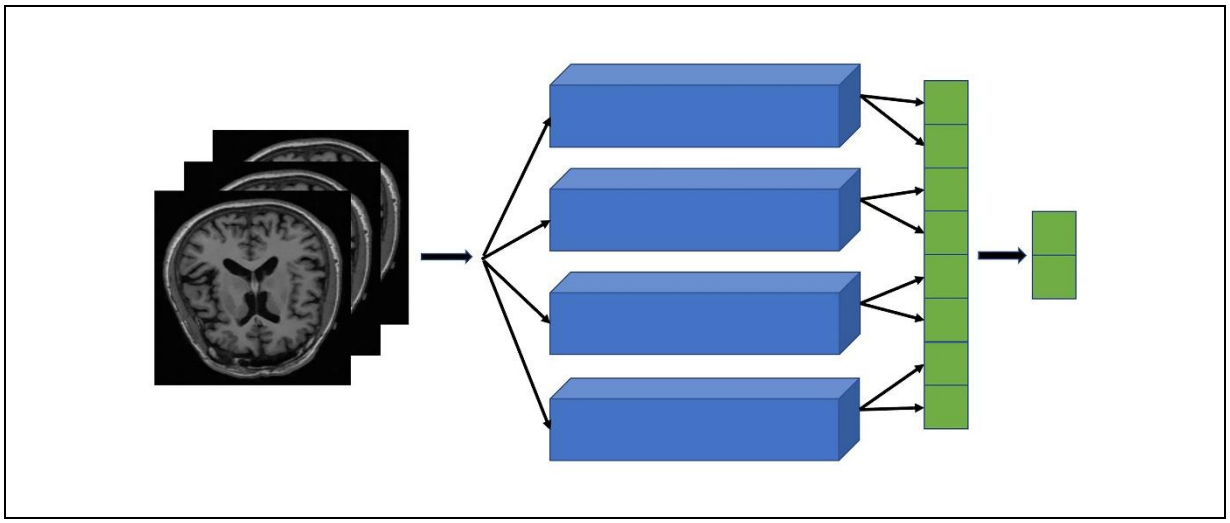
HOCHSCHULE LANDSHUT
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

Fakultät für Elektrotechnik und Wirtschaftsingenieurwesen

Bachelorarbeit zum Thema

CLASSIFICATION OF 3D MRI HEAD SCANS USING PRETRAINED CNNS – COMPARISON, OPTIMIZATION AND EVALUATION OF APPROACHES



Vorgelegt von
Roland Stolz
aus Landshut

eingereicht:

Betreuer: Prof. Dr. Stefanie Remmele

HOCHSCHULE LANDSHUT
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

Fakultät für Elektrotechnik und Wirtschaftsingenieurwesen

Erklärung zur Bachelorarbeit

(gemäß § 9 d, Abs. 3 APO)

Name, Vorname der/des

Student(in)en: .....

Hochschule Landshut
Fakultät Elektrotechnik und Wirtschaftsingenieurwesen

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

…… ……

(Datum) (Unterschrift der/des Student(in)en)

HOCHSCHULE LANDSHUT
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

Fakultät für Elektrotechnik und Wirtschaftsingenieurwesen

FREIGABEERKLÄRUNG DER/DES STUDENT(IN)EN

Name, Vorname der/des Studentin/en:

.....

Hiermit erkläre ich, dass die vorliegende Bachelorarbeit in den Bestand der Hochschulbibliothek aufgenommen werden kann und

- ☐ ohne Sperrfrist

oder nach einer Sperrfrist von

- ☐ 1 Jahr
- ☐ 2 Jahren
- ☐ 3 Jahren
- ☐ 5 Jahren oder länger

über die Hochschulbibliothek zugänglich gemacht werden darf.

....
(Datum) (Unterschrift der/des Student(in)en)

Table of Content

1 Purpose

This work focuses on providing a convolutional neural network (CNN) architecture for the classification of 3D MRI head scans. In order to offer a comprehensible recommendation for an architecture, this work has four main objectives:

- Implementation of two CNN architectures based on transfer learning, for the classification of 3D MRI head scans
- Optimization of both architectures to provide a fair comparison
- Comparison of both architectures with relevant criteria for the usage of the CNNs in medical diagnosis
- Evaluation of the effects of transfer learning on both architectures

