SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence in Informatics

Learning on Cortical Meshes with Surface Vision Transformers (SiT)

Moritz Schüler

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence in Informatics

Learning on Cortical Meshes with Surface Vision Transformers (SiT)

Lernen auf Kortex-Oberflächen mit Surface-Vision-Transformern (SiT)

Author:	Moritz Schüler
Supervisor:	Prof. Dr. Christian Wachinger
Advisor:	Fabian Bongratz, M.Sc.
Submission Date:	15.06.2023

I confirm that this master's thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Garching, 15.06.2023 Moritz Schüler

Acknowledgments

I would like to start by thanking Fabian Bongratz for the weekly support, the explanations and guidance during my thesis. Also thanks to Ignacio Sarasúa for his support until the end of his Phd. Have fun in Barcelona! Further, I want to thank the AI-Med team and especially Prof. Christian Wachinger for welcoming me to this project. Thank you all for your feedback, ideas and overall the interesting discussions we had. Big thanks to all my friends in Munich, who made the studies more fun and the group projects always a success. Happy to enjoy more time with you guys in Munich! Special shout out to Markus for lending me his couch during appartment search and to Paula for the support during my writing of the thesis.
Finally, I want to thank my family for the countless proof-reading, for them always believing in me and supporting me in whichever idea comes next. Especially if it involves an attic appartment without an elevator. New record to be set for climbed floors!

Abstract

The analysis of cortical meshes from brain structural magnetic resonance imaging (MRI) scans is essential for the early diagnosis and treatment of many brain-related diseases, like Alzheimer's Disease (AD). Many techniques to classify different neurological degeneratives already exist, but usually a lot of domain knowledge and fine-tuned preprocessing steps are needed to get results for a specific disease. In this work, we try to address these issues by evaluating the transfer abilities of a promising method, called Surface Vision Transformer (SiT), that works on 3D MRI scans, requiring only standardized preprocessing by established components (FreeSurfer). SiT uses a projection onto the spherical manifold in combination with a vision transformer architecture to model long-range associations in the data and overcome the limitations of graph convolutions. We evaluate the performance of the method on three brain MRI datasets on the tasks of brain age prediction and Alzheimer's disease classification and show, that competitive results are possible in the regression setting. However, more work is needed to tackle the state of the art for the classification task. The code is available from Github.

Keywords: Geometric Deep Learning, Surface Transformer, Alzheimer, Cortical Analysis, Deep Learning, Attention-based Modelling, Neuroimaging

Contents

1 Introduction

Alzheimer's Disease (AD) is a degenerative neurological disorder that progressively destroys memory and cognitive abilities. The symptoms can become so severe that patients require support with basic everyday tasks. It is the most common form of dementia, accounting for up to 70% of all dementia cases [1]. AD is a major public health issue with prevalence increasing as the population ages. It is among the ten leading causes of death worldwide together with heart diseases and cancer [2]. Over 50 million people, dated 2020, are already living with dementia and statistics show that the amount is almost doubling every 20 years [3], [4]. The *Alzheimer's Disease International* even claims that there are nearly ten million new cases each year [5] despite most people having not received formal diagnosis [6]–[8].
This is especially important as the *World Alzheimer Report* 2011 shows that earlier diagnosis is a very important mechanism for a good course of the disease [9]. Fortunately, researchers made substantial progress in this era due to the identification of several biomarkers [10], [11], that improve the detection of cortical atrophy in its initial stages [12]–[14] and help to diagnose Mild Cognitive Impairment (MCI), a pre-form of AD.

In general, these biomarkers are measurements taken from 3D representations of the brain, which are generated using MRI. Exemplary markers for brain analysis include cortical thickness, gyri, sulci, volume or myelin maps, which shows demyelination that can be a sign for certain neurological conditions, such as multiple sclerosis [15] or AD [16]. In figure 1.1, representative magnetic resonance (MR) images illustrate the neurodegenerative changes present in a severe AD patient, a patient with MCI and a healthy person's brain. The way these images are built is by leveraging different tissue properties, like water concentration [17] when exposed to varying magnetic fields, e.g. longer or shorter frequency pulses (T1-/T2-weighing) [12]. Longer pulses produce images with good contrast between different tissue types, whereas shorter pulses help to differentiate fluid-filled spaces better [17].

Even with the naked eye, it's possible to distinguish between a healthy brain and a severely degenerated one. However, to distinguish between a patient with MCI and a healthy individual, reliable biomarkers are required because the differences are subtle. In order to ensure accurate measurements and maintain high quality, a standardized procedure has been developed. The initial step involves eliminating artifacts and noise

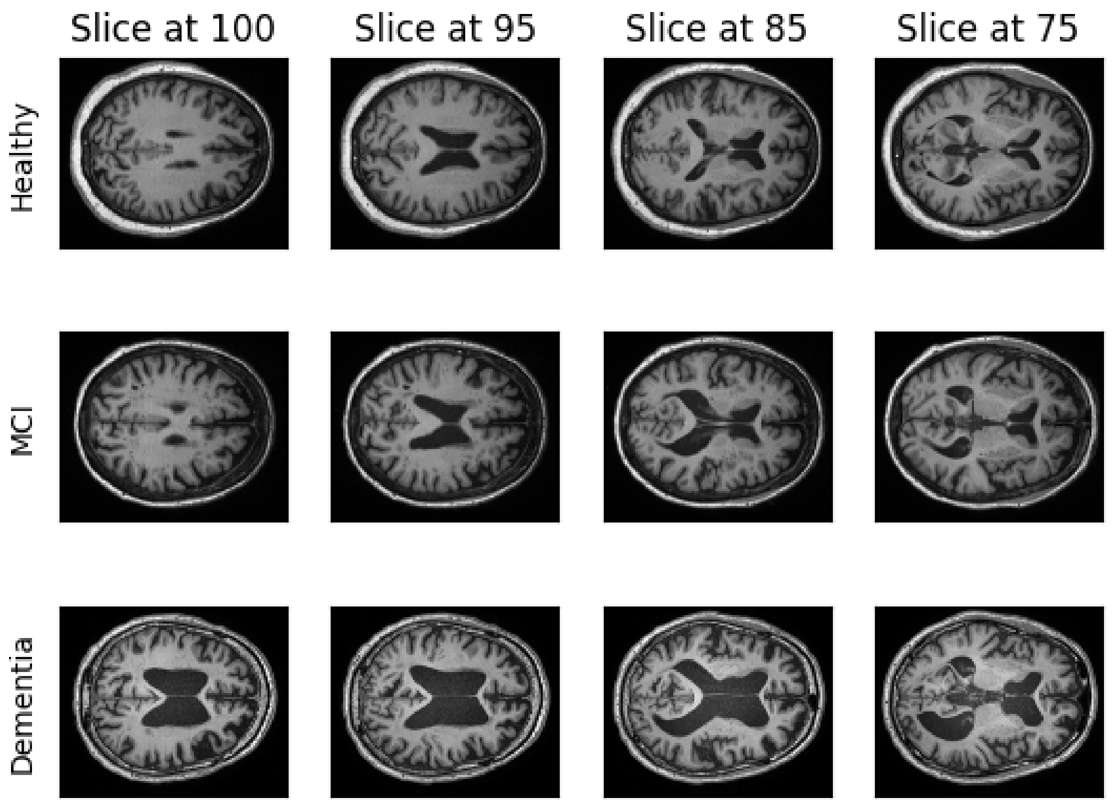


Figure 1.1: Axial cross sections from T2-weighted MRI images from the Alzheimer's Disease Neuroimaging Initiative (ADNI), available under https://adni.loni.usc.edu/. The top row shows a Cognitive Normal (CN) brain, the middle row a MCI and the bottom row a brain affected by AD. The cortical atrophy of an AD patient (bottom) compared to a healthy person's brain (top) is clearly visible, however, the difference between the healthy person and the patient with MCI is only subtle.

from the MR image, followed by segmentation. The brain is split by tissue type and brain structure, like gray matter, white matter and cerebrospinal fluid. For more detail, we refer to chapter 3.1, where the relevant types of brain tissue are described. In the early stages this was a manual process but thanks to the recent advancements in image processing several computation-based pipelines have been proposed [18]–[21], allowing to automate and accelerate this step.
Furthermore, this progress enabled the practical execution of large cohort studies and generating large databases. Together with the advent of Machine Learning (ML) and Deep Learning (DL) this led to the development of algorithms to support medical professionals on diagnosing brain diseases in their daily work.
In a first step, researchers extracted features for further analysis from the brain scans [22], [23], but soon, methods that leverage the full advantage of DL were trained end-to-end [24], [25]. First, standard image methods, like Convolutional Neural Network (CNN) were applied to 2D slices of the brain scans, while later even more sophisticated approaches, that work with the detailed 3D representation were developed [26]–[30]. These Geometrical Deep Learning (gDL) methods transfered the well-known convolution operation to non-Euclidian space and achieve to incorporate the geometrical information of the highly-folded surface into the model to improve the accuracy of the state of the art.
Particularly, the work of [30] combines the latest findings of gDL and Computer Vision (CV) to create a model working on the surface representation of the brain, whilst using the attention mechanism.

In this work, we first evaluate the performance of the SiT [30] approach on the task of brain age prediction from cortical surface metrics derived from the data of the Developing Human Connectdome Project (dHCP). Additionally we test the generalization properties on the datasets of Human Connectdome Project (HCP) and ADNI. Later, we adapt the architecture of the model to the classification problem of AD prediction using the dataset from ADNI and testing the transfer properties of the SiT architecture. Further, we show that the application of such a general, but complex models is not straightforward and elaborate on the pitfalls and solutions to successfully use these models for new tasks.

In summary, the main contributions of this work are the following:

- This paper proposes an adaption of the SiT model [30] to regression and classification problems.
- We evaluate the generalization capability of the SiT architecture on three different tasks, namely phenotype prediction on the dHCP dataset, brain age prediction

on the HCP and ADNI dataset, as well as AD prediction on the ADNI dataset.

- We show that the adapted SiT architecture achieves competitive results in the regression setting while being able to be applied on different datasets.

Research Question

The main objective of our work is to generalize the framework of the SiT to classification problems and evaluate the generalization performance of the architecture on different age groups and tasks.

2 Related Work

This chapter summarizes existing research efforts related to our method. Relevant work stems mostly from the field of image-based Deep Learning and Geometrical Deep Learning (gDL).

Deep learning-based methods have been widely applied in the field of image analysis, particularly for the task of image classification and object detection. These methods, which include convolutional neural networks (CNNs) [31], residual neural networks (ResNets) [32], recurrent neural networks (RNNs) [33]–[35], autoencoders [36], Graph Neural Network (GNN) [37] and transformers [38], [39], have been successful in improving the accuracy and efficiency of computer vision tasks. In the first chapter, we will explore two subcategories of deep learning-based methods: image-based methods and geometrical deep learning methods. Note, that this chapter focuses on the CV domain in general, while section 2.3 concentrates on related works for medical imaging.

2.1 Deep Learning-Based Methods for Computer Vision

The field of CV is a relatively young research field, dating back to 1957, when the first image was transformed into a grid of numbers [40]. Since then, continuous progress has been made, first connecting a camera to a computer [41], [42] and later adding face detection to it [43]. The rise of DL in the past years led to a huge jump in research and applications of CV when moving from hand-crafted features to data-extracted ones thanks to higher compute power and more efficient algorithms. The foundation for the first major breakthrough dates back to 1998, when Yann LeCunn introduced the CNN architecture, which is still one of the most dominant ones nowadays [31]. In 2012, AlexNet dominated the ImageNet challenge with a scaled up version of this architecture marking the start of the era of DL [44].

In the following sections, we review the existing literature in the field of CV and discuss the related works that are most relevant to our proposed method. We divide the related works into categories based on their application, namely Image Classification and Image Regression.

2.1.1 Image Classification

In Image Classification the task is to train a classifier, s.t. it can recognize and categorize certain objects in an image. The goal is to assign each input image a label or category based on the content of the image. We now further investigate important research in that domain starting by [31], [44] as the first milestones.
In 1998, LeCunn introduced the well-known CNN architecture, which made it possible to process images with neural networks. More specifically, it allowed data-driven features instead of handcrafted ones on the high-dimensional image domain. Previous architectures, e.g. fully connected neural networks were unfeasible to work on such high dimensional data because the amount of parameters was too big, making them untrainable. What LeCunn did to levitate that issue was to combine two techniques to reduce the number of parameters drastically. Firstly, one filter per layer is learned independent of the image size, secondly pooling layers aggregate information from several pixels, thus reducing the spatial dimensions of the feature maps [31]. By stacking several of those layers, a neural net can learn low level features, such as edges in the first few layers to also very complex features like shapes in the later layers.
This power was first demonstrated by [44] in 2012 for the ImageNet challenge. They used several CNN layers, as proposed by LeCunn, to win the challenge with a big lead. The main finding of the author was that if a high amount of data is present stacking several layers leads to an improved accuracy for the task of image classification. Inspired by that the authors of the VGGNet further increased the amount of layers but kept the overall number of parameters manageable by reducing the filter size [45]. Whilst creating State of the art (SOTA) results and winning the ImageNet challenge, they also showed that training deep models is challenging because of exploding and vanishing gradients [46].
[32] showed that deep networks struggle to learn the identity function because of the numerous non-linearities that are applied after each layer. Therefore, the authors idea was to introduce so called skip connections that allow the model to circumvent a non-linearity if needed. This architecture is called Residual Network (ResNet) and was the first to achieve super human performance in image classification. Additionally, this architecture solves the vanishing gradient problem and allows to train very deep models, even beyond 100 layers for the first time.
In the following years, mainly bigger networks were trained and compute power was the main factor for SOTA results, when in 2019 the authors of [47] came up with a new and improved scaling method for CNN achieving new benchmark records whilst also being more efficient. They discovered, that the interaction between width, depth and resolution has a huge impact on the network accuracy and thus introduced the compound scaling method, which is widely adopted in the computer vision domain

nowadays.
In the same year another very famous paper was published introducing the transformer architecture [38]. Despite being targeted at the Natural Language Processing (NLP) domain the idea of the paper was so successfull that it quickly got adapted to suit all domains. In NLP the main challenge was to process long text sequences as the last word might adhere to the first word of a sentence leading to long-range dependencies. Another problem in that domain was that for models such as Recurrent Neural Network (RNN) or Long Short Term Memory Network (LSTM) the computations are not easy to parallelize [38]. The Transformer solves these problems by using self-attention mechanisms, which allow the model to dynamically weigh the importance of different elements in a sequence. This idea translates to the image domain in that different pixels in an image could attend to one another whilst being further away than the filter size of a CNN filter.
However, translating the idea to the image domain remains challenging as the dimensionality of the data is larger that in the text domain. To solve this issue the authors of [39] published a variation for the algorithm to work on patches instead of pixels of images. The algorithm achieved SOTA performance on the benchmarking datasets of ImageNet [48], COCO [49] and Pascal VOC [50] and showed high transfer learning capabilities, making it a great backbone model for various computer vision tasks. An overview of the architecture is depicted in figure 2.1 below.

2.1.2 Image Regression

Despite classification there is also another really important task, which is regression. Here, the objective is to train a model, that predicts a continuous value instead of a specific class. In the realm of images, this means to either forecast some meta data of an image or to regress the pixel values for object segmentation, where the task is to find all pixels belonging to an object on the image [51]. In the context of this thesis, we will only concentrate on the first part and leave out the details on object segmentation apart from mentioning some resources for the interested reader [52]–[56]. The same architectures mentioned in the previous section can be used to predict meta data of an image, when applying some minor adjustments.
Firstly, the last layer needs to be adapted to output a continuous value instead of a softmax distribution. This is achieved by shrinking the size of the final layer to one and leaving out the activation function. Secondly, the loss function also needs to be changed to suit the different problem statement. The most common loss function for regression problems is the mean squared error, but others include the mean absolute error or the Huber loss [57]. Lastly, the evaluation metrics between classification and regression problems also differ a lot. On one hand, one can use a confusion matrix and

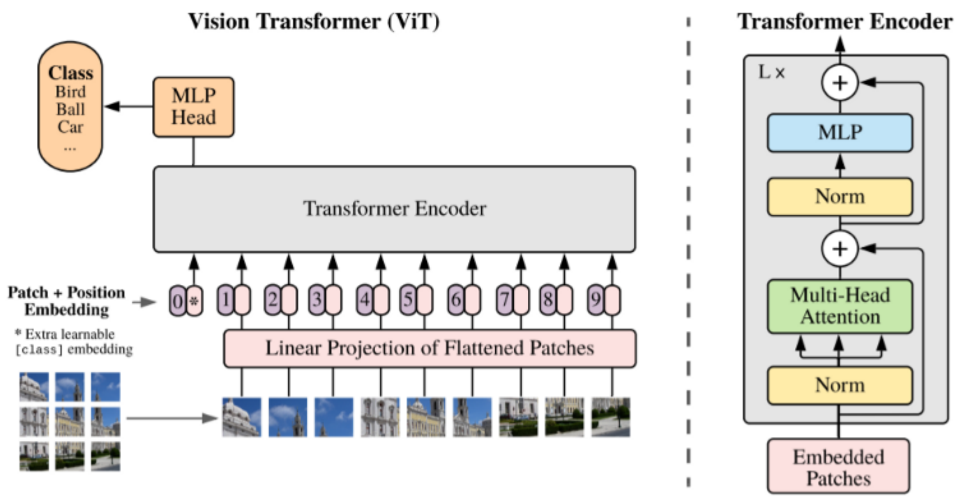


Figure 2.1: Model architecture of the vision transformer. Images are split into fixed-sized patches before feeding into a standard transformer encoder. Embedding 0 is the class label, which is needed to perform classification tasks. Image taken from [39].

calculate all sorts of metrics like precision, recall or accuracy in a classification szenario, whereas one has to resort to losses that minimize the distance between prediction and real value in the regression setting, like root mean squared error or R-squared [58].

2.2 Geometric Deep Learning

In chapter 2.1.1, we discussed the most influential work for image classification, however, images represent only 2D data and many applications handle more complex data, like graphs. In fact, the world wide web can be seen as a hyper text graph and also human friendship relations can be modeled as such. To process this data new approaches are needed, which are covered under the topic of geometric deep learning. In general, geometric deep learning focuses on learning representations and patterns in graph-structured and non-Euclidean data, such as point clouds, graphs, and manifolds. The first noteworthy work to create a neural network to process graph data was by [37]. The authors developed a model that takes into account information from neighboring nodes to update the own state. This recurrent idea propagates through the whole graph and captures the local and global relationships between the nodes in the graph. The graph can at the end be understood as some divided feature space where a classical

neural net architecture can be used to generate predictions.
Thanks to the success of convolutions in traditional deep learning the idea of using the locality of a node to improve the quality was quickly put into practice by [59], [60]. They introduced the Graph Convolutional Network (GCN) which represents the graph via an adjacency matrix and a feature matrix. By transforming the adjacency matrix the weighted sum of the neighboring nodes can be achieved by a simple matrix multiplication which is comparable to how a traditional CNN works and enabled the application on a wide range of new tasks, like social-media analysis or protein interactions [61], [62].
In 2017, the authors of [63] found a way to generalize the method for learning representations of nodes in graph-structured data, which they called "neural message passing." The main idea behind this approach is that each node in a graph sends messages to its neighboring nodes, which are then used to update the node's representations. The process of message passing is repeated iteratively, allowing the model to propagate information through the graph and build up a more complete understanding of the graph structure. The underlying idea is the same as for a GNN, but this generalized process allows to easily adapt the architecture to different tasks as it is trained end-to-end using a loss function that measures the quality of the graph representations generated by the model. Nowadays, if talking about GNN the model usually possesses the neural message passing architecture. The message passing operation is usually followed by a non-linear transformation, such as a neural network layer, which maps the node representation to a new representation and allows to generate predictions.

2.3 Deep Learning-Based Medical Imaging

The great success of the previously presented methods led to them being applied in a variety of fields. In the coming sections, we will concentrate specifically on the application in the medical imaging domain. Further detail will be put on the task of disease classification and how the task of brain age prediction can help for that. Most of the presented methods work on 2D representations, like MRI scans, which are a common practice used widely in the treatment and diagnosis process of head injuries or brain related diseases. In recent years, however, techniques developed that also work with 3D representations. Many of them were applied in the medical imaging domain, as 3D brain MRI scans are taken in the same clinical assessment and are thus readily available.

2.3.1 Image-Based Deep Learning

Image-based methods refer to approaches that are based on the analysis of images, such as CT scans or MRIs. These methods are often used to extract features from the images and classify them based on characteristics that are indicative of certain diseases or abnormalities.
The methods from the previous chapter can be directly applied in that szenario, therefore the improvements in the CV domain transfer directly to the medical domain. Example applications where CNNs were applied are the segmentation of different tissues in medical images, such as brain tissue or tumor tissue. They have also been used to classify medical images based on characteristics that are indicative of certain diseases or conditions, such as Alzheimer's disease or breast cancer [64]–[68]. Additionally, glsplCNN have been used to detect abnormalities in medical images, such as lesions or tumors [69]–[71]. The main innovation in the field stemmed from the transition of hand crafted specifically designed features [72], [73] to learning the features from the data automatically and having a generalization over several tasks [74], [75]. This change sped up the diagnosis process as well as improved the accuracy and efficiency.
With the rise of attention-based models the accuracy of certain applications could be drastically improved, as the model could focus on certain parts of the input image, rather than processing the entire input equally. This is especially helpful for tasks, in which pixelwise predictions are needed, like segmentation of tumor tissue [76]–[78].
Another approach for such tasks is to switch from a 2D to a 3D representation, as the data usually is available from the same clinical assessment. This often helps to improve the accuracy and robustness as more information is provided to the model in cost of computational power.

2.3.2 Geometrical Deep Learning

These 3D datapoints are usually handled in a graph representation and processed with gDL methods. The idea of those methods is to apply convolutions to 3D data. This means to apply location equivariant filters which translates to formulate a convolution in a way that it is rotation equivariant. Depending on the domain, where the convolution is computed on, these formulations can be classified into spectral and spatial methods.
The latter slides the filter over the data, which only leads to an approximate mathematically correct convolution as the curved surface leads to missorientation of the filter. These methods are expressive, but not fully rotation-equivariant. The spectral variants on the other hand work on the Fourier transform, which makes them accurate, but expensive to compute. They are also rotation-equivariant, but the even bigger downside

of this formulation is the expressiveness [79]. The main applications for gDL in the medical domain are analyzing anatomical shapes of organs, like the brain [30], [79], [80]. Further, [81], [82] showed that the cortical shape alone is a bad indicator of many facets of cognition and behaviour, therefore several features are extracted and the shape can be simplified to a sphere to ease computation and handling, which will be used in this work as well.
In the following the two tasks of brain age prediction and Alzheimer's disease prediction are highlighted and recent works in their applications are presented.

