




Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Information Technology – Robotics and Signal Processing	Spring semester, 2019 Open/ Confidential
Author: Simen Norrheim Larsen	 (signature of author)
Programme coordinator: Trygve Eftestøl Supervisor(s): Postdoc researcher Ketil Oppedal	
Title of master's thesis: Data-assisted differential diagnosis of dementia by deep neural networks	
Credits: 30	
Keywords: Deep Learning, Deep Neural Networks, Convolutional Neural Networks, Dementia Classification, Alzheimer's Disease, Dementia with Lewy Bodies	Number of pages: 73 + supplemental material/other: 6 Stavanger, 29 July 2019

Title page for Master's Thesis
Faculty of Science and Technology



Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Data-assisted differential diagnosis of dementia by deep neural networks

Master's Thesis in Robotics and Signal Processing

by

Simen Norrheim Larsen

Supervisors

Ketil Oppedal

Trygve Eftestøl

June 29, 2019

Abstract

There are currently 50 million people suffering from dementia worldwide. With an increasing life expectancy of the elderly, this number is expected to increase drastically over the next decade. With today's diagnosis of dementia being highly dependent on the expertise of clinical personnel, there is thus a pressing need for readily available and reliable diagnosis systems.

In this thesis, the potential for computer-aided diagnosis systems based on deep neural networks and structural magnetic resonance imaging is investigated. An ensemble model was designed and trained on 690 class-balanced brain scans for the differentiation of subjects diagnosed with Alzheimer's disease and dementia with Lewy bodies, as well as normal control subjects. All scans were initially skull-stripped and spatially normalized to remove unwanted information.

A final accuracy of 71.9% was reported for the three class differentiation of 171 test subjects. The results presented in this thesis fall a little short in comparison with those of related work, but indicates, nonetheless, a potential for this type of deep learning-based diagnosis systems.

Acknowledgements

This thesis marks the end of my Master's degree in Robotics and Signal processing at the University of Stavanger. I want to thank the lecturers, personnel, and co-students at the University of Stavanger for two exciting years filled with new knowledge, challenges, and good memories. I would also like to give a special thanks to Trygve Eftestøl and, in particular, my primary supervisor, Ketil Oppedal, for their valuable advice and feedback during this last semester. A final thank goes out to Theodor Ivesdal for his technical support concerning the use of the university's UNIX-system.

Contents

Abstract	ii
Acknowledgements	iv
Abbreviations	viii
1 Introduction	1
1.1 Dementia	1
1.2 Alzheimer’s disease	2
1.3 Dementia with Lewy bodies	3
1.4 Motivation	4
1.5 Thesis objective	5
1.6 Deep learning and neuroimaging	5
1.7 Thesis outline	6
2 Background	8
2.1 Magnetic resonance imaging	8
2.1.1 MRI markers	9
2.2 Preprocessing	11
2.2.1 Spatial normalization	12
2.2.2 Brain extraction	12
2.2.3 Data normalization	12
2.3 Artificial neural networks	13
2.3.1 Multi-layer perceptron	13
2.3.2 Feedforward neural networks	15
2.3.3 Convolutional neural networks	16
2.3.4 Pooling	18
2.3.5 Loss functions	19
2.3.6 Backpropagation	20
2.3.7 Optimizers	20
2.3.8 Activation functions	23
2.3.9 Overfitting	25
2.3.10 Regularization	28

2.3.11	Data augmentation	31
2.3.12	Batch normalization	31
2.3.13	Hyperparameter tuning	32
2.3.14	Evaluation metrics	34
2.4	Software	36
2.4.1	Pytorch	36
2.4.2	SciKit-learn	36
2.4.3	Docker	37
2.4.4	NiPype	37
3	Materials and method	39
3.1	Dataset construction	39
3.2	Preprocessing implementation	42
3.3	Model design	44
4	Experiments and Results	46
4.1	Experiments	46
4.1.1	Experimental layout	48
4.1.2	Experiment - Learning rate	49
4.1.3	Experiment - Dropout	49
4.1.4	Experiment - Bayesian optimization	51
4.1.5	Experiment - Augmentaion	52
4.2	Final evaluation	55
5	Discussion	57
5.1	Dementia classification	57
5.2	Limitations	59
5.2.1	Dataset	59
5.2.2	Preprocessing	60
5.2.3	Classifier	60
6	Conclusion and future directions	62
6.1	Conclusion	62
6.2	Future directions	63
	List of Figures	63
	List of Tables	65
A	Python code	66
A.1	fit.py	66
A.2	main_setup.py	67
A.3	system_resources.py	67
A.4	test.py	68

A.5	<code>data_resources.py</code>	69
A.6	<code>NormalizeSkullStripPipeline.py</code>	69
Bibliography		70

Abbreviations

AD	A lzheimer's D isease
DLB	D ementia with L ewy B odies
NC	N ormal C ontrol
CAD	C omputer- A ssisted D iagnosis
MRI	M agnetic R esonance I maging
sMRI	structural M agnetic R esonance I maging
MTL	M edial T emporal L obe
ML	M achine L earning
DL	D eep L earning
ANN	A rtificial N eural N etwork
FC	F ully C onnected
DNN	D eep N eural N etwork
CNN	C onvolutional N eural N etwork
SVM	S upport V ector M achine
SD	S tandard D eviation
GPU	G raphical P rocessing U nit
GD	G radient D escent
SGD	S tochastic G radient D escent
ReLU	R ectified L inear U nit
CV	C ross- V alidation
BN	B atch N ormalization
SPM	S tatistical P arametric M apping
FSL	F MRIB S oftware L ibrary
E-DLB	E uropean D ementia with L ewy B odies consortium

ADNI Alzheimer's Disease Neuroimaging Initiative

Chapter 1

Introduction

1.1 Dementia

Dementia is a collective name for progressive brain syndromes that affect memory, thinking, behavior, and emotion[1]. There are many risk factors associated with dementia and with age being the strongest among them, the increase in life expectancy for the elderly[2] causes the number of people affected by dementia to increase drastically. Currently, around 50 million people have dementia worldwide, with nearly 10 million new cases every year. The total number of people living with dementia is projected to reach 82 million in 2030 and 152 million in 2050[3]. The economic strain on society caused by dementia was in 2015 calculated to 818 million USD, about 1,1% of the global gross domestic product. This number was projected to reach 1 trillion USD by 2018 and 2 trillion USD by 2030[4].

The condition is very demanding for the patients themselves, their families, and their caretakers. The psychological, and emotional stress dementia inflicts on families can be severe and may lead to the need for additional assistance from public health-, social-, financial-, and legal systems. There currently exists no cure for dementia[3].

1.2 Alzheimer's disease

Alzheimer's disease (AD) was first described by Alois Alzheimer in 1906 and is today the most commonly diagnosed form of dementia, accounting for 50% to 70% of all cases[5]. The disease causes nerve cells to die, resulting in symptoms such as loss of short and eventually long term memory, degraded ability

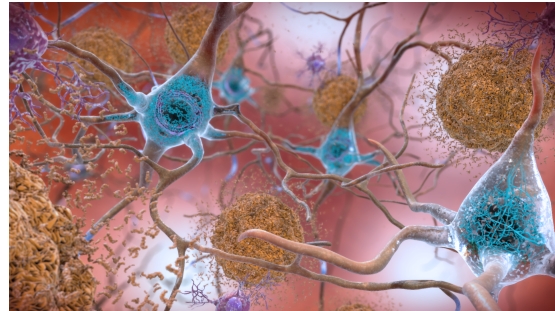



Figure 1.1: Plaque and tangles¹

to communicate, and compromised spatial and temporal orientation[6]. Alzheimer's disease is irreversible and manifests in the brain long before any cognitive problems appear[7]. The condition gradually worsen until symptoms are clear enough for a diagnosis to be made and does not stop until the patient dies, normally four to ten years after initial diagnosis[6].

The substantially higher buildup of plaque and tangles in the brains of people suffering from AD is suspected to be causing the death of nerve cells associated with the disease. The plaque is deposits of a protein fragment called beta-amyloid and builds up in the space between nerve cells. The tangles are twisted fibers of a protein called tau which builds up inside the cells themselves. Figure 1.1 illustrates how plaque and tangles manifests in the brain. Exactly how these buildups affects the brain is not known, but scientists believe that they block signals between the nerve cells and disrupts processes needed for the cells to survive[8].

There are no known causes for AD, but common risk factors are old age and degraded hearing, as well as those associated with cardiovascular disorders such as high blood pressure, diabetes, obesity, and high cholesterol. Age is the most dominant risk factor, with 3% of people above 65 and 12% to 15% of people above

¹ This work is by the National Institute on Aging, NIH, and is licensed under a [Creative Commons Attribution-NonCommercial 2.0 Generic License](https://creativecommons.org/licenses/by-nc/2.0/).

80 being affected by the disease. Age is a consistent factor as well, with 95% of all people suffering from AD being more than 65 years old[6].

Diagnosis is considered to be reasonably certain when indicative symptoms are present and typically consist of a multitude of clinical tests, including cognitive tests, neuroimaging, and blood analysis[6].

1.3 Dementia with Lewy bodies

Behind AD and vascular dementia, Dementia with Lewy bodies(DLB) is estimated to be the third most common form of dementia accounting for 5% to 10% of all cases[9]. In 1912, Friederich Lewy was working in Dr. Alois Alzheimer's lab when he discovered alpha-synuclein protein buildups to be disrupting the normal brain function of people with Parkinson's disease[10]. These buildups are today known as Lewy-bodies, shown in Figure 1.2. Lewy bodies are not only found in the brains of people with DLB but also other brain disorders like AD and Parkinson's disease dementia[9].

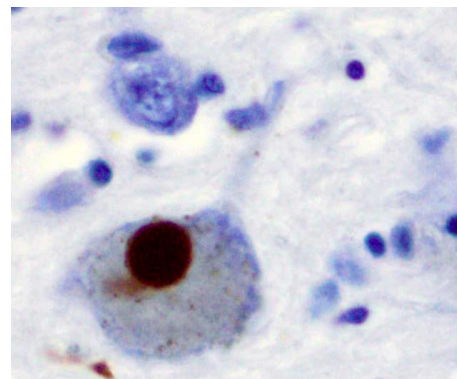


Figure 1.2: Intraneural Lewy-body²

A large part of DLB diagnosis is the differentiation to other cognitive diseases, as there might be multiple overlapping symptoms. Early-stage diagnosis is especially hard as initial symptoms of DLB resemble those of AD, causing DLB to be frequently misdiagnosed or even missed altogether. A total of 10% to 15% of autopsy reports on dementia patients describe DLB pathology. These reports also indicate that comorbidity is a problem, showing that the pathological changes associated with DLB might show up together with those of AD[11].

² This work is from Wikimedia Commons and is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). Author: Marvin 101

Getting a correct initial diagnosis of patients suffering from DLB is essential as treatment and care differs from that of AD. Prescribing the correct medication for DLB patients is especially vital as some antipsychotic drugs occasionally prescribed to AD patients can give serious side effects like sudden changes in consciousness, impaired swallowing, acute confusion, episodes of deletion, and appearance or worsening of Parkinson's syndrome³for people with DLB[9]. Today, diagnosis is performed clinically by looking at symptoms, conducting physical examinations and cognitive tests. Known risk factors of DLB are being older than 60, male, and having family members with DLB or Parkinson's disease[12].

1.4 Motivation

Both AD and DLB are, as mentioned, currently diagnosed clinically. Diagnosis relies on a set of tests and evaluations, and the quality of diagnosis is thus dependent on the procedures used and the experience of the medical personnel. Misdiagnosis of dementia may lead to delayed treatment, complications, and low quality of life for those affected. It is also unfortunately not uncommon[14, 15]. The financial savings of correct early diagnosis have been estimated to be quite substantial for both patients and healthcare systems as well[16].

Deep learning(DL) techniques such as convolutional neural networks(CNNs) have, in recent years, gained much attention for their high performance in solving computer vision tasks such as object detection and image classification[17]. With nonintrusive medical imaging techniques such as magnetic resonance imaging(MRI) being commonly available, DL-based computer-assisted diagnosis(CAD) systems could potentially contribute to increased reliability of clinical diagnosis. Such CAD systems may also free up precious time for experts and provide better assessments for institutions lacking thereof.

³Parkinsonian symptoms include tremors, slowed movement, rigid muscles, impaired posture and balance, degraded ability to perform unconscious movements like blinking, smiling etc. and speech and writing changes[13]

1.5 Thesis objective

The objective of this thesis is to explore the potential of deep neural network(DNN)-implementations for structural MRI(sMRI⁴)-based computer-assisted differential diagnosis of subjects suffering from AD or DLB, as well as normal control(NC) subject.

1.6 Deep learning and neuroimaging

Many different types of machine learning(ML) techniques have been proposed and investigated for dementia classification, with most being developed for differential diagnosis of subjects suffering from AD or mild cognitive impairment, as well as NC subjects[18, 19].

Different variations of the support vector machine(SVM) have been widely researched and reported to deliver promising performance for dementia classification[20–23]. However, most of the proposed SVM-based implementations rely on gathering features either by hand or through extensive preprocessing. These processes of feature extraction can be both time-consuming and require expertise. The quality of handcrafted features is also dependent on the person performing the procedure, which introduces variability to the overall system and limits reproducibility. The fact that these features are predetermined might also be limiting the performance of these methods[24].

In 2012, A. Krizhevsky et al.[25] illustrated the promising potential of CNNs for computer vision tasks. The ability for these networks to learn task-specific image features makes them good candidates for various neuroimaging tasks. The technology has therefore gained popularity in the neuroimaging community over the last few years. Promising results in dementia classification has been made using multiple types of DL techniques on different image modalities[18, 19]. The

⁴The term structural magnetic resonance imaging refers to the use of a standard MRI technique without the use of contrast.

performance of DNNs is, however, generally dependent on access to large amounts of data, which in the case neuroimaging often is limited[18, 26].

Much less effort has been directed towards MRI-based differential diagnosis of different types of dementia and DLB diagnosis. Two papers using quite different approaches showed, however, promising performance for MRI-based differential diagnoses of subjects suffering from AD or DLB, and NC subjects. Wada et al.[27] proposed a six-layer CNN for classification of MR connectome maps[28], and Oppedal et al.[29] used texture analysis of MRI scans and a random forest classifier.

1.7 Thesis outline

Chapter 2 - Background

Chapter two will give an introduction to relevant background theory for this thesis. MRI-based biomarkers, artificial neural networks, data preprocessing, and utilized software will be reviewed.

Chapter 3 - Materials and method

This chapter will cover the data used in this study, as well as how it is handled and processed. The model proposed for image classification is presented in section 3.3.

Chapter 4 - Experiments and results

Experiments and results are reported chronologically in this chapter as conducted experiments are based on the results from the previous experiments. A final model is evaluated in section 4.2.

Chapter 5 - Discussion

In this chapter, the results from chapter four will be discussed and compared to related work. Limitations of this work will be covered in section [5.2](#).

Chapter 6 - Conclusion and future directions

Chapter six will conclude this thesis and propose directions for future research.

Chapter 2

Background

In this chapter, relevant background theory for this thesis will be introduced.

2.1 Magnetic resonance imaging

In order to examine and differentiate the brains of subjects suffering from different types of dementia in a noninvasive way, the medical imaging technique known as magnetic resonance imaging, will be utilized. In this section, this commonly available technology will be presented, along with some biomarkers of dementia that these images might display.