2 Motivation

2.1 The relevance of artificial intelligence in medical imaging

In recent years artificial intelligence (AI) is obtaining an increasing interest in many different scientific areas. Especially in medical healthcare, there are a lot of possible applications, in which AI can provide assistance. For example, an algorithm for detecting atrial fibrillation in an Electrocardiogram (ECG) reached a sensitivity of 82.3% and a specificity of 83.4% [1]. Another field of intense interest is the automatic classification of lung nodules in computed tomography (CT) lung cancer screening. A neural network from 2017, designed for this task, achieved an error rate of 4.59% [2]. A more recent model from 2019 for predicting the risk of lung cancer on CT volumes, claims to achieve a better performance compared to six different radiologists in a situation, where there weren't any previous CT scans of the patient available [3]. Additionally, more abstract usage for machine learning can be found in medicine, e.g. predicting the risk of suicide for patients with a history of self-injury. This model can forecast a patient's suicide seven days prior, with accuracy under the curve (AUC) of 84% [4].

With an increasing interest in AI, the demand for medical imaging examination was also growing over the last two decades. The annual increase in image examinations in the USA and Canada rose continuously from 2000 to 2016, with annual growth rates of 11.6% from 2000 to 2006 and 3.7% from 2013 to 2016 [5]. Although the diagnostic quality benefits from an increasing amount of data, on the other hand, they significantly increase the workload of radiologists analyzing them. The average time expended by a radiologist for the complete diagnosis of a magnetic resonance imaging (MRI) head scan is about 18 minutes (Appendix A) [6]. An increasing number of imaging examinations, resulting in less time for the radiologist to analyze the data, or longer working shifts, could lead to higher error rates of their diagnosis [7]. Figure 1 shows that examinations of the head make up the largest portion of CT- and MRI-examinations in Germany in 2009.

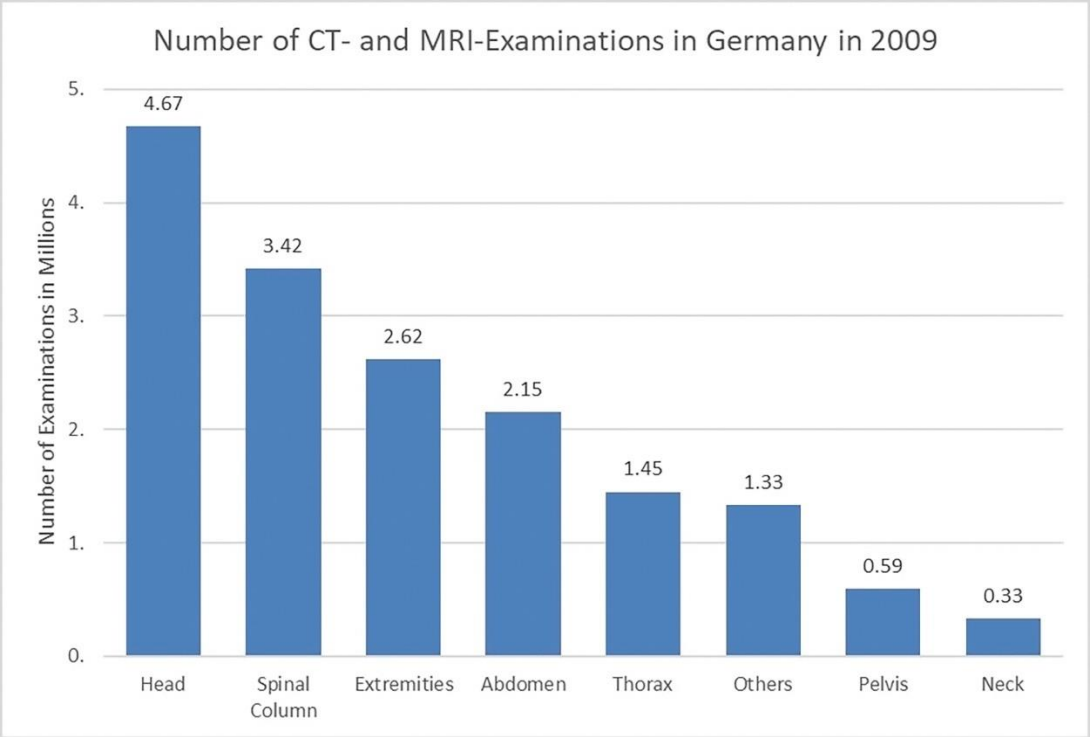


Figure 1. The number of CT- and MRI-Examinations in Germany by region in 2009 [8].

