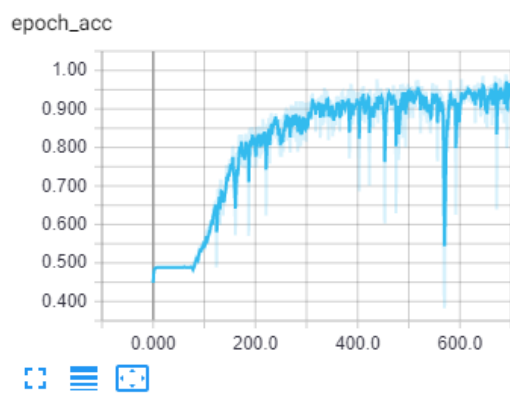
```
Finished estimating CNN score. Used: 6.667112999999517 seconds  
acc: 93.67%  
Total time used: 13878.509484799999
```

Figure 29: A high accuracy percentage acquired during training. Graph of epoch accuracy and loss during training of a classifier with over 90% accuracy

---

epoch\_acc

---



epoch\_loss

---

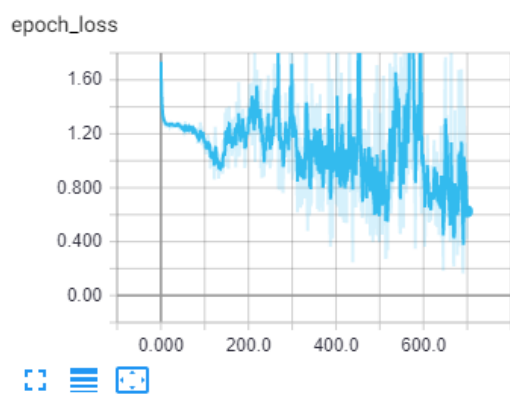
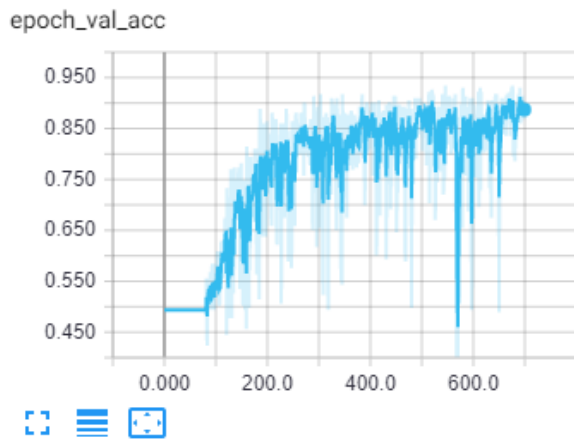


Figure 30: Graph of epoch accuracy and loss during training of a classifier with over 90% accuracy

---

epoch\_val\_acc

---



---

epoch\_val\_loss

---

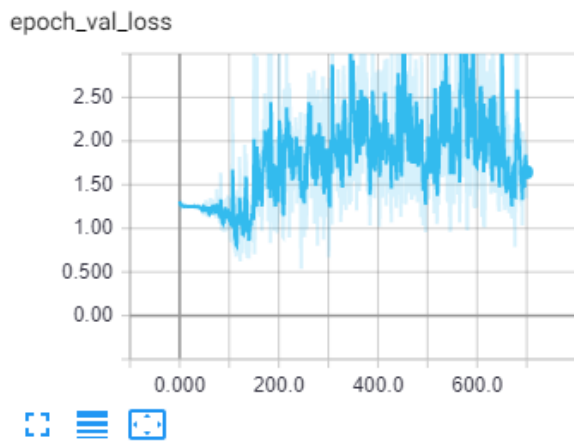
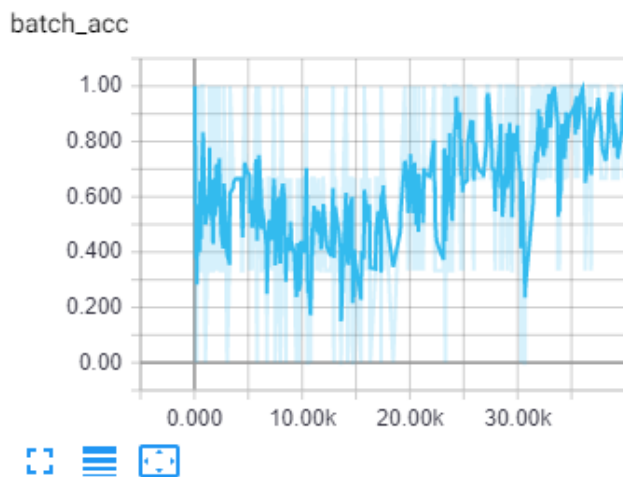