2.3.3 Brain Age Prediction

Brain age prediction is a method used to estimate the biological age of a person's brain based on various factors such as their genetics, lifestyle, and health history. Traditional methods used to extract features from MRIs, like curvature, volume, gray and white matter, as well as cerebrospinal fluid before regressing the actual brain age [83], [84]. Newer methods, however, can work on the brain scans directly, which mitigates the loss of information due to preprocessing [85].
Training a model to return the age of a patient is in most cases only a surrogate component. Most of the times the Predicted Age Difference (PAD) (difference between the estimated and chronological age) is calculated and used as a feature for more thorough analyis, like Alzheimer's disease classification. Several studies showed that structural changes related to ageing correlate with reduced mental and physical abilities of the patients and can be further linked to increased risk of Alzheimer's and Parkinson's disease [85]–[87].

There are a variety of algorithms that have been used for the task of brain age prediction in medical imaging. One common approach is to use machine learning techniques, such as support vector machines (SVMs) [88], random forests [89], or artificial neural networks [90], to predict brain age from imaging data. These techniques can be trained on a large dataset of brain scans and associated ages, and then used to predict the age of a new brain scan.
In the following, some recent works in that field are highlighted and their approaches are explained. We specifically focus on research, which is relevant for this work and that will be used for comparison in chapter 5 later. First, we want to explain the works by [91]–[93], which use classical approaches, meaning that they work on 2D slices or extracted features of MRI scans. Afterwards, we explain the rational behind some research that operates on 3D representations of the data [23], [26]–[29].

In [91], they compared several machine learning techniques for brain age prediction with publicly available datasets (HCP, Cam-CAN and IXI). The authors followed a classical approach, where the features were generated by applying the Desikan-Killiany parcelation [94] on the T1-weighted MRI scans to derive morphological measurements (cortical thickness, surface area, subcortical volume, and total intracranial volume). These attributes were fed to different machine learning models, which will be quickly explained below.

Least Absolute Shrinkage and Selection Operator (Lasso) is a linear algorithm that adds a $L1$ regularization term to the linear regression equation. The penalty term is the absolute value of the sum of the regression coefficients multiplied by a tuning parameter λ. The larger the value of λ, the greater the penalty for having larger coefficients, which can cause the model to favor simpler models with fewer variables [95].

Support Vector Regression (SVR) The objective of SVR is to find a function $f(x)$ that predicts the value of a continuous variable y based on a set of predictor variables x. $f(x)$ is constructed using a subset of the data called support vectors, which are the data points closest to the hyperplane, that maximizes the margin between the predicted values and the actual values of the response variable [96].

Elastic Net is a regression technique, which adds a $L1$ and $L2$ penalty term to the linear regression to combine the strengths of both, leading to small coefficients and a sparse solution. Elastic Net is used to handle situations where there are many predictors and some of them are highly correlated. In these situations, Lasso may arbitrarily choose one of the correlated predictors and ignore the others, whereas Ridge regression may shrink the coefficients of all predictors, including the important ones. Elastic Net can overcome this limitation by allowing for the selection of a group of correlated predictors [97].

Decision Tree (DT) is a very simple algorithm that creates a tree-like structure of decisions that leads to a prediction. The tree is constructed by recursively partitioning the data into subsets based on the values of the predictors, until a stopping criterion is met. Once the tree is built, it can be used to make predictions by following the path down the tree based on the values of the predictors. The prediction is given by the class or value associated with the leaf node reached at the end of the path [98].

Random Forest (RF) is a machine learning technique, that uses the ensemble method combining several different DTs to improve accuracy and robustness, whilst reducing overfitting. Each tree in the forest is trained on a randomly selected subset of the data, and a random subset of the predictors is considered at each split in the tree. The final output is aggregated over all sampled trees to create the collective and stable result [99].

Gradient Boosting Machine (GBM) is an ensemble method that combines multiple weak learners (usually decision trees) to create a stronger, more accurate model. The basic idea behind GBM is to iteratively add decision trees to the model, with each subsequent tree fitting the residual errors of the previous tree. This process continues until the desired level of accuracy is achieved, or until a stopping criterion is met [100].

The authors of [92] used a similar approach. They used the gray volume maps of the MRI scans mapped with a non-linear Dartel algorithm to extract intensity values. After concatenating them to a feature matrix, the measures are fed to several machine learning algorithms, which will be shortly described below.

Gaussian Process Regression (GPR) is a non-parametric method that models the relationship between the predictors and the response variable as a probability distribution over functions, rather than a single point estimate. The basic idea behind GPR is to use a Gaussian process to model the distribution of possible functions that could relate the predictors to the response variable. This distribution is defined by a mean function and a covariance function, which specify the average value and the degree of similarity between different points in the predictor space, respectively. To make a prediction for a new observation, GPR calculates the posterior distribution of the response variable given the observed data and the Gaussian process prior [101].

Ridge Regression adds a $L2$ penalty to the linear regression equation to combat overfitting and handle multicollinearity between predictors. This is achieved by penalizing large values and encouraging small and smooth solutions, which also helps to work with noisy data and outliers [102].

CNN A Convolutional Neural Network (CNN) is a artificial neural network, that adds an inductive bias by aggregating the locality of the input, like images, and sharing the weights of the so called kernel. The kernel is a filter with shared and learned weights, that slides over the input to calculate the local interactions and extract the underlying data structure from it. It is found in a so called convolutional layer, which is described in more detail in section 3.3.1. The learned filters are able to detect simple meaningful patterns, such as edges, corners, or textures and by stacking them also increasingly complex and abstract features [31].

Instead of using a CNN architecture on precomputed features, [93] operated the model directly on slices of the brain scans. The only preprocessing was to rescale the images to the same size for them to work with the CNN architecture. They used three separate models, that were working on the different orientations of the slices (axial, coronal, sagital), which were followed by a linear regression model to compute the brain age. Lastly, the output of all three models was averaged to produce the final prediction.

The previous presented methods were all working on features or 2D data, whereas the following tried to implement different variations of convolutional filters that work on surfaces or 3D representations in general.
One simple option to use the additional available information is to use 3D convolutions to process the whole MR image. [90] used a readily available DenseNet architecture [56] that makes use of 3D convolutions to process the MRI scans. The only preprocessing needed was some resampling and reshaping to guarantee fixed-sized inputs for the CNN based architecture to work. A similar approach was used by [23], who also used 3D convolutions to process the brain scans and predict the age, however, they used this calculated information as a surrogate marker to classify conversion to MCI.
Expanding normal convolutional filters to accept 3D data only preserves the location-equivariance in Euclidean space, however, in medical imaging the data often lies outside the Euclidean space. To be able to preserve this property outside of Euclidian space, the applied filter needs to be rotation-equivariant instead of translation-equivariant, hence more sophisticated approaches are needed.
[26], [28] transfer the problem to the Fourier domain, where a convolution operation transforms to a simple multiplication, whereas [27], [29] slide a localised filter over the surface, similar to the 2D convolution. The latter one, also known as spatial methods, are easier to calculate and more expressive, whilst having the disadvantage of not being mathematically rotation-equivariant. On the other hand, the so called spectral methods guarantee rotation-equivariance, but to be able to compute their outcome the spectral filters are often approximated via polynomials, hindering their expressiveness [79].

The authors of *Spherical UNet* [29] use two tricks to get meaningful convolutional filters on the brain surface. First, they map the cortical surface onto a sphere, which ensures a consistent structure to slide a filter over, because the standard sphere usually is generated from a regular icosahedron [18]. Secondly, they phrase the convolutional filter as an aggregation operation over direct neighbors of a vertex on that sphere, exploiting the regular structure and formulating the problem similar to a standard 2D convolution.

Another spatial method generalizes the convolution to any non-Euclidean space instead of only a sphere [27]. Generally, the *MoNet* architecture also applies an aggregation over the neighbors of each vertex, but the neighbors are represented by a

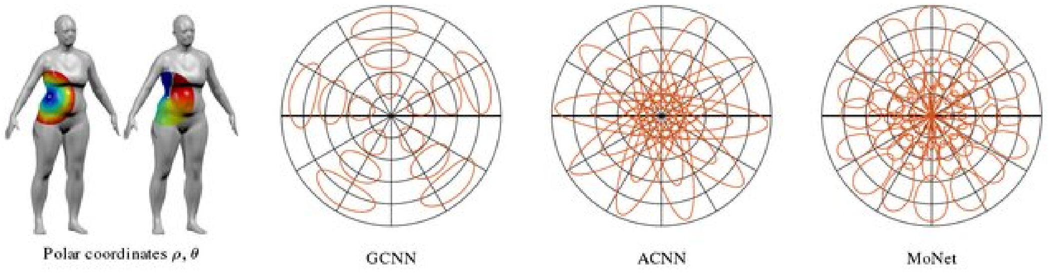


Figure 2.2: Comparison of the coverage of different convolution operations on non-Euclidean space. On the left the use of simple polar coordinates show the uneven distribution over the surface of the body of a woman. With the GCN a similar distribution to polar coordinates can be achieved, while the two approaches on the right show a more even distribution with higher focus on the center of the circle. Image taken from [27].

d-dimensional vector of pseudo-coordinates. The aggregation is then achieved by a parametrized kernel function inside of the pseudo space. This added degree of freedom enables the generalization of the approach to different settings. In the paper, the authors show, that CNN, GCN and others can be obtained from their framework by using specific pseudo-coordinates and kernel functions. A comparison of the coverage of the space or surface between different methods, showing the superiority of the approach, can be seen in figure 2.2.

Despite their easy interpretability and good expressiveness, the former explained methods do not guarantee rotation-equivariance, therefore researchers developed convolutions in the spectral domain. One such approach by [28] formulates the convolution in Fourier space, where the complex convolution is expressed as a simple element-wise Hadamard product. One challenge in the spectral domain, however, is the limitation of the convolution to a specified local neighborhood. To achieve a consistent local operation and reduce the computational complexity, the authors approximate the spectral filter via a K-th order polynomial of the Laplacian. Further optimization for the *ChebNet* architecture is needed, to apply the filter on a whole graph, which is achieved by using a Chebyshev polynomial to approximate the Fourier basis for the transformation into Fourier space [28].

Applying approximations enables the use of spectral filters in non-Euclidean space, but limiting expressiveness. The authors of the *Spherical CNN* [26] limit themselves to the sphere, but achieve spectral filtering without compromises on the expressiveness of the method. Their approach exploits several mathematical properties of Fourier transformation and the $SO(3)$ group [103], which mitigates the problem of rotation

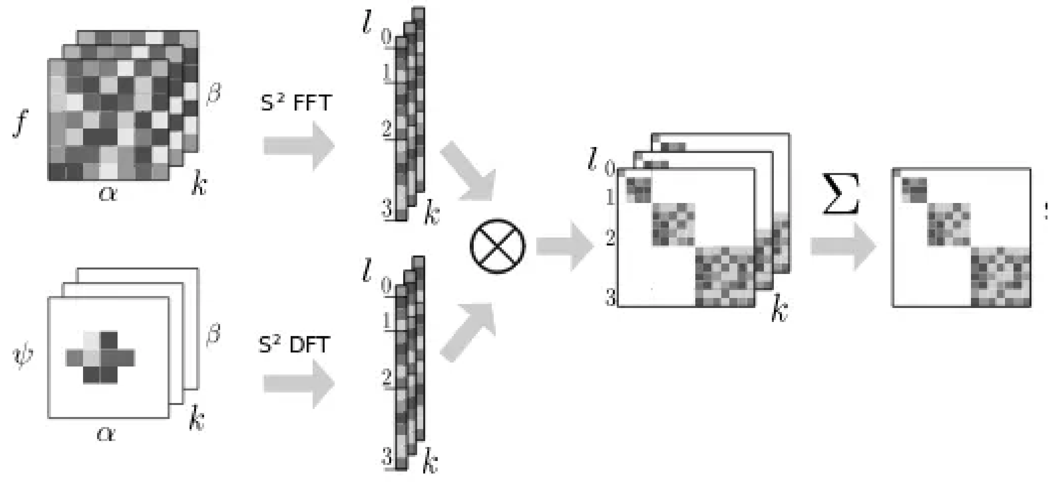


Figure 2.3: Spherical CNN. Transforming signal and filter into Fourier space to simplify the convolution operation to a matrix multiplication and projected back afterwards. Image taken from [26].

of a spherical filter in $SO(3)$ space as symmetrical properties for translations of pixel grids doesn't hold on the manifold of a sphere. In general, they first use a Fast Fourier Transform (FFT) to project the input and the filter into the Fourier Space, where the convolution operation simplifies to a matrix multiplication. After aggregation, the result is projected back to give a rotation-equivariant and expressive output. This schema is also shown in image 2.3.

2.3.4 Alzheimer's Disease Classification

AD is a progressive neurodegenerative disease that is characterized by the accumulation of several biomarkers, namely amyloid plaques and tau tangles in the brain, as well as a decline in cognitive function. These biomarkers can be used for an early diagnosis of AD, which is important because early treatment and interventions may slow the progression of the disease [104]. Therefore, many publications which try to classify patients into three classes of CN, MCI and AD exist. Their goal being to achieve a reliable detection of transitions between the states of the disease as well as the MCI class in general. Traditional methods build up on medical research, that searched for causes and biomarkers in patients, like increased amyloid plagues [104], but due to the dominance of DL techniques all newer publications use neural nets in their approaches. [22] use a combination of DL and ML, where the normalized MRI scans are first run

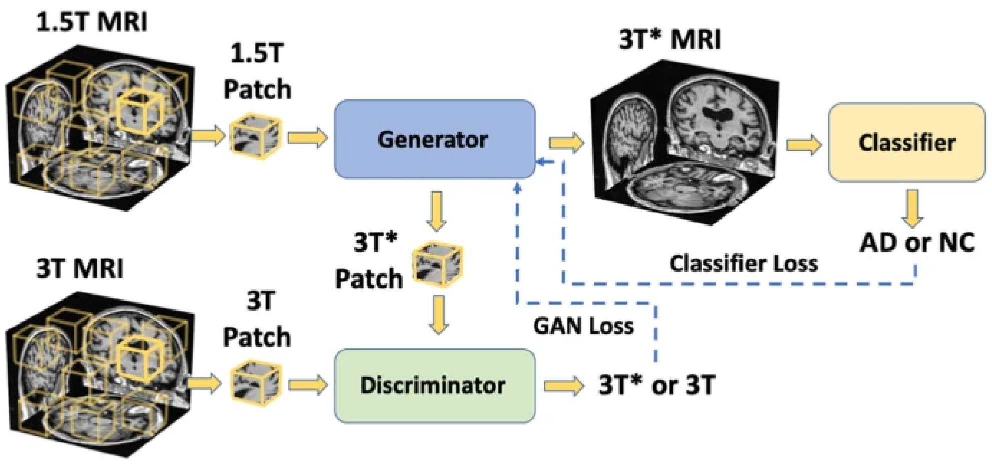


Figure 2.4: A GAN generates images that are used to train a fully convolutional network to predict AD status. Both models are trained simultaneously on their respective losses. Image taken from [105].

through a VGGNet to extract meaningful features and afterwards a combination of ML algorithms is trained as an ensemble method to increase accuracy and robustness. Others tried to use the brain-PAD as a guiding feature for the diagnosis of AD [23]. They first pretrained a CNN for the task of brain age prediction to generate meaningful features before using it together with the brain-PAD estimate for AD classification. Instead of calculating special surrogate features the authors of [24], [25] used different CNN architectures on image slices of the 3D MRI scans to separate the inputs into the wanted classes and achieved very good results. Lastly, we want to mention the work by [105], which uses a generative approach. The idea is to simultaneously train a Generative adversarial network (GAN) and a CNN model to first generate scans with higher resolution to base the classification on afterwards, as shown in figure 2.4. Also, the previously presented methods for brain age prediction can be used for AD classification, but the greater complexity of the 3D models might make them hard to train.

In this related work chapter, we have reviewed the existing literature in the field of computer vision and discussed the related works that are most relevant to our evaluated method. We have compared the traditional methods and the deep learning-based methods and discussed their strengths and weaknesses. In the following sections, we present the theoretical background to understand the presented methods, our used method and its experimental results.

3 Theoretical Background

This chapter will provide an overview of the primary theoretical foundations upon which our method and experiments are based on. Specifically, we will provide a brief description of the brain's structure, discuss the fundamental principles of meshes and surfaces, and review pertinent principles of deep learning. Furthermore, we will briefly explain important biomarkers in the realm of medical modelling.

3.1 Brain structure

The brain is a complex organ that is responsible for a wide range of functions, including sensation, perception, movement, cognition, and emotion. It is made up of a variety of different cell types, including neurons and glial cells, and it is organized into different structures that perform specific functions [106].

One way to understand the structure of the brain is to consider it at different levels of organization, from the microscopic to the macroscopic. At the microscopic level, the brain is made up of cells, including neurons and glial cells. Neurons are the primary cells of the nervous system, and they are responsible for transmitting information throughout the brain. Glial cells are non-neuronal cells that provide support and protection for neurons [107]. At the macroscopic level, the brain is divided into different structures, each of which performs specific functions.

The brain is comprised of Gray Matter (GM), a thin layer of neural tissue consisting of neurons, which is highly folded, as well as White Matter (WM) and Cerebrospinal Fluid (CSF). The folds of the GM are known as gyri, while the grooves between them are called sulci. In between is the WM connecting several cortical regions and the CSF, which fills the space and helps cushioning the brain. The WM is composed primarly of axons, that are covered on myelin to speed up the transmission of nerve impulses. The boundaries between the different tissue is of significant interest as a wide variety of cortical biomarkers can be computed from them, which will be explained in more detail in chapter 3.4. A cropped MRI scan with the tissues labeled can be seen in figure 3.1, showing the white matter surface (which separates the white and gray matter) and the pial surface (which separates the GM and CSF) [108].

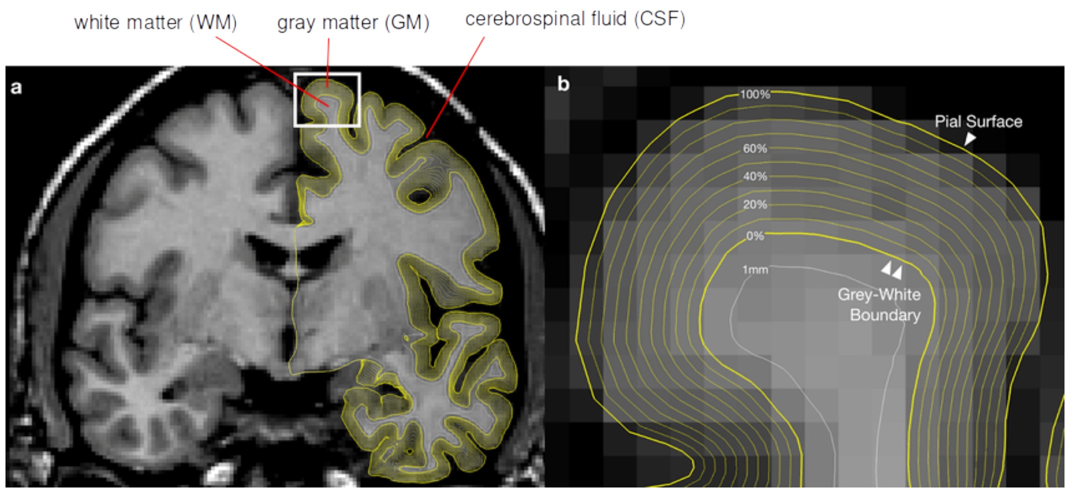


Figure 3.1: The brain contains neural tissue, which can be classified into white matter and gray matter. It is enveloped by cerebrospinal fluid. The surfaces dividing them are called pial surface and white matter surface (or Grey-White Boundary). Image taken from [109]

3.2 Graphs, Meshes and Surfaces

A standard way to display and model 3D objects and surfaces are graphs or meshes. They both resemble a collection of points connected by edges, however, differ in their structure and purpose. The relevant notations, concepts and differences will be outlined in the following.

3.2.1 Geometric Definitions

First, we want to give a basic understanding of the terms *surface, mesh* and *graph* according to the definitions in [110].

Surface A surface is viewed as a two-dimensional manifold which is continious, orientable and embedded in $\mathbb{R}^3$. This means that a space $S \subset \mathbb{R}^3$ is locally represented in $\mathbb{R}^2$ as a surface Ω, where each point $p \in S$ has a matching point on the surface Ω. Note, that when approximating a body B in the space S the matching is not a 1-to-1 matching. Every point p on the surface Ω corresponds to a $3D$ point in the space S, but not necessarily to a point q on the body [110].
As an example the brain, more precisely the cortical surface, can be used. It is topologically equivalent to a sphere, meaning each point on the surface of the sphere corresponds to a point in $3D$ space, but not every point on the surface

of the sphere corresponds to a point on the surface of the cortex. In fact, the cortical surface is much more complex and irregular than the smooth surface of the sphere [111].
There are many ways to represent surfaces in mathematical terms, for now we want to focus on the two main representations to model surfaces, which are implicit and explicit (or parametric) surface representations. The latter one describes the surface as a function mapping points from 2D to 3D:

$$\begin{aligned} f &: \mathbb{R}^2 \rightarrow \mathbb{R}^3 \\ S_\Omega &= f(\Omega) \end{aligned}$$

Here, the function f maps the points $p \in \Omega$ to their 3D counterpart $q \in S$. This mapping gives the benefit of easy sampling, because generating points on the surface S means sampling points in the easier 2D domain of Ω and mapping them to the surface via the function f. The downside is that generating such a function can be very complex and changes in the surface are reflected both in needed adaptions in f and Ω as well [110].
The second main way to represent a surface is by its implicit form, which means that each point is characterized by either lying inside, outside or exactly on the surface S that bounds the object. In example, the surface can be defined by a radial basis function or a discrete voxelization. Generally, this means that there is a mapping $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ that classifies every 3D point to the zero-level isosurface S of the scalar valued function F. One benefit of the implicit representation is an easy inside / outside evaluation, which boils down to simple computing the function F. Negative values of F are regarded as inside, whereas positive values lie on the outside of the object. This further opens up the possibility to combine multiple implicit functions to construct more complex objects by simply taking the *min* and *max* of the individual functions. Furthermore, in contrast to explicit representation adapting the surface is quite trivial as it corresponds to just changing the function value of F locally. But, there are also drawbacks to this group of representation as sampling points on the surface is difficult because the roots of the function must be computed for this. Consequently just rendering the surface is a challenge [110].
In general, the purpose of using a surface to approximate a physical object is to simplify the geometry of the object while still capturing its essential features. The surface may not perfectly match the shape of the object, but it can be a useful approximation for many purposes.