A statistic from the German radiology Mühleninsel in Landshut (Appendix B) shows that 56.36% of their MRI head scans and 60.20% of their CT head scans were normal findings. A different study from India indicates an even higher percentage of 78.2% for CT head scans [9]. As a result, radiologists lose much time in diagnosing normal findings, especially for examinations of the head, which they could spend on pathological diagnosis instead. Having a machine learning algorithm, that assists radiologists in their daily workflow, could lead to an improvement of temporal efficiency for diagnosing medical imaging scans, especially when it comes to automatic diagnosis of normal findings. Another benefit of a computer aided diagnosis (CAD) system, would be the constant availability of a second opinion for the radiologist. Two studies [10,11] demonstrated the benefit of a radiologist's second opinion when it comes to classification accuracy and patient care.

There is already research being conducted for the analysis of MRI head scans with AI. For example, Dong et al. developed an algorithm for the segmentation and detection of brain tumors that reached an accuracy of 86% in tumor segmentation on the 2015 Multimodal Brain Tumor Segmentation Challenge (BRATS) [12]. A different algorithm for the diagnosis of Alzheimer's disease (AD) in MRI head scans was developed by Lian et al. and reached an accuracy of 90% [13]. Other examples for research on the classification of MRI head scans with machine learning algorithms are from Suk et al. [14], Lin et al. [15] and Khvostikov et al. [16]. Despite many publications, there is no commercially available CAD system on the market right now, which uses a machine-learning algorithm for the classification of MRI head scans.

2.2 CNN - a powerful neural network for computer vision

All of the previously mentioned examples of machine-learning algorithms have one thing in common: They are based on artificial neural networks (ANN). More specifically, when it comes to medical image classification, almost all models inherit the architecture of a convolutional neural network (CNN). The modern CNN architecture was first introduced by Yann LeCun and Patrick Haffner in the paper: "Object Recognition with Gradient-Based Learning" in 1999 [17]. The model invented by the authors, called "LeNet-5", consisted of only two convolutional layers but managed to achieve an error rate of 0.7% on the MNIST dataset. Concluding this paper, they stated: "For 2D shape recognition, convolutional neural networks have been shown to eliminate the need for hand-crafted feature extractors" [17], already realizing the potential CNNs have for computer vision. Although the "Le-Net" architecture had proven to be comparable or even better than feature extracting algorithms, the real breakthrough for CNNs happened in 2012, with the introduction of a deep convolutional neural network called "AlexNet". The model consisted of five convolutional layers and classified an image input with a size of 224 x 224 x 3 into 1000 different classes. At the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) from 2010, the model managed to achieve first place, with top-1 and top-5 error rates of 37.5% and 17.0% [18].

Since then, multiple new models with different architectures were published, which continuously push the performance of CNN's even further. Two examples for that are the "Inception Network" or also called "GoogLeNet", which introduced the Inception Module [19], and the "ResNet", which established way deeper networks with 110 or even 1202 layers [20]. An issue that arises with deep neural networks is that the amount of data required for training a CNN rises with the depth of the neural network. This issue creates a problem for neural networks in medical applications due to the lack of publicly available data.

2.3 The problem of data shortage with medical images

As earlier stated, there has been a general upward trend in the usage of medical imaging over the last two decades. Therefore, every year, more and more medical scans should be available, which could function as training data for CNNs. Although these data exist, most of them are not accessible due to patient privacy and security policies [21]. The Health Insurance Policy (HIPAA) [22] of the USA and the Regulation 2016/679 [23] of the EU limit the storage and accessibility of medical data for the public. In other fields of computer vision, there are large publicly available datasets, like the Kinetics 400 dataset. It consists of 306 245 video clips from 400 different action classes [24] that were used by Google DeepMind to pretrain the "Two-Stream Inflated 3D ConvNet" [25]. Datasets for medical images are generally smaller; E.g., the Open Access Series of Imaging Studies (OASIS) dataset consists of 2 168 medical scans [26] and the Alzheimer Disease Neuroimaging Initiative (ADNI) dataset consists of about 1 500 MRI head scans altogether [27]. Another challenge for utilizing a large number of medical imaging scans is the generalizability of data. There are apparent differences in contrast between different MRI sequences, even for the same pathologies, and even differences between the same sequences from different MRI scanners. As a result, it is challenging to combine different MRI scans for building a large benchmark dataset [28].

So especially compared to datasets used for the training of CNNs in other areas, for medical image classification, there are not enough data publicly available to train a CNN from the beginning. Too little data could lead to overfitting, which results in a worse performance of the network. Using the weights of pretrained CNNs with transfer learning can compensate for the lack of available data [29].