Figure 31: Graph of epoch validation accuracy and loss during training of a classifier with over 90% accuracy

batch\_acc



batch\_loss

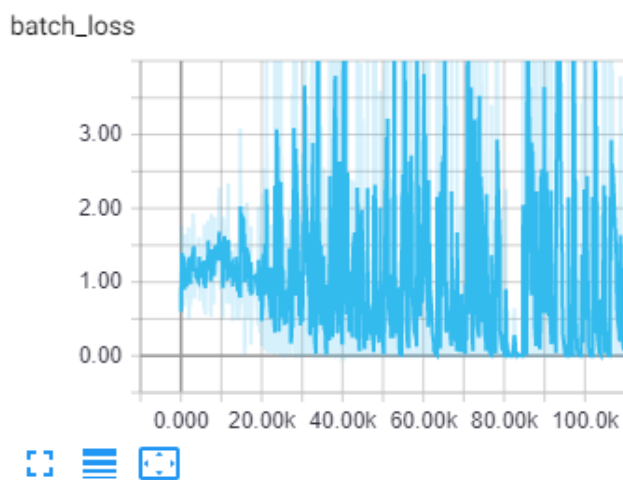


Figure 32: Graph of batch epoch accuracy and loss during training of a classifier with over 90% accuracy

As seen in figure 27, sometimes the loss during training increased, when the accuracy increased. This behaviour was not expected since it is common for the loss to increase in value when the accuracy increases. An explanation for this can be that the neural activations is not easily separable. When the models tries to predict a label for a data point it might be that there is a

low probability that it is the correct label. However, since the probability for the correct label to be predicted is higher than the others, it still predicts correctly. This will explain why the loss and the accuracy can be increasing at the same time.

Lastly, we will look at a confusion matrix taken from a training example of the 3D-CNN model using 906 neural images, with a 60% training split. The vertical columns represent the different classes and the horizontal columns represent the predictions. Each cognitive task and the default mode network is associated with a number.

- 0 = Mental Arithmetic
- 1 = Mental Rotation
- 2 = Default Mode Network
- 3 = Working Memory