Genus 0 Surface Genus is a term used in topology to describe the topological properties of a surface. It is a measure of the number of holes or handles on a surface, and it is defined as the number of times the surface must be punctured or cut before it can be smoothly deformed into a sphere.

A genus 0 surface is a topological space that has the topological properties of a closed, orientable surface with no holes. It is a surface with no handles or other topological features, and it is topologically equivalent to a sphere [112]. In example, the cortical surface does not contain any holes thus making it a genus 0 surface and allowing us to treat it as a sphere to approximate the properties and simplyfing calculation [111].

Barycentric Interpolation Barycentric interpolation is a method for interpolating (estimating) the value of a function at a given point based on the values of the function at a set of known points. It is a type of Lagrange interpolation, which means that it uses a polynomial to approximate the function [113]. It is, in example, used to rescale a triangulated sphere to different number of vertices to increase the number of nodes and allow for higher precision or reduce the amount of vertices to ease computation.

To understand how barycentric interpolation works, consider a set of n points (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) and a target point x. The goal of barycentric interpolation is to estimate the value of the function at the target point x based on the known values of the function at the points (x_1, y_1), (x_2, y_2), ..., (x_n, y_n).

To do this, barycentric interpolation constructs a polynomial of degree n-1 that passes through all the known points. This polynomial is given by the following formula:

$$p(x) = \sum_{i=1}^{N} y_i \cdot w_i(x) \tag{3.1}$$

where $w_i(x)$ is a weighting function that is defined as:

$$w_i(x) = \frac{(x - x_1) \cdot (x - x_2) \cdot ... \cdot (x - x_{i-1}) \cdot (x - x_{i+1}) \cdot ... \cdot (x - x_n)}{(x_i - x_1) \cdot (x_i - x_2) \cdot ... \cdot (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdot ... \cdot (x_i - x_n)} \tag{3.2}$$

The value of $p(x)$ is the estimated value of the function at the target point x [114].

There are favorable characteristics associated with barycentric interpolation, including the fact that it is well-conditioned, meaning that it is not sensitive to small perturbations in the input data, and that it can be computed efficiently using a stable algorithm. However, it is worth noting that barycentric interpolation is generally less accurate than other interpolation methods, such as spline interpolation or polynomial interpolation, when the input data is noisy or has a large degree of curvature [115].

Mesh A polygon mesh, or short mesh, is defined as a collection of points or vertices (also called nodes) V, faces F and the connecting edges E. This tuple $M = (V, F, E)$ stores the information about the geometry (V) and the connectivity ($E \iff (V, F)$). In some cases further information (properties) can be stored such as texture. Mathematically, the nodes represent Cartesian coordinates $V \subset \mathbb{R}^3$ with n being the number of vertices $n = |V|$. Each face corresponds to a connected series of those vertices: $F \subseteq \{1, ..., n\}^3$ where the connectivity information is stored in the set of edges $E \subseteq \{1, ..., n\}^2$. In general, a polygon resembles a planar shape, that is defined through a set of connected vertices. A special case is using the minimum of only three points to form a polygon, also called triangular mesh, as it guarantees that the nodes lie in one plane [116]. Barycentric interpolation, as described above, can be used to easily up- or downsample a triangular mesh. In fact a triangular mesh is defined by a barycentric parametrization, more precisely consisting of piecewise linear functions. This leads to the property, that the approximation error of a triangulation for a sufficiently smooth surface is quadratic to the maximum edge lenght h. In more practical terms, this means halving the edge length the error reduces to about a quarter. On the other side, however, the number of triangles grows by the factor four as depicted in figure 3.2. Generally speaking, the quality of the approximation depends on the curvature of the underlying surface, but with clever sampling over the surface a good approximation is possible with a moderate mesh complexity [110]. Another common way to reduce the data complexity is to model surfaces as a graph instead of a mesh. This means representing the object only by the vertices and their connections (edges). This is usually done in the field of deep learning to allow many applications to be able to handle the amount of data in the first place, like in GNNs for example. Note, that using this simplified representation has the drawback, that the information about the orientation is lost, which is stored withing the faces.

Graph A graph $G(V, E)$ is, similar to a mesh, a collection of vertices V and edges $E \subseteq V \times V$. It is, however, missing the accompanying faces, which store the geometric information like the orientation. On the other hand it makes it more versatile, because it can represent a broad kind of structure consisting of entities and relationships, like a friendship graph [116]. There exist several extensions to

the graph representation, like weighted and attributed graphs. In those cases, the definition is extended to $G = (V, E, A)$, where A is either a matrix of weights w that are assigned to each edge in the graph. This is used to show the tie strength between two relationships to determine the concept of friendship or in more related deep learning cases, it is used to have attention-based weights on the edge to better cluster the embedding space (see section 3.3.2). The more general case is to have any kind of attribute associated with the graph. In our work each vertex has several medical features stored in addition, which are explained in more detail in chapter 3.4.
Some important properties of graphs are the neighbors of a vertex v_i, which are denoted as:

$$N(i) = \{v_j \mid (i,j) \in E\}, \tag{3.3}$$

where v_j is a neighbor of v_i, if the edge $(i,j) \in E$. This relationship is often stored in an adjacency matrix, which is defined as follows:

$$A_{ij} = \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise, \end{cases} \tag{3.4}$$

with $A \in \mathbb{R}^{n \times n}$. The neighbors can be calculated by simple taking the row or column of the adjacency matrix. Furthermore, the degree of a node gives an implication of how connected a node is, meaning how many connection or neighbors does it have, which is also encoded in A_{ij}:

$$deg(i) = |N(i)| = \sum_i A_{ij}, \tag{3.5}$$

which is often used in a matrix representation like:

$$D_{ij} = \begin{cases} deg(i) & if\ i = j \\ 0 & otherwise \end{cases} \tag{3.6}$$

The adjacency matrix is an important part in graph representation as a lot of information is encoded within. Further properties are described in [117], [118].

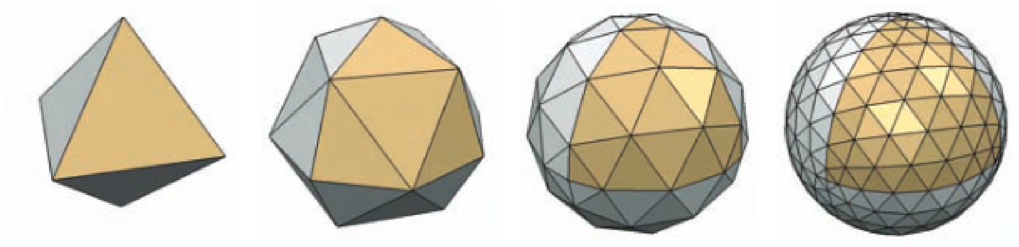


Figure 3.2: Halving the edge length of the triangle tesselation, grows the amount of faces by a factor of four, but reduces the error of the approximation to around a quarter compared to previously. Image taken from [110].

3.3 Deep Learning

With the increase of compute power many different deep learning based approaches were applied and led to state-of-the-art performance in various fields like natural language processing [38], [119], computer vision [32], [39] and most importantly medical imaging [120], [121]. For further details about the specific applications and more depth regarding the history and development of the different topics, we refer to the textbooks of Bishop [122], Murphy [102], Goodfellow *et al.* [51], Chollet [123], Jurafsky *et al.* [124] and Goldberg [125]. This work will provide an overview of the fundamental concepts that are relevant to our study, which primarily emphasizes transformers and geometric deep learning.

3.3.1 Convolutional Neural Network

A neural net is a universal function approximator, that can approximate any function to any degree if enough resources and data are available. Expressed in mathematical terms this just means:

$$f_\theta = X \rightarrow Y, \tag{3.7}$$

where f is the parametrized function by θ, X is the input and Y the output domain. The initial version included the combination of linear multiplications with non-linearities, the so called *feed forward neural network*. However, as described above, the approximation works under the constraint of the resources, which quickly led to a problem when working with high-dimensional data, such as images. Therefore, more efficient calculations are needed, which resulted in the convolution operation. Instead of linearly combining all input pixels, the convolution operation restricts the linear combination to

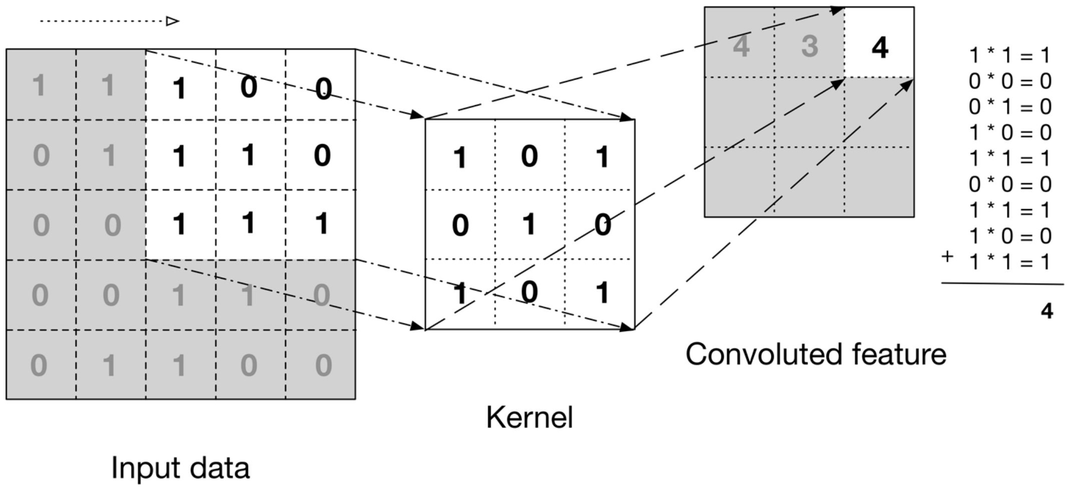


Figure 3.3: Convolution operation for a simple 2D example. The kernel slides over the input data and calculates the weighted sum between the receptive field and the filter. The result is stored in the convoluted feature.
Image taken from [126].

a local neighborhood using a kernel. To further reduce computational complexity the learned filter stays the same over the whole input. The number of convolution kernels present in a convolutional layer of a CNN is commonly referred to as channels. It is worth noting that a single layer can contain multiple kernels, e.g. one per color channel of the image and that the kernel weights are included in the overall model parameters θ, which are optimized during the training process.

The above mentioned tricks enabled applications in the image domain, whilst not sacrificing the performance. In figure 3.3, the usage of a convolution operation is shown. The kernel slides over the input and calculates the weighted sum locally, in mathematical terms it can be shown that this indeed resembles a convolution. For completeness, the equations from [51] are as follows:

$$s(t) = (x * w)(t) = \sum_{a=-\inf}^{\inf} x(a) \cdot w(t-a), \tag{3.8}$$

which in discretized form boils down to:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \tag{3.9}$$

The equation can be generalized to higher dimensions by adding further summations over the added dimensions and increasing the kernel size accordingly. One important thing to note is, that a convolution operation changes the data size, as can be seen in image 3.3 as well. This helps to extract the essential features from e.g. images for classification tasks, on the other hand creating a challenge for image segmentation, where a pixel-to-pixel mapping is needed.
To get to a full blown CNN more operations are required, namely *non-linearities* and *pooling*. For segmentation tasks the reverse operations of *unpooling* and *transpose convolution* are required. Additionally, normalization [127], skip connections [32] and regularization [128] can be applied. To keep it brief, we will only mention the functionality of non-linearities and pooling. A non-linearity, like a rectified linear unit (ReLU) [129], is usually added after each layer, such as a feed forward or convolutional layer, to enable the universal approximation theorem. As the neural network layers usually only resemble linear functions, a sequence of them can be rewritten as a single linear multiplication again, rendering the layering useless:

$$\begin{aligned} y &= W_3 \cdot (W_2 \cdot (W_1 \cdot X)) && (3.10)\\ &= (W_3 \cdot W_2 \cdot W_1) \cdot X && (3.11)\\ &= W \cdot X, \text{ with } W = W_3 \cdot W_2 \cdot W_1 && (3.12) \end{aligned}$$

With the addition of non-linearities, this property doesn't hold anymore and the layering allows to add more parameters, increasing the capacity of the network and with that also the approximation abilities.
Pooling on the other hand is a simple operation, that is used to further reduce the required computational power needed to process the data. It is similar to the convolution operation in that, it takes the local neighborhood into account and slides over the input. This time, however, it doesn't compute a weighted sum between some learned filter and the receptive field, it is a simple fixed function, such as the mean or the max operation. The last component to building a CNN model is the output layer, which is either a fully connected layer or an equivalent 1x1 convolution. This is needed to map the embedding space to the output space, such as image categories and perform the last step of the mapping. In figure 3.4 a common CNN model can be seen, together with the learned features to get a better understanding how to use the convolution operation and what it learns.
Lastly, we want to mention the important properties of CNNs, namely their sequential learning and the translation-eqivariance. The first property is visible in the learned feature of figure 3.4, which shows, that the early layers learn simple visual features such as edges, and combine those to more complex representations in later layers, like circles. The second property comes from the fact that the kernel parameters are shared

over the whole input. Mathematically, this means that if the input changes, the output changes in the same way or more practically expressed the model doesn't care where the relevant object in the image is [51].

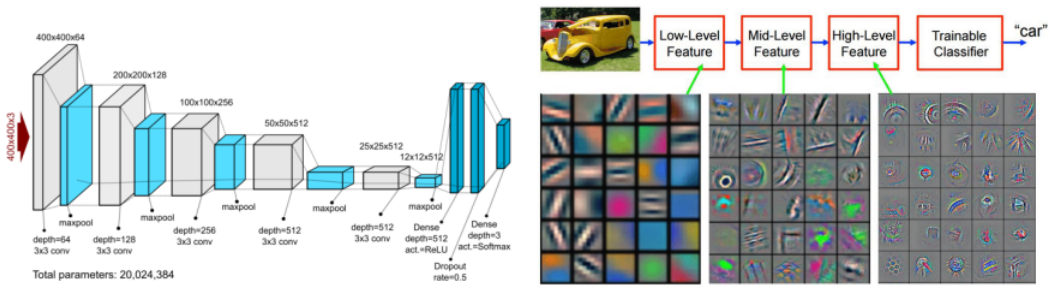


Figure 3.4: (Left) CNN model architecture, displaying the several used layers, consisting of convolutions, max pool operations and fully connected layers for classification at the end.
(Right) Visualization of the features of a model trained on ImageNet, categorized in low, mid and high level features. First showing the learned edges and further the more complex filters like circles.
Images taken from [130], [131].

3.3.2 Attention

The challenge of computational complexity, that was faced in image processing, was also present in the NLP domain. However, in language topics the local neighborhood is not sufficient to express the relationship between the words. Oftentimes a word in the beginning and the end of a sentence belong together to understand the meaning, making it a daunting task for CNNs. This led to the invention of the attention mechanism, which has the idea to learn the relation between the words and specifically "attend" to the needed words, remaining computationally efficient whilst giving the best performance. To better understand this, figure 3.5 illustrates the concept.

In the image the relationship between the translations can be seen, showing that it is not always a one-to-one mapping and especially that the order is different. With attention the prediction of the next most probable word is now simplified in that the model gets all words as input, rated by their attention scores for the current prediction, making it easier for the model to predict the next word. More precisely, the model gets the information that the word "zone" is more important than any order word, when it is trying to predict "Area".

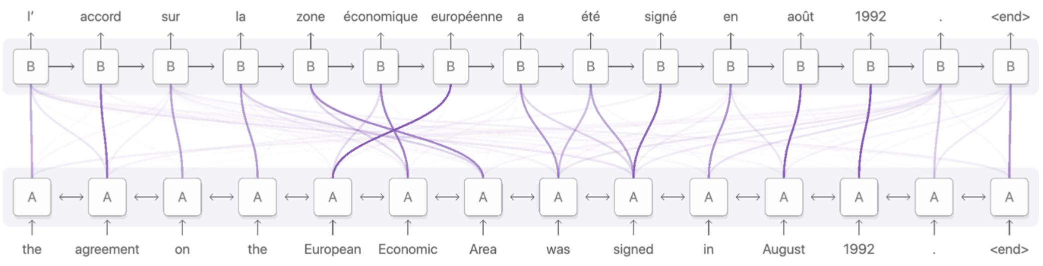


Figure 3.5: Attention mapping shown for a language translation task from French to German. Image taken from [132].

3.3.3 Transformer

Many different models use the attention mechanism, especially in the NLP domain, like the Seq2Seq adaption from [133]. However, one specific model, namely the Transformer by [38], led to a major breakthrough in the field, similar to CNN in the image domain. The main difference to the previous variations was to get rid of the sequential nature of the proposed attention mechanism and formulate it in a way that it can be computed in parallel. In other words, previously you needed to translate the sentence word by word, whereas now the sentence can be computed in one go, as there are no inter dependencies between the words anymore. This speedup, compared to previous methods, led to the ability to train models faster and to scale them to almost arbitrary sizes. The initial model from [38] had 213 million parameters, whereas the latest model from OpenAI has 175 billion parameters, nearly 1000 times as much, whilst still relying on the same ideas.

The underlying attention mechanism introduced by [38] is following:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \cdot V \tag{3.13}$$

He proposes three matrices Q, K and V, that are called the queries, keys and values. By the multiplication of the query and key matrix an alignment is checked, because we know from linear algebra that:

$$q \cdot k = |q| \cdot |k| \cdot \cos(\alpha) \tag{3.14}$$
$$q \cdot k = 0, \iff q \perp k \tag{3.15}$$

In words, if two vectors are perpendicular to each other their dot product is zero. For the attention score this means, that we basically take a query to find the corresponding key, namely the specific row of K that is aligned the most with the query. The scaling by d_k is only to prevent extreme points of the softmax function that is applied next, which in turn only transforms the output to a proper distribution, that sums to one. The last part of the equation is the multiplication with the value matrix V, which means to select the corresponding value for the key that aligned the most with the query. Generally speaking, the proposed attention mechanism is learning a look up table that can be queried and returns the fitting value. An accompanying illustration to the above mentioned explanation can be found in figure 3.6.

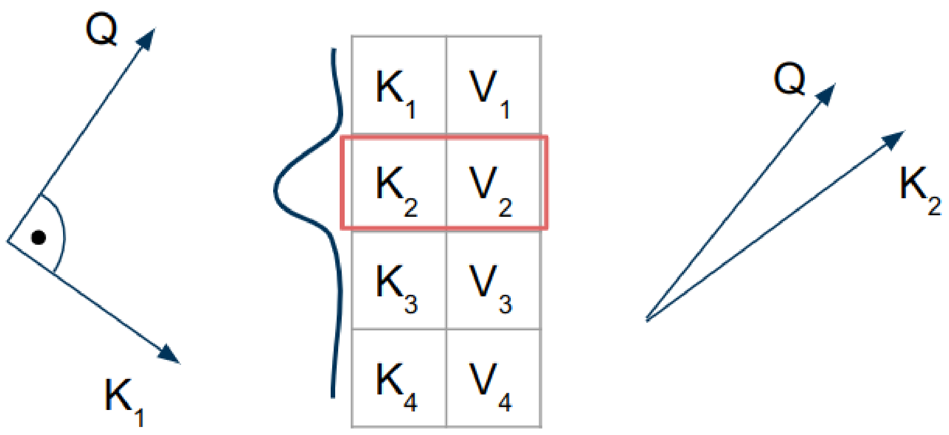


Figure 3.6: Two exemplary configurations of vectors for the query and key with the according key value table shown. On the left of the table the softmax distribution over the query-key dot product is shown and the key-value pair with the highest attention is marked. Image derived from [134].

In addition to the attention mechanism, some other key elements play a role in the success of the transformer architecture, which are the high quality embeddings, the positional encodings of the words as well as how the ouput of the attention layer is used. Figure 3.7 displays the complete model architecture.