MRI was first tested on a living human in 1976 by Peter Mansfield[30]. Almost a decade later, in 1984, the first MRI scanner was implemented for clinical use. These machines have since then played a large part in the field of medical imaging[31].

MRIs are acquired by utilizing the magnetic properties of the hydrogen nucleus. Hydrogen was chosen as it is found in abundance throughout the body. As hydrogen has one single proton in its nucleus, each nucleus has one magnetic north and south pole which is aligned with its spinning axis. Under normal circumstances, all the axes of the body's hydrogen nuclei are oriented randomly. During an MRI scan, the subject's body is exposed to a strong magnetic field causing these axes to align,

creating a magnetic vector along the main magnetic field of the MRI scanner. By introducing a radio wave to this magnetic field, these axes begin to resonate about the magnetic axis of the scanner. Incrementally small changes in the magnetic field throughout the scanned area cause the axes to resonate at different frequencies. When the radio wave is switched off, the axes of the nuclei return to align with the axis of the main magnetic field. During this process, the axes will resonate, causing the nuclei to emit energy in the form of radio waves. These waves are picked up by the machine, interpreted, and processed to produce magnetic resonance images[32].

2.1.1 MRI markers

Diagnosis of both AD and DLB is, as mentioned today performed clinically, and there has yet to be determined a single set of pathological markers good enough for definitive diagnosis[33, 34]. This section will present some proposed biomarkers which might be perceivable on sMRI scans and potentially could be utilized for dementia diagnosis. Coronal images of subjects diagnosed with AD or DLB are shown in Figure 2.4, as well as an NC subject.

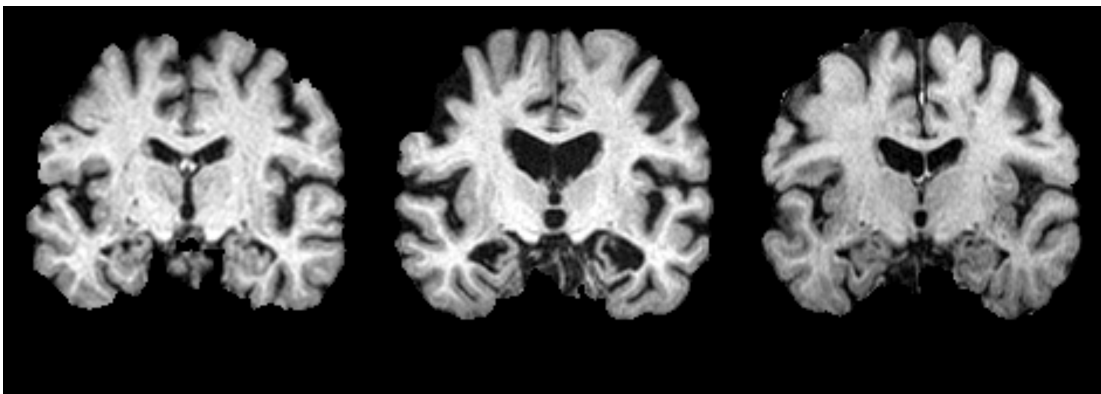


Figure 2.1: NC

Figure 2.2: AD

Figure 2.3: DLB

Figure 2.4: Brains of subjects belonging to NC, AD or DLB. All subjects are of about the same age

Common for the markers presented below is that they are all based on cerebral atrophy. Atrophy is, by definition, the loss of cells and in the case of cerebral atrophy, the loss of nerve cells and their connections[35]. This loss of nerve cells is

as mentioned a characteristic of dementia and might be observed as a loss of brain matter or volume on an sMRI. Most mentioned regions are visualized in Figures 2.5 and 2.6.

Alzheimer's disease

AD is characterized by causing widespread atrophy throughout the brain. Some regions are, however, more distinctive for AD than other neurological disorders. One of these regions is the medial temporal lobe(MTL)[36], with the hippocampus being one of the most prominent and earliest occurring AD markers[37]. Atrophy of the hippocampus has, in fact, been projected to diverge from atrophy caused by normal aging before the age of 40. The lateral ventricles and the amygdala diverges shortly thereafter[38].

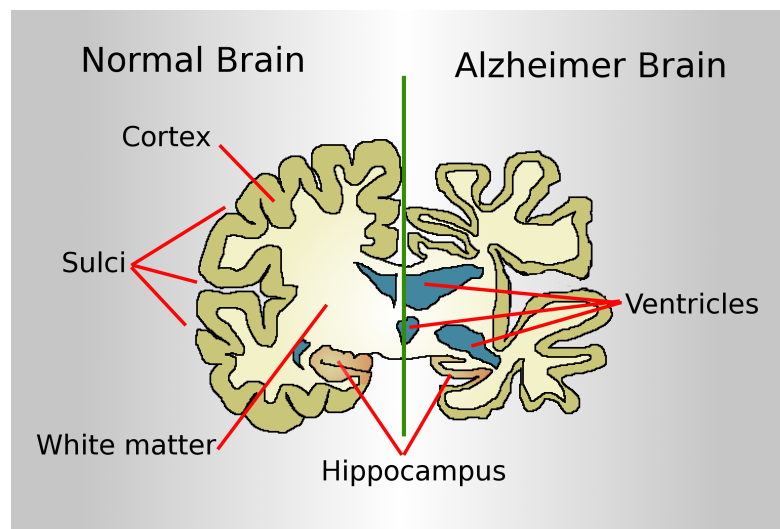


Figure 2.5: Healthy and AD affected brain indicating relevant areas¹

Dementia with Lewy bodies

DLB is characterized by generally lower rates of atrophy than AD. This difference is especially apparent when looking at the MTL, where it has been proposed as a marker for differentiation of the two diseases[39, 40]. In particular, the

¹ This image is from Wikimedia Commons and is in the public domain.

hippocampus, for which atrophy is highly associated with AD, has been shown to be better preserved in DLB subjects[41]. Even though there is generally less atrophy involved with DLB, there are still some areas that are characteristically affected by the disease. Some of these areas are the midbrain and putamen, which has been shown to be even more atrophic than in AD[36, 42]. It has also been shown that the degeneration of white matter is most prominent in the early stages of DLB, while it continues throughout the progression of AD[43].

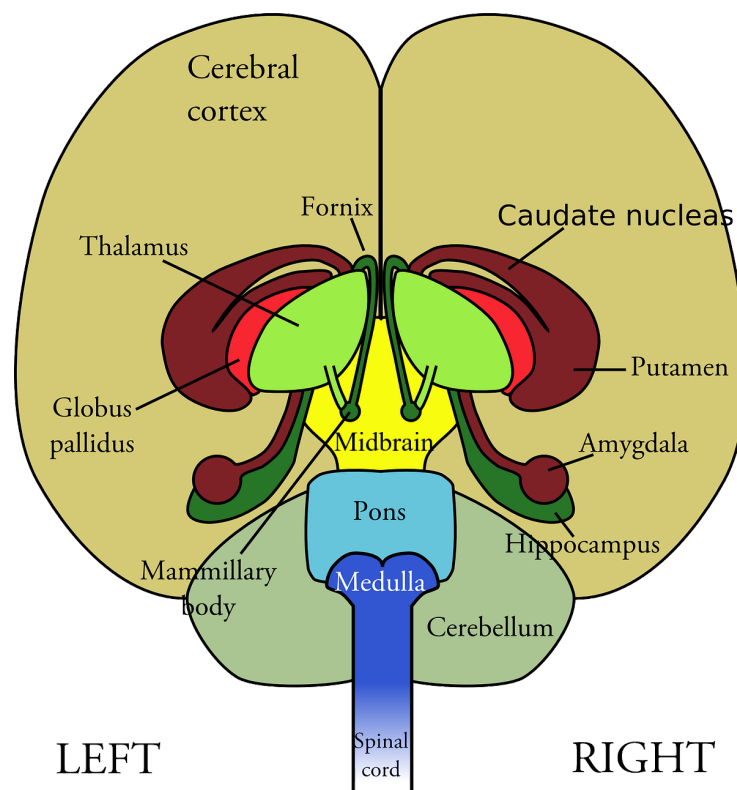


Figure 2.6: Parts of the brain indicated on coronal slice²

2.2 Preprocessing

To optimize the training of a DL model, reducing the amount of irrelevant information in the dataset is desirable as it increases the probability of a model learning more general features. Below some preprocessing steps used to reduce unwanted characteristics of MRI scans are introduced.

² This image from [Pixabay](https://pixabay.com/) and is in the public domain

2.2.1 Spatial normalization

Spatial normalization addresses the issue of difference in size, spatial location, rotation, translation, and warping of brain matter between images. The procedure involves fitting the data to a standardized template, resulting in a dataset with all images sharing the same coordinate system. Spatial normalization procedures might also correct variations in image intensity caused by the varying strength of the magnetic field during an MRI scan.

2.2.2 Brain extraction

As the name implies, brain extraction allows for the exclusion of the parts of a head scan which are not brain matter.

2.2.3 Data normalization

When working with neural networks, it is common practice to normalize the input data. This means that the data is altered as shown in Equation 2.1 to have a mean of zero and a unit standard deviation(SD). Natural images generally have values within a specific range and are thus often just zero centered using the mean value per channel.

$$\hat{y} = \frac{y - mean}{SD} \quad (2.1)$$

The normalization of the data is done to speed up training by stabilizing the distribution of the input layer[44]. The idea is that by ensuring all inputs are of the same distribution and relatively similar scale, it becomes easier for the model to learn relevant patterns for the given task.

2.3 Artificial neural networks

For the classification of brain scans, a highly popular technology known as artificial neural networks(ANNs) will be utilized.

ANNs exist in many different shapes and sizes and can be categorized in many different ways. This thesis will be focused on what is called supervised learning, meaning that an ANN-model is created using data for which we already know the desired model output. A dataset used for supervised learning thus consists of a set of input data, as well as the correspondingly wanted outputs. Such a dataset can, therefore, be interpreted as a description of the desired model. These known outputs are often referred to as the ground truth, targets or, in a classification scenario, labels. Training an ANN means to expose the network to a set of samples and optimize its characteristics to make its predictions as equal to the corresponding ground truths as possible. This process can also be regarded as fitting the network to the desired model description, as described by the dataset.

As the name implies, ANNs are inspired by the inner workings of the human brain. It is, however, important to emphasize that these networks are based on a very loose analogy to the biological brain. This analogy is none the less a useful tool for understanding the concept on which ANNs are based.

2.3.1 Multi-layer perceptron

A simplified description of the human brain could be a network-like structure of biological neurons. Each biological neuron receives signals from the nearby neurons of the network and processes them, before passing on its own resulting signal. Figure 2.7 illustrates a biological neuron and some of its components. As we are all aware, such biological neural networks are capable of representing very powerful models and can be trained to perform a variety of complex tasks.

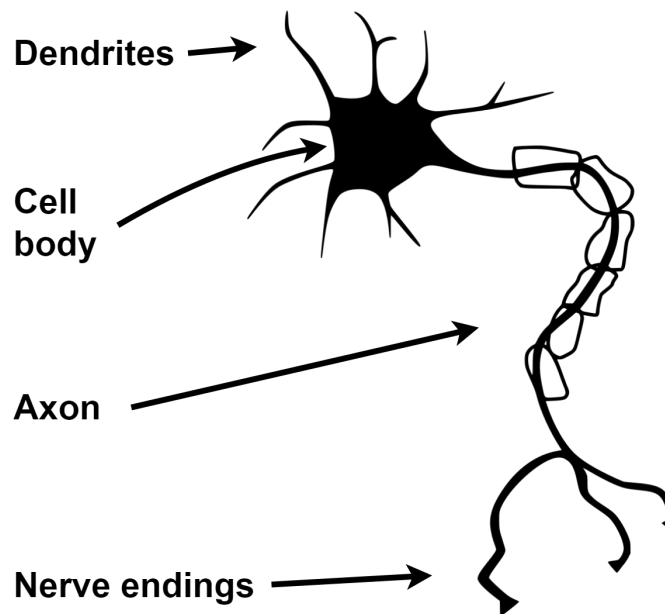


Figure 2.7: Illustration of a biological neuron³

The artificial neuron mimics the abovementioned description of the biological neuron. Each artificial neuron consists of a set of weights and a nonlinear activation function. All inputs of the neuron have one assigned weight, and the first step of calculating a neuron's output is to take the sum of all inputs, multiplied by their corresponding weight. This sum, plus a bias term, is fed to the activation function, giving the final output of the neuron. As with biological networks, the idea behind ANNs is to create systems capable of learning complex models by exposing them to data.

An early example of an artificial neuron is the perceptron, a binary linear classifier. The perceptron uses a unit step activation function, causing the output to be either a one or a zero corresponding to whether the input is above or below a linear threshold. This threshold is determined by the input weights and the bias of the neuron. Combining multiple perceptrons in a layered network structure makes it possible to create complex non-linear models, composed by the different discriminant functions represented by each perceptron. Such networks are commonly known as multi-layer perceptrons.

³ This illustration is from [SVG Silh](#) and is in the public domain

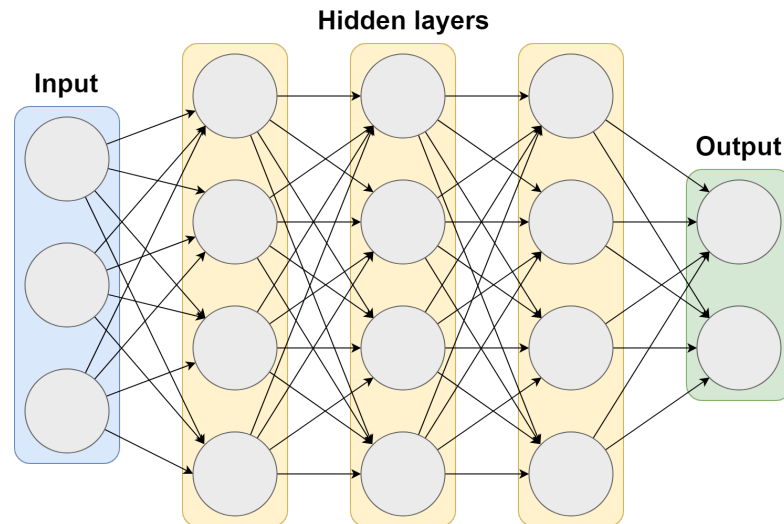


Figure 2.8: Multi-layer perceptron

The general structure of multi-layer perceptrons consists of multiple layers of neurons and is often visualized as graphs like the one shown in Figure 2.8. These graphs are characterized by their input layer, output layer, the number of layers in between, called hidden layers, as well as the number of nodes in each layer. These nodes represent the neurons of the network, with the exception of the input layer, in which the nodes represent each value of the input data. The nodes of the network are connected by lines, which can be interpreted as input weights of the next neuron. If all outputs of the previous layer are connected to a given layer, the layer is said to be fully connected(FC).

2.3.2 Feedforward neural networks

The multi-layer perceptrons can, in many ways, be considered the early version of what is today referred to as feedforward neural networks. These networks are a category of ANNs where all information is being fed from one end of the network to the other, without creating any cycles.

The size and complexity of the models that can be represented by feedforward neural networks make them very popular when solving complex tasks. Training such networks can however require very large amounts of data and might be

computationally demanding. These networks have, therefore, first gained popularity in recent years, with the rise of computational power and available data.