2.4 The problem of computational power for 3D CNNs

Another limiting factor for training deep neural networks is the requirement of computational power. Most smaller research institutions do not have the financial resources to build a powerful computer with many graphics processing units (GPU). When compared to 2D images, CT or MR scans are acquired with multiple slices through the patient, which results in a 3D Volume [30]. To point out the size difference between 2D and 3D data, e.g., the images used by Szegedy et al. [19] have the size 224 x 224, which results in 50 176 pixels. An MRI scan from the ADNI dataset has the dimensions 192 x 192 x 160, which amounts to 5 898 240 voxels. As a result, more computational power is needed to process this data. As an example of the required computational power, the 3D "ResNext" model was trained on the Kinetics dataset with 8 Tesla P100 NVIDIA GPUs [31], each costing about 5000 – 6000 €. The earlier mentioned "Two-Stream Inflated 3D ConvNet", was trained with 64 GPUs [25]. As a comparison, for the models used in this work, there is only a single NVIDIA GTX 1080ti GPU available.

Transfer learning utilizes pretrained CNNs to reduce the amount of training data needed [32], which also comes with less computational expense for training the CNN. There are many pretrained 2D CNNs that are publicly available, e.g., the Inception-V3, Inception-V4, ResNet, or the VGG16 network in Keras, which were trained on the ImageNet dataset [33]. For 3D data, there are less pretrained CNNs available due to the earlier mentioned lack of data and computational power. There are two different strategies that make use of pretrained neural networks for the classification of MRI head scans, which are explained in sections 4.2.1 and 4.2.2. The purpose of this work is to compare these two strategies with respect to the quality and applicability of the resulting CNNs. This comparison is meant to support the foundation of developing commercially available CAD software that provides assistance for radiologists in the analyzation and classification of 3D MRI head scans.

3 A theoretical basis for understanding CNNs

CNNs are state of the art for computer vision tasks. They are a specialized architecture of the artificial neural network (ANN). Therefore, in order to understand CNNs, first, the theory of ANNs has to be introduced.

3.1 The artificial neural network

The ANN is a machine learning algorithm that tries to imitate the architecture of a human brain to be able to learn and store knowledge. It processes an input X to an output Y [34]. The core structure of an ANN, the artificial neuron, was first modeled in 1958 by Frank Rosenblatt [35]. Its implementation is based on the architecture of a real neuron. Visualized in Figure 2, the neuron consists of multiple inputs $\vec{x} = [x_1, x_2, \dots, x_n]$, which all have a corresponding weight $\vec{\omega} = [\omega_1, \omega_2, \dots, \omega_n]$.

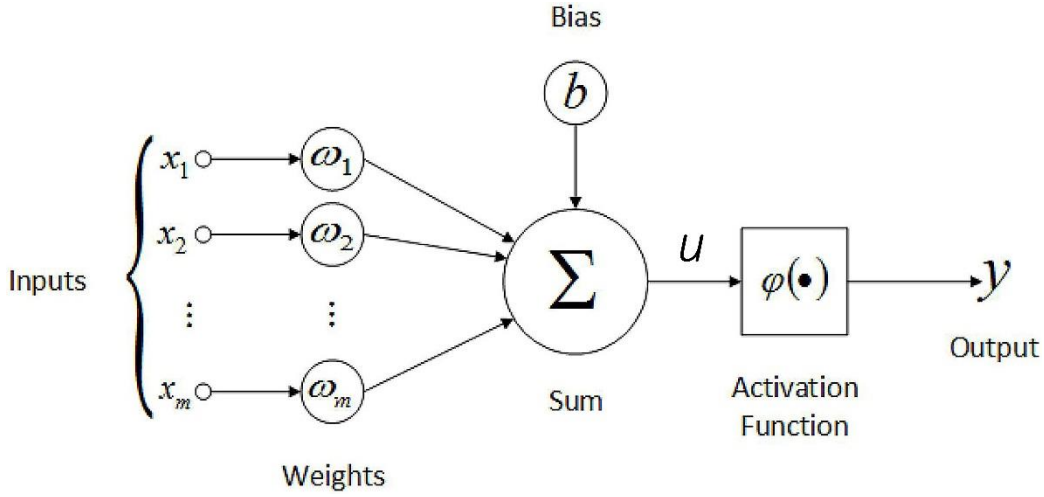


Figure 2. The architecture of an artificial neuron [36].

To compute the output y of an artificial neuron, all inputs are multiplied with their respective weights, and afterward, the sum of them is formed. After that, the bias term b