As the model was proposed for a NLP task, the words had to be first mapped to a computer understandable format, which is done by the input embedding. Further details on how the embedding works, can be found in the paper [38], as it is not relevant for this work. The positional encoding was added to help the model to distinguish between homonyms, like the bank to sit on and bank as the financial institution.
The bigger block on the left displays the encoder, which encodes the information of the sentence into the above explained value matrix V, whereas the decoder block on the right side tries to pick the correct value out of this matrix by learning the fitting

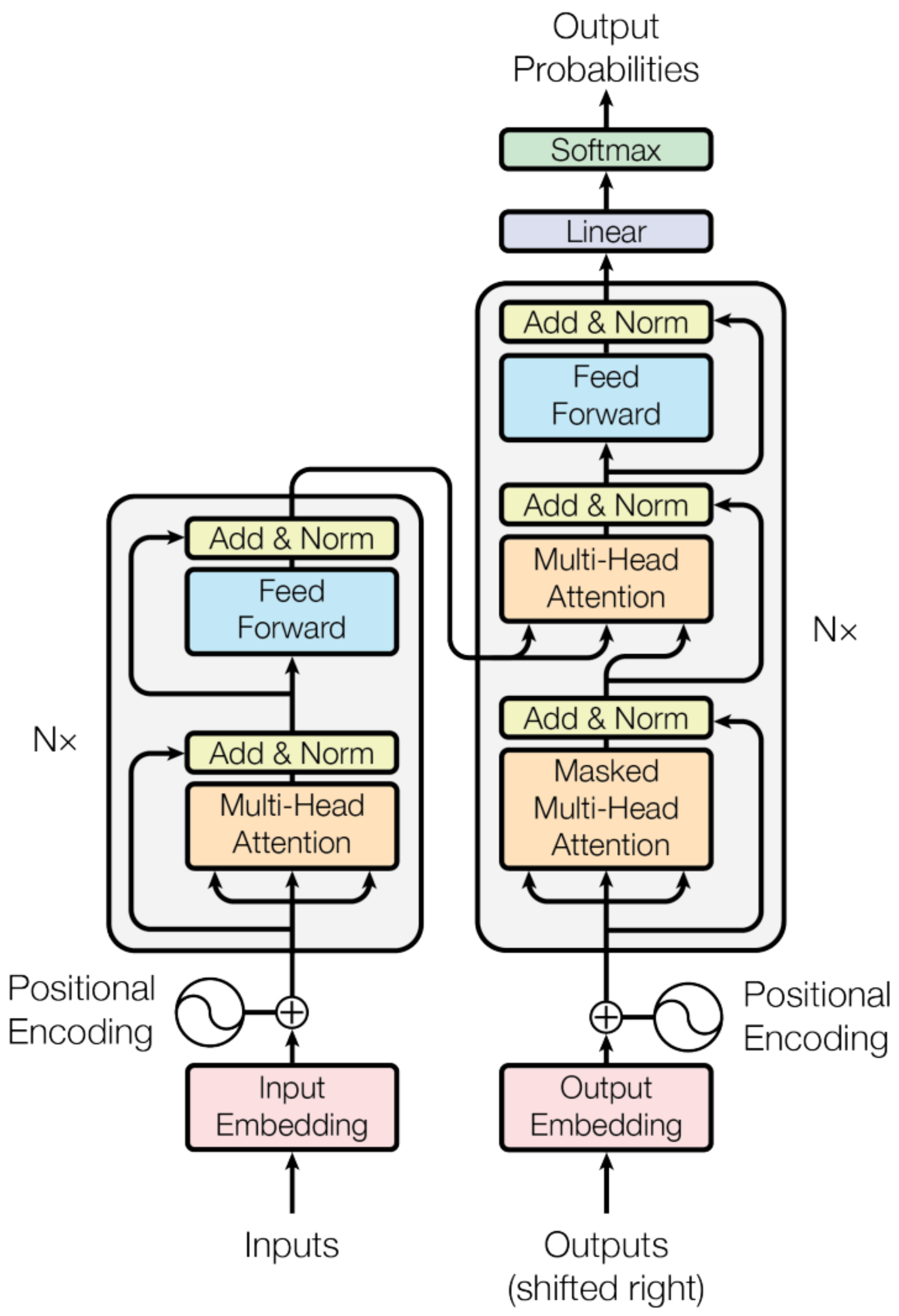


Figure 3.7: Transformer architecture consisting of encoder and decoder block. Image taken from [38].

key and query matrices. To improve the performance, the architecture can be scaled by increasing the number of both attention blocks simultaneously as depicted by the "Nx" on the side of the schematic. We will only explain the encoder block as it is mainly relevant for our work and the decoder follows a similar functionality.
The multi-head attention uses the explained attention mechanism, the twist here is, that it learns h different representations to attend to different subspaces and positions in the data, which further boosts the performance with no additional cost. The h attention scores are simply concatenated and projected to the correct dimension before being fed through the linear layer.
For the task of the paper, namely neural machine translation, the encoded information needs to be decoded to the target language, but in many following applications of the architecture only the encoder is used, for example to output image classes.

3.3.4 Vision Transformer

Whilst possessing a huge popularity in the language domain, the attention mechanism posed several challenges in the image domain. The simple application on a pixel level results in the same problem, that fully connected nets faced as the complexity of the self-attention layer scales quadratic with the sequence length, here the number of pixels. This hinders already the application on small images of size 256 by 256. As a solution, the authors suggest a patching mechanism to shrink the dimensionality of the data, as already presented in the model architecture in figure 2.1.

The encoder is expecting a word embedding sequence of dimension $\mathbb{R}^{N \times E}$, where N is the sequence length and E is the embedding of the words. Therefore, the image, which is of dimension $\mathbb{R}^{H \times W \times C}$, needs to be reshaped. Here (H, W) stands for the image pixels, where C denotes the color channels. The solution is to rearrange the images in a sequence of flattened 2D patches, where $E = P^2 \cdot C$ is the image embedding, resulting in a sequence of $N = HW/P^2$, with (P, P) being the patch dimension. The linear projection displayed in the architecture overview is needed to map to the latent dimension, that the encoder expects, which might be different from E. This addition allows to scale the architecture to different, also medium sized images by increasing the patch dimensions. For reference, the authors used 16x16 pixels to divide the images. Similar to the BERT [119] embeddings used in the original transformer architecture, the authors added a [CLS] token to the beginning of the embedding. Instead of it being computed via the whole input sequence, they just learn a linear layer to map the wanted image classes to the correct dimensions to append them to the image embeddings. Furthermore, they added simple 1D positional encodings to tell the model from which part of the image the patch came from and were then able to use the standard transformer to get encodings, which they turned into class labels by

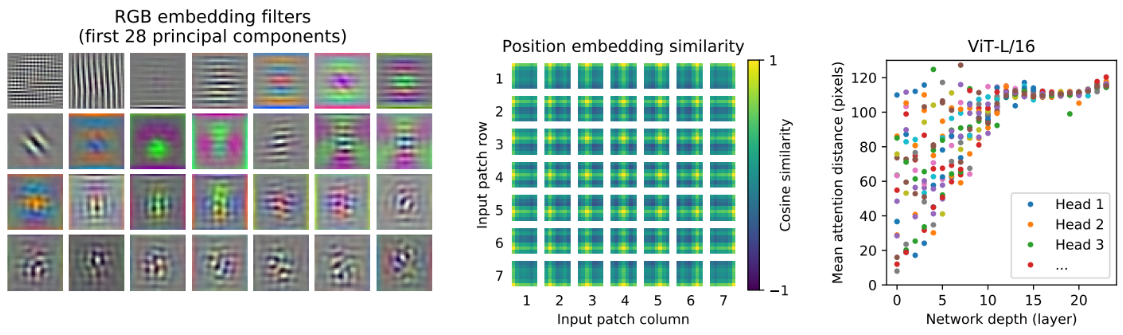


Figure 3.8: Visualization of the embedding filters (left). Similarity of the position embeddings (center). Attention distribution over network depth (right). Image from [39].

appending a multi-layer perceptron head [39].

Interesting to note are also the comparisons the authors make towards CNNs. Where convolutions add an inductive bias because of the locality and square filter shape, as well as the added translation invariance, the vision transformer overcomes this by scaling the information of the patches by the global attention. This makes it more flexible as the only image-specific operations are the patching and positional embeddings, which are both learned and can be easily exchanged. To further analyze the similarities and differences have a look at figure 3.8.

Similar to what the low level and mid level layers of a CNN learn can be seen on the left side. What is displayed are the principal components of the learned embeddings, which are fed into the transformer encoder. This shows, that also hybrid architectures are possible, where the embeddings are gained from a pretrained CNN. On the right, the mean attention distance over the network depth is visible. In other words, the graph shows that already in the first layer, pixels can attend to any other pixel in the image, whereas in a CNN the attention is only to the local neighboorhood the size of the kernel and grows with the increasing receptive field with the network depth. This shows, that by using attention, the network depth can be reduced whilst still maintaining the relationship of pixels across the whole image. For completeness, the center image shows the position embedding. It is easy to see, that the model itself came up with some kind of 2D information even though only a 1D positional encoding was used [39].

3.3.5 Geometric Deep Learning

While breakthroughs in the text domain can be transfered to the image domain with reasonable effort, applying them to non-Euclidian data like graphs poses a bigger challenge. The so called Graph Neural Network (GNN) generalizes the application of linear [37], convolutional [59], [60] and even attention layers [135] to mesh structures. We will focus on their underlying mechanism, the so called message passing, as GCN and Graph Attention Network (GAT) are mainly extension of it. For more details on gDL we refer to [136]–[138].

Every node in a GNN is represented by some embedding, which can just be the attributes of the node, but also some computed value. For example, a graph consisting of images to do deep metric learning is usually preprocessed with a CNN to obtain the embeddings for later refinement with a GNN [139]. This tweaking procedure consists of two steps. First, the information of the neighbors of a node are collected, the so called message $m_v^{(k)}$, which are afterwards used to calculate a new node representation $h_v^{(k)}$. Exemplary for a single node update, the procedure is shown in figure 3.9, all other nodes are updated accordingly.

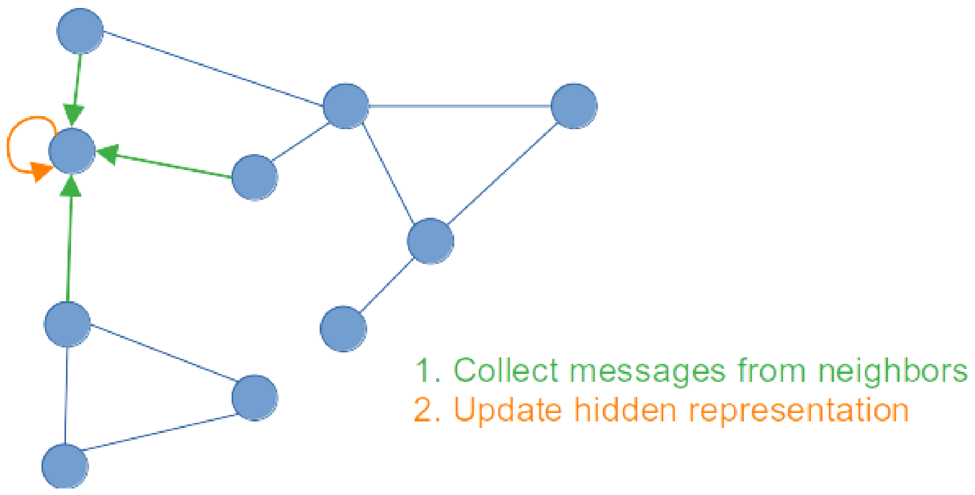

1. Collect messages from neighbors
2. Update hidden representation

Figure 3.9: Visualization of the procedure of a message passing step for a single node. First, all the information from the neighboring nodes are gathered. This information is then aggregated to update the own hidden representation of the node.

The message gathering step can be described as taking the current node representation as well as the influence of the neighborhood into account:

$$m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)},\, h_u^{(k-1)},\, E_{vu}) \tag{3.16}$$

$$= \sum_{u \in N(v)} \frac{1}{d_v} (W^{(k)}\, h_u^{(k-1)} + b^{(k)}), \tag{3.17}$$

where M is just a standard fully connected layer, that calculates the average over the neighbor embeddings. This aggregated information is then used to update the embedding of node h_v:

$$h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)}) \tag{3.18}$$

$$= relu(Q^{(k)} h_v^{(k-1)} + p^{(k)} + m_v^{(k)}), \tag{3.19}$$

where U is as well just a regular fully connected layer, with a residual connection of the previously computed message $m_v^{(k)}$ before adding a non-linearity.
Note, the superscript k stands for the k-th layer, as this procedure is applied repeatedly, where k-iterations means the node v getting information from neighbors of distance k. The matrices and vectors $W^{(k)}, b^{(k)}, Q^{(k)}$ and $p^{(k)}$ are the trainable parameters of the k-th layer. After the last iteration the GNN produces refined embeddings of each node, which further can be used for regression or classification for the given task at hand, similar to how CNN produces embeddings.

To extend the presented algorithm, it is necessary to exchange the update step of the message passing framework. Equation 3.16 takes every neighbor into account with the same weight, however, by adding a learned weight it is possible to prioritize the messages of certain neighboring nodes, which resembles the attention mechanism, leading to the GAT [135]. The authors of [59] show, that with another adaption of the formula also convolutions on graphs can be represented.

3.3.6 Supervised Learning

In supervised learning, a machine or deep learning model is trained by getting feedback or supervision. The training consists of optimizing the parameters θ of a model f_θ, that maps a certain input X to the respective output Y: $f_\theta : X \rightarrow Y$. This mapping is evaluated by a loss function L, where the objective is to find the best parameters to minimize its loss:

$$\theta^* = argmin_\theta \, L(f(X), Y), \tag{3.20}$$

where L represents the guidance or supervision for the model, as it evaluates the fit of the function f. More precisely, each input $x \in X$ needs to have a ground truth label $y^{gt} \in Y$, that can be compared to the model prediction $y^p = f(x)$ via the loss function L. During training, a set of training data $X_{train} \in X$ is presented to the model together with their labels y_{train}. In each step, it makes a prediction y_i^p, compares it to the current label y_i^{gt} of the sample x_i and calculates the error. This procedure is repeated for several epochs e, where one epoch means to have seen the whole training data once. To find the best parameters, each sample propagates its error from the loss back to the input, via the so called backpropagation algorithm [140]. This is an optimization technique, that is described in section 3.3.7. A schematic overview of supervised learning is presented in figure 3.10.

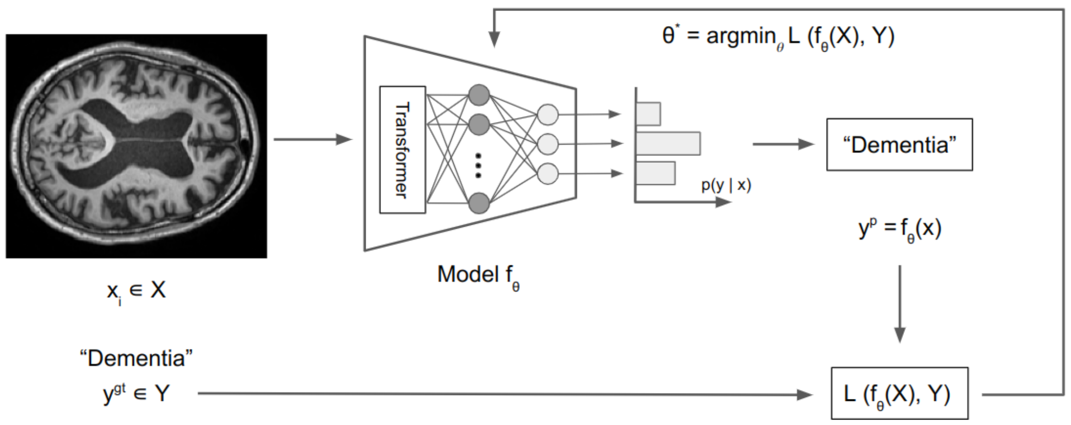


Figure 3.10: Overview of the training procedure of a neural net in a supervised fashion using backpropagation.

3.3.7 Optimization

Training a neural network is the mathematical equivalent of optimizing the model's parameters in order to minimize the used loss function. More precisely, it aims to change the parameters, such that the error between the predictions and the labels of the training data decreases. This automated and data driven method is the main reason of why neural networks can be adapted to so many tasks. It also helps them to generalize to unseen data, however, it is important that they were trained with data from the same data distribution than the new data.

The general process to optimize a given loss function is to use a Taylor approximation to fit a function f to the given data points. As the data can get relatively complex and the number of parameters is very high in neural networks, one relies on an iterative approach to fit the data. For that, so called first or second order methods are commonly used. First order methods approximate the gradients of the function and update the initial parameters of the neural network to reflect that, whereas second order methods also take the curvature of the loss function into account. There are several benefits for both methods, but for neural networks, first order methods are the most popular ones, as they are computationally more efficient and simple to implement, however usually take longer to converge to a minimum of the loss function as they are less accurate.

The most used ones are stochastic gradient descent (SGD) [141], [142], RMSProp [143], [144], Momentum [145]–[147] and Adam [148]. Overall, every algorithm takes the gradient of the loss function with respect to the previous layer's parameters to calculate an update vector, s.t. the model parameters get adapted in a way that the loss moves in the direction of decent, hence also the name gradient descent methods. This layer update is then propagated until the first layer of the network to form the well known backpropagation algorithm. An overview, how such an update process looks like in a 3D loss landscape can be seen in figure 3.11.

One simplification that is often used in neural network training, is to update the parameters more frequently, but only use a fraction of the training data for that, a so called *mini-batch*. This further reduces the computational complexity and speeds up the convergence process in most cases. Additionally, it helps to overcome plateaus in the loss function due to the noisy updates applied on the parameters. Another way, to cope with this challenge is the concept of momentum [145], which takes previous update steps into account to calculate a "velocity", that scales with the steepness of the gradient updates and helps to move away from plateaus by using the momentum to get out of it, like a marble ball on a course.

Despite plateaus in the loss function, there are many more challenges in optimizing the weights of a neural network. The process of optimization plays an important role and many factors can affect the speed of convergence or even if it converges at all.

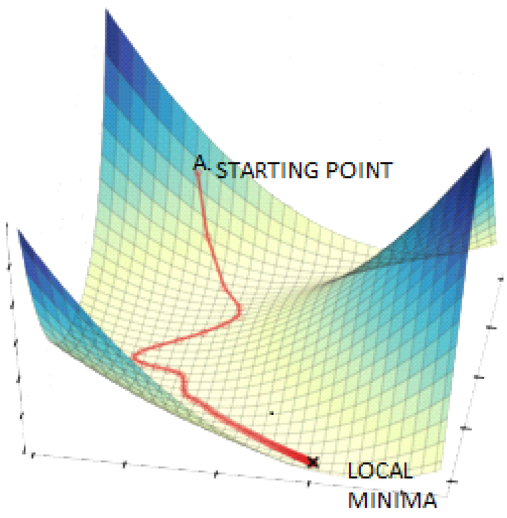


Figure 3.11: Overview of the gradient updates during optimization of a neural net. The path follows the direction of steepest gradient using SGD until a local minima is reached. Image taken from [149].

Therefore, the careful choice of many hyperparameters, such as the used algorithm to cope with plateaus, the size of a mini-batch to balance speed and accuracy of the updates, the initial configuration of the parameter space, the architecture of the neural network itself and most importantly the quantity and quality of the data is essential.

3.4 Bio Markers

As the data is one of the most important parts in having a data-driven optimization framework, such as neural networks, it is essential to clarify the characteristics of the datasets we utilized. Since these features are derived from the medical field, not everyone may be well-versed with their precise definitions and functions.

T1w/T2w myelination T1-weighted (T1w) and T2-weighted (T2w) imaging refer to two different types of magnetic resonance imaging (MRI) sequences that are commonly used to visualize the brain and other parts of the body. These sequences are based on the way that different tissues in the body respond to a magnetic field [150].

Myelination refers to the process by which cells in the central nervous system called oligodendrocytes produce a fatty substance called myelin, which wraps around the axons (long, thin extensions) of neurons. Myelin acts as an insulating layer, helping to speed up the transmission of nerve impulses and protect the axons from damage [151].

T1w MRI sequences are generally better at visualizing tissues with higher amounts of water and fat, such as muscle and white matter, whereas T2w MRI sequences are generally better at visualizing tissues with higher amounts of water, such as CSF and gray matter [152], [153].

[154] proposed to divide T1-weighted images by T2-weighted images to create a relative myelin map in order to assess tissue microstructure and better map the intra-cortical myelin.

Myelin Maps These myelin maps are images or representations of the myelin content in the brain or spinal cord. In medical imaging, myelin maps can be used to study the structure and function of the brain and spinal cord, as well as to diagnose and monitor a range of neurological conditions, such as multiple sclerosis, Alzheimer's disease, and brain injuries. They can also be used to monitor the progression of neurological conditions over time and to evaluate the effectiveness of treatments [151], [155].

Sulcal Depth Sulcal depth is a measure of the depth of the sulci, which are the grooves or indentations on the surface of the brain. The sulci are formed by the folds in the cerebral cortex, which is the outer layer of the brain responsible for higher cognitive functions such as thinking, learning, and memory. It is of interest in the study of neurological disorders and brain development. For example, changes in sulcal depth have been observed in individuals with conditions such as schizophrenia and Alzheimer's disease, and have also been found to change over the course of normal brain development [156].

Cortical Thickness Cortical thickness refers to the thickness of the cerebral cortex, which is the outer layer of the brain that plays a key role in higher brain functions such as thinking, learning, and perception, as also described in chapter 3.1. In addition, [157]–[159] has suggested that changes in cortical thickness may be linked to various brain disorders, including schizophrenia, Alzheimer's disease, and depression.

Curvature One way that curvature can be used as a biomarker is by measuring the curvature of brain structures such as the cortex (the outer layer of the brain). The cortex has a characteristic folded shape, with a series of grooves (sulci) and ridges (gyri) that help to increase its surface area. Changes in the curvature of the cortex can be indicative of various brain disorders or conditions, such as developmental abnormalities, brain injury, or neurodegenerative diseases [160], [161].

Average Curvature Average curvature is an important measure because it provides information about the overall shape and structure of the brain. It is calculated by taking the average of all the curvature measurements in a specific region of the brain, or across the entire brain.

Average curvature is useful because it helps to identify patterns or trends in brain shape and structure that may not be apparent when looking at individual curvature measurements. For example, if the average curvature of a specific region of the brain is consistently higher or lower than expected, it may indicate an underlying structural or functional abnormality and can be used to to compare brain shapes or structures between different groups of individuals, such as healthy controls versus patients with a specific brain disorder [162].

Pial Curvature Pial curvature refers to the curvature of the pial surface of the brain, which is formed by the pial cells, which are specialized cells that form the outermost layer of the brain. It is related to help researchers and clinicians better understand and treat brain disorders and injuries, similar to the overall and average curvature [163].

Volume One way that volume is used as a biomarker is to compare the volume of different brain regions or structures between individuals or groups. For example, studies [164], [165] have found that the volume of certain brain regions, such as the amygdala or the hippocampus, is correlated with certain psychological or psychiatric conditions.

Volume can also be used to track changes in brain structure over time. For example, studies have shown that the volume of certain brain regions can decrease with age or with the onset of certain conditions such as Alzheimer's disease. By measuring volume over time, researchers can track the progression of these changes and potentially identify early markers of these conditions [166], [167].

Area The area of a brain structure or region can be used as a biomarker in much the same way that volume is used [168], [169]. For example, studies [170], [171] have found that the area of certain brain regions, such as the prefrontal cortex, is correlated with certain cognitive abilities or psychological traits.

4 Method

This chapter presents the whole setup of the used approach to learn various tasks on cortical surface meshes, including the data preprocessing, model architecture and implementation details.

4.1 Architecture Overview

The overall architecture is based on the works of SiT and starts with a MRI as input, which gets passed through FreeSurfer [18] (see 4.2.1) to segment the brain images and calculate biomarkers, such as volume, surface area or cortical thickness (see 3.4). The outcome is a 3D mesh made out of 163,842 vertices per hemisphere with the aforementioned biomarkers calculated for each point. After several standard preprocessing techniques, like cleaning or balancing (more details in 4.4), the data gets projected onto a unit sphere. This simpler, but mathematically equivalent representation [111] allows a straightforward approach to resample the data to adapt to the given resources, especially compute power and amount of data. To feed the data into the neural network model, which is inspired by the vision transformer architecture [39], the mesh is tessellated with uniform triangles to generate "image patches". A final Multi Layer Perceptron (MLP) layer transforms the generated embedding space into predictions for the task at hand (age or AD prediction). An overview of the whole architecture with exemplary input is shown in figure 4.1. Below, the individual building blocks will be described in further detail, as well as depicting the tools used to implement the architecture.