2.3.3 Convolutional neural networks

CNNs are currently a very popular class of DNNs. DNNs are referred to as ANNs that have multiple hidden layers and generally implies models of high complexity. In recent years CNNs have gotten much attention in the computer vision community as they have shown great promise in tasks such as image classification, image segmentation, and object detection. Even though much of the theory and many of the concepts behind CNNs have been known for quite some time, the technology's popularity has first risen during the last decade, with the improvement of graphical processing units (GPUs). Modern GPUs allow for large-scale parallel computing, which is ideal for matrix operations used in CNNs.

Convolutional layer

CNNs are to some extent based on the human, or animal, visual system in that they mimic the use of local receptive fields to evaluate visual inputs. These receptive fields are represented by filters or kernels. A convolutional layer operates by sliding a filter over a matrix-shaped input, convolving it with the part of the matrix that it covers. The convolution term is used quite loosely in this instance as the procedure does not correspond to classical convolution but is, in fact, more like the dot product between the filter coefficients and the values of the given part of the image. The convolution procedure is illustrated for a 3D matrix input in Figure 2.9.

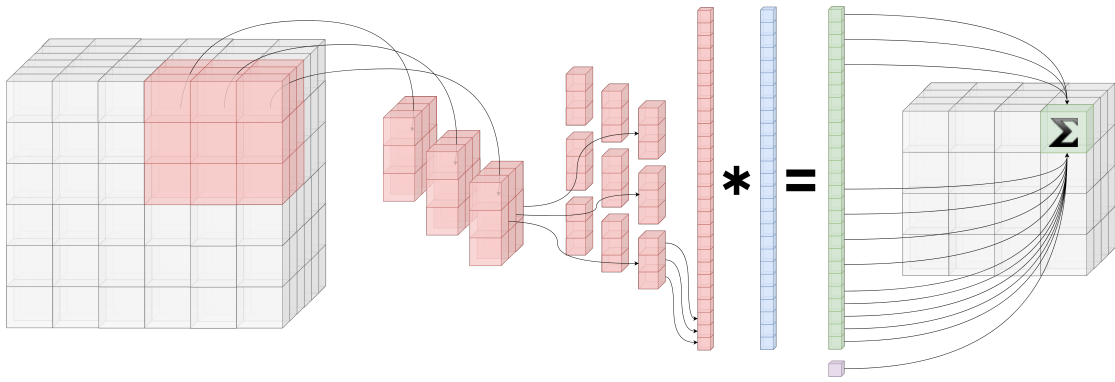


Figure 2.9: Illustration of 3D convolution operation with a kernel of 3x3x3(blue), bias(purple), stride of one and no padding. * indicates element wise multiplication

When sliding the filter over an image, convolving the filter with a given part of the image gives the filter-activation at the image location covered by the center value of the filter. For this reason, the filter has to be of an odd length, width, depth, etc., depending on the dimension of the input. The result of all these convolutions is saved as a matrix which is often called a feature map. The feature map indicates how much the receptive field represented by a given filter was activated at each point of the input. The output of any convolutional layer is a number of feature maps, corresponding to the number of filters in the given layer. The nonlinearity of convolutional layers is introduced by applying an activation function to each value of the feature map independently. The size of the receptive field of a convolutional layer is defined by the filter size, which is a parameter set by the network designer. Such parameters are called hyperparameters.

A filter can not be applied at the edges of a matrix input as it would cover parts outside of the matrix bounds. This means that the convolutional operation can only be applied over image locations with a certain distance from its edges. The filter size determines this distance. The result is a feature map that is smaller than the input. How much smaller is determined by the size of the filter. In general, a feature map will shrink along a given dimension by the filter size of the same dimension, minus one. A common way to avoid this effect is by using what is referred to as padding. Padding means that the input image is expanded, or padded, along its edges by a number of values. How many values that are added at each location along the edges is a hyperparameter of the convolutional layer. A

popular choice is to pad images using values of zero, but any value can theoretically be used.

Another important hyperparameter of convolutional layers is the stride. The stride defines how the algorithm slides the filter over the input matrix and can be set for each dimension separately. As an example, the algorithm can be set to apply the filter convolution at every other value of the input along one dimension, and every third along another. This will reduce the size of the feature map and the number of calculations.

During the training of a neural network, the filter coefficients of each convolutional layer are altered to better represent features, or parts of features, relevant for the task at hand. By stacking multiple convolutional layers, a network can learn increasingly complex features. Higher-level convolutional layers learn activation patterns of the feature maps of the lower-level layers, creating a hierarchical structure of low to high-level feature representations.

2.3.4 Pooling

Pooling is referred to as the act of combining or grouping a set of values. In neural network scenarios, pooling typically means to represent a set of values by one value and is used in so-called pooling layers. Pooling layers works somewhat similar to convolutional layers in that they slide a filter of a given size over the inputs and returns a value which somehow represents the part of the input covered by the filter. These filters are often called windows when working with pooling layers. As with convolutional layers, pooling layers have hyperparameters such as kernel size, stride, and padding. Pooling layers are often used between convolutional layers of a CNNs as a tool to reduce the spatial size of data, and thus also the memory consumption and computational strain on the system. One commonly used form of pooling is max pooling. Max pooling represents the windowed part of the input by its maximum value.

2.3.5 Loss functions

Training any form of ANN requires some measure of how well the model performs, for which the network can be optimized. During supervised training, the network is exposed to data at the input for which we know the desired output. The data is propagated through the network, producing the model's corresponding output. This process is called a forward pass. The performance measure is usually calculated by comparing the output of the model with the desired output. There are many different techniques designed specifically for this purpose, and they are implemented as what is called loss functions. Loss functions are designed to quantify how wrong a model is in its prediction. This quantity is called the model loss and can be interpreted as a model's penalty for being wrong.

Different ANN applications require different loss functions. Examples of such applications are regression, segmentation, and classification. This section will introduce a common loss function used for classification tasks, the cross-entropy loss function.

Cross-entropy loss

In a classification scenario, the loss calculated by the cross-entropy loss function can be considered as being based on the model's confidence in its prediction, meaning that the model is not only penalized according to whether its output was right or wrong, but also by how confident it is in its prediction. This property makes the cross-entropy loss function popular, as it does not stop improving the model even if it classifies everything correctly.

The function is based on the idea of maximizing the likelihood of the model's prediction representing the distribution of the correct class labels by altering the model's characteristics. The model's characteristics are optimized indirectly by determining the model's loss.

2.3.6 Backpropagation

When training an ANN, the goal is to minimize the loss for any given input by changing the parameters of the network. This is normally done by altering the parameter values according to their gradient, calculated with respect to the model loss.

Following a forward pass, backpropagation is the procedure of propagating the calculated loss back through the network. Starting from the output, the algorithm works by recursively applying the chain rule, calculating the gradient with respect to each parameter moving through every node and branch of the network. A simple example of how the chain rule can be applied to is shown in Equation 2.2, where f could represent the network's output function.

$$\frac{\partial Loss}{\partial \mathbf{w}} = \frac{\partial Loss}{\partial f} \frac{\partial f}{\partial \mathbf{w}} \quad (2.2)$$

For backpropagation to be possible, all intermediate values needed to calculate the gradient have to be saved during the forward pass. The use of the chain rule simplifies the computation of the gradient as the derivatives of each parameter can be calculated using the intermediate values from the forward pass and the gradient of the previous parameter, moving down the network.

2.3.7 Optimizers

The algorithms responsible for updating the parameter values of a network according to the calculated gradient are called optimizers. In this section two commonly used optimizers are presented.

Stochastic gradient descent

Stochastic gradient descent (SGD) is a well-established optimizer which is, as most of the commonly used algorithms are, based on the original gradient descent (GD)

algorithm. To get an intuitive interpretation of how these algorithms work, the general loss corresponding to different network parameters is often thought of as a graph or a landscape. An example of a two-dimensional loss landscape is illustrated in Figure 2.10.

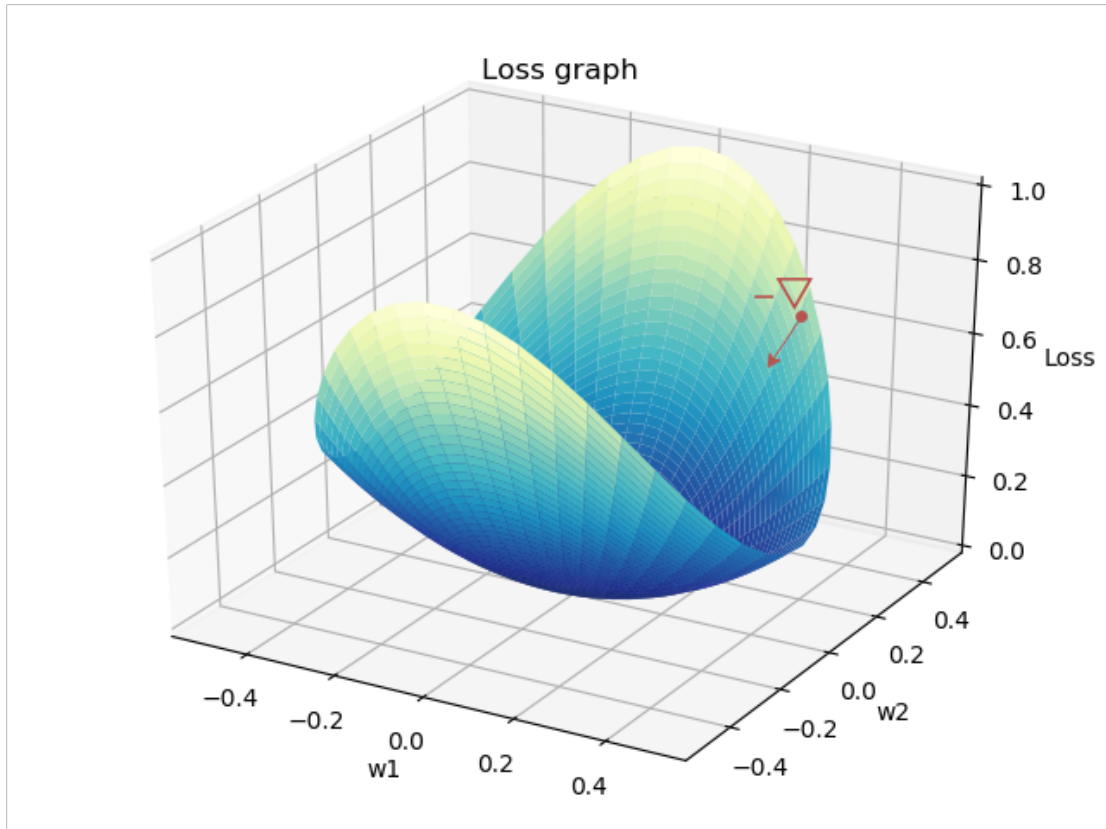


Figure 2.10: Loss over a two dimensional system parameter space. $-\nabla$ indicates the direction of the negative of the gradient

The GD algorithm attempts to optimize the network’s performance by altering its parameters in the opposite direction of their gradient. Equation 2.3 shows how the system parameters \mathbf{w}^4 are updated according to the gradient, which is calculated from the loss corresponding to a given input \mathbf{x} . In this equation, η represents the learning rate, an important hyperparameter of all gradient-based optimizers. The learning rate scales the magnitude of the parameter updates.

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \cdot \frac{\partial}{\partial \mathbf{w}_i} \text{Loss}(\mathbf{x}_i, \mathbf{w}_i) \quad (2.3)$$

⁴The system parameters are often denoted with \mathbf{w} as most of them are weights of the different layers of the network.

The idea is that the network, when exposed to enough data, will reach parameter values for which the gradient is zero. In a loss-landscape, a point with a gradient of zero will either be a minimum, a maximum or a saddle point. Moving in the negative direction of the gradient, the parameter values at a minimum point could represent the lowest possible loss. The point corresponding to the lowest possible loss is known as the global minimum, and it represents the optimal model for the given task.

When using the original GD optimizer, the gradients are calculated and summed for all samples in the dataset before the parameters are updated. This means that as the size of the dataset increases, the number of calculations before each parameter update increase with it. These computations can become time-consuming, causing the model to have a slow progression towards the global minimum.

SGD differentiates from GD by updating the parameter values according to the gradient calculated by one single input. SGD does not necessarily move as directed towards the same minimum as GD, but the more frequent updates can reduce the time of computation drastically and thus speed up training.

Another, more commonly used version is the mini-batch GD. Mini-batch GD calculates the gradient using a small batch of the dataset rather than just one sample, as with SGD. This method has the advantage of getting a better estimate of the overall gradient without increasing the number of computations significantly. The size of the mini-batches is a hyperparameter determined by the designer.

For all these three methods, a problem arises with the presence of local minima or saddle points. At these points, the gradient will become zero, and parameter updates will thus stop before reaching the global minimum. Even if the algorithm does not lead to one of these points exactly, parameter values near them will cause training to slow down drastically. To overcome this problem, the notion of momentum was proposed. The momentum can be regarded as an accumulation of the gradients calculated as the optimizer iterates through the different batches of the dataset. Each update is the result of a decayed running average of the previous

gradients, as well as the current gradient. This makes the algorithm more robust and able to recover from saddle points and small local minima.

Adam

The Adam optimizer is a gradient-based optimization algorithm with adaptive parameter updates dependent on the decayed average of the gradients and their square. When compared to SGD with momentum, the inclusion of the squared gradients improves the algorithm's robustness to large relative differences between the derivatives of the different system parameters. A simple example is illustrated in Figure 2.10, where w_2 highly dominates the illustrated gradient. The introduction of the squared term can be used to limit dominant parameters with the intuition that it will direct parameter updates more efficiently towards the global minima. Adam has been shown to speed up training in most applications and occasionally lead to better final performance[45].

2.3.8 Activation functions

The activation function can be seen as the main characteristic of a network's nodes. All the hidden nodes of a network should contain a non-linear activation function. It is this nonlinearity that makes the network capable of learning complex nonlinear models. In this section some common activation functions will be introduced.

Sigmoid

The sigmoid activation function can be interpreted as a saturated firing rate for a neuron as it truncates its output between zero and one according to Equation 2.4.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.4)$$

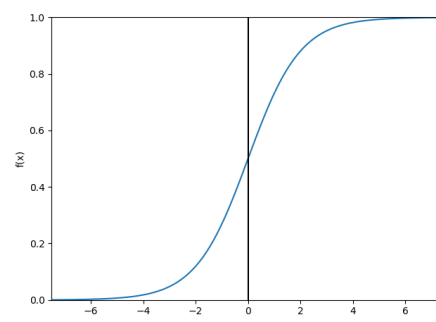


Figure 2.11: Plot of sigmoid function

The function was one of the earlier activation functions used with ANNs but has lost its popularity as some drawbacks of the function has become apparent.

A significant problem with this activation function is the saturation of the output. When the input of the function has a large absolute magnitude the neuron saturates to either zero or one, as shown in Figure 2.11. This means that the gradient with respect to these points becomes close to zero. In a chain of multiple neurons with sigmoid activation functions, the gradients are thus likely to become very close to zero before they are propagated back to the input layer. This problem is known as vanishing gradients and results in very slow progress during the training of deep structures, especially in layers close to the input.

ReLU

The rectified linear unit (ReLU) is currently a popular activation function. For all inputs of either negative values or zero, the function outputs zero, while all positive inputs are passed through. The ReLU is plotted in Figure 2.12.