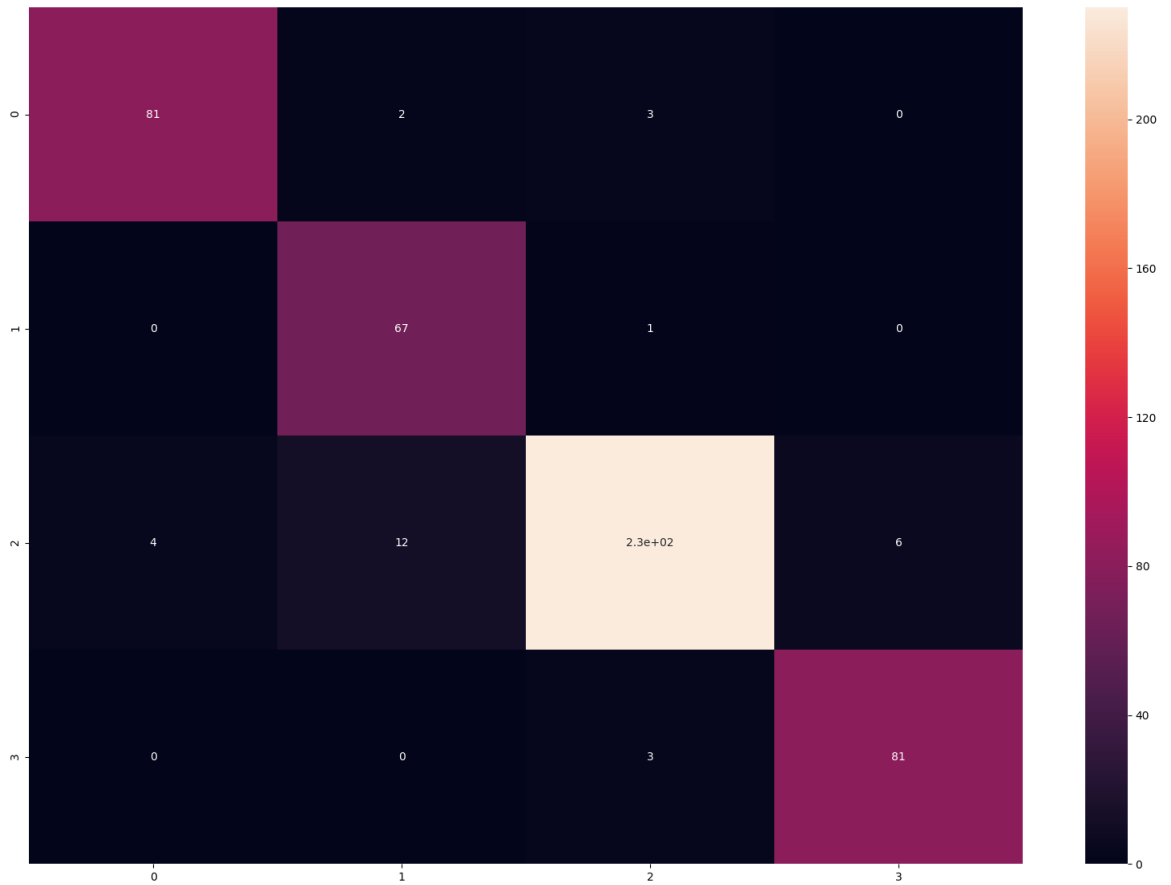


Figure 33: Confusion matrix of predictions done on the fMRI-BOLD images

## 8.4 Answering the Research Questions

In the Section 2.3, the research questions were formulated as follows: First, how do EMN and DMN interplay in each individual participant? Here the focus will be on each cognitive tasks in conjunction with the DMN and how separable they are.

Secondly, How can the experiment be optimized, i.e. shortened in time without the loss off accuracy- to develop an easy-to-include experiment for future clinical settings or multicentre studies. Here the focus will be on implementing a machine learning approach to the fMRI analysis. It was decided in the beginning of this project to use a three-dimensional convolutional neural network (3D-CNN) as the model for this implementation.

From the results above we can clearly see that the the cognitive neural activations are separable from each other, and the Default Mode Network. This is represented in figure 33, which answers the first research question.

Through this thesis we have looked at the implementation of the 3D-CNN from early iteration of the implementation to the final model. Ways of improving the performance of the model and increasing the accuracy have been discussed and the 3D-CNN suggested in this project can be easy to include in future studies on similar data, which answers the second question.

## 9 Conclusion

In this final section, further work that could be done in this project is discussed, and finally the conclusion of this thesis is presented.

### 9.1 Further Work

Working with deep learning on an fMRI project, as mentioned, can be very time consuming. Large part of the time put in to the project goes in to data collection, pre-processing, machine learning model selection, parameters tuning, training, and testing. Because of this there are several things that could been done in this project that, unfortunately, was not inside the time frame of this project. Given more time, the classifier constructed in this project would be tested against the extrinsic mode network (EMN). This could in turn support he argument for the existence of the EMN and inspire new approaches to identifying this network.

It could also be of interest to use the classifier on other cognitive or sensory tasks to observe how well the classifier is generalized. Doing so might also lead to observations on how to improve the network suggested in this thesis, or other network that might be constructed in the future.

The classifier should also be compared to standard methods of identifying brain activations at Haukeland. It would be interesting to see where, and if, the classifier suggested in this thesis would perform better or worse than the standard methods. In addition, the 3D-CNN classifier could be compared and evaluated against other similar method using NiftyNet[3] or DeepMedic[4].

Other steps during pre-processing could also be tested. During development, using histogram equalization or other similar contrast methods was discussed. As mentioned in Section 5, pre-processing is a crucial part in differentiating the different areas that we are interested in, and could therefore have a direct effect in improving the accuracy of the classifier.

## 9.2 Conclusion

From this project, we have showed that 3D convolutional neural network can be applied efficiently to BOLD neural images from an fMRI analysis. The classifier suggested in this thesis have proven to be able to differentiate and classify different cognitive brain activation with a high accuracy percentage. The applications for the results found in this thesis could be applied and built upon in several other scenarios. It can be a great tool for fMRI analysis in general, and with more testing can lead the way for new approaches for this type of analysis. In additions to this, better understanding of 3D convolutional neural networks on complex 3D images given in this thesis could be applied and extended upon in other project with a machine learning approach, and can serve as a comparison to similar projects. As mentioned, even though several changes could be made to the classifier, and more research should be done related to this thesis, the work on this thesis is completed.