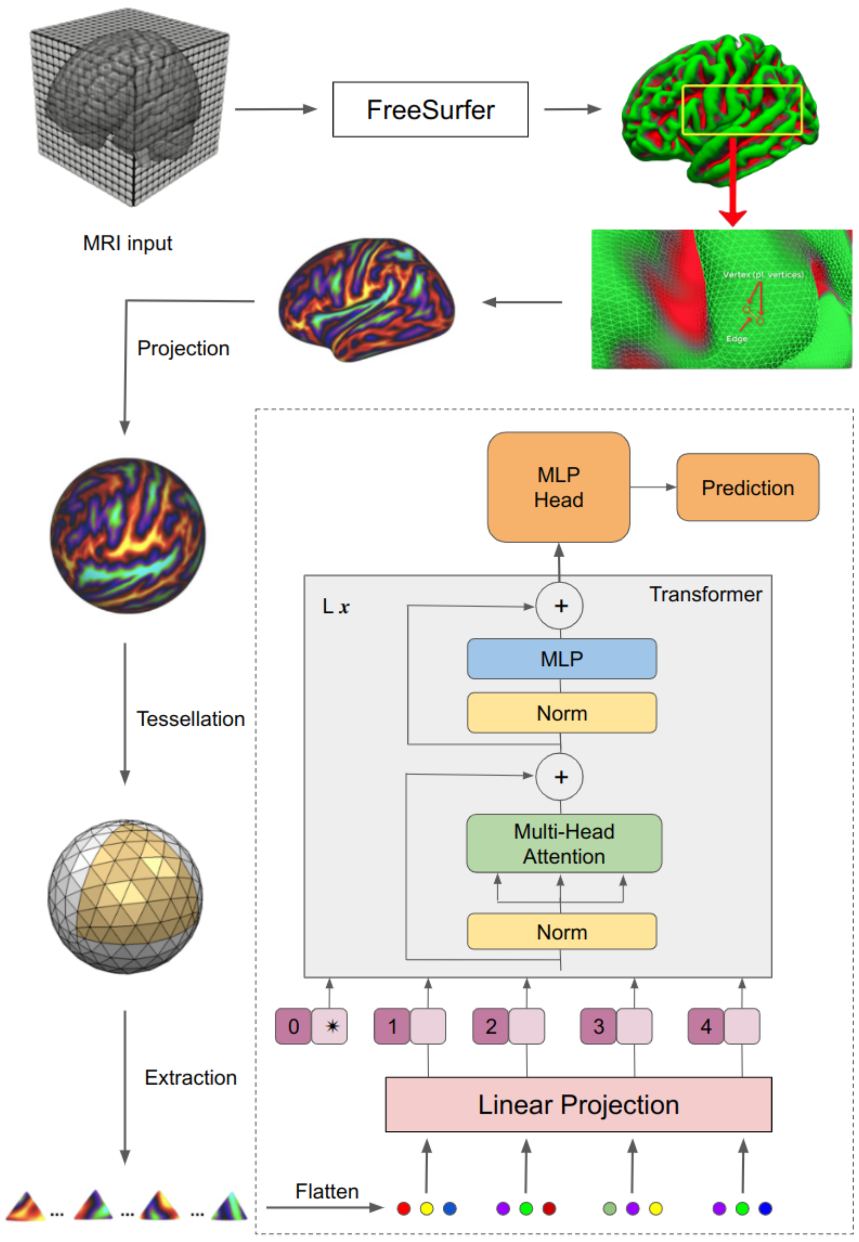


Figure 4.1: Overview of the architecture, depicting the preprocessing and neural network (inside dashed box). We convert MRI scans into meshes with the help of FreeSurfer before projecting them onto a unit sphere. In order to feed the data to a vision transformer architecture, we extract triangular patches and flatten them. Image derived from [30], [172]

.

4.2 Data Preprocessing

4.2.1 FreeSurfer

FreeSurfer is a software suite for analyzing, processing and visualizing structural and functional neuroimaging data from longitudinal and cross-sectional studies, such as MRI and positron emission tomography scans. It includes a full processing stream involving skull-stripping, bias field correction, registration, and anatomical segmentation as well as cortical surface reconstruction, registration, and parcellation. In our work, we use the already preprocessed datasets of ADNI, dHCP and HCP, which optimized the pipeline using FreeSurfer for the data at hand. Details on the exact steps conducted can be found in their respective papers [173]–[175].

4.2.2 Data Split and Normalization

Following to the generation of features from the 3D data, the samples are divided into a train (60%), validation (20%) and test set (20%). To guarantee meaningful results in the medical domain, it is good practice to balance the distribution of gender and age in the used data. Furthermore, if diseases are analyzed, it is advised to ensure a statistically identical distribution of affected and healthy patients in each set. We achieve this by using the method introduced by [176], which first estimates the propensity score with a logistic regression. Next, they construct an empirical quantile-quantile plot from which the maximum deviation from the 45-degree line as a measure of imbalance is computed. Prior to transforming the data, each biomarker, computed by FreeSurfer, is normalized, s.t. the data has a mean of zero and a standard deviation of one for each channel.

4.2.3 Projection

For the projection we used the method of barycentric interpolation (see 3.2.1), provided by the HCP workbench software [177] to resample the cortical data from its template resolution to an sixth order icosphere. The representation of an icosahedral icosphere exhibits a predominantly regular hexagonal arrangement across the sphere, which enables convenient downscaling and upscaling procedures, facilitated by the iterative generation of icospheres at varying resolutions, as depicted in figure 4.2. The selection of the sixth order was based on its alignment with the template resolution of 32,492 vertices, resulting in the smallest approximation error among the options, as it closely matched its resolution with 40,962 equally-spaced vertices.

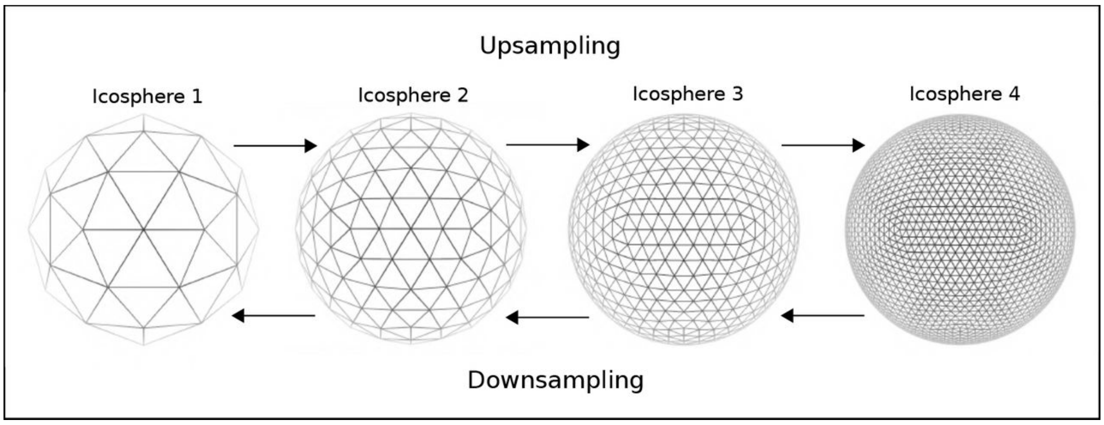


Figure 4.2: Efficient up- and downscaling of icosahedrons. Image taken from [79].

4.2.4 Tesselation

The usage of an icosahedron not only facilitates the scaling of cortical data, but also enables the extraction of 2D patches from the 3D sphere through a regular triangular tessellation. An icosahedron is, by definition, a polyhedron (a solid with flat faces) with 20 triangular faces, that is inscribed within a unit sphere, such that the vertices of it touch the surface of the sphere. These 20 triangular faces can then be divided into smaller triangles, resulting in a more fine-grain tessellation of the unit sphere. In our case, 320 triangular patches were extracted from the sixth order icosphere to feed them into the neural network architecture. These 2D patches correspond to the image patches in the original vision transformer architecture by [39].

4.3 Model Architecture

After the description of the preprocessing the following sections show the details of the underlying neural network model and its components.

4.3.1 Embedding Generation

Equivalently to the Vision Transformer, we flatten the generated patches transforming the 2D high resolution data X to a N-dimensional sequence $\hat{X} \in \mathbb{R}^{N\times(VC)}$ with V being the number of vertices and C being the number of channels, in our case the number of biomarkers at each vertex. To match the needed dimension D to employ the underlying transformer architecture [178], a learnable linear layer is used to transform the features to the embedding space of dimension D. Furthermore, an additional embedding is prepended, serving the purpose of BERT's [$class$] token. Lastly, to retain the positional information encoded in the rich 3D representation, each transformed patch gets a learned positional encoding of where on the sphere it belonged to. These embeddings are then used to finetune a pretrained vision transformer [178].

4.3.2 Data-efficient Transformer (DeiT)

The used vision transformer is a data-efficient transformer introduced by [178]. They use the same architecture as proposed by [39], but employ different training strategies to ensure lightweight training. Through a student-teacher-approach as well as distillation, the authors were able to improve the image througput by over two times whilst keeping the number of parameters the same (86 million). Additionally, they introduced two smaller models with 22 million and five million parameters respectively, that have a similar performance to the vision transformer base model. In our work, we mainly used the provided tiny model due to timing and computational constraints. More information about the internal layers are already described in section 3.3.2 and following.

4.3.3 Prediction

For the prediction, a MLP layer is appended to the transformer architeture, mapping the processed class embedding back to an actual class or continious value depending on the task at hand, similar to how it is used in the NLP domain. Therefore, the information encoded in the patches needs to be shared with the class embedding for prediction. During training time, this leads to a bottleneck, as the supervision is handled only by the class embedding [178].

4.4 Regularization

Transformers are powerful models, but many previous studies showed that they also require huge amounts of training data [39], [179], [180]. This poses a challenge in the medical domain, as the datasets are relatively small compared to datasets in other areas. One choice to mitigate this problem is to use data-efficient transformers, that use a combination of novel training techniques to improve the transfer learning abilities of the transformer architecture. Further, several regularization methods can be employed to limit the expressiveness of the model, but help with convergence. In this work, we tried several regularization mechanisms, spanning the training procedure, as well as the data and model generation, which are shortly highlighted below.

4.4.1 Training

During training, there are many factors that influence the model performance and can work as regularizers.

Optimizer The choice of the optimizer is crucial for a deep learning task. The general idea is to use the first derivative to calculate the parameter updates, however, due to the complex loss landscape, the model might get stuck at some saddle points, which can be overcome with the correct optimizer. However, the use of optimizers like Adam might loose the regularization properties of stochastic gradient descent, which allows to restrict the model's capacity by adapting the batch size.

Batch Size The batch size i.e. can be used to escape saddle points, because of the noise induced through the stochastic behavior of varying samples. Additionally, it works similar to a L2-regularization, because larger batch sizes introduce more noise into the parameter updates, reducing the risk of overfitting.

Scheduler Changing the learning rate over time also allows an indirect regularization, as the learning rate determines the speed of convergence and with that also the speed at which the model learns. This not only allows to do bigger update steps in the beginning and speed up the convergence overall, but also allows the model to explore the solution space more thourouhgly and thus possibly find solutions with better generalization.

Early Stopping As the name suggests, this method stops the training if the generalization gap gets worse, meaning that the validation loss rises again while the training error still goes down. This also limits the model's capacity as the parameters can not be updated often enough to fit every single data point.

Weight Decay is a technique, that adds a penalty term to the loss function, which hinders the optimizer in making large parameter updates. This stabilizes the training and also hinders the model in finding an exact mapping from the training data to the labels by keeping the weights small. Additionally, as weight decay employs a L2 regularization, it hinders the model to put full or no importance to a certain input feature.

Temperature is a parameter added to the softmax function, which is used in most classification tasks. The parameter softens the target distribution produced by the model introducing noise on the class prediction, which makes the task harder and forces the model to generalize better, decreasing the risk of overfitting.

4.4.2 Data

Independent of the training of a neural net, the data can be manipulated on its own to work in a regularizing fashion.

Shuffling Mixing up the order of the training samples, prior to each training epoch, adds extra randomness and reduces the dependence of the order of the training samples. This removes any bias that is incoporated in the training data, such as the data being sorted by class labels. Additionally, this stochasticity reduces overfitting as the model can memorize the order of the training data leading to better generalization.

Adding Noise Similar to the noise introduced by shuffling the data, a random sampled, gaussian distributed vector can be added to the data before feeding it to the model. The effect being similar to the shuffling, because every training epoch will be different. Furthermore, the robustness of the model is increased, as small deviations from the seen training data will not cause the model to change its prediction.

Balancing By either up- or downsampling some samples in the data the distribution of the classes can be improved, leading to a balanced representation of all the classes. If one class is overrepresented by a huge margin the model tends to always predict this class, which is mitigated by balancing it out. However, using downsampling and removing a lot of samples might make the overall amount of data too scarce and loosing valuable information, whereas upsampling has the problem of introducing a large bias towards the few seen samples of the class that get duplicated. In general, if the imbalance is big, it makes sense to check the data quality of the underrepresented class, to not duplicate samples that don't belong to the correct distribution of that data. Overall, it can be a valuable technique to regularize the model to not focus on a single class only.

4.4.3 Model

Lastly, one can incorporate regularization already in the model design itself.

Model Size The size of the model, which can be characterized by the number of layers, significantly impacts the model's capacity. Larger models may be able to capture more complex patterns in the data, but are also more prone to overfitting. Additionally, Occam's razor says, that simpler explanations are more likely to be correct. Meaning a simpler model restricts the capacity, but can lead to better generalization and less computational effort.

Dropout Another way to regularize a model on a layer basis is dropout. It involves randomly setting a proportion of the model's neuron units to zero during training. This forces the model to rely on a subset of its neurons at each training step, which has several benefits. First, this reduces the co-adaption of several neurons that always fire together, making the model more flexible without the need of more parameters. Further, the dropping out of some neurons introduces noise into the network, which also helps to reduce the risk of the model memorizing the data. In general, the mechanism can be seen as an ensemble training method, because many different, but similar models are trained simultaneously and combined during inference.

Batch Normalization Batch normalization is primarily used as a technique to improve the training process and address issues related to internal covariate shift. However, it can also have a regularization effect on neural networks. Overall, it normalizes the activations of each mini-batch, improving training stability and accelerating convergence. However, with the normalization it introduces additional noise, which reduces the reliance on specific-values within a mini-batch making it more robust to variations in the input.

For more available techniques or the mathematical background of the presented solutions, we refer to the paper [181].

4.5 Tools

In this next section, we want to highlight some tools, that were used during this project, which greatly helped to conduct meaningful research and thus deserve some mention.

Pytorch Lightning *PyTorch Lightning* [182] is an open-source Python library that provides a lightweight and flexible interface for organizing, scaling, and standardizing PyTorch code. It is designed to simplify the process of training complex machine learning models while improving their readability and maintainability. It separates the data handling and the model to ensure no data leaking and allows easy exploration by defining flexible data pipelines and model selection. Furthermore, it takes care of many best practices in the background, like switching between train and evaluation mode of the model, providing a fail-safe and readable codebase.
By leveraging PyTorch Lightning, we write more modular and reusable code as it promotes a clear separation between the research and engineering aspects of deep learning projects. We benefit by easier experimentation with different models, hyperparameters, and data pipelines and encouraged to use best practices together

with a standardized structure, which facilitates collaboration among researchers and developers.

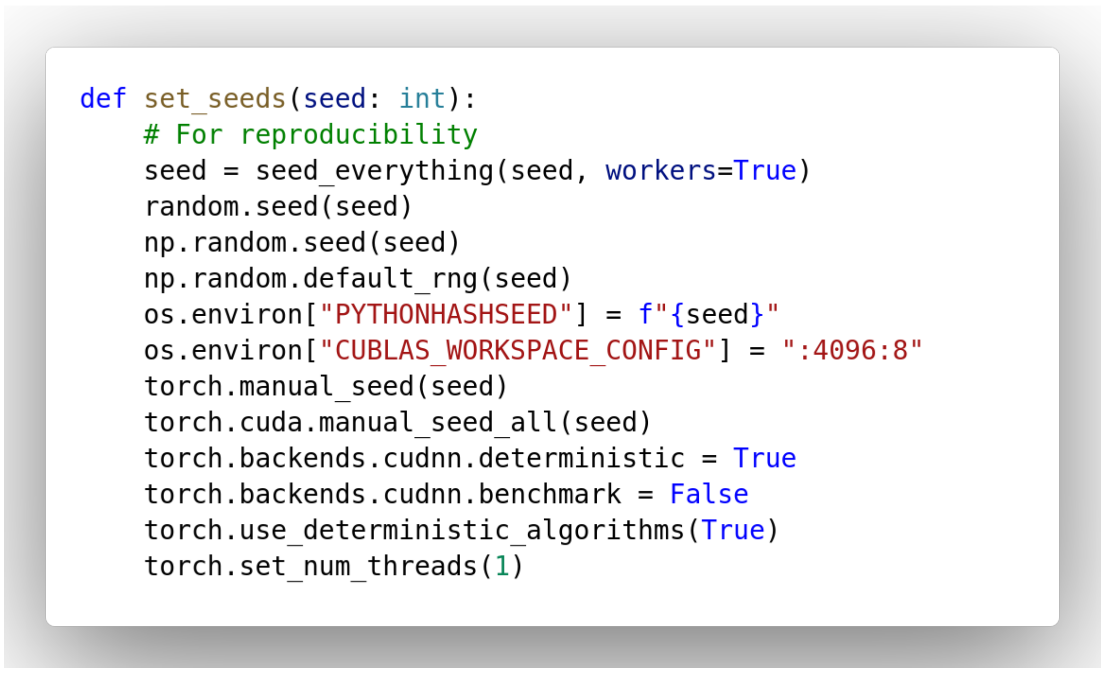
Weights and Biases Another tool we use to improve the quality of our research is *Weights and Biases* [183]. It is a machine learning experiment tracking and visualization platform providing tools and infrastructure to help researchers and data scientists track, organize, and visualize their machine learning experiments, models, and datasets.
The advantage of W&B over other popular tools like TensorBoard is the extensive experiment tracking. This tool allows to not only show training curves, but also stores entire datasets and models all linked together to reproduce the exact same results again. Furthermore, it supplies the same ease of use in creating graphs and other visualization, but enhances them by allowing some manipulation post-run. Especially, allowing to create new plots, combine or cluster them on all available metrics logged within the run. Even post-hoc analysis of different runs to compare the results is possible, supporting in selecting the best experiments to conduct further research by not only looking at single numbers emphasizing a more hollistic view on model performance.

Ray Tune Lastly, we want to mention *RayTune* [184], which is an open-source Python library that provides a scalable and efficient framework for hyperparameter tuning and distributed training. It offers the ability to write clean and reusable code, that can be run on a single machine, but also on a large compute cluster without requiring any modification. It further eases the hyperparameter search by providing a wrapper on top of many well-known deep learning frameworks, such as Pytorch Lightning. Similarly, it also helps to employ best practices, like random search, which is prefered over grid search for large search spaces [185]. Additionally, the seemless integration with Weights and Biases opens up easy ways to keep track of the individual tuning runs and even allows thourough comparison.

4.6 Reproducibility

Each tool individually offers many benefits, the most important ones for us being reproducibility and clean, reusable code, but when combined these advantages become even larger. Therefore, we now want to quickly highlight and show how these tools were used to get reproducible results.
First, the importance of reproducibility will be revisited. Nowadays when starting with a research topic, there are usually many methods and results available. Being able to re-

produce and verify existing works helps in exploring new ideas and advancing research. Furthermore, a reproducible experimental setup allows to quickly detect errors, which is common when developing new and untested networks. Also, comparing different versions and hyperparameters with deterministic results helps in understanding the model performance and the underlying factors of it to explore new ideas more effectively. Lastly, we want to underline the importance of reproducibility, when sharing the results, as the transparency and openness of models, data and methodologies helps to accelerate the collective progress in the field.
For this reason, we will now show how we achieved a reproducible setup, that can be reused by other researchers to help the whole community forward in achieving impactful, high quality results. The affected areas are subdivided in data, code, hyperparameters and hardware and software. In general, one important aspect for all these areas is fixing the random number generators, also called seeding, which impacts the data splitting, weight initialization and model training. Each library, that uses random number generators usually provides a way to set a seed to get deterministic results, however, often the interconnection between different libraries is neglected. In example, that *Pytorch* and *Pandas* depend on *Numpy*, which in turn depends on the Python base library *Random*. In many deep learning related repositories only the *Pytorch* variant is called, ignoring the downstream dependencies of the other libraries, leading to a non-deterministic behavior. *Pytorch Lightning* provides its own seed function ***seed_everything***(), which already includes the dependencies making the setup deterministic, when run on the same system every time. However, many models today are very big, requiring (distributed) compute engines, often in the cloud, where it cannot be guaranteed to always run on the same hardware. Furthermore, if other researchers try to reproduce the results, it is very unlikely, that they have access to the same hardware configuration. Therefore, in figure 4.3, a way to set seeds and operating system flags to ensure reproducible results on varying hardware resources is shown.