The function has some specific features which make it popular. One important feature is that it does not saturate the

output and thus avoid the vanishing gradient problem. Another important factor is that the function is computationally efficient, which contributes to both faster training and execution. The ReLU function is presented in Equation 2.5.

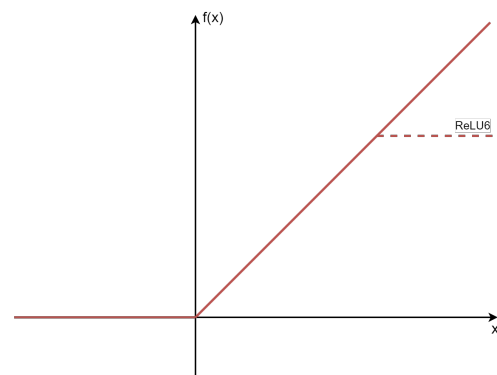


Figure 2.12: Plot of ReLU and ReLU6

$$f(x) = \max(0, x) \quad (2.5)$$

A commonly used version of the ReLU is the ReLU6. This function has a hard maximum of six, as illustrated in Figure 2.12. ReLU6 has been shown to have

some fortunate characteristics like encouraging the model to learn more sparse features and reducing memory consumption[46].

2.3.9 Overfitting

During supervised training, an ANN is optimized to create a model that fits the training data for a given task. A problem arises when this data is not a perfect representation of the general data distribution for this task. An example of such a general distribution could be all possible images of animals when training an animal classifier. Perfect datasets does not exist in practice as they would require an infinite amount of noise-free data samples.

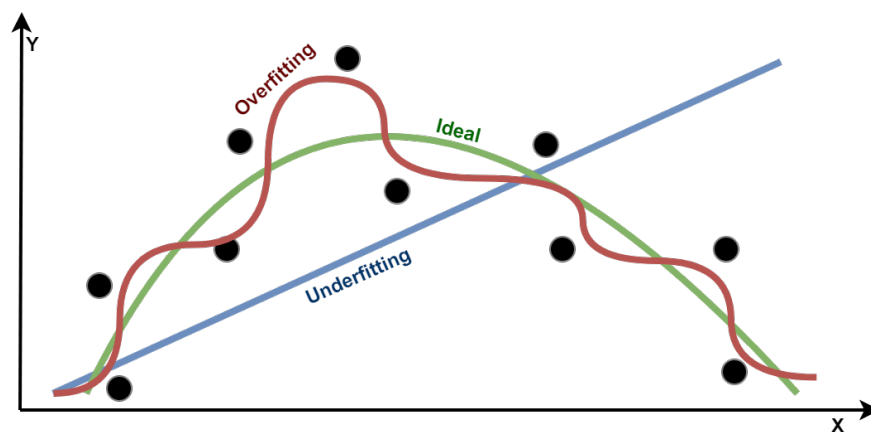


Figure 2.13: Model over- and underfitting on regression problem

A high complexity model has a large representation power, also referred to as its capacity. Such models might be able to fit a given data distribution quite precisely. A model with very high complexity might, in fact, fit the training data too well, meaning that it learns patterns applicable to the less than perfect training data, and not the general data distribution. This is called overfitting, and the result is a model that performs well on the training data and less so on new data from the same general distribution, so-called unseen data.

Models with very low complexity will, on the other hand, have a lower capacity. Such models might not be sufficiently complex to properly represent the nature of

the task at hand, resulting in what is referred to as underfitting. Figure 2.13 shows an example of over- and underfitting for a two-dimensional regression problem. Overfitting is a common challenge when designing DNNs as they have the potential to represent very complex models.

Model bias and variance

As mentioned, models with large capacity might overfit a given training set. This means that if multiple such models were trained on different subsets of the same general distribution, the resulting models would vary greatly. Such models are, for this reason, said to have a high variance. Oppositely, models with very low capacity might create a simple representation of the desired model. With all the data stemming from the same general distribution, these models will tend to become very similar and are thus said to have a low variance. As an example, changing one point in Figure 2.13 would significantly alter the red model and not the blue.

How much a model generally differs from the true model of the given task is defined as the model bias. When solving complex tasks, most low capacity models tend to become very simple versions of the desired model and are thus said to have a high bias. High capacity models tend to have a lower bias as they are able to resemble the complex nature of the problem better. Figure 2.14 illustrates the connection between the concepts described above. It is important to emphasize that it is impossible to measure the model bias and variance as this would require knowledge of the true model.

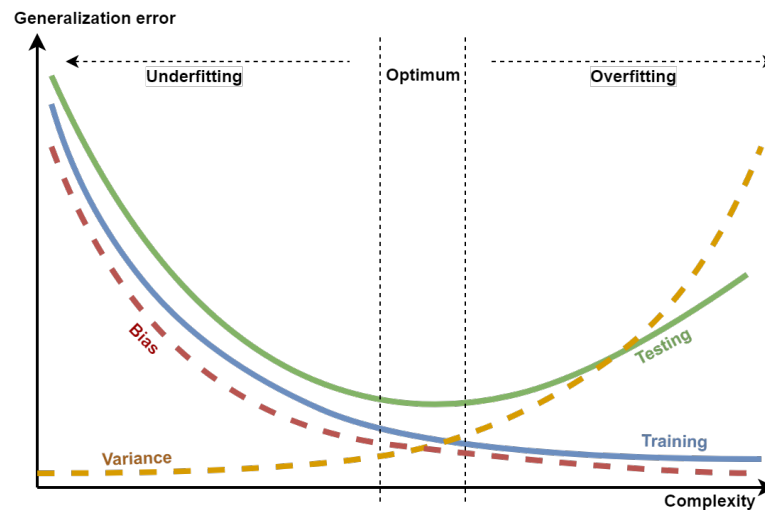


Figure 2.14: Bias/Variance trade-off

Designing a model of the correct complexity is not always a trivial task. Too low complexity means that there is unused potential in the data. Too high complexity may result in overfitting. Both scenarios will lead to suboptimal generalization performance. As it is difficult to reduce the variance without increasing the bias, and vice versa, balancing a model's complexity is often referred to as the bias-variance trade-off.

Validation and test

In order to estimate how well a model performs on unseen data, it is common practice to set aside a part of the available data before training the model. In doing so, the model can be trained on one subset of the data and its performance validated on another. The validation performance will then give an estimate of how well the model generalizes, meaning how well it fits the general data distribution for the given task. To reduce the chance of overfitting, hyperparameters, and other model design aspects can be optimized for the model's performance on the validation set rather than the training set.

Even though this means that the model is not trained on the validation data directly, there is still a probability of overfitting as the model is designed to better fit the validation set. A third set can be used to estimate the final model's performance

on unseen data. This dataset is called the test set and should contain data which in no way has influenced the construction of the model⁵.

Cross-validation

Because of large model variance, estimating model generalization can be quite challenging when working with complex models like DNNs. Cross-validation(CV) is a tool used during model design to improve these estimates.

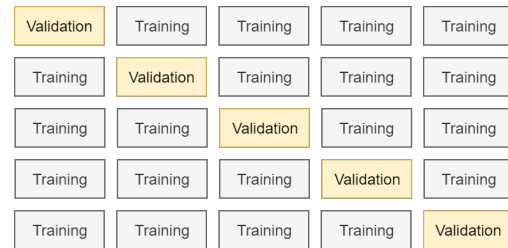


Figure 2.15: Illustration of 5-fold cross-validation operation

When performing CV, a dataset is split

into a given number of subsets, called folds. Selecting one fold as a validation set, a given model is trained on all remaining folds before being evaluated on the selected fold. Training one model for each such split, a better estimate of the model's generalization can be made by combining the performance metrics calculated for each validation fold. Figure 2.15 shows how the algorithm splits up the dataset and iterates through the different folds.

2.3.10 Regularization

When trying to combat overfitting, there are generally two ways to go. Either increase the chance of the model prioritizing features specific to the general distribution or limit the models capacity. Below some techniques used to reduce a model's capacity will be introduced.

⁵There is an inconsistency of the terminology used for these datasets. In some cases, switching meanings of the validation and test terms can be observed. In this thesis, the terminology will be used as described above

L2 regularization

L2 regularization tries to limit a model's capacity by penalizing the parameter updates according to the model's current weights. A penalty is added to the loss as a regularization term⁶. This is shown in Equation 2.6, where m is the total number of weights in the network.

$$Loss + \lambda \sum_i^m w_i^2 \quad (2.6)$$

The technique is especially hard on large weights as each parameter update is penalized by the current parameters' square product, scaled by the regularization hyperparameter, λ . The technique is sometimes referred to as weight decay as it will force the weights towards zero as the model iterates during training.

There are multiple intuitions behind why L2 regularization reduces the model capacity. One being that features of the general distribution might tend to be more subtle than those specific to the training set. Penalizing higher weights might then reduce the chance of the network prioritizing these features.

Dropout

There are many aspects to consider when designing a model, e.g., which optimizer and loss function to choose, how to set the hyperparameters and structural elements like the model's size and shape. Different configurations will result in different models. These models will perform differently for any given task, each having individual characteristics, strengths, and weaknesses. Intuitively, combining the strengths of multiple different models into one might provide higher performance than simply choosing the best one among them. This may, however, be impractical when working with DNNs as each of these models would have to be tuned, stored, and ran at each execution of the final model.

⁶There is another type of regularization called L1 regularization, where the regularization term is based on the non-squared weights. This type of regularization is, however, not as commonly used

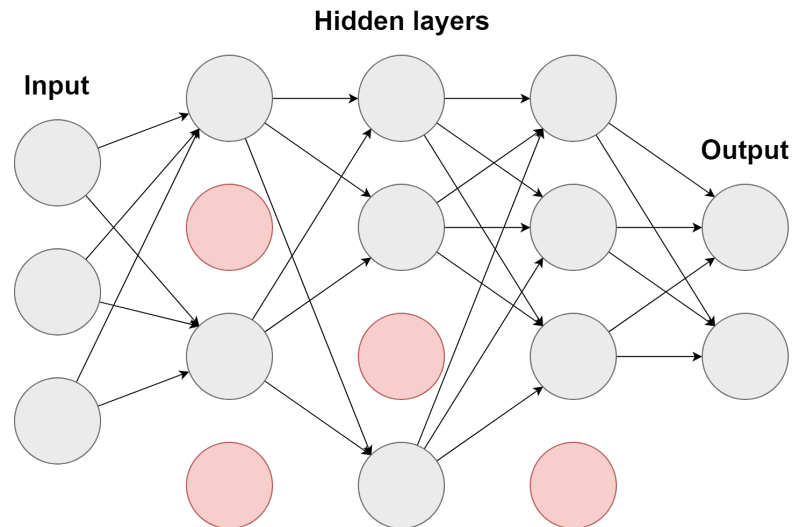


Figure 2.16: Dropout implemented in a multi-layer perceptron

The main idea of dropout is to estimate this combination of multiple models using only one. The intuition is that by randomly dropping layer outputs and connections during training, the resulting model can be interpreted as a combination of multiple thinned out versions of the original model. The dropping of layer outputs is illustrated in Figure 2.16. The probability of an output being dropped is a hyperparameter set by the system designer.

For each iteration, the system only updates the parameters of the thinned out model. In order to combine these thinned-out versions of the network during validation and testing, an equally weighted geometric average of the models is estimated. This is done by scaling the layer outputs which were dropped during training by their probability of being dropped.

When tuned well, dropout has been shown to reduce overfitting and thus improve generalization in most applications[47]. Dropout is also said to have a somewhat regularizing effect in that it introduces random noise within the network structure, encouraging the model to learn more general features.

2.3.11 Data augmentation

Improving a training set's representation of the general data distribution for a given task might increase the given model's generalization. Increasing the number of unique samples in the training set is usually a good place to start as the most common features across all samples then will be more likely to represent the general distribution. Increasing the number of unique samples in a given dataset is, however, not always possible.

In such scenarios, data augmentation might be a good option. For classification implementation, data augmentation generally refers to the altering of data in ways that do not change its ground truth. As an example, rotating or flipping an image of a cat does not alter the fact that it pictures a cat, but provides another different representation of it. With the same argumentation as in the previous paragraph, increasing the size of the dataset with such augmented samples might improve model generalization.

2.3.12 Batch normalization

Batch normalization(BN) builds on the same principle of input normalization as mentioned in section [2.2.3](#) but brings it into the network structure itself.

When training a neural network, the input of each hidden layer, as well as the output layer, receives the output of a previous layer. Normalization of the input data ensures that the first layer always receives data of the same distribution. As the data propagates through the layers of the network, the distribution of the activations is, however, not guaranteed to be consistent. This is where BN comes in.

A BN layer works much like the normalization of the input data in that it normalizes the layer inputs by their mean and standard deviation. However, having data of zero mean and a standard deviation of one may not always be desirable inside the network. For this reason, the data is scaled by two learnable parameters, β and γ ,

as shown in Equation 2.7, where \hat{x} represents the normalized inputs.

$$y = \gamma\hat{x} + \beta \quad (2.7)$$

These parameters become the new mean and standard deviation of the given data. As with other learnable parameters, these are updated for every mini-batch when using a mini-batch gradient-based optimizer. The consequence is that as the model is trained, iterating through the dataset, these parameters change slightly. This change in distribution can be considered as noise, which, as with dropout, might have a regularizing effect. BN has been shown to speed up the learning process quite drastically and potentially improve the final performance of the network[48].

2.3.13 Hyperparameter tuning

As the system itself does not optimize hyperparameters, these need to be determined by the system designer. The difference in performance between a poorly tuned and a well-tuned model can be substantial. Choosing the right hyperparameters is thus a crucial part of creating a DNN model. There is, unfortunately, no uniquely correct way of finding these optimal hyperparameters. There are, however, some techniques that can be used to search a given hyperparameter space methodically. Common for all methods is that they sample from a set of hyperparameters, which all are within a bounded range.

Grid search

A commonly used approach is grid search. By constructing a grid of different hyperparameter values, the system can iterate through these combinations, train models, and record the corresponding performance. This method works fine for smaller dimensional hyperparameter spaces, and particularly in combination with models that have short evaluation time.

Random search

When using the random search approach, one set of hyperparameters is chosen randomly for each iteration. Random search has been shown to be more efficient than grid search, in most scenarios[49].

Bayesian optimization

Training and evaluating a DNN can be a time-consuming task, and tuning hyperparameters using methods like grid and random search can thus become close to infeasible. Bayesian optimization is a technique used to determine which hyperparameters to test based on prior knowledge.

Finding the right hyperparameters can be considered an optimization problem, much like the training of a neural network. The model representing the relationship between the different sets of hyperparameters and the corresponding network performance is, however, often considered as unknown. There is also no way of calculating the gradients of the hyperparameters as they are considered to have no direct mathematical connection to the model's performance.

Bayesian optimization is a technique that can be used for black-box optimization problems, like the one described above. The idea is to consider the connection between the hyperparameters and the model performance as a random function and build a probabilistic model of this function using a prior distribution. When iterating through different sets of hyperparameters, the prior distribution is updated using the previous results. The prior at a given iteration is used to estimate the next set of hyperparameters. Different prior distributions can be used. A popular choice is to use a Gaussian distribution. Bayesian optimization has been shown to find well-performing hyperparameters using significantly fewer iterations than manual or random search[50].

2.3.14 Evaluation metrics

The evaluation of a model's performance is an essential part of any ML design process, and neural networks are no exception. There exist many different performance metrics made for different purposes and applications. In this section, some of the more commonly used performance metrics for classification applications will be introduced.