## 9.3 Classifier

The classifier that was implemented in this thesis can be found on GitHub: <https://github.com/eivindKo/3DCNN>

## References

- [1] Hugdahl,K., Raichle,M.E., Mitra, A., Specht,K. *On the existence of a generalized non-specific task-dependent network*, "<https://www.frontiersin.org/articles/10.3389/fnhum.2015.00430/full>" August 2015 Last accessed 02.06.2019
- [2] Huettel,S.A., Song,A.W., McCarthy,G. *Functional Magnetic Resonance Imaging* Some publisher 2nd edition 2008
- [3] Gibson, E. Li, W. Sudre, C. Fidon, L. Shakir, D. Wang, G. Eaton-Rosen, Z. Gray, R. Doel, T. Hu, Y. Whyntie, T. Nachev, P. Modat, M. Barrat, D. Ourselin, S. Cardoso, M.J. Vercauteren, T. *NiftyNet: a deep-learning platform for medical imaging*, "<https://www.sciencedirect.com/science/article/pii/S0169260717311823>" January 2018, Last accessed 02.06.2019
- [4] Kamnitsas,K. Ledig,C. Newcombe,V. Simpson,J. Kane,A. Menon,D. Rueckert,D. Glocker,B. *Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation*, "<https://www.sciencedirect.com/science/article/pii/S1361841516301839>" October 2016, Last accessed 02.06.2019
- [5] Vu,H. Kim,H. Lee,J. *3D convolutional neural network for feature extraction and classification of fMRI volumes*, "[https://www.researchgate.net/publication/326796284\\_3D\\_convolutional\\_neural\\_network\\_for\\_feature\\_extraction\\_and\\_classification\\_of\\_fMRI\\_volumes](https://www.researchgate.net/publication/326796284_3D_convolutional_neural_network_for_feature_extraction_and_classification_of_fMRI_volumes)" June 2018 Last accessed, 02.06.2019
- [6] Marshland,S *Machine learning An Algorithmic Perspective* Chapman & Hall/CRC 2nd edition October 8 2014
- [7] Berger,A. *Magnetic resonance imaging*, "<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1121941/>" January 2001 Last accessed, 02.06.2019
- [8] Pooja. Sharma,Aa. Sharma,An, *Machine Learning A Review of Techniques of Machine Learning* "[https://www.researchgate.net/publication/329609597\\_Machine\\_Learning\\_A\\_Review\\_of\\_Techniques\\_of\\_Machine\\_Learning](https://www.researchgate.net/publication/329609597_Machine_Learning_A_Review_of_Techniques_of_Machine_Learning)", December 2018 Last accessed 02.06.2019
- [9] Abadi,M. Agarwal,A. Barham,P. Brevdo,E. Chen,Z. Citro,C. Corrado,G. 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. Mane, 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. Viegas, F. Vinyals, O. Warden, P. Wattenberg, M. Wicke, M. Yu, Y. Zheng, X. *TensorFlow Large-Scale Machine Learning on Heterogeneous Distributed Systems*, November 2015 Last accessed 02.06.2019
- [10] *Keras: The Python Deep Learning library*, "<https://keras.io/>" Last accessed 02.06.2019
- [11] *Using GPUs*, "[https://www.tensorflow.org/guide/using\\_gpu](https://www.tensorflow.org/guide/using_gpu)" Last accessed 02.06.2019
- [12] *Develop, Optimize and Deploy GPU-accelerated Apps*, "<https://developer.nvidia.com/cuda-toolkit>" Last accessed 02.06.2019
- [13] *NVIDIA cuDNN*, "<https://developer.nvidia.com/cudnn>" Last accessed 02.06.2019
- [14] *Mohn Medical Imaging and Visualization Centre*, "<https://mmiv.no/>" Last accessed 02.06.2019
- [15] Hermans, E.J. *SPM12 Starters' Guide*, "<https://www.sciencedirect.com/science/article/pii/S1361841516301839>" 2002-2016. Last accessed, 02.06.2019
- [16] Andronache, A. Rosazza, C. Sattin, D. Leonardi, M. D'Incerti, L. Minati, L. *Impact of functional MRI data preprocessing pipeline on default-mode network detectability in patients with disorders of consciousness*, "<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3749435/>" August 2013. Last accessed, 02.06.2019
- [17] Brownlee, J. *A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks*, "<https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>", January 2019. Last accessed, 02.06.2019
- [18] Shiruru, K. *An Introduction to Artificial Neural Networks*, "[https://www.researchgate.net/publication/319903816\\_AN\\_INTRODUCTION\\_TO\\_ARTIFICIAL\\_NEURAL\\_NETWORK](https://www.researchgate.net/publication/319903816_AN_INTRODUCTION_TO_ARTIFICIAL_NEURAL_NETWORK)", September 2016. Last accessed, 02.06.2019

- [19] Shealand,K. Nash,R *An Introduction to Convolutional Neural Networks*, "[https://www.researchgate.net/publication/285164623\\_An\\_Introduction\\_to\\_Convolutional\\_Neural\\_Networks](https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks)", November 2015. Last accessed, 02.06.2019