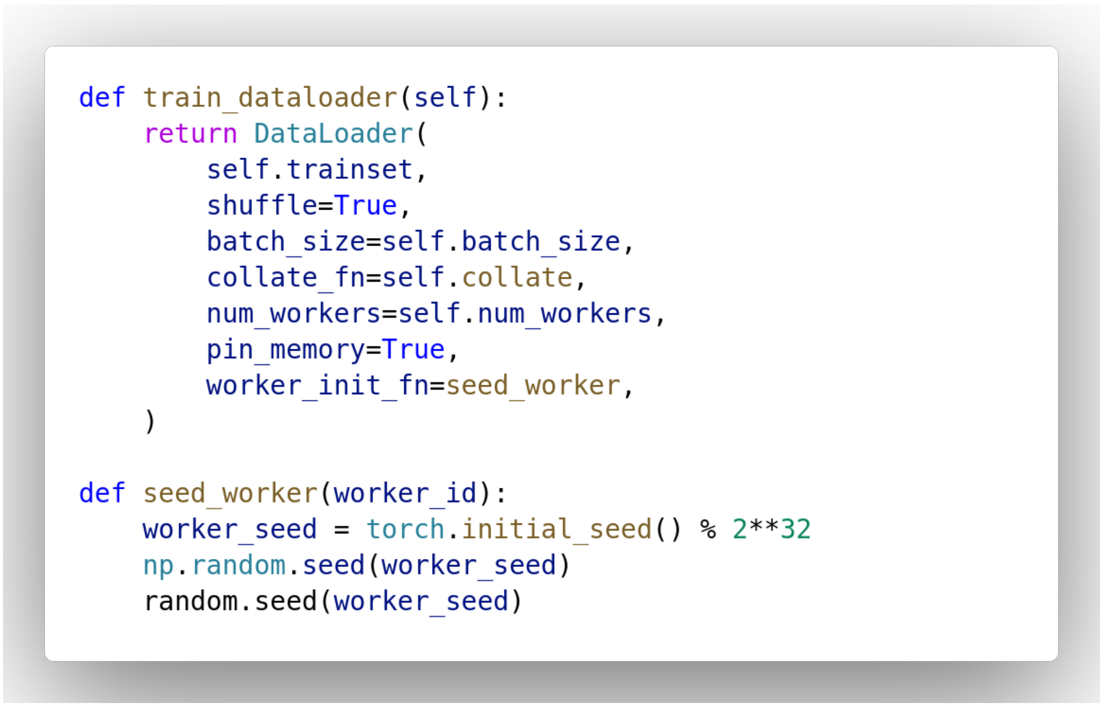


```python
def set_seeds(seed: int):
    # For reproducibility
    seed = seed_everything(seed, workers=True)
    random.seed(seed)
    np.random.seed(seed)
    np.random.default_rng(seed)
    os.environ["PYTHONHASHSEED"] = f"{seed}"
    os.environ["CUBLAS_WORKSPACE_CONFIG"] = ":4096:8"
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
    torch.use_deterministic_algorithms(True)
    torch.set_num_threads(1)
```

Figure 4.3: Code snippet to achieve a deterministic behavior of the *Pytorch* framework.

Note, that for this to work, the data needs to be deterministic as well. Therefore, in the following we show, how to process the data to guarantee a deterministic behavior. If all seeds are set correctly, the algorithms to split the data behave deterministically, but if multiple machines are used in distributed training, it might still happen that the allocation of the data follows a random pattern depending on the workload on the used nodes. This can be mitigated by providing a initialization function to the dataloader as shown in figure 4.4.

Lastly, we want to highlight again the tool *Weights and Biases*, as it helps to store all the remaining needed components for a deterministic, reproducible setup. As an additional safety mechanism, the preprocessed data, as well as the splits are stored with every run, reflecting even the slightest changes in preprocessing. The data itself is publicly available and the code and hyperparameters are also kept in sync with the run experiments by *Weights and Biases*. The important part here is the clever mechanism, that the data is tagged with a version, as well as the code and the run experiments. This allows to store all changes in data, code or hyperparameters together to be able to easily re-run an exact copy of an experiment even if it dates back a long time ago. This flexibility enables to re-run old setups with new insights to quickly combine new findings into promising new experiments and accelerate the research. Furthermore,



```
def train_dataloader(self):
    return DataLoader(
        self.trainset,
        shuffle=True,
        batch_size=self.batch_size,
        collate_fn=self.collate,
        num_workers=self.num_workers,
        pin_memory=True,
        worker_init_fn=seed_worker,
    )

def seed_worker(worker_id):
    worker_seed = torch.initial_seed() % 2**32
    np.random.seed(worker_seed)
    random.seed(worker_seed)
```

Figure 4.4: Code snippet to loading the data in a deterministic fashion even on multiple varying machines.

every run stores the hardware and software configuration, as well as the used command to start the experiment.

Overall, these tools and software snippets achieve a transparent and reproducible research setup, that was used to generate the results presented in the upcoming chapter. With this explicit mention, we hope to motivate more researchers to thrive for a similar setup, helping others to easily reproduce their results.

5 Results

In the following, the results of the conducted research are shown, starting with describing the experimental setting. Afterwards, we explain our conducted experiments and the reasoning behind them. Finally, a comparison of our method with related approaches on multiple datasets in terms of age prediction and Alzheimer's disease classification is presented.

5.1 Experimental Setting

This section gives a brief overview over the used datasets, evaluation metric and hyperparameters for this project.

5.1.1 Datasets

This research is based on three individual datasets, namely Developing Human Connectdome Project (dHCP), Human Connectdome Project (HCP) and Alzheimer's Disease Neuroimaging Initiative (ADNI). All datasets were preprocessed as described in the previous chapter 4.2. The model was adapted depending on the task, where age prediction was executed on all datasets, whereas Alzheimer's disease prediction was only run with the ADNI data.

dHCP The Developing Human Connectdome Project (dHCP) dataset contains structural and functional MRI data collected from a large cohort of neonatal human subjects (23 to 42 weeks). After preprocessing, the data includes the features of cortical thickness, curvature, sulcal depth and a myelin map and was split into train, validation and test set. The splits are balanced according to their age and sex and are of respective size: 350 train subjects and 117 subjects each for validation and test. This dataset was mainly used to verify the previous work by [30].

HCP A similar project, which focuses on adults is the Human Connectdome Project (HCP). It offers a collection of structural and functional MRI of over 1200 healthy human subjects (aged 22 to 37 years). After preprocessing and balancing again according to age and sex, the data is split into 724 train subjects, and 241 samples each for validation and testing. Each sample includes the same features as the dHCP dataset and the main focus for this data was to test the transfer abilities of the model towards adults.

ADNI The Alzheimer's Disease Neuroimaging Initiative (ADNI) project provides MRI scans collected from older individuals (55 to 90 years) with Alzheimer's disease, mild cognitive impairment (MCI), and healthy controls. The ADNI dataset was collected as part of a multi-year effort to study the progression of Alzheimer's disease and to identify early biomarkers of the disease. The incorporated features are volume, area, curvature, average curvature, curvature pial, sulcal depth and Cortical Thickness. For more information about the features, see section 3.4. In the experiments, we used two versions of the dataset. For the age prediction task, we only included healthy patients, whereas for the task of Alzheimer's disease classification all subjects were taken into account. The data was balanced according to age, sex and diagnosis before being split into train, validation and test sets with a ratio of 0.6, 0.2 and 0.2 respectively. Because the ADNI dataset was collected to track the progression of the disease multiple scans per patient exist, but we only used the initial scans in the beginning of the study. The main focus of this dataset in our research was to test the transfer abilities of the model architecture to new tasks.

5.1.2 Evaluation Metrics

To evaluate the performance of our model and compare the results to current state of the art research, we use the following metrics, which are split into regression and classification respective to the task the model is used for.

Regression

For the regression task of age prediction the below described metrics are used.

MAE The Mean Absolute Error (MAE) is defined by the L1-distance average over the number of samples N:

$$MAE(Y, \hat{Y}) = \frac{1}{N} \cdot \sum |y_i - \hat{y}_i| \tag{5.1}$$

It provides a measure of the average magnitude of errors without considering their direction, making it less sensitive to outliers compared to metrics such as the Mean Squared Error (MSE).

MSE The Mean Squared Error (MSE) measures the average squared difference between the predicted and actual values. It is given by:

$$MSE(Y, \hat{Y}) = \frac{1}{N} \cdot \sum |y_i - \hat{y}_i|^2 \tag{5.2}$$

Due to the squaring of the error term it emphasizes large errors more, meaning the model is trying to predict less values at the border of the distribution.

Classification

For the Alzheimer's disease classification the subsequent metrics were used to assess model performance.

Confusion Matrix A confusion matrix is a tabular representation that provides a comprehensive view on the performance of a classification model. It summarizes the predictions made by the model on a set of data by comparing them to the actual class labels, stating the amount of correct and wrong predictions the model made representing all different combinations of predicted and actual class labels. The confusion matrix consists of four different values:

- True Positive (TP): The model correctly predicted the positive class when the actual class was positive. It indicates the number of instances truly belonging to the positive class that were correctly identified as positive.
- False Positive (FP): The model incorrectly predicted the positive class when the actual class was negative. It represents the number of instances wrongly classified as positive.
- False Negative (FN): The model incorrectly predicted the negative class when the actual class was positive. It represents the number of instances wrongly classified as negative.
- True Negative (TN): The model correctly predicted the negative class when the actual class was negative. It indicates the number of instances truly belonging to the negative class that were correctly identified as negative.

These values can be used to calculate further metrics that express the performance of the model with a single number to more easily compare different approaches.

Accuracy The Accuracy of a model is the proportion of correctly classified instances over the total number of samples. It can be calculated by:

$$\boldsymbol{Accuracy} = \frac{\text{Number of Correctly Classified Instances}}{\text{Total Number of Instances}} \tag{5.3}$$

$$= \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FP + TN + FN}} \tag{5.4}$$

Accuracy alone may not be sufficient in certain cases, like in a highly imbalanced dataset where the majority class heavily outweighs the minority class. A classifier that predicts all instances as the majority class would still achieve a high accuracy due to correctly classifying the majority class instances. In such cases, additional evaluation metrics like precision, recall, or F1 score may provide a more comprehensive understanding of the model's performance.

Precision The precision of a model returns the accuracy of positive predictions. More precise, it measures the proportion of correctly predicted positive instances out of the total instances predicted as positive. This can be seen in the mathematical definition as well:

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} \tag{5.5}$$

Recall The recall, also known as sensitivity, measures the proportion of correctly predicted positive instances out of the total actual positive instances. It is given by:

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \tag{5.6}$$

Recall indicates the model's ability to find and include all positive instances without missing any.

F1 Score The F1 score considers both precision and recall, making it useful in cases where there is a trade-off between the two metrics or an imbalance in the dataset. It rewards models that can achieve high precision and recall simultaneously and is defined by the harmonic mean of the two aforementioned metrics:

$$\boldsymbol{F1\text{-}Score} = \frac{\boldsymbol{2 \cdot Precision \cdot Recall}}{\boldsymbol{Precision + Recall}} \tag{5.7}$$

5.1.3 Hyperparameters

For the task of brain age prediction, we used the found parameters from [30] to focus on the transfer abilities of the architecture. For completeness, the parameters can be seen in table 5.3. In the case of Alzheimer's disease classification, a thourough hyperparameter tuning was run using the *RayTune* library, presented in chapter 4.5. We evaluated the parameters based on the validation loss of the model and conducted 100 trials, while restricting the number of epochs to a maximum of 150 due to time limitations. An overview of the configuration used for the tuning can be found in table 5.1.

Parameter	Values
Batch size	4, 8, 16, 24, 32
Learning rate	10^{-7} to 10^{-2}
Architecture	tiny, small, base
Load weights	None, ImageNet
Dropout	$[0, 1]$
Optimizer	SGD, Adam, AdamW
Scheduler	None, CosineDecay, StepLR, ReduceLRonPlateau
Warmup	True, False

Table 5.1: Hyperparameters and their respective ranges used for tuning.

During the progress of the research for the classification task additional hyperparameters were added to the tuning process to regularize the model further, which are explained in more detail in chapter 5.2. All other parameters, such as the β values of Adam, have been selected based on existing deep learning literature.
For the training, the number of epochs was increased to 250 using the best hyperparameters that were computed during the tuning process. The best model, hereby, was chosen based on the best validation metric, mean absolute error for the age prediction and balanced accuracy for the classification respectively. More details on the training process and adaptions necessary to avoid overfitting, especially in the case of Alzheimer's disease prediction is presented in the following chapter.

5.2 Conducted Experiments

In this section, we present the different experiments, that were conducted, highlighting the successes, failures and adaptions to overcome these, in order to help future researchers, when faced with similar challenges.

5.2.1 Brain Age Prediction

First, the experiments reproducing the results from [30], as well as testing the transfer capabilities of the architecture to brain scans from young adults and adults will be presented.

Brain Age Prediction on neonatal Subjects

With the provided SiT model by [30], it was possible to achieve similar results to their respective numbers in the paper, creating a good baseline for our own implementation. As this is mainly a verification of the results reported by [30], we only briefly present the hyperparameters and the results in the two tables below.

Methods	Gestinal Age
SiT paper results	1.69
SiT implementation	1.66
Our method	1.61

Table 5.2: Comparison of SiT results with our method on the dHCP dataset predicting the gestinal age. All values are in weeks and lower is better for MAE.

Parameter	Values
Batch size	32
Learning rate	$\mathbf{10^{-5}}$
Architecture	tiny
Pretrained	ImageNet
Dropout	0
Optimizer	SGD
Scheduler	None
Warmup	No

Table 5.3: Hyperparameters for brain age prediction on the datasets of dHCP, HCP and ADNI.

Brain Age Prediction on Adults

After verification of our method against the works by [30], the more interesting experiment was the application on further datasets to test the transfer capabilities of the selected architecture. To highlight the difficulties present in the new datasets, we will have a look at the characteristics of them shown in table 5.4. Note, that the brain evolves and changes continiously throughout the life of a human.

Characteristic	dHCP	HCP	ADNI
# of samples	584	1206	632
Mean Age	37.2	28.8	72.6
Std Age	4.4	3.7	6.0
Age Range	23.7 - 42.1	22.0 - 37.0	55.1 - 89.6
Men	321	556	270
Women	263	650	362

Table 5.4: Characteristics of the datasets dHCP, HCP and ADNI. Note, that data for dHCP is in weeks, whereas the others are in years.

Not only the size and structure of the brain scans in the used datasets is different, but also the overall characteristics. HCP has a similar range and and standard deviation compared to the dHCP data, however, the unit is years instead of weeks, making it a challenging task to transfer to. For ADNI, there is even a higher variance, as well as a wider range in the data, creating additional difficulties. Nevertheless, it was possible to achieve competitive results, which are shown in table 5.7, where they also get compared to other state of the art methods. A more complete comparison follows in chapter 5.3.

5.2.2 Alzheimer's Disease Classification

Due to positive results from the previous executed experiments, further analysis was made to test the transfer to a classification setting on the ADNI dataset. For this to work, the model was adapted by applying a different head with three output neurons, one for each class, as well as a new loss function. The previously employed mean squared error was changed to a cross entropy loss. In figure 5.1 the data distribution of the classes is shown to get a better understanding of the task at hand.

There are three classes: AD or Dementia, Healthy or CN and MCI, with the number of patients with Dementia being underrepresented. Firstly, a naive training run with the parameters from the regression task showed mixed results. The loss curves, shown in figure 5.2 indicate some learning progress with the train loss slightly decreasing and a valley for the validation loss. However, the accuracy clearly shows that the model is not learning, with a fluctiation between zero and one for training and a static validation performance.

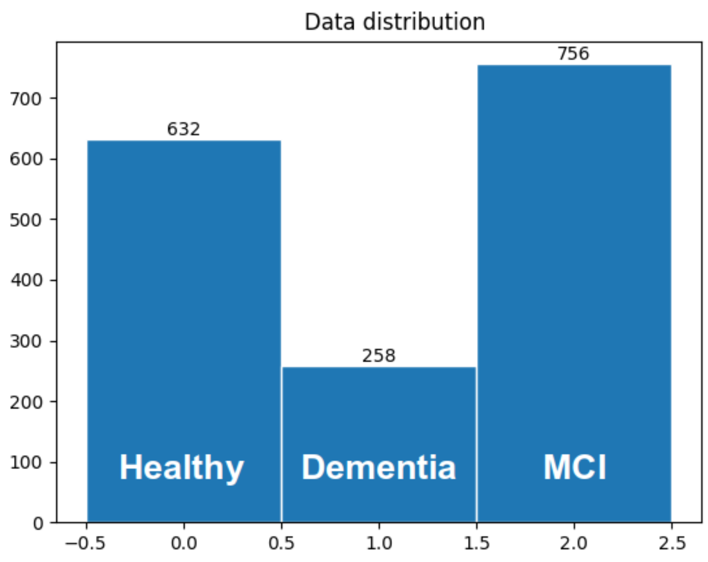


Figure 5.1: Data distribution of the ADNI dataset for the classification task of disease prediction. Showing the amount of samples on the y-axis plotted against the encoding of the three classes.

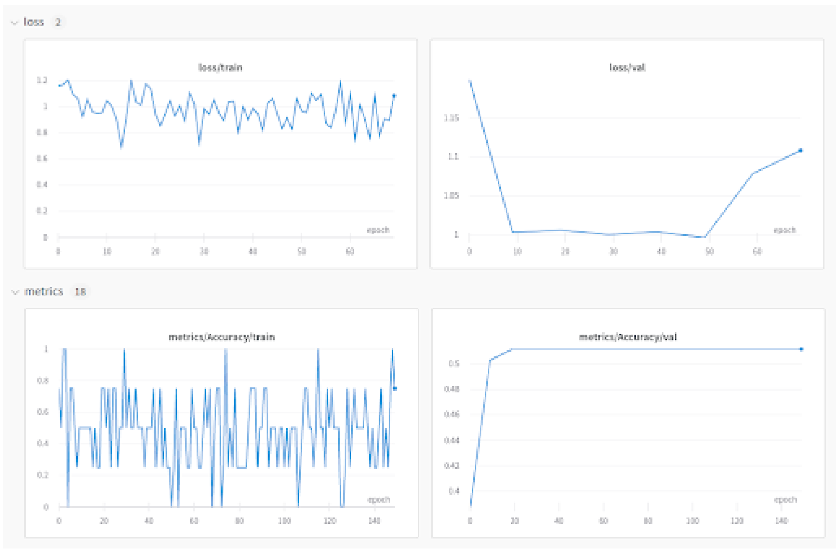


Figure 5.2: Loss and accuracy curves for initial training setup.

To verify, that this issue is not specific to the used parameters, a tuning run was made, which allowed the model to use different optimizers, adapt the learning rate over time using a scheduler, as well as training from scratch instead of using weight initialization from ImageNet. Furthermore, the batch size was varied to get a more stable training curve. For completeness and to get a better understanding of how the different parameters influence the model the additional parameters, portrayed in table 5.1, were also included in the random search. The results of one exemplary run of the first tuning process are shown in the image 5.3 below.

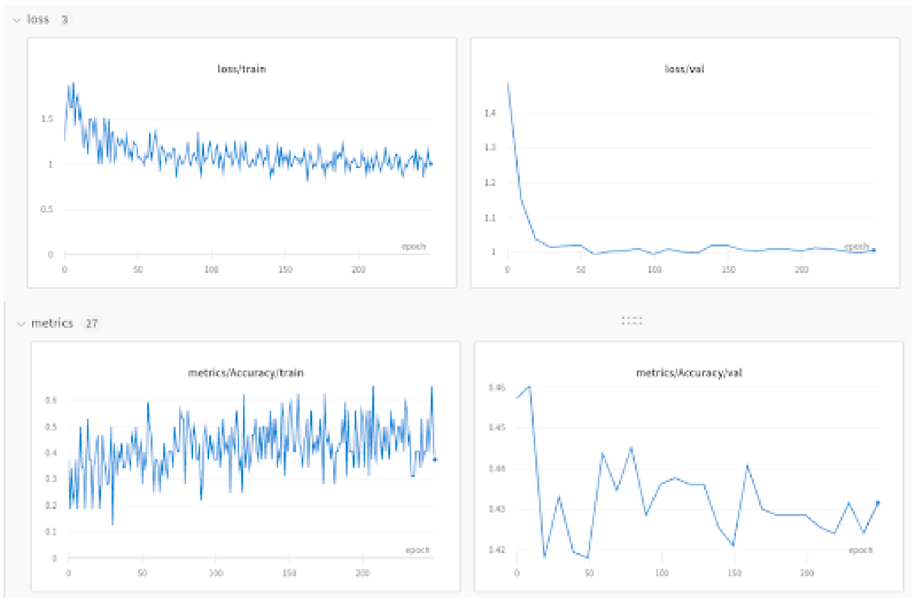


Figure 5.3: Loss and accuracy curves for an exemplary run of the first hyperparameter tuning.

A big improvement in the loss curves is visible, which show that over time the error for training and validation shrinks. Also, no upwards trend in the validation loss is noticeable towards the end of the tuning run, expressing that longer training could further boost the results. The metrics look better as well, indicating an upwards trend in training, but stagnate on the validation data, showing slight signs for overfitting. Hence, as a next step further regularization, as described in chapter 4.4, was introduced, as well as allowing different data splits to balance out the amount of training and validation data. The new additions are summarized in table 5.5.

Restricting the model in combination with added noise and different splits helped to combat overfitting, which can also be seen in figure 5.4. The loss curves remained similar to the previous example, showing that the model learns. But this time, the

Parameter	Values
Weight Decay	0.015 to 0.06
Noise	$\mathbf{10^{-3}}$ to $\mathbf{10^{-1}}$
Architecture	xxtiny, xtiny, tiny
Temperature	$\mathbf{10^{-2}}$ to $\mathbf{10^{2}}$
Splits	70 / 15 / 15 and 80 / 10 / 10

Table 5.5: Addtional hyperparameters used for classification.

metrics also reflect this with increasing over time for both training and validation. The overall performance, however, could still be improved, with a validation accuracy of 56% on a classification task with three classes. A comparison to possible results for this specific task will be covered in the next chapter, showing the room for improvement for our model.

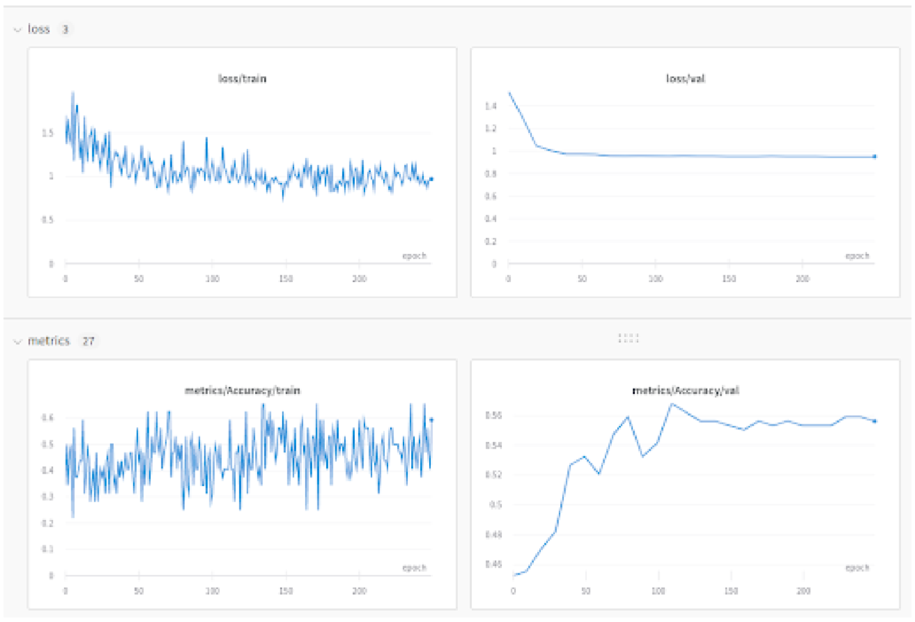


Figure 5.4: Loss and accuracy curves for an exemplary run of the second hyperparameter tuning.