Loss

When training an ANN the value of a given loss function can be used as an estimate of model performance. Depending on the type of loss function, the loss can be more or less a rough estimate of performance and is usually only considered while training a model.

Accuracy

Accuracy is one of the most commonly used performance metrics. The measure is the percentage of the total data which was correctly classified. A limitation of this measure is reliability when having imbalanced amounts of data between the different classes.

Confusion matrix

The confusion matrix is a powerful tool for inspecting how the model's predictions are in comparison with the corresponding labels. Represented as a square matrix of length and width corresponding to the number of classes, one axis represents the predicted values and the other the true labels. An example is shown in Figure 2.17. Some performance metrics can be derived from inspecting the confusion matrix. Examples are accuracy, precision, recall, and F-beta score.

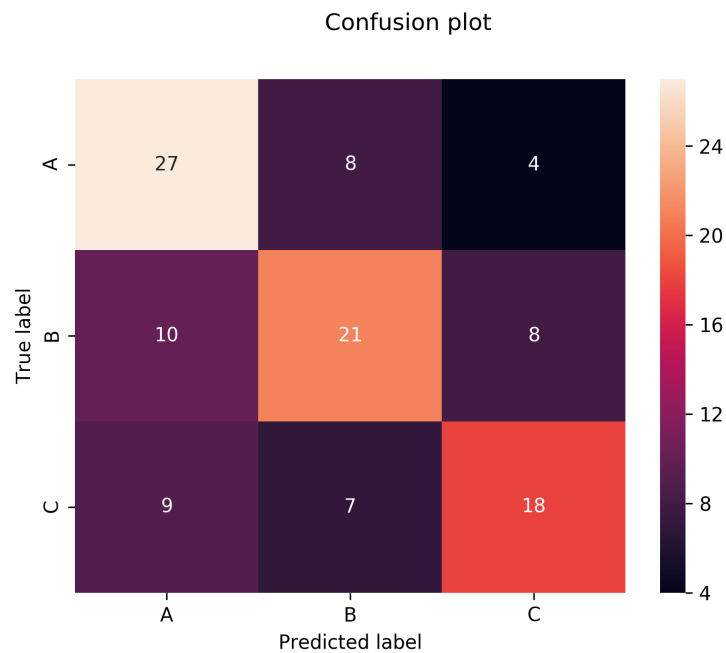


Figure 2.17: Example of confusion matrix

Precision

Precision is defined as the share of the data predicted to be of one specific class that actually belongs to the given class. A typical scenario where this measure becomes relevant is when having imbalanced amounts of data between the classes.

Recall

The share of the data that belongs to a given class that is correctly classified is referred to as a model's recall. As with precision, this can give a broader sense of a model's performance and might be especially relevant when working with imbalanced datasets. Another typical scenario where recall might be a valuable measure is when there is a high risk associated with misclassifying a given class.

F-beta score

The F-beta score is a performance measure based on the harmonic mean between precision and recall. The score is, e.g., useful when comparing different models by their precision and recall. The measure is calculated according to Equation 2.8.

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \quad (2.8)$$

The beta value determines a weighting of the algorithm toward either precision or recall. A beta value between zero and one increases the weight of precision, while a beta value between one and infinite yields an increased weight of recall. A special case of the F-beta score is when beta is set to one. This is called the F1 score and is the harmonic mean between precision and recall as they are weighted equally.

2.4 Software

Some of the libraries used in this thesis will be introduced below.

2.4.1 Pytorch

For this thesis, the PyTorch platform was used for deep learning implementation. PyTorch is based around tensors and is an open-source deep learning platform written in the Python programming language. The platform was chosen for its flexibility and its use of the standard Python debugger. PyTorch also includes Nvidia GPU support and can parallelize between multiple GPUs, which helps speed up network training.

2.4.2 SciKit-learn

SciKit-learn is a popular open-source ML platform built on the Python programming language. The platform delivers a large selection of tools for many different aspects

of machine learning, such as system setup, different types of models, performance evaluation and visualization.

2.4.3 Docker

Docker is a software that allows for the utilization of custom system images on most common operating systems. The software accesses the host's operating system and instantiates a Linux kernel that can run what is referred to as images. These images are instances of a system set up, which means that the same system image can be loaded on multiple machines with different operating systems, ensuring compatibility across platforms. As an image is a saved instance of a system set up at a given time, software used in the image will not implement the developers delivered updates. This helps with compatibility over time.

2.4.4 NiPype

An issue when comparing different ML neuroimaging studies is the lack of reproducibility and the use of different types of preprocessing and data[51]. Some attempts have been made to standardize the way of conducting ML neuroimaging studies with dedicated software platforms[52, 53], Clinica⁷. One such platform is NiPype[54], a standardized preprocessing pipeline for neuroimaging.

Most of the software used for preprocessing of neuroimages today are provided by different parts of the neuroimaging research community. This software is thus often made by people with good technical background, but which are not software engineers. Even though the software delivers good results, it might not be particularly efficient, user-friendly, or robust. To handle some of these challenges, NiPype provides a Python user interface for most of the preprocessing software commonly used with neuroimaging today. Popular software such as SPM12, FreeSurfer, FSL, and ANTs are implemented in NiPype.

⁷Clinica software platform (www.clinica.run) developed by the ARAMIS Lab (www.aramislab.fr)

SPM

The Statistical Parametric Mapping (SPM) software was created by members and collaborators of the Wellcome Centre for Human Neuroimaging at University College London. The software provides functionality for processing of neuroimages like realignment, normalization to standard templates, smoothing and more.

FMRIB Software Library

FMRIB Software Library (FSL) delivers a multitude of tools for processing and analysis of different types of neuroimaging modalities[55]. Their software can only be run on macOS or Linux.

Chapter 3

Materials and method

This chapter will go through the different steps and methods used to handle and process the data used in this thesis, as well as the proposed model for this thesis.

3.1 Dataset construction

The data used in this thesis consists of T1 weighted sMRI from both the European Dementia with Lewy bodies consortium(E-DLB) and the Alzheimer’s Disease Neuroimaging Initiative(ADNI) databases. Out of the 853 MRI scans provided by the E-DLB consortium, 288 were of subjects having a DLB diagnosis, 171 had an AD diagnosis, and 146 were NC subjects. As the E-DLB dataset was the only source of DLB diagnosed scans, AD and NC subjects from this study were supplemented with data from ADNI. To improve the chance of subjects having a correct diagnosis, only ADNI scans from baseline studies or initial screening was chosen. The intuition behind this being that these subjects had a clinical diagnosis within a reasonably short time of the scan being performed, increasing the reliability of the image labels. The dataset was determined to be split into two subsets of 80% for training and 20% for testing.

Age and gender matching

In the dementia section, common risk factors for dementia were introduced. Some of these factors might be more or less present throughout the dataset. As neural networks are statistical models, any known underlying patterns in the dataset which are not general enough for the differentiation of these diseases should be either removed or matched as well as possible. Doing so increases the probability of the model generalizing better.

Both the E-DLB and ADNI datasets were supplied with metadata containing relevant information about their subjects. This metadata was used to match the subjects. Removing subjects from the already relatively small dataset would be unfortunate and was avoided to a reasonable extent. As the ADNI database contains greater amounts of data than the available DLB data, the DLB data was chosen as a template to which the AD and NC subjects were matched. Relevant risk factors listed in the metadata were age, gender, years of education, and other conditions such as vascular diseases, etc. After ruling out partial metadata, age and gender were chosen for matching. The age distribution of all available subjects is shown in Figure 3.1.

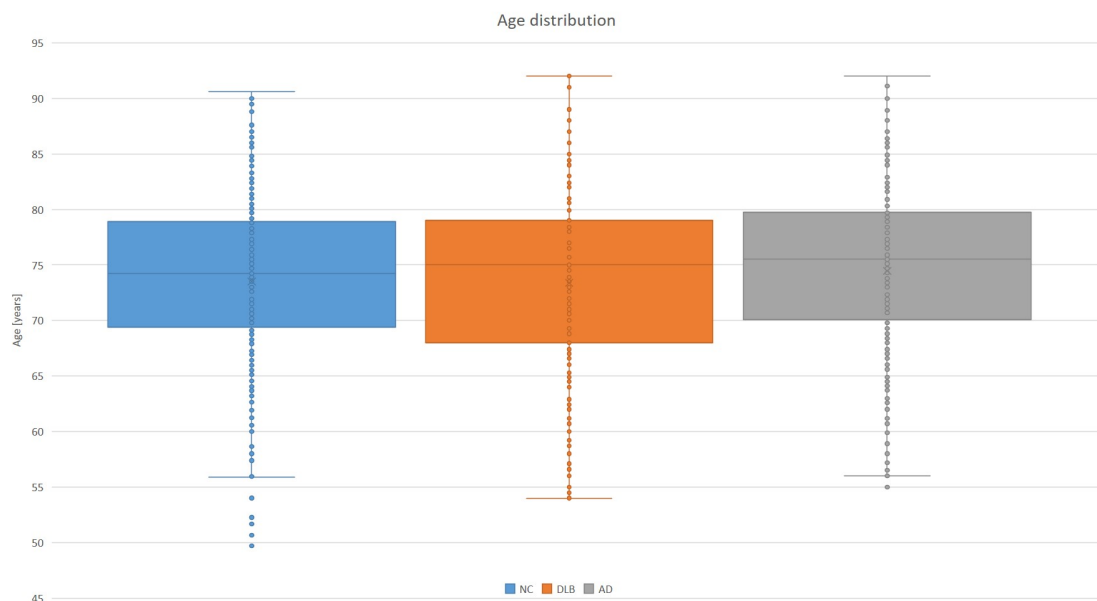


Figure 3.1: Age distribution in original dataset

Age is the most dominant risk factor for dementia. Having different age distributions between the different classes during training might be unfortunate and result in a biased model. Gender is not as strong a risk factor, but its characteristics might be present in the scans. Any pattern in data perceivable by the model, but which are not disease-related might misguide the model during training and result in a reduced generalization.

Table 3.1: Characteristics of the final dataset

Dataset characteristics	Training (80.1%)			Testing(19,9%)		
	NC	DLB	AD	NC	DLB	AD
Mean age	74.48	74.49	74.50	69.51	68.88	74.34
age SD	6.72	6.72	6.72	11.24	11.05	9.51
Count	230	230	230	57	57	57
Males	131	131	131	45	45	45
Females	99	99	99	12	12	12

Gender is binary by nature and was thus matching for each subject individually. As it is during training that a neural network learns viable patterns, the training dataset was prioritized during matching. The result was a training set where all subjects having an AD or NC diagnosis were matched to the DLB subjects with an age difference within half a year. The subjects belonging to the test set were age-matched as well as possible and were at least within two standard deviations of the age distribution of the DLB subjects from the training set. The resulting age distributions for the training and test sets are presented in Table 3.1. One subject from the DLB data was initially discarded, being more than three standard deviations from the mean age of the DLB dataset.

3.2 Preprocessing implementation

As the MRI scans used in this thesis originates from different MRI machines, studies and are of people with different demographics, image preprocessing was considered necessary in order to increase the likelihood of the model learning features specific to the different diseases, rather than the different image acquisition processes. In order for the data to be compatible with preprocessing software, all scans were initially converted to the commonly used NIfTI format.

All neuroimaging specific preprocessing was conducted using the NiPype platform. NiPype delivers its own official docker image¹, which was utilized in this thesis to ensure a fully working environment and reproducibility. The steps of the used preprocessing pipeline are shown in Figure 3.2 and explained below.

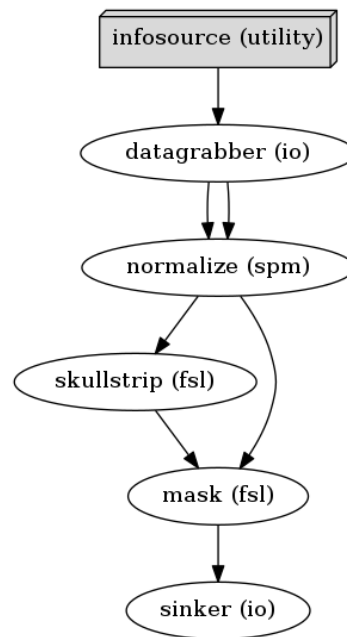


Figure 3.2: Nipype workflow for image preprocessing

Spatial Normalization

Spatial normalization was conducted using the SPM12 normalization software, which is based on[56]. In addition to performing spatial normalization, the software also corrects image intensity variations due to varying strength in the magnetic field throughout the images. Most of the images used in this thesis were originally of a voxel size of about 1x1x1 mm. To retain as much detail as possible, this size was chosen as the standard for the resulting normalized images. Images that were not of the correct voxel size were either downsampled or upsampled using b-spline interpolation of degree four[57]. The result of conduction spatial normalization is a dataset that is more structurally and spatially comparable.

¹NiPype docker-image ID: sha256:2444418803137c4e1554074d9e6edd66d5a6a682e38d18d3c3f0f8ece3a673e0, date:13.03.2019

Brain extraction

As dementia mainly manifests in the brain, the information in the form of eyes, ears, skin, and bone is generally undesirable. The brains of the normalized images were thus extracted using the FSL brain extraction(BET) software. This software implementation is based on [58] which reported segmentation of brain matter with high performance. The resulting images were of size 157x189x156 voxels.

Dataset normalization

As the voxel values were drastically different between images in the dataset, the first step of the data normalization was to scale all images down to a distribution between zero and one. Next, all images were zero-centered by the voxelwise subtraction of the mean image, which was calculated for the training data. This was chosen over subtracting the over-all mean voxel value of the dataset as the images consist of only one channel, and the variance of a large part of the voxels was relatively small. For instance, with the brains being of a somewhat elliptical shape, a significant number of voxels located in the corners of each image were zero across all scans. Scaling all images by the average voxel value of a single channel image would simply shift up or down the image. This linear transformation is not very powerful.

3.3 Model design

This section presents a model inspired by the conventional CNN structure of multiple convolutional and max-pooling layers, followed by fully connected layers. The complete model is visualized in Figure 3.3. ReLU6 was chosen as the activation function for all layers.

The MRI-based biomarkers of dementia presented in section 2.1.1 indicates the importance of volumetric measures when utilizing MRI scans for dementia diagnosis. Several newer studies have reported good performance using 3D convolutional layers for their feature extraction [59, 60], and in their comparison [61] found that 3D convolution outperformed 2D for MRI classification. 3D convolution was thus chosen for the model’s convolutional layers. All convolutional layers were set up with a filter size of 3x3x3 and a stride of one. Layer inputs were padded with values of zero in order to not lose information from the images’ edges.

As often done in conventional CNN structures, the spatial resolution of the network was downsampled between each convolutional layer using max-pooling layers. Max pooling was utilized with a kernel size of 3x3x3 and a stride of two. As the size of each feature map decreased, moving up through the layers, an increasing number of filters were chosen. The number for filters were chosen in such that the number of computations was somewhat consistent throughout the convolutional layers.

The convolutional layers were followed by a set of three FC layers. These layers interpret the features maps of the final convolutional layer to determine the diagnosis of a given input. Training time becomes an important aspect when designing a

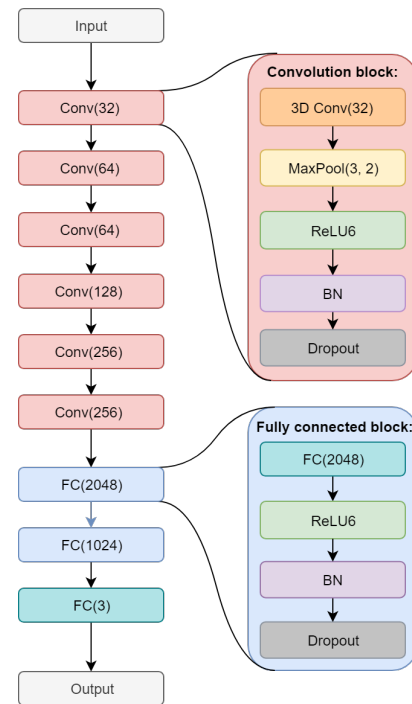


Figure 3.3: Proposed network structure. Parentheses indicate number of filters in a given layer

deep model. To speed up training and potentially increase performance, BN was implemented for every layer of the network.

Chapter 4

Experiments and Results

In this chapter, all performed experiments and corresponding results will be presented chronologically. This approach was chosen as later experiments were based on the experience gained from earlier results.

4.1 Experiments

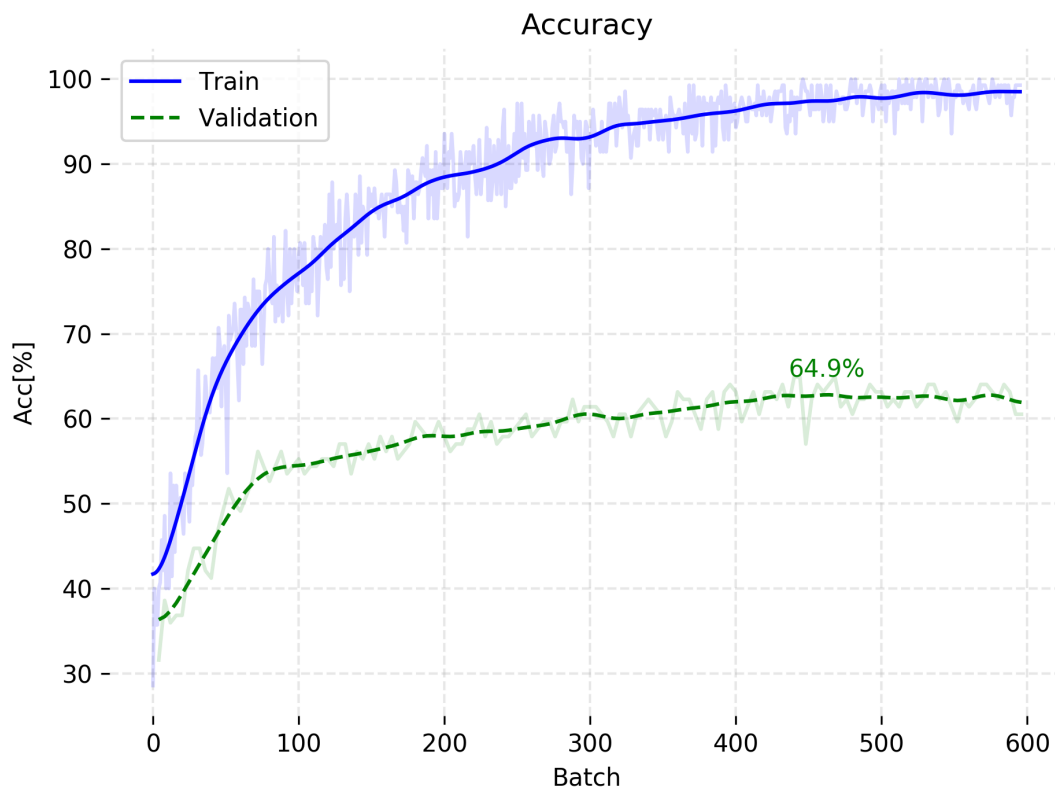
Initial testing

Initial testing was performed to determine how the planned experiments should be conducted. The variation in performance as a result of random initialization of network parameters and different training/validation splits were of particular interest as this would determine if the general performance could be estimated using a single training/validation split, or if the use of CV would be necessary.

Two six-fold CV runs were thus conducted with equal configurations and without any regularization. Both runs were executed with a learning rate of $1 \cdot 10^{-6}$ and the remaining system configurations according to Table 4.2. The results of both runs are reported in Table 4.1. Figure 4.1 shows learning curves for a fold with a close to average maximum validation accuracy.

Table 4.1: Results of two initial cross-validation executions with a learning rate of $1 * 10^{-6}$

	Run A	Run B	Run A	Run B
Fold	Loss	Loss	Accuracy[%]	Accuracy[%]
1	0.800934296	0.795536779	61.53846359	60.68376541
2	0.794067509	0.796742983	61.53846359	62.39316559
3	0.795691714	0.828704454	64.91228485	63.15789413
4	0.694504805	0.665960576	70.17543793	71.92982483
5	0.783715405	0.837599628	66.66666412	64.91228485
6	0.847598903	0.855287593	59.64912415	59.64912415
Mean	0.786085439	0.796638669	64.08007304	63.78767649
SD	0.050105925	0.068152329	3.926560029	4.39687365

**Figure 4.1:** Learning curves for initial testing

As expected, initial testing resulted in substantial overfitting, indicated by the difference in accuracy between the training and validation curves in Figure 4.1. This high variance problem was also verified by the significant standard deviations reported in Table 4.1, showing a performance highly dependent on the training/validation split of each fold.

4.1.1 Experimental layout

It became apparent, after initial testing, that the use of CV would be necessary in order to get performance measures consistent enough for hyperparameter tuning. Splitting the data into six folds was chosen as this allowed for a good trade-off between the generality of the average results and execution time. The folds contained 115 subjects each, resulting in the dataset being split into 83.3% training and 16.7% validation. One model was trained for each split, saving the model with best-recorded accuracy. During training, each model was validated after every passed epoch. Training data was shuffled between each epoch to avoid a constant pattern in the parameter updates, hopefully improving generalization. Accuracy was chosen as the main performance measure during experimentation as the dataset was class-balanced, as well as age- and gender-matched.

The cross-entropy loss function and the Adam optimization algorithm was used for model optimization. The Adam optimizer was chosen for its robustness and efficiency, while the cross-entropy loss function was chosen for its persistent optimization characteristics.

To reduce unnecessary time consumption, training was stopped after a given number of epochs had passed without improved validation performance. This number of epochs is often referred to as patience. The patience was reset by an improved running average of either loss or accuracy over the last five epochs. Training progression was logged every tenth iteration. The system configuration for all the following experiments is presented in Table 4.2.

Table 4.2: System configuration used during experiments

Batch size	Optimizer	Loss function	Patient	Cross-validation folds	Log interval	Shuffle data
14	Adam	Cross-Entropy	15	6	10	Yes

4.1.2 Experiment - Learning rate

In order to get a sense of the hyperparameter space, different learning rates were tested without any form of regularization.

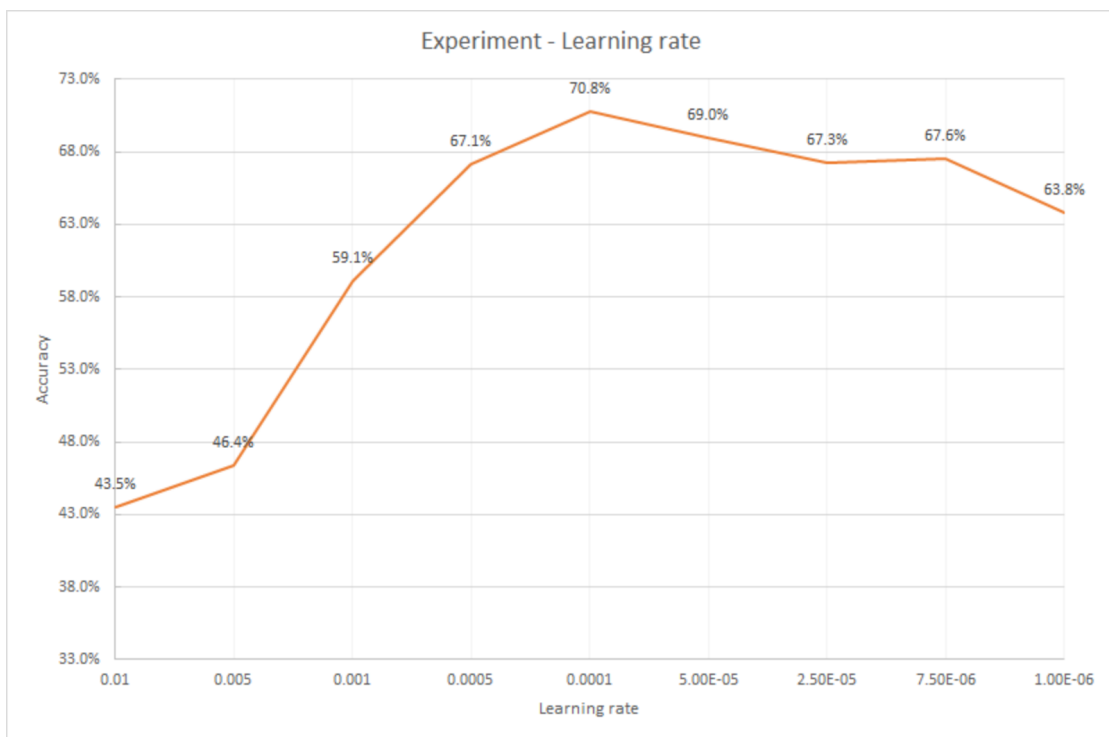
**Figure 4.2:** Result from the learning rate experiment

Figure 4.2 indicates a pretty clear peak in performance at a learning rate of about $1 \cdot 10^{-4}$. These results were used to determine a reasonable span for the learning rates used in the experiments following.

4.1.3 Experiment - Dropout

The results from the initial testing indicated significant amounts of overfitting, and regularization was thus deemed necessary. As the use of dropout might alter the

efficacy of training, different combinations of dropout probabilities and learning rates were tested using random search. The initial range of the dropout values was set up to 0.9 to verify that using a high dropout probability indeed result in lower performance, as indicated by [47] and especially when combined with BN[48]. The search was also tuned in towards both lower learning and dropout rates, to check performance when looking at a smaller span. For this first part of the experiment, dropout was implemented only in the FC layers of the network.

For the second part of the experiment, dropout was implemented in the convolutional layers as well. [47] indicated that dropout should be implemented with a lower probability in lower-level layers. The probability of dropping a feature map was on this basis set to be half of that for the linear layers. Results from both experiments are shown in Figures 4.3 and 4.4, where decreasing bubble diameter indicates decreasing loss, and brighter yellow bubbles indicate increasing accuracy.

Results

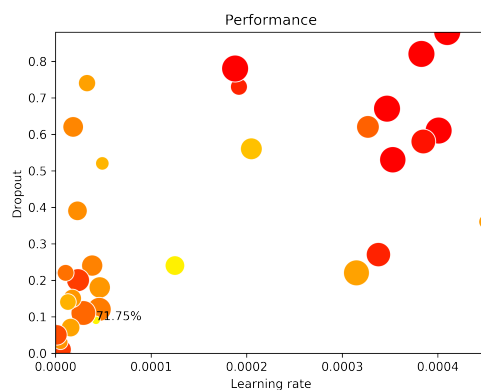


Figure 4.3: Results implementing dropout in only FC layers

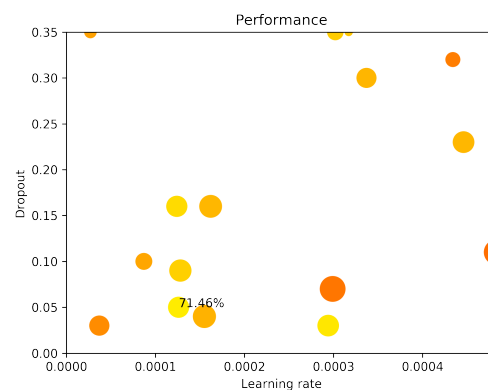


Figure 4.4: Results implementing dropout in all layers

The results shown in Figure 4.3 indicates that larger dropout probabilities generally decrease validation performance. Overall comparison of Figures 4.3 and 4.4 suggest that implementing dropout layers in both convolutional and FC layers might be favorable during further experimentation.

4.1.4 Experiment - Bayesian optimization

Hyperparameter tuning is a major part of any DNN design process. As hyperparameters might be more or less correlated, comparing different setups should thus be conducted by searching over a reasonable span of relevant hyperparameters. Searching the hyperparameter space using techniques like grid search or random search might be quite time consuming, especially when using CV. In the case of the linear-layer dropout experiment, the average elapsed time for each execution was 8 hours and 53 minutes when running on one Nvidia V100 GPU.

A Bayesian hyperparameter technique provided by SciKit-learn was therefore utilized to hopefully increase the efficiency of the hyperparameter search. This algorithm was used with a Gaussian prior distribution. The hyperparameters chosen for tuning was the L2 regularization parameter, the learning rate, and the dropout probability. Dropout was utilized in all layers. Data augmentation was implemented as well, with images being smoothed with a 3x3x3 Gaussian kernel and a sigma of one. Each image was given a probability of being smoothed which was tuned by the Bayesian tuning algorithm as well, allowing for a continuous scale. Used span for all mentioned hyperparameters is presented in Table 4.3.

Table 4.3: Hyperparameter space used for Bayesian optimization experiments

Learning rate	Dropout probability	L2 regularisering	Smoothing probability
$1 \cdot 10^{-5} - 1 \cdot 10^{-3}$	0 - 0.5	0 - 0.4	0 - 1

Results

Table 4.4: Results from Bayesian optimization experiments

Iteration	Learning Rate	Dropout probability	L2	Smoothing probability	Accuracy[%]	Loss
5	2.97e-05	0.20	0.155	0.67	73.2	0.729
8	8.47e-05	0.40	0.062	0.11	70.0	0.732
3	8.21e-05	0.22	0.376	0.78	68.7	0.746
2	7.71e-05	0.11	0.214	0.91	68.4	0.802
6	1.51e-05	0.43	0.177	0.13	66.7	0.762
7	0.000493	0.45	0.094	0.01	48.9	1.036
1	0.000253	0.37	0.255	0.78	48.6	1.037
4	0.000743	0.42	0.125	0.52	45.9	1.063
0	0.000987	0.47	0.051	1.00	45.8	1.052

Table 4.4 presents the results of all iterations from the Bayesian optimization experiment.

4.1.5 Experiment - Augmentation

A major challenge when training DNNs on neuroimages is the lack of available data. An experiment using additional data augmentation was therefore conducted.

The biomarkers presented in section 2.1.1 are all based on atrophy of different parts of the brain. With all images being fit to the same template during preprocessing, this would mean that most of these markers probably appear in approximately the same location and with roughly the same orientation throughout the dataset. The choice of augmentation technique was based on the idea that keeping relevant features as similar as possible between images would make them more likely to be learned by the model. For this reason, experiments using techniques like rotation, flipping, and cropping were not pursued.

With at least two important biomarkers, the hippocampus and the ventricles, being somewhat symmetrical about the sagittal plane, each scan was split in two along this exact plane. The different sectional planes of the human brain are shown in Figure 4.5. With the mentioned symmetry in mind, half of the resulting images were flipped along the transverse axis to hopefully increase the similarity between the features found in each half. The experiment was otherwise conducted in the same fashion as the Bayesian experiment.