Lastly, we want to mention, that we also tried to balance the dataset further by employing different strategies of over- and undersampling to keep the model from resorting to predicting the majority class. This approach, however, did not lead to any improvements in the model performance and thus further details about it are neglected.

In table 5.6, we collected the hyperparameters used to achieve the best result.

Parameter	Values
Seed	864
Batch size	32
Learning rate	0.00000719293780196941
Architecture	tiny
Pretrained	No
Dropout	0.20699865730499045
Embedding Dropout	0.15556303467928978
Optimizer	SGD
Scheduler	None
Warmup	No
Weight Decay	0.03178989086424102
Noise	0
Temperature	1
Splits	70 / 15 / 15

Table 5.6: Best hyperparameters used for classification.

5.3 Comparison with Related Work

In this section the best results obtained from the various conducted experiments are compared with current state of the art to put the numbers into perspective.

5.3.1 Brain Age Prediction

We compare the mean absolute error of our model against two machine learning models, a Lasso and Elastic Net model [91], [92], and a CNN model [92] on the HCP and ADNI datasets. Table 5.7 shows, that both machine learning models slightly outperform our method on the HCP data and more clearly on ADNI. It is important to note, however, that the compared models solely work on features extracted by FreeSurfer without incoporating any spatial knowledge of the data, as well as the use of an additional dataset for the training on ADNI. Furthermore, it needs to be stated, that no hyperparameter tuning was conducted for our model on the specific datasets, leaving room for improvement, whereas the other models had a thorough preprocessing and feature selection process to achieve these results. Therefore, it can be observed, that compute power can achieve similar performance as domain knowledge with careful feature selection. On the other hand, this also shows, that large neural networks are not always the best option to solve a problem.

	HCP		ADNI	
Method	MAE	MAE / Std	MAE	MAE / Std
Ours	3.054	0.825	4.47	0.745
Lasso regression	2.757	0.745	3.532	0.589
Elastic Net	2.792	0.755	3.012	0.502
CNN	-	-	3.076	0.513

Table 5.7: Comparison of our method with several other state of the art approaches on the HCP and ADNI datasets. All values are in years and lower is better for MAE.

5.3.2 Alzheimer's Disease Classification

For the classification, we analyze the accuracies of our model against three models from [24], namely a CNN, as well as the two popular architectures ResNet-50 [32] and VGG-16 [45] on the ADNI dataset. In contrast to our method, the other approaches use heavy preprocessing, such as skull stripping as well as intensity correction before extracting a 2D image from the 3D MRI scan. The results are summarized in table 5.8, where it can be seen, that our method is outperformed by all other approaches. Interestingly, VGG-16 achieves the best performance, even though, the introduction of skip connection with the ResNet model architecture usually boosts image task performance.

	Accuracy	
Method	Training	Validation
Ours	0.656	0.556
CNN	0.876	0.727
ResNet-50	0.916	0.767
VGG-16	0.949	0.807

Table 5.8: Comparison of our method with several state of the art approaches for Alzheimer's disease classification on the ADNI dataset. All values are in percent and higher is better for accuracy.

6 Discussion

In the following chapter, the limitations of our method will be outlined, as well as directions for future research are given. Furthermore, a short disclaimer about the practical usability of research like this is given.

6.1 Limitations

Dataset size A big challenge in the medical domain often is the dataset size, which becomes even more difficult, if large neural networks are transfered from the image domain to test their capabilities on medical data. Especially, transformers need vast amount of data to be trained, which gets better as recent results by [178] show, however, the current situation still remains challenging. Additionally, for the classification task, the uneven class distribution further complicated the training procedure, as cutting out even more data aggravates the first problem, whereas oversampling one class several times introduces new problems.

Feature set The idea of the model is to use 3D structure with some additional information to predict the wanted target variable, however, the added features present in the ADNI dataset were mainly structural as well. Works, like [155], state, on the other side, that biomarkers like myelin are crucial indicators for Alzheimers. Also, amyloid plaques and tau tangles are cited to help in recognizing AD [104].

Compute resources Laslty, the needed compute to train a 3D model and a transformer model in general is limiting this research. Most experiments were executed on the tiny model or even adjusted smaller variants to be able to run on our hardware in a reasonable amount of time. Especially, the tradeoff between model size and batch size was a problem, as a larger batch size indicated promising results, but were limited by the given hardware. Also using a higher order icosphere might boost performance, but could not be investigated. Lastly, a design choice of the underlying vision transformer architecture was not followed due to hardware constraints. The hidden dimension of the original ViT architecture is calculated by the patch size and the channels of the input, which differs in our case, leading to interpolation inaccuracies.

6.2 Potential negative Impact

If algorithms achieves promising results, like the state of the art methods mentioned in section 5.3, using them for practical assessment is the logical next step. Therefore, we want to mention, that a single number, such as accuracy can be misleading, especially for unbalanced data. A more thorough analysis of the results would be needed, including other metrics, as presented in section 5.1.2, to allow for medical usage. Furthermore, the trust should not be put solely into the prediction of a model, but rather help the trained professionals as an indication to start further examinations. In general, the data analyzed in this study only includes a tiny subset of the population, with one specific morphological degeneration. Hence, for new unseen data, we can not guarantee meaningful predictions and new special models might be needed. Furthermore, no investigations on the topic of fairness, bias and privacy were conducted in this research, but are crucial components in gaining authorization.

6.3 Future Work

Based on the conducted research in this paper, promising improvements could be achieved in future work, focussing on the following areas.

Additional features As mentioned in the limitations, it might be worth computing the myelin maps for the ADNI dataset to enhance the feature set by providing a biomarker that is not included in the structure already. Furthermore, a hybrid model might be interesting, computing the structural information with our method, extended by a linear classifier, trained on biomarkers, that are fed together into the final head to make the prediction.

Dataset size One promising direction might also be to increase the dataset size by either combining multiple datasets or by implementing an augmentation technique on 3D data to generate more training samples. A possible dataset, that is often used together with ADNI is the dataset from Beijing Aging Brain Rejuvenation Initiative (BABRI), which is also a longitudinal study about dementia in China with over 1200 subjects.

Graph Neural Network (GNN) Lastly, an idea to further investigate is to use a GNN around the current transformer model. The output of our model is generally a high dimensional space, in which the appended head tries to fit boundaries to separate the different classes. By using a graph network, the embedding space can be further refined using the message passing process, which together with attention allows to update the intra cluster dependencies, improving the latent space. This idea already achieved state of the art results in the image domain for the task of similarity learning [139].

7 Conclusion

In this work, we presented an adaption to the SiT architecture and tested its transfer abilities on different datasets, as well as new tasks. The performance for brain age prediction on new datasets suggests some generalization capabilities, however further experiments are needed to test if the current state of the art can be reached. Also, a thorough comparison of the latest research on standardized datasets might be interesting to evaluate the generalization on unseen data.
The performance for the classification task was far away from the current results of state of the art models and it is still unclear, whether the model actually learned a meaningful mapping or just predicted the majority class. Therefore, it does not seem as the proposed method is able to push the current state of the art forward in the classification setting, denying the phrased research objective.
Overall, we can say, that it makes sense to put more emphasize on the data preparation, as the comparison with other research showed, that significantly simpler models were able to outperform our approach.

List of Figures

List of Tables

Acronyms

GM	Gray Matter. 18
GNN	Graph Neural Network. 5, 9, 22, 33, 34, 66
GPR	Gaussian Process Regression. 13
HCP	Human Connectdome Project. 3, 4, 42, 52, 58, 62
Lasso	Least Absolute Shrinkage and Selection Operator. 12
LSTM	Long Short Term Memory Network. 7
MAE	Mean Absolute Error. 53
MCI	Mild Cognitive Impairment. 1, 2, 14, 16, 58, 68
ML	Machine Learning. 3, 16, 17
MLP	Multi Layer Perceptron. 40, 44
MR	magnetic resonance. 1, 3, 14
MRI	structural magnetic resonance imaging. iv, 1, 9, 11, 12, 13, 14, 16, 17, 18, 40, 42, 52, 53, 63
MSE	Mean Squared Error. 53, 54
NLP	Natural Language Processing. 7, 27, 28, 29, 44
PAD	Predicted Age Difference. 11
ResNet	Residual Network. 6
RF	Random Forest. 12
RNN	Recurrent Neural Network. 7
SGD	stochastic gradient descent. 36, 37, 69
SiT	Surface Vision Transformer. iv, 3, 4, 40, 57
SOTA	State of the art. 6, 7
SVR	Support Vector Regression. 12

TN True Negative. 54

TP True Positive. 54

WM White Matter. 18

Bibliography

[1] WHO, *Dementia*, Sep. 2022.

[2] J. Elflein, *Leading causes of death worldwide in 2019(in millions)*, Statista, May 2022.

[3] M. Guerchet, M. Prince, and M. Prina, "Numbers of people with dementia worldwide," Alzheimer's Disease International, Tech. Rep., Nov. 2020.

[4] J. Elflein, *Projected numbers of older people with alzheimer's in the u.s. from 2020 to 2060 (in millions)*, Statista, Apr. 2022.

[5] A. Wimo, G.-C. Ali, M. Guerchet, M. Prince, M. Prina, and Y.-T. Wu, "World alzheimer report 2015: The global impact of dementia: An analysis of prevalence, incidence, cost and trends," Alzheimer's Disease International, Tech. Rep., Sep. 2015, p. 33.

[6] A. Dias and V. Patel, "Closing the treatment gap for dementia in india," *Indian Journal of Psychiatry*, vol. 51, no. Suppl1, pp. 93–97, Jan. 2009.

[7] A. E. Nakamura, D. Opaleye, G. Tani, and C. Ferri, "Dementia underdiagnosis in brazil," *The Lancet*, vol. 385, pp. 418–419, Jan. 2015.

[8] S. Jitapunkul, S. Chansirikanjana, and J. Thamarpirat, "Undiagnosed dementia and value of serial cognitive impairment screening in developing countries: A population-based study," Geriatrics Gerontology International, Tech. Rep., Feb. 2009.

[9] C. Ferri, M. Prince, and R. Bryce, "World alzheimer report 2011: The benefits of early diagnosis and intervention.," Alzheimer's Disease International, Tech. Rep., Sep. 2011.

[10] M. W. Bondi, E. C. Edmonds, and D. P. Salmon, "Alzheimer's disease: Past, present, and future," *Journal of the International Neuropsychological Society*, vol. 23, no. 9-10, pp. 818–831, 2017.

[11] G. Perry, J. Avila, M. Tabaton, and X. Zhu, "Alzheimer's disease: A journey from amyloid peptides and oxidative stress, to biomarker technologies and disease prevention strategies—gains from aibl and dian cohort studies," *Journal of Alzheimer's disease*, vol. 62, no. 3, pp. 965–992, 2018.

[12] G. B. Frisoni, N. C. Fox, C. R. Jack, P. Scheltens, and T. P. M., "The clinical use of structural mri in alzheimer disease," *Nature Reviews Neurology*, vol. 6, no. 2, pp. 67–77, 2010.

[13] J. P. Lerch, J. C. Pruessner, A. Zijdenbos, H. Hampel, S. J. Teipel, and A. C. Evans, "Focal decline of cortical thickness in alzheimer's disease identified by computational neuroanatomy," *Cerebral Cortex*, vol. 15, no. 7, pp. 995–1001, 2005.

[14] E. C. Edmonds, J. Eppig, M. W. Bondi, K. M. Leyden, B. Goodwin, L. Delano-Wood, C. R. McDonald, and A. D. N. Initiative, "Heterogeneous cortical atrophy patterns in mci not captured by conventional diagnostic criteria," *Neurology*, vol. 87, no. 20, pp. 2108–2116, 2016.

[15] E. M. Frohmann, M. K. Racke, and C. S. Raine, "Multiple sclerosis — the plaque and its pathogenesis," *The New England Journal of Medicine*, vol. 354, pp. 942–955, 2006.

[16] E. Papuć and K. Rejdak, "The role of myelin damage in alzheimer's disease pathology," *Arch Med Sci*, vol. 16, pp. 345–351, 2018.

[17] T. Watanabe, X. Wang, Z. Tan, and J. Frahm, "Magnetic resonance imaging of brain cell water," *Nature*, vol. 9, no. 5084, 2019.

[18] B. Fischl, "Freesurfer," *Neuroimage*, vol. 62, no. 2, pp. 774–781, 2012.

[19] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm," *IEEE Transactions on Medical Imaging*, vol. 20, no. 1, pp. 45–57, 2001.

[20] D. W. Shattuck and R. M. Leahy, "Brainsuite: An automated cortical surface identification tool," *Medical Image Anal.*, vol. 6, no. 2, pp. 129–142, 2002.

[21] F. Bongratz, A.-M. Rickmann, S. Pölsterl, and C. Wachinger, *Vox2cortex: Fast explicit reconstruction of cortical surfaces from 3d mri scans with geometric deep neural networks*, 2022. DOI: `10.48550/ARXIV.2203.09446`.

[22] Y. F. Khan, B. Kaushik, C. L. Chowdhary, and G. Srivastava, "Ensemble model for diagnostic classification of alzheimer's disease based on brain anatomical magnetic resonance imaging," en, *Diagnostics*, vol. 12, no. 12, p. 3193, 2022. DOI: `10.3390/diagnostics12123193`.

[23] F. Gao, Y. Zhu, W. Lin, Y. Qin, H. Zhang, S.-L. Ding, and D. Shen, "Ad-net: Age-adjust neural network for improved mci to ad conversion prediction," *NeuroImage: Clinical*, vol. 27, p. 102 290, 2020. DOI: `10.1016/j.nicl.2020.102290`.

[24] B. Y. Lim, K. W. Lai, K. Haiskin, K. A. S. H. Kulathilake, Z. C. Ong, Y. C. Hum, S. Dhanalakshmi, X. Wu, and X. Zuo, "Deep learning model for prediction of progressive mild cognitive impairment to alzheimer's disease using structural mri," *Frontiers in Aging Neuroscience*, vol. 14, 2022, ISSN: 1663-4365. DOI: `10.3389/fnagi.2022.876202`.

[25] R. Jain, N. Jain, A. Aggarwal, and D. J. Hemanth, "Convolutional neural network based alzheimer's disease classification from magnetic resonance brain images," en, *Cognitive Systems Research*, vol. 57, pp. 147–159, 2019. DOI: `10.1016/j.cogsys.2019.04.002`.

[26] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," in *International Conference on Learning Representations*, 2018.

[27] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, *Geometric deep learning on graphs and manifolds using mixture model cnns*, 2016. DOI: `10.48550/ARXIV.1611.08402`.

[28] S. Tang, B. Li, and H. Yu, *Chebnet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations*, 2019. DOI: `10.48550/ARXIV.1911.05467`.

[29] F. Zhao, S. Xia, Z. Wu, D. Duan, L. Wang, W. Lin, J. H. Gilmore, D. Shen, and G. Li, *Spherical u-net on cortical surfaces: Methods and applications*, 2019. DOI: `10.48550/ARXIV.1904.00906`.

[30] S. Dahan, A. Fawaz, L. Z. J. Williams, C. Yang, T. S. Coalson, M. F. Glasser, A. D. Edwards, D. Rueckert, and E. C. Robinson, *Surface vision transformers: Attention-based modelling applied to cortical analysis*, 2022. DOI: `10.48550/ARXIV.2203.16414`.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Institute for Cognitive Science, University of California, San Diego, California, Tech. Rep., Sep. 1985.

[34] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943. DOI: `10.1007/BF02478259`.

[35] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[36] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. DOI: `10.1126/science.1127647`.

[37] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: `10.1109/TNN.2008.2005605`.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, T. Hilsenbeck, T. Unterthiner, M. Dehghani, R. Cipolla, T. Hazelden, J. Uszkoreit, *et al.*, "An image is worth 16x16 words: Transformative attention for image recognition," *arXiv preprint arXiv:2010.11929*, 2020.

[40] N. B. of Standards and Technology, *First digital image*, Accessed on [09.01.2023], 2022.

[41] S. Papert, "The summer vision project," 1966.

[42] A. B. Margaret, *Mind as Machine: a history of cognitive science*. Clarendon Press, 2006.

[43] P. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, IEEE, vol. 1, 2001, pp. I–511.

[44] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2014.

[46] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012.

[47] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," pp. 248–255, 2009. DOI: 10.1109/CVPR.2009.5206848.

[49] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312.

[50] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision (IJCV)*, vol. 111, no. 1, pp. 98–136, 2015. DOI: 10.1007/s11263-014-0733-5.

[51] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[52] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[53] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[54] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2013. DOI: 10.48550/ARXIV.1311.2524.

[55] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. DOI: 10.48550/ARXIV.1505.04597.

[56] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, 2016. DOI: 10.48550/ARXIV.1608.06993.

[57] V. Yathish, *Loss functions and their use in neural networks*, Towards Data Science, Aug. 2022.

[58] I. Bernardo, *4 metrics to evaluate your regression models*, Towards Data Science, Nov. 2021.

[59] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2016. DOI: 10.48550/ARXIV.1609.02907.

[60] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016. DOI: 10.48550/ARXIV.1606.09375.

[61] L. Wu, P. Sun, R. Hong, Y. Fu, X. Wang, and M. Wang, *Socialgcn: An efficient graph convolutional network based model for social recommendation*, 2018. DOI: 10.48550/ARXIV.1811.02815.

[62] V. Gligorijević, P. D. Renfrew, and T. e. a. Kosciolek, "Structure-based protein function prediction using graph convolutional networks," *Nature Communications*, vol. 12, p. 3168, 2021. DOI: `10.1038/s41467-021-23303-9`.

[63] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry*, 2017. DOI: `10.48550/ARXIV.1704.01212`.

[64] W. Lin, T. Tong, Q. Gao, D. Guo, X. Du, Y. Yang, G. Guo, M. Xiao, M. Du, X. Qu, and, "Convolutional neural networks-based mri image analysis for the alzheimer's disease prediction from mild cognitive impairment," *Frontiers in Neuroscience*, vol. 12, 2018, ISSN: 1662-453X. DOI: `10.3389/fnins.2018.00777`.

[65] J. B. Bae, S. Lee, W. Jung, S. Park, W. Kim, H. Oh, J. W. Han, G. E. Kim, J. S. Kim, J. H. Kim, and K. W. Kim, "Identification of alzheimer's disease using a convolutional neural network model based on t1-weighted magnetic resonance imaging," *Scientific Reports*, vol. 10, no. 1, p. 22 252, 2020, ISSN: 2045-2322. DOI: `10.1038/s41598-020-79243-9`.

[66] M. Desai and M. Shah, "An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (mlp) and convolutional neural network (cnn)," *Clinical eHealth*, vol. 4, pp. 1–11, 2021, ISSN: 2588-9141. DOI: `https://doi.org/10.1016/j.ceh.2020.11.002`.

[67] F. Gao, T. Wu, J. Li, B. Zheng, L. Ruan, D. Shang, and B. Patel, "Sd-cnn: A shallow-deep cnn for improved breast cancer diagnosis," *Computerized Medical Imaging and Graphics*, vol. 70, Mar. 2018. DOI: `10.1016/j.compmedimag.2018.09.004`.

[68] A. K. Bairagi, S. A. Alanazi, M. M. Kamruzzaman, M. N. Islam Sarker, M. Alruwaili, Y. Alhwaiti, N. Alshammari, and M. H. Siddiqi, "Boosting breast cancer detection using convolutional neural network," *Journal of Healthcare Engineering*, vol. 2021, p. 5 528 622, 2021, ISSN: 2040-2295. DOI: `10.1155/2021/5528622`.

[69] M. Cullell-Dalmau, S. Noé, M. Otero-Viñas, I. Meić, and C. Manzo, "Convolutional neural network for skin lesion classification: Understanding the fundamentals through hands-on learning," *Frontiers in Medicine*, vol. 8, 2021, ISSN: 2296-858X. DOI: `10.3389/fmed.2021.644327`.

[70] A. Chattopadhyay and M. Maitra, "Mri-based brain tumour image detection using cnn based deep learning method," *Neuroscience Informatics*, vol. 2, no. 4, p. 100 060, 2022, ISSN: 2772-5286. DOI: `https://doi.org/10.1016/j.neuri.2022.100060`.

[71] D. C. Febrianto, I. Soesanti, and H. A. Nugroho, "Convolutional neural network for brain tumor detection," *IOP Conference Series: Materials Science and Engineering*, vol. 771, no. 1, p. 012 031, 2020. DOI: 10.1088/1757-899X/771/1/012031.

[72] K. Blennow and H. Zetterberg, "The past and the future of alzheimer's disease fluid biomarkers," *Journal of Alzheimer's Disease*, vol. 62, pp. 1125–1140, 2018, ISSN: 1875-8908. DOI: 10.3233/JAD-170773.

[73] W. Lin, K. Hasenstab, G. Moura Cunha, and A. Schwartzman, "Comparison of handcrafted features and convolutional neural networks for liver mr image adequacy assessment," *Scientific Reports*, vol. 10, no. 1, p. 20 336, 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-77264-y.

[74] J. Huang, S. Wang, G. Zhou, W. Hu, and G. Yu, "Evaluation on the generalization of a learned convolutional neural network for mri reconstruction," *Magnetic Resonance Imaging*, vol. 87, pp. 38–46, 2022, ISSN: 0730-725X. DOI: https://doi.org/10.1016/j.mri.2021.12.003.

[75] D. Anand, R. Patil, U. Agrawal, R. V, H. Ravishankar, and P. Sudhakar, "Towards generalization of medical imaging ai models: Sharpness-aware minimizers and beyond," in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, 2022, pp. 1–5. DOI: 10.1109/ISBI52829.2022.9761677.