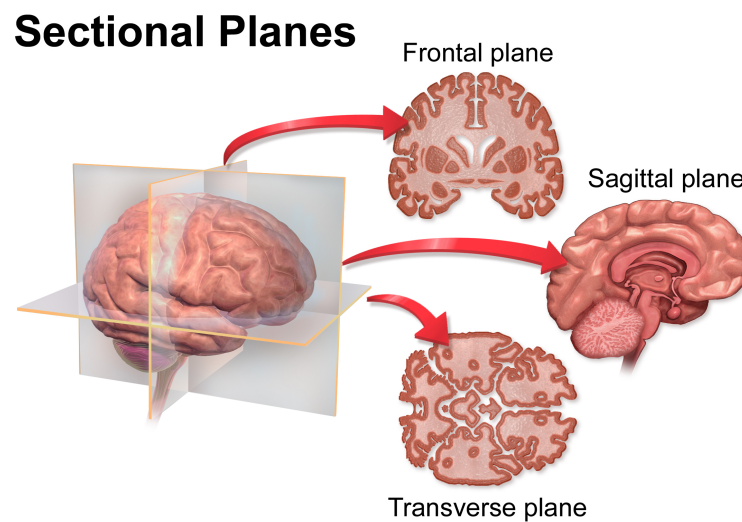



Figure 4.5: Sectional planes of the human brain¹

¹ This work is licensed under a [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by/3.0/). Author: [Bruce Blaus](https://www.blausen.com/) Blausen.com staff (2014). "Medical gallery of Blausen Medical 2014". WikiJournal of Medicine 1 (2).DOI:10.15347/wjm/2014.010. ISSN 2002-4436

Results

Table 4.5: Results from augmentation experiment

Iteration	Learning Rate	Dropout probability	L2	Smoothing probability	Accuracy[%]	Loss
4	1.31e-05	0.34	0.237	0.67	69.2	0.718
1	7.71e-05	0.11	0.214	0.91	68.9	0.765
5	2.97e-05	0.20	0.155	0.67	68.5	0.761
2	0.000270	0.40	0.037	0.52	68.0	0.741
3	8.21e-05	0.22	0.376	0.78	66.4	0.763
8	0.000253	0.37	0.255	0.78	55.2	0.960
0	0.000494	0.45	0.094	0.01	54.8	0.963
6	0.000987	0.47	0.051	1.00	51.6	0.996
9	0.000743	0.42	0.125	0.52	50.1	1.023
7	0.000537	0.41	0.332	0.27	48.7	1.031

Table 4.5 presents average CV results from the augmentation experiment. The best accuracy of the augmentation experiment was 69.2%.

Table 4.6: CV results from the fourth iteration of the augmentation experiment

Model	Accuracy[%]	Loss
1	70.9	0.685
2	66.2	0.775
3	68.0	0.724
4	72.4	0.672
5	68.9	0.759
6	68.9	0.694
Mean	69.2	0.718
SD	2.17	0.042

4.2 Final evaluation

The six models which reported the best average CV validation performance during the previous experiments were chosen for final evaluation on the test set. These models were found during the Bayesian experiment. The models were all trained on different combinations of the same training samples, but none were evaluated on the same validation set. With an intuition similar to that of dropout, combining all these models could thus potentially increase performance. An ensemble model was therefore created using a majority voting rule. This meant that the predictions of all six models were compared for each subject, and the diagnosis which was predicted by most models was chosen for that given subject. Ties were broken with an equal probability game of chance.

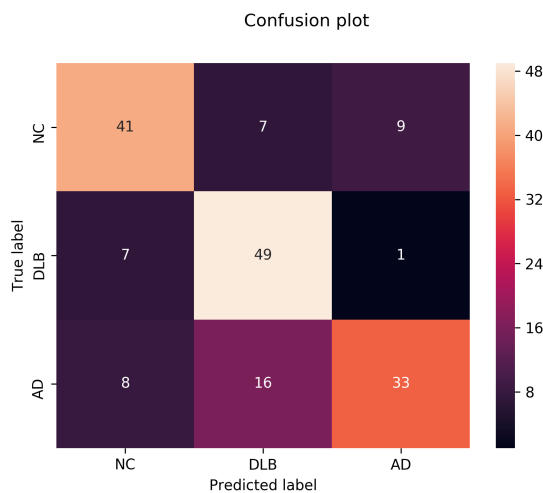


Table 4.7: Confusion matrix for final results

Table 4.8: Performance of final model

	NC	DLB	AD	Mean
Precision	0.732	0.681	0.767	0.727
Recall	0.719	0.860	0.579	0.719
F1	0.726	0.760	0.660	0.715

The final model diagnosed subjects with an overall accuracy of 71.9%. Table 4.8 and Figure 4.7 further describes the performance of the final classifier.

With the final model determined, the test performance for each of the six models could be established. The validation and test results for each fold are displayed in Tables 4.9 and Table 4.10 respectively.

Table 4.9: Final model CV validation results

Model	Accuracy[%]	Loss
1	67.5	1.000
2	77.8	0.571
3	76.3	0.652
4	71.1	0.838
5	72.8	0.796
6	73.7	0.790
Mean	73.2	0.729
SD	3.69	0.150

Table 4.10: Final model CV test results

Model	Accuracy[%]	Loss
1	69.6	0.842
2	69.6	0.795
3	66.1	0.972
4	66.7	0.764
5	60.2	0.923
6	70.8	0.744
Mean	67.2	0.840
SD	3.85	0.091

Chapter 5

Discussion

This chapter will discuss the results presented in chapter four, the final model's performance in regard to similar work, and the challenges and limitations encountered during this thesis.

5.1 Dementia classification

The proposed model of this thesis delivered an accuracy of 71.9%, and an average precision and recall of 0.73 and 0.72, respectively. Further analysis of Table 4.8 shows that the model reports a quite stable precision over the three classes, but struggles to recall AD subjects. Figure 4.7 indicates that the model confuses subjects suffering from AD with those diagnosed with DLB. Confusion of AD and DLB subjects could be expected with the possible presence of comorbidity and similarity between biomarkers. It does, however, in this case, seem to be a one-way street, with only one DLB subject being predicted as having AD. This might suggest that some features common for both AD and DLB are perceived as belonging to DLB. It is not obvious what would cause this, but one possibility is that the DLB subjects of the training set, for some reason, hold some general markers of dementia more prominently than the AD subjects. It could also be a pattern of noise in either the training or test set.

As shown by Tables 4.9 and 4.10, the test performance is generally lower than for the validation performance for the CV models, indicating that the models are able to overfit the validation set during training. The SD of the accuracy is, however, about the same for both the validation and the test results. This, together with the fact that the different models' relative performance on the validation set does not seem to relate to that reported on the test set might suggest that each model picks up some of the same, but also different features of the general distribution. The fact that the model ensemble performed substantially better than the average of the CV models further strengthens this hypothesis.

The average accuracy of the augmentation experiments did not live up to those of the Bayesian and was thus not chosen for the final evaluation. Table 4.6 shows, however, that the SD of the best augmentation CV experiment is significantly smaller than that of the chosen Bayesian experiment. This indicates that more data, or potentially, harder regularization might result in better generalization. Both the observations of this and the previous paragraph indicates that there might be unused potential in the presented setup.

As mentioned earlier, little effort has been directed towards MRI-based differential diagnosis of different types of dementia, and there are thus not many reports on the classification of these diseases. The two earlier mentioned studies[27, 29] have, however, reported results for the same three class problem as investigated in this thesis. The reported results and dataset characteristics of these reports, as well as those of this thesis, can be seen in Table 5.1.

Table 5.1: Results of this thesis and related work

	Accuracy[%]	Avg. precision	Avg. recall	Size of dataset
This thesis	72	0.73	0.72	861
[27]	73	0.78	0.73	48
[29]	87	0.88	0.87	109

Comparing studies of this sort is not a straight forward task as they often use different models, different datasets, and different preprocessing. As the two earlier

mentioned papers used approaches and datasets different from the one of this thesis, a direct comparison of results should be made with reservations. The overall performance reported in this thesis looks, nevertheless, to be almost on par with the performance reported by Wada et al.[27] and below that of Oppedal[29] et al. It is also worth mentioning that these studies report average CV-results, which might further weaken the comparison.

5.2 Limitations

This section will cover some limitations of this study.

5.2.1 Dataset

The use of a cross-center dataset could be considered a strength of this study as it, with its lower likelihood of consistent noise-specific patterns, might improve generalization. The fact that all data labeled as DLB were from the E-DLB consortium could, however, bias the model as all subjects provided by E-DLB are from Europe and all ADNI subjects are from North-America. The differences between scans from these two demographics are, however, probably not substantial. There is also no good way to check if this affects the model without access to a demographically balanced dataset.

During supervised training, a classifier is optimized to model the relationship between the input data and the corresponding labels. For dementia classification, this means that the optimal model would classify subjects with the same accuracy as the procedures of the clinics represented in the dataset. As clinical diagnosis is imperfect and comorbidity not uncommon, there is a high probability of multiple images in the used dataset containing biomarkers from both AD and DLB or even worse, having an altogether wrong diagnosis. If the relative amount of such compromised data is substantial, the generalization of a model could be significantly reduced as these samples would weaken the patterns of the general distribution in

the dataset. There is, however, no viable way of knowing how much of a problem this is. Expert assessments could be used to correct labels and discard uncertain images, but comorbidity might still be a problem as it could be hard to determine clinically.

5.2.2 Preprocessing

Both training and test scans used for the experiments conducted in this thesis were preprocessed in order to reduce the amount of unwanted information. Random inspection of some preprocessed images showed brains that appeared to be of similar size, rotation, translation, and crop. There is, nonetheless, a chance of inconsistencies in the preprocessing affecting the final results. The spatial normalization, e.g., might have caused a somewhat volumetric stretching and warping as it fits brains of different sizes and shapes into a fixed template. Manual inspection of all images could potentially be used to verify the consistency of the preprocessed dataset and remove possible outlier. This type of manual inspection was, unfortunately, not an option with the limited time available while writing this thesis. There is also a probability of some unfortunate effects of the spatial normalization being hard to perceive by an untrained eye.

5.2.3 Classifier

The classifier proposed in this thesis was inspired by a conventional CNN setup. The presented structure was determined and then set for all experiments conducted. As this model indicated a high variance problem, it was, as common practice in DL, attempted regularized by multiple means. This is, however, not the only way to combat overfitting. The structure itself was never altered, and this could potentially have limited the final result.

Any ANN design procedure can be considered an optimization problem, and with close to infinitely many possible combinations of network structures, hyperparameters, augmentation techniques, etc. to choose from, time becomes an essential

factor. With the use of time-consuming techniques like 3D convolution and CV, the experiments conducted during the course of this thesis came to be of a limited scope. The Bayesian and augmentation experiments, in particular, could potentially yield significantly better models with more time, as their Bayesian optimization algorithms were allowed only a minimal amount of iterations.

Chapter 6

Conclusion and future directions

This chapter will conclude this thesis and suggest directions for future work.

6.1 Conclusion

In this thesis, the potential of DL-based systems for reliable differential diagnosis of AD and DLB was explored. The proposed CNN model consisted of six 3D convolutional layers with corresponding max pooling layer, all followed by three FC layers. The model was trained on a class-balanced dataset of 690 T1 weighted sMRI scans. All samples were both gender- and age-matched to further balance the dataset. Preprocessing, in the form of skull-stripping and spatial normalization, was conducted to reduce the amount of undesired information or noise in the dataset.

A set of experiments was conducted in the search for an optimal classifier, including experiments with regularization, data augmentation, and hyperparameter optimization. With overfitting being the main challenge, all experiments were executed using six-fold CV.

The resulting ensemble model was evaluated on a separate, unseen, and balanced test set of 171 scans. A majority voting rule was used to combine the test results

of the six models that yielded the best average validation performance. A 71.9% overall accuracy, a 0.73 average precision, and a 0.73 average recall was reported. The results of this thesis indicate that there indeed is a potential in the use of DL-based CAD systems for differential diagnosis of dementia.

6.2 Future directions

The limited amount of data is a central challenge for most DL-based neuroimaging research. Using some form of a pre-trained model has shown promising results[59, 62] and should be investigated further.

An approach that might be worth exploring is to initially pre-train a model to predict age, before fine-tuning the model to dementia classification using datasets like the one presented in this thesis. With age being a significant risk factor associated with dementia, a model pre-trained to predict age on a large dataset could learn relevant features for dementia classification. As most studies register the subject's age when performing an MRI scan, a relatively large dataset could possibly be obtained.

Another challenge of dementia classification is the quality of the labeled data currently available. Even with large-scale studies like ADNI, OASIS¹, AILB², and E-DLB providing large amounts of high-quality images will the performance of CAD systems still be limited to mimicking that of current day clinical procedures. Using scans of subjects with an autopsy-confirmed diagnosis would most likely improve the quality of such datasets substantially. With the currently increasing amount of available data, creating such datasets could eventually be feasible, and would hold great promise.

¹Open Access Series of Imaging Studies

²The Australian Imaging, Biomarker & Lifestyle Flagship Study of Ageing

List of Figures

1.1	Plaque and tangles associated with AD	2
1.2	Intraneural Lewy-body	3
2.1	Brain of a healthy subject	9
2.2	Brain of a subject with AD diagnosis	9
2.3	Brain of a subject with DLB diagnosis	9
2.4	Brains of subjects belonging to NC, AD or DLB	9
2.5	Healthy and AD affected brain	10
2.6	Parts indicated on coronal slice of the brain	11
2.7	Illustration of a biological neuron	14
2.8	Multi-layer perceptron	15
2.9	3D convolution operation	17
2.10	Loss landscape	21
2.11	Plot of sigmoid function	23
2.12	Plot of ReLU and ReLU6	24
2.13	Model over- and underfitting on regression problem	25
2.14	Bias/Variance trade-off	27
2.15	5-fold cross-validation	28
2.16	Dropout implemented in a MLP	30
2.17	Example of a confusion matrix	35
3.1	Age distribution in dataset	40
3.2	Nipype workflow for image preprocessing	42
3.3	Proposed network structure	44
4.1	Learning curves during initial testing	47
4.2	Result of learning rate experiment	49
4.3	Dropout experiment result. Dropout only in FC layers	50
4.4	Dropout experiment result. Including dropout in convolutional layers	50
4.5	Sectional planes of the human brain	53

List of Tables

3.1	Characteristics of the final dataset	41
4.1	Results of initial testing	47
4.2	Experiment system configuration	49
4.3	Hyperparameter space for Bayesian optimization experiment	51
4.4	Results from Bayesian optimization experiment	52
4.5	Results from augmentation experiment	54
4.6	CV results from the fourth iteration of the augmentation experiment	54
4.7	Confusion matrix for final results	55
4.8	Performance of final model	55
4.9	Final model CV validation results	56
4.10	Final model CV test results	56
5.1	Results of this thesis and related work	58

Appendix A

Python code



The Python scripts used over the course of this thesis.

A.1 `fit.py`

This file contains the main setup for executing a model evaluation. A model is trained until the best validation performance is achieved. The file contains a single function, `model_fit`, which takes a python dictionary as input. The dictionary should contain a certain set of parameters describing the given execution. The main characteristics of the function are listed below.

- Allows for the training of a single test/validation split or the use of CV
- Supports the use of Nvidia GPUs
- Can automatically create a compatible dataset on Numpy format
- Can be set to perform classification of two or three classes
- Saves all results systematically to a given path

Supported hyperparameters:

- Batch size for training and validation
- Shuffle of data during training
- Number of CV folds
- Probability of smoothing the input
- Whether to use BN in the model structure or not
- Maximum number of epochs
- Learning rate
- L2 regularization
- Patience
- Dropout probability
- Patience

A.2 `main_setup.py`

File for executing experiments. Setup to accommodate the use of the `model_fit` function. Includes a predefined dictionary setup for all parameters compatible with the `model_fit` function.

A.3 `system_resources.py`

This file contains the functionality used by `fit.py` and `fit.py`. The model proposed in this thesis implemented in this file. The different functions and classes found in this file are listed below with a short description.

- **Simen_net:**
The proposed model

- **Split_net:**
A copy of Simen_net which accommodates the used of split images
- **train:**
Trains a given model with set up with the provided parameters
- **cross_validation:**
Shuffles and splits dataset, trains and evaluates models
- **validaton:**
Validates a given model on a given dataset
- **device_setup:**
Sets up connection with Nvidia GPUs
- **create_dataset:**
Constructs a compatible numpy dataset
- **nii_to_numpy:**
Collects and converts images on the NIFTI format before saving them
- **load_data_folders:**
Loads data for a single random split excecution
- **evaluation:** Computes and saves model performance from targets and predictions
- **training_plot:**
Saves plots of training statistics
- **cv_results:**
Computes the performance of a CV execution

A.4 test.py

File containing a setup for testing a given model. Contains functions from `system_resources.py`.

A.5 data_resources.py

A file containing code for creating a file containing the specific metadata needed for the *create_dataset* function. The file contains two functions, one for matching subjects by optimizing for a given mean and SD, and one for matching subjects individually.

A.6 NormalizeSkullStripPipeline.py

The code for running the NiPype pipeline used during preprocessing is found in this file. The file is set up to preprocess data from a single folder. The given folder should contain one subfolder for each subject.

Bibliography

- [1] About dementia | alzheimer's disease international. <https://www.alz.co.uk/about-dementia>. (Accessed on 06/17/2019).
- [2] Life expectancy - our world in data. <https://ourworldindata.org/life-expectancy>. (Accessed on 06/17/2019).
- [3] Dementia. <https://www.who.int/en/news-room/fact-sheets/detail/dementia>. (Accessed on 06/17/2019).
- [4] Martin James Prince. *World Alzheimer Report 2015: the global impact of dementia: an analysis of prevalence, incidence, cost and trends*. Alzheimer's Disease International, 2015.
- [5] Alzheimer's disease | alzheimer's disease international. <https://www.alz.co.uk/info/alzheimers-disease>, . (Accessed on 06/17/2019).
- [6] Alzheimers sykdom - store medisinske leksikon. https://sml.snl.no/Alzheimers_sykdom, . (Accessed on 06/17/2019).
- [7] Alzheimer's disease fact sheet. <https://www.nia.nih.gov/health/alzheimers-disease-fact-sheet>, . (Accessed on 06/17/2019).
- [8] What is alzheimer's | alzheimer's association. <https://www.alz.org/alzheimers-dementia/what-is-alzheimers>, . (Accessed on 06/17/2019).
- [9] Lewy body dementia (lbd) | symptoms & treatments. <https://www.alz.org/alzheimers-dementia/what-is-dementia/types-of-dementia/lewy-body-dementia>, . (Accessed on 06/17/2019).

-
- [10] What is lewy body dementia? <https://www.nia.nih.gov/health/what-lewy-body-dementia>, . (Accessed on 06/17/2019).
- [11] Demens med lewylegemer - store medisinske leksikon. https://sml.snl.no/demens_med_lewylegemer. (Accessed on 06/17/2019).
- [12] Lewy body dementia - symptoms and causes - mayo clinic. <https://www.mayoclinic.org/diseases-conditions/lewy-body-dementia/symptoms-causes/syc-20352025>, . (Accessed on 06/17/2019).
- [13] Parkinson's disease - symptoms and causes - mayo clinic. <https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055>. (Accessed on 06/24/2019).
- [14] Thien Kieu Thi Phung, Birgitte Bo Andersen, Lars Vedel Kessing, Preben Bo Mortensen, and Gunhild Waldemar. Diagnostic evaluation of dementia in the secondary health care sector. *Dementia and geriatric cognitive disorders*, 27(6):534–542, 2009.
- [15] Thomas G Beach, Sarah E Monsell, Leslie E Phillips, and Walter Kukull. Accuracy of the clinical diagnosis of alzheimer disease at national institute on aging alzheimer disease centers, 2005–2010. *Journal of neuropathology and experimental neurology*, 71(4):266–273, 2012.
- [16] Craig A Hunter, Noam Y Kirson, Urvi Desai, Alice Kate G Cummings, Douglas E Faries, and Howard G Birnbaum. Medical costs of alzheimer's disease misdiagnosis among us medicare beneficiaries. *Alzheimer's & Dementia*, 11(8):887–895, 2015.
- [17] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

- [18] G Zaharchuk, E Gong, M Wintermark, D Rubin, and CP Langlotz. Deep learning in neuroradiology. *American Journal of Neuroradiology*, 39(10):1776–1784, 2018.
- [19] Md Rishad Ahmed, Yuan Zhang, Zhiquan Feng, Benny Lo, Omer T Inan, and Hongen Liao. Neuroimaging and machine learning for dementia diagnosis: Recent advancements and future prospects. *IEEE reviews in biomedical engineering*, 12:19–33, 2018.
- [20] Stefan Klöppel, Cynthia M Stonnington, Carlton Chu, Bogdan Draganski, Rachael I Scahill, Jonathan D Rohrer, Nick C Fox, Clifford R Jack Jr, John Ashburner, and Richard SJ Frackowiak. Automatic classification of mr scans in alzheimer’s disease. *Brain*, 131(3):681–689, 2008.
- [21] Emilie Gerardin, Gaël Chételat, Marie Chupin, Rémi Cuingnet, Béatrice Desgranges, Ho-Sung Kim, Marc Niethammer, Bruno Dubois, Stéphane Lehéricy, Line Garnero, et al. Multidimensional classification of hippocampal shape features discriminates alzheimer’s disease and mild cognitive impairment from normal aging. *Neuroimage*, 47(4):1476–1486, 2009.
- [22] Jongin Kim and Boreom Lee. Automated discrimination of dementia spectrum disorders using extreme learning machine and structural t1 mri features. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1990–1993. IEEE, 2017.
- [23] Lauge Sørensen, Mads Nielsen, Alzheimer’s Disease Neuroimaging Initiative, et al. Ensemble support vector machine classification of dementia using structural mri and mini-mental state examination. *Journal of neuroscience methods*, 302:66–74, 2018.
- [24] Lauge Sørensen, Christian Igel, Akshay Pai, Ioana Balas, Cecilie Anker, Martin Lillholm, Mads Nielsen, Alzheimer’s Disease Neuroimaging Initiative, et al. Differential diagnosis of mild cognitive impairment and alzheimer’s disease using structural mri cortical thickness, hippocampal shape, hippocampal texture, and volumetry. *NeuroImage: Clinical*, 13:470–482, 2017.

- [25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [26] Sandra Vieira, Walter HL Pinaya, and Andrea Mechelli. Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. *Neuroscience & Biobehavioral Reviews*, 74:58–75, 2017.
- [27] Akihiko Wada, Kohei Tsuruta, Ryusuke Irie, Koji Kamagata, Tomoko Maekawa, Shohei Fujita, Saori Koshino, Kanako Kumamaru, Michimasa Suzuki, Atsushi Nakanishi, et al. Differentiating alzheimer’s disease from dementia with lewy bodies using a deep learning technique based on structural brain connectivity. *Magnetic Resonance in Medical Sciences*, pages mp–2018, 2018.
- [28] Teng Xie and Yong He. Mapping the alzheimer’s brain with connectomics. *Frontiers in psychiatry*, 2:77, 2012.
- [29] Ketil Oppedal, Trygve Eftestøl, Kjersti Engan, Mona K Beyer, and Dag Aarsland. Classifying dementia using local binary patterns from different regions in magnetic resonance images. *Journal of Biomedical Imaging*, 2015:5, 2015.
- [30] Mr-undersøkelse - store medisinske leksikon. <https://sml.snl.no/MR-unders%C3%B8kelse>. (Accessed on 06/17/2019).
- [31] Peter mansfield - store norske leksikon. https://snl.no/Peter_Mansfield. (Accessed on 06/17/2019).
- [32] Abi Berger. Magnetic resonance imaging. *BMJ*, 324(7328):35, 2002. ISSN 0959-8138. doi: 10.1136/bmj.324.7328.35. URL <https://www.bmj.com/content/324/7328/35>.
- [33] IG McKeith, Dennis W Dickson, J Lowe, M Emre, JT O’Brien, H Feldman, J Cummings, JE Duda, C Lippa, EK Perry, et al. Diagnosis and management

- of dementia with lewy bodies: third report of the dlb consortium. *Neurology*, 65(12):1863–1872, 2005.
- [34] Marilyn S Albert, Steven T DeKosky, Dennis Dickson, Bruno Dubois, Howard H Feldman, Nick C Fox, Anthony Gamst, David M Holtzman, William J Jagust, Ronald C Petersen, et al. The diagnosis of mild cognitive impairment due to alzheimer’s disease: Recommendations from the national institute on aging-alzheimer’s association workgroups on diagnostic guidelines for alzheimer’s disease. *Alzheimer’s & dementia*, 7(3):270–279, 2011.
- [35] Cerebral atrophy information page | national institute of neurological disorders and stroke. <https://www.ninds.nih.gov/Disorders/All-Disorders/Cerebral-atrophy-Information-Page>. (Accessed on 06/17/2019).
- [36] Federica Agosta, Sebastiano Galantucci, and Massimo Filippi. Advanced magnetic resonance imaging of neurodegenerative diseases. *Neurological Sciences*, 38(1):41–51, 2017.
- [37] KM Gosche, JA Mortimer, CD Smith, WR Markesbery, and DA Snowdon. Hippocampal volume as an index of alzheimer neuropathology: findings from the nun study. *Neurology*, 58(10):1476–1482, 2002.
- [38] Pierrick Coupé, José Vicente Manjón, Enrique Lanuza, and Gwenaëlle Catheline. Lifespan changes of the human brain in alzheimer’s disease. *Scientific reports*, 9(1):3998, 2019.
- [39] EJ Burton, R Barber, EB Mukaetova-Ladinska, J Robson, RH Perry, E Jaros, RN Kalaria, and JT O’Brien. Medial temporal lobe atrophy on mri differentiates alzheimer’s disease from dementia with lewy bodies and vascular cognitive impairment: a prospective study with pathological verification of diagnosis. *Brain*, 132(1):195–203, 2008.
- [40] R. Barber, A. Gholkar, P. Scheltens, C. Ballard, I.G. McKeith, and J.T. O’Brien. Medial temporal lobe atrophy on mri in dementia with lewy bodies. *Neurology*, 52(6):1153–1153, 1999. ISSN 0028-3878. doi: 10.1212/WNL.52.6.1153. URL <https://n.neurology.org/content/52/6/1153>.

-
- [41] Elijah Mak, Li Su, Guy B Williams, Rosie Watson, Michael Firbank, Andrew Blamire, and John O'Brien. Differential atrophy of hippocampal subfields: a comparative study of dementia with lewy bodies and alzheimer disease. *The American Journal of Geriatric Psychiatry*, 24(2):136–143, 2016.
- [42] Prashanthi Vemuri, Gyorgy Simon, Kejal Kantarci, Jennifer L Whitwell, Matthew L Senjem, Scott A Przybelski, Jeffrey L Gunter, Keith A Josephs, David S Knopman, Bradley F Boeve, et al. Antemortem differential diagnosis of dementia pathology using structural mri: Differential-stand. *Neuroimage*, 55(2):522–531, 2011.
- [43] Michael J Firbank, Rosie Watson, Elijah Mak, Benjamin Aribisala, Robert Barber, Sean J Colloby, Jiabao He, Andrew M Blamire, and John T O'Brien. Longitudinal diffusion tensor imaging in dementia with lewy bodies and alzheimer's disease. *Parkinsonism & related disorders*, 24:76–80, 2016.
- [44] Simon Wiesler and Hermann Ney. A convergence analysis of log-linear training. pages 657–665, 12 2011.
- [45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [46] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7), 2010.
- [47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [49] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

- [50] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.
- [51] Jorge Samper-Gonzalez, Ninon Burgos, Sabrina Fontanella, Hugo Bertin, Marie-Odile Habert, Stanley Durrleman, Theodoros Evgeniou, Olivier Colliot, Alzheimer’s Disease Neuroimaging Initiative, et al. Yet another adni machine learning paper? paving the way towards fully-reproducible research on classification of alzheimer’s disease. In *International Workshop on Machine Learning in Medical Imaging*, pages 53–60. Springer, 2017.
- [52] Eli Gibson, Wenqi Li, Carole Sudre, Lucas Fidon, Dzhoshkun I Shakir, Guotai Wang, Zach Eaton-Rosen, Robert Gray, Tom Doel, Yipeng Hu, et al. Niftynet: a deep-learning platform for medical imaging. *Computer methods and programs in biomedicine*, 158:113–122, 2018.
- [53] Andrew Beers, James Brown, Ken Chang, Katharina Hoebel, Elizabeth Gerstner, Bruce Rosen, and Jayashree Kalpathy-Cramer. Deepneuro: an open-source deep learning toolbox for neuroimaging. *arXiv preprint arXiv:1808.04589*, 2018.
- [54] Krzysztof Gorgolewski, Christopher D Burns, Cindee Madison, Dav Clark, Yaroslav O Halchenko, Michael L Waskom, and Satrajit S Ghosh. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Frontiers in neuroinformatics*, 5:13, 2011.
- [55] Mark W Woolrich, Saad Jbabdi, Brian Patenaude, Michael Chappell, Salima Makni, Timothy Behrens, Christian Beckmann, Mark Jenkinson, and Stephen M Smith. Bayesian analysis of neuroimaging data in fsl. *Neuroimage*, 45(1):S173–S186, 2009.
- [56] John Ashburner and Karl J Friston. Unified segmentation. *Neuroimage*, 26(3):839–851, 2005.
- [57] Gary D Knott. *Interpolating cubic splines*, volume 18. Springer Science & Business Media, 2012.

-
- [58] Stephen M Smith. Fast robust automated brain extraction. *Human brain mapping*, 17(3):143–155, 2002.
- [59] Ehsan Hosseini-Asl, Georgy Gimel'farb, and Ayman El-Baz. Alzheimer's disease diagnostics by a deeply supervised adaptable 3d convolutional network. *arXiv preprint arXiv:1607.00556*, 2016.
- [60] Silvia Basaia, Federica Agosta, Luca Wagner, Elisa Canu, Giuseppe Magnani, Roberto Santangelo, Massimo Filippi, Alzheimer's Disease Neuroimaging Initiative, et al. Automated classification of alzheimer's disease and mild cognitive impairment using a single mri and deep neural networks. *NeuroImage: Clinical*, 21:101645, 2019.
- [61] Adrien Payan and Giovanni Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*, 2015.
- [62] Marcia Hon and Naimul Mefraz Khan. Towards alzheimer's disease classification through transfer learning. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1166–1169. IEEE, 2017.