[76] J. Wang, Y. Peng, Y. Guo, D. Li, and J. Sun, "Ccut-net: Pixel-wise global context channel attention ut-net for head and neck tumor segmentation," in Jan. 2022, pp. 38–49, ISBN: 978-3-030-98252-2. DOI: 10.1007/978-3-030-98253-9_2.

[77] M. A. Islam, V. Vibashan, V. M. Jose, N. C. Wijethilake, Utkarsh, and H. Ren, "Brain tumor segmentation and survival prediction using 3d attention UNET," in *International MICCAI Brainlesion Workshop*, Springer, 2019, pp. 262–272. DOI: 10.1007/978-3-030-46640-4_26.

[78] C. Liu, W. Ding, L. Li, Z. Zhang, C. Pei, L. Huang, and X. Zhuang, "Brain tumor segmentation network using attention-based fusion and spatial relationship constraint," *arXiv preprint arXiv:2010.15647*, 2020.

[79] A. Fawaz, L. Z. J. Williams, A. Alansary, C. Bass, K. Gopinath, M. da Silva, S. Dahan, C. Adamson, B. Alexander, D. Thompson, G. Ball, C. Desrosiers, H. Lombaert, D. Rueckert, A. D. Edwards, and E. C. Robinson, "Benchmarking geometric deep learning for cortical segmentation and neurodevelopmental phenotype prediction," *bioRxiv*, 2021. DOI: 10.1101/2021.12.01.470730. eprint: https://www.biorxiv.org/content/early/2021/12/02/2021.12.01.470730.full.pdf.

[80] P. Besson, E. Rogalski, N. P. Gill, H. Zhang, A. Martersteck, and S. K. Bandt, "Geometric deep learning reveals a structuro-temporal understanding of healthy and pathologic brain aging," *Frontiers in Aging Neuroscience*, vol. 14, 2022, ISSN: 1663-4365. DOI: 10.3389/fnagi.2022.895535.

[81] M. F. Glasser, T. S. Coalson, E. C. Robinson, C. D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C. F. Beckmann, M. Jenkinson, *et al.*, "A multi-modal parcellation of human cerebral cortex," *Nature*, vol. 536, no. 7615, pp. 171–178, 2016.

[82] B. Fischl, N. Rajendran, E. Busa, J. Augustinack, O. Hinds, B. T. T. Yeo, H. Mohlberg, K. Amunts, and K. Zilles, "Cortical folding patterns and predicting cytoarchitecture," *Cerebral cortex*, vol. 18, no. 9, pp. 1973–1980, 2008.

[83] C. Kondo, A. Nakagawa, J. Mizutani, K. Fujimoto, Y. Sato, K. Togashi, and H. Matsuda, "An age estimation method using brain local features for t1-weighted images," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, IEEE, 2015, pp. 666–669.

[84] J. Wang and Y. Wang, "Age estimation using cortical surface pattern combining thickness with curvatures," *Med. Biol. Eng. Comput.*, vol. 52, no. 4, pp. 331–341, 2014.

[85] J. H. Cole, R. P. Poudel, D. Tsagkrasoulis, M. W. A. Caan, C. Steves, T. D. Spector, and G. Montana, "Brain age predicts mortality," *Mol. Psychiatry*, vol. 23, p. 1385, 2018.

[86] F. Liem, G. Varoquaux, J. Kynast, F. Beyer, S. Kharabian Masouleh, J. M. Huntenburg, L. Lampe, M. Rahim, A. Abraham, R. C. Craddock, *et al.*, "Predicting brain-age from multimodal imaging data captures cognitive impairment," *NeuroImage*, vol. 148, pp. 179–188, 2017.

[87] A. Abbott, "A problem for our age," *Nature*, vol. 475, no. 7355, S2, 2011.

[88] L. Baecker, E. Sörkmez, H. Walter, S. Erk, and S. Mohnke, "Brain age prediction: A comparison between machine learning models using region- and voxel-based morphometric data," *Human brain mapping*, vol. 42, no. 8, pp. 2332–2346, 2021. DOI: 10.1002/hbm.25368.

[89] P. F. Da Costa, J. Dafflon, and W. H. L. Pinaya, "Brain-age prediction using shallow machine learning: Predictive analytics competition 2019," *Frontiers in Psychiatry*, vol. 11, 2020, ISSN: 1664-0640. DOI: 10.3389/fpsyt.2020.604478.

[90] D. A. Wood, S. Kafiabadi, A. A. Busaidi, E. Guilhem, A. Montvila, J. Lynch, M. Townend, S. Agarwal, A. Mazumder, G. J. Barker, S. Ourselin, J. H. Cole, and T. C. Booth, "Accurate brain-age models for routine clinical mri examinations," *NeuroImage*, vol. 249, p. 118 871, 2022, ISSN: 1053-8119. DOI: `https://doi.org/10.1016/j.neuroimage.2022.118871`.

[91] J. Han, S. Y. Kim, J. Lee, and W. H. Lee, "Brain age prediction: A comparison between machine learning models using brain morphometric data," *Sensors*, vol. 22, no. 20, 2022, ISSN: 1424-8220. DOI: `10.3390/s22208077`.

[92] W. Huang, X. Li, H. Li, W. Wang, K. Chen, K. Xu, J. Zhang, Y. Chen, D. Wei, N. Shu, and Z. Zhang, "Accelerated brain aging in amnestic mild cognitive impairment: Relationships with individual cognitive decline, risk factors for alzheimer disease, and clinical progression," *Radiology: Artificial Intelligence*, vol. 3, no. 5, e200171, 2021. DOI: `10.1148/ryai.2021200171`. eprint: `https://doi.org/10.1148/ryai.2021200171`.

[93] P. L. Ballester, L. T. da Silva, M. Marcon, N. B. Esper, B. N. Frey, A. Buchweitz, and F. Meneguzzi, "Predicting brain age at slice level: Convolutional neural networks and consequences for interpretability," *Frontiers in Psychiatry*, vol. 12, 2021, ISSN: 1664-0640. DOI: `10.3389/fpsyt.2021.598518`.

[94] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, *et al.*, "An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest," *Neuroimage*, vol. 31, no. 3, pp. 968–980, 2006.

[95] R. Tibshirani, "Regression shrinkage and selection via the lasso," en, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. DOI: `10.1111/j.2517-6161.1996.tb02080.x`.

[96] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, vol. 9, 1997, pp. 155–161.

[97] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," en, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. DOI: `10.1111/j.1467-9868.2005.00503.x`.

[98] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, en. Berlin/Heidelberg, Germany: Springer, 2009, vol. 2. DOI: `10.1007/978-0-387-84858-7`.

[99] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[100] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," en, *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. DOI: 10.1214/aos/1013203451.

[101] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, en. Cambridge, MA, USA: MIT Press, 2006, ISBN: 026218253X.

[102] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, en. Cambridge, MA, USA: The MIT Press, 2012, ISBN: 978-0262018029.

[103] G. B. Folland, *A Course in Abstract Harmonic Analysis*. CRC Press, 1995.

[104] N. I. on Aging, *What happens to the brain in alzheimer's disease?*

[105] X. Zhou, S. Qiu, P. S. Joshi, C. Xue, R. J. Killiany, A. Z. Mian, S. P. Chin, R. Au, and V. B. Kolachalama, "Enhancing magnetic resonance imaging-driven alzheimer's disease classification performance using generative adversarial learning," en, *Alzheimer's Research & Therapy*, vol. 13, no. 1, p. 60, 2021, ISSN: 1758-9193. DOI: 10.1186/s13195-021-00797-5.

[106] N. I. of Neurological Disorders and Stroke, *Brain basics: Know your brain*.

[107] N. I. of Neurological Disorders and Stroke, *Brain basics: The life and death of a neuron*.

[108] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C. J. Honey, V. J. Wedeen, and O. Sporns, "Mapping the structural core of human cerebral cortex," *PloS biology*, vol. 6, no. 7, e159, 2008.

[109] M. Fouquet, N. Traut, A. Beggiato, R. Delorme, T. Bourgeron, and R. Toro, "Increased contrast of the grey-white matter boundary in the motor, visual and auditory areas in autism spectrum disorders," Sep. 2019. DOI: 10.1101/750117.

[110] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*, 1st. A K Peters/CRC Press, 2010. DOI: https://doi-org.eaccess.ub.tum.de/10.1201/b10688.

[111] D. W. Shattuck and R. M. Leahy, "Automated graph based analysis and correction of cortical volume topology," *Signal and Image Processing Institute, Department of Electrical Engineering-Systems, University of Southern California*, Jun. 2001.

[112] P. Popescu-Pampu, *What is the Genus?* (Lecture Notes in Mathematics). Springer International Publishing, 2016, ISBN: 9783319423128.

[113] K. Hormann and N. Sukumar, *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*. CRC Press, 2017, ISBN: 9781498763615.

[114] M. Kim and K. Shimada, *Geometric Modeling and Processing - GMP 2006: 4th International Conference, GMP 2006, Pittsburgh, PA, USA, July 26-28, 2006, Proceedings* (Lecture Notes in Computer Science). Springer, 2006, ISBN: 9783540367116.

[115] A. Ungar, *Barycentric Calculus In Euclidean And Hyperbolic Geometry: A Comparative Introduction*. World Scientific Publishing Company, 2010, ISBN: 9789814464956.

[116] H. Hege and K. Polthier, *Visualization and Mathematics III* (Mathematics and Visualization). Springer Berlin Heidelberg, 2013, ISBN: 9783662051054.

[117] R. Bapat, *Graphs and Matrices* (Universitext). Springer London, 2010, ISBN: 9781848829817.

[118] U. Brandes and T. Erlebach, *Network Analysis*. Springer, 2008, ISBN: 9783540807872.

[119] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: `1810.04805 [cs.CL]`.

[120] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: `1505.04597 [cs.CV]`.

[121] K. Kamnitsas, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation," *Medical Image Analysis*, vol. 36, pp. 61–78, 2017, ISSN: 1361-8415. DOI: `https://doi.org/10.1016/j.media.2016.10.004`.

[122] C. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Springer, 2006, ISBN: 9780387310732.

[123] F. Chollet, *Deep Learning with Python*. Manning Publications Company, 2017, ISBN: 9781617294433.

[124] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (Prentice Hall series in artificial intelligence). Pearson Prentice Hall, 2009, ISBN: 9780131873216.

[125] Y. Goldberg, "A primer on neural network models for natural language processing," *CoRR*, vol. abs/1510.00726, 2015. arXiv: `1510.00726`.

[126] M. Mandal, *Introduction to convolutional neural networks (cnn)*, Analytics Vidhya, May 2022.

[127] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: `1502.03167 [cs.LG]`.

[128] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[129] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, ACM, 2010, pp. 807–814.

[130] J. Jaworek-Korjakowska, P. Kleczek, and M. Gorgon, "Melanoma thickness prediction based on convolutional neural network with vgg-19 model transfer learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2748–2756. DOI: `10.1109/CVPRW.2019.00333`.

[131] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[132] C. Olah and S. Carter, "Attention and augmented recurrent neural networks," *Distill*, 2016. DOI: `10.23915/distill.00001`.

[133] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: `1409.0473 [cs.CL]`.

[134] Y. Kilcher. "Attention is all you need," Youtube. (2018), [Online]. Available: `https://www.youtube.com/watch?v=iDulhoQ2pro#t=19m5s`.

[135] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: `1710.10903 [stat.ML]`.

[136] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017. DOI: `10.1109/MSP.2017.2693418`.

[137] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. DOI: `https://doi.org/10.1016/j.aiopen.2021.01.001`.

[138] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *CoRR*, vol. abs/2104.13478, 2021.

[139] J. Seidenschwarz, I. Elezi, and L. Leal-Taixé, *Learning intra-batch connections for deep metric learning*, 2021. arXiv: `2102.07753 [cs.CV]`.

[140] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," en, *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, ISSN: 1476-4687. DOI: `10.1038/323533a0`.

[141] A.-L. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées," *C. R. Acad. Sci.*, vol. 25, p. 536, 1847.

[142] H. E. Robbins, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[143] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks*, 1993, 586–591 vol.1. DOI: `10.1109/ICNN.1993.298623`.

[144] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[145] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," *The Journal of Finance*, vol. 48, no. 1, pp. 65–91, 1993.

[146] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[147] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *Proceedings of the 30th international conference on machine learning*, vol. 28, pp. 1139–1147, 2013.

[148] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: `1412.6980 [cs.LG]`.

[149] R. Jain, *Types of gradient descent algorithms*, Hackerearth, Mar. 2017.

[150] D. C. Preston, *Magnetic resonance imaging (mri) of the brain and spine: Basics*, Case Western Reserve University, 2006.

[151] K.-A. Nave and H. B. Werner, "Myelination of the nervous system: Mechanisms and functions," *Annual review of cell and developmental biology*, vol. 30, pp. 503–33, 2014. DOI: `10.1146/annurev-cellbio-100913-013101`.

[152] M. N. Uddin, C. R. Figley, A. V. Avram, D. J. Chong, A. S. Barnett, J. He, T. E. Nichols, P. R. Szeszko, E. A. Hazlett, Y. Zhang, and A. W. Song, "Comparisons between multi-component myelin water fraction, t1w/t2w ratio, and diffusion tensor imaging measures in healthy human brain structures," *Scientific reports*, vol. 9, no. 1, p. 2500, 2019. DOI: `10.1038/s41598-019-39199-x`.

[153] D. Kawahara and Y. Nagata, "T1-weighted and t2-weighted mri image synthesis with convolutional generative adversarial networks," *Reports of practical oncology and radiotherapy : journal of Greatpoland Cancer Center in Poznan and Polish Society of Radiation Oncology*, vol. 26, no. 1, pp. 35–42, 2021. DOI: `10.5603/RPOR.a2021.0005`.

[154] M. F. Glasser and D. C. Van Essen, "Mapping human cortical areas in vivo based on myelin content as revealed by t1- and t2-weighted mri," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 31, no. 32, pp. 11 597–616, 2011. DOI: `10.1523/JNEUROSCI.2180-11.2011`.

[155] E. Papuć and K. Rejdak, "The role of myelin damage in alzheimer's disease pathology," *Archives of medical science : AMS*, vol. 16, no. 2, pp. 345–351, 2018. DOI: `10.5114/aoms.2018.76863`.

[156] I. Lyu, H. Lee, J. Lee, N. Jahanshad, K.-M. Cho, S. I. Kim, S. H. Lee, H. Park, J.-H. Kim, D.-H. Kang, *et al.*, "Sulcal depth-based cortical shape analysis in normal healthy control and schizophrenia groups," *Proceedings of SPIE–the International Society for Optical Engineering*, vol. 10574, p. 1 057 402, 2018. DOI: `10.1117/12.2293275`.

[157] E. Busovaca, G. M. Peavy, C. G. Wong, L. Delano-Wood, and M. W. Bondi, "Is the alzheimer's disease cortical thickness signature a biological marker for memory?" *Brain imaging and behavior*, vol. 10, no. 2, pp. 517–23, 2016. DOI: `10.1007/s11682-015-9413-5`.

[158] B. C. Dickerson, D. A. Wolk, and Alzheimer's Disease Neuroimaging Initiative, "Mri cortical thickness biomarker predicts ad-like csf and cognitive decline in normal adults," *Neurology*, vol. 78, no. 2, pp. 84–90, Jan. 2012, ISSN: 0028-3878. DOI: `10.1212/wnl.0b013e31823efc6c`.

[159] J. S. Suh, M. A. Schneider, L. Minuzzi, G. M. MacQueen, S. C. Strother, and S. H. Kennedy, "Cortical thickness in major depressive disorder: A systematic review and meta-analysis," *Progress in neuro-psychopharmacology & biological psychiatry*, vol. 88, pp. 287–302, 2019. DOI: `10.1016/j.pnpbp.2018.08.008`.

[160] L. Ronan, C. Scanlon, C. P. Doherty, W. H. Backes, P. M. Dockree, J. A. Buckley, J. O'Muircheartaigh, M. Moran, M. Carton, J. J. Foxe, and R. B. Reilly, "Cortical curvature analysis in mri-negative temporal lobe epilepsy: A surrogate marker for malformations of cortical development.," *Epilepsia*, vol. 52, no. 1, pp. 28–34, 2011. DOI: `10.1111/j.1528-1167.2010.02895.x`.

[161] M. Deppe, J. Marinell, J. Krämer, T. Duning, T. Ruck, O. J. Simon, F. Zipp, H. Wiendl, and S. G. Meuth, "Increased cortical curvature reflects white matter atrophy in individual patients with early multiple sclerosis," *NeuroImage: Clinical*, vol. 6, pp. 475–487, 2014, ISSN: 2213-1582. DOI: `https://doi.org/10.1016/j.nicl.2014.02.012`.

[162] R. Pienaar, B. Fischl, V. Caviness, N. Makris, and P. Grant, "A methodology for analyzing curvature in the developing brain from preterm to adult.," *International journal of imaging systems and technology*, vol. 18, no. 1, pp. 42–68, 2008. DOI: 10.1002/ima.v18:1.

[163] L. Ronan, R. Almeida, C. J. Stagg, V. Morash, S. Manohar, M. Husain, J. Driver, and N. Weiskopf, "Intrinsic curvature: A marker of millimeter-scale tangential cortico-cortical connectivity?" *International journal of neural systems*, vol. 21, no. 5, pp. 351–366, 2011. DOI: 10.1142/S0129065711002948.

[164] J.-Y. Rotge, N. Langbour, D. Guehl, B. Bioulac, N. Jaafari, M. Allard, B. Aouizerate, P. Burbaud, J. v. d. Plas, J.-F. Tignol, *et al.*, "Meta-analysis of brain volume changes in obsessive-compulsive disorder," *Biological psychiatry*, vol. 65, no. 1, pp. 75–83, 2009. DOI: 10.1016/j.biopsych.2008.06.019.

[165] C. Royer, N. Delcroix, E. Leroux, A. Razafimandimby, P. Brazo, and S. Dollfus, "Functional and structural brain asymmetries in patients with schizophrenia and bipolar disorders," *Schizophrenia research*, vol. 161, no. 2-3, pp. 210–214, 2015. DOI: 10.1016/j.schres.2014.11.014.

[166] E. Geuze, E. Vermetten, and J. D. Bremner, "Mr-based in vivo hippocampal volumetrics: 1. review of methodologies currently employed," *Molecular psychiatry*, vol. 10, no. 2, pp. 147–159, 2005. DOI: 10.1038/sj.mp.4001580.

[167] S. S. K. Nair and K. Revathy, "Quantitative analysis of brain tissues from magnetic resonance images," in *2009 International Conference on Digital Image Processing*, 2009, pp. 57–61. DOI: 10.1109/ICDIP.2009.34.

[168] D. Zhang, Y. Wang, L. Zhou, H. Yuan, and D. Shen, "Multimodal classification of alzheimer's disease and mild cognitive impairment," *NeuroImage*, vol. 55, no. 3, pp. 856–867, 2011, ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2011.01.008.

[169] I. Garali, M. Adel, S. Bourennane, and E. Guedj, "Histogram-based features selection and volume of interest ranking for brain pet image classification," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–12, 2018. DOI: 10.1109/JTEHM.2018.2796600.

[170] L. Bonilha, F. Cendes, C. Rorden, M. Eckert, P. Dalgalarrondo, B. Lafer, and J. C. Soares, "Neurocognitive deficits and prefrontal cortical atrophy in patients with schizophrenia," *Schizophrenia research*, vol. 101, no. 1-3, pp. 142–151, 2008. DOI: 10.1016/j.schres.2007.11.023.

[171] R. H. B. Benedict, J. M. Bruce, M. G. Dwyer, N. Abdelrahman, S. Hussein, B. Weinstock-Guttman, F. Munschauer, and G. Phillips, "Frontal cortex atrophy predicts cognitive impairment in multiple sclerosis," *The Journal of neuropsychiatry and clinical neurosciences*, vol. 14, no. 1, pp. 44–51, 2002. DOI: 10.1176/jnp.14.1.44.

[172] Andy Jahn, *FreeSurfer Tutorial: Basic Terms*, Andy's Brain Book, 2019.

[173] A. Makropoulos and et al., "The developing human connectome project: A minimal processing pipeline for neonatal cortical surface reconstruction," *NeuroImage*, vol. 173, pp. 88–112, 2018. DOI: 10.1016/j.neuroimage.2018.01.054.

[174] M. F. Glasser, S. N. Sotiropoulos, J. A. Wilson, T. S. Coalson, B. Fischl, J. L. Andersson, J. Xu, S. Jbabdi, M. Webster, J. R. Polimeni, D. C. Van Essen, and M. Jenkinson, "The minimal preprocessing pipelines for the human connectome project," *Neuroimage*, vol. 80, pp. 105–124, 2013.

[175] R. C. Petersen and et al., "Alzheimer's disease neuroimaging initiative (adni): Clinical characterization," *Neurology*, vol. 74, no. 3, pp. 201–9, 2010. DOI: 10.1212/WNL.0b013e3181cb3e25.

[176] D. E. Ho, K. Imai, G. King, and E. A. Stuart, "Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference," *Political Analysis*, vol. 15, no. 3, pp. 199–236, 2007. DOI: 10.1093/pan/mpl013.

[177] D. S. Marcus, J. Harwell, T. Olsen, M. Hodge, M. F. Glasser, F. Prior, M. Jenkinson, T. Laumann, S. W. Curtiss, and D. C. Van Essen, "Informatics and data mining tools and strategies for the human connectome project," *Frontiers in neuroinformatics*, vol. 5, p. 4, 2011.

[178] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *CoRR*, vol. abs/2012.12877, 2020. arXiv: 2012.12877.

[179] J. Beal, H.-Y. Wu, D. H. Park, A. Zhai, and D. Kislyuk, *Billion-scale pretraining with vision transformers for multi-task visual representations*, 2021. arXiv: 2108.05887 [cs.CV].

[180] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, *How to train your vit? data, augmentation, and regularization in vision transformers*, 2022. arXiv: 2106.10270 [cs.CV].

[181] J. Kukačka, V. Golkov, and D. Cremers, *Regularization for deep learning: A taxonomy*, 2017. arXiv: 1710.10686 [cs.LG].

[182] W. Falcon *et al.*, *Pytorch lightning*, version 1.6.3, 2019.

[183] Weights and Biases, *Weights and biases*, version 0.12.16, 2017.

[184] R. Liaw, E. Liang, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, and I. Stoica, “Tune: A research platform for distributed model selection and training,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2020. DOI: `10.1145/3394486.3403342`.

[185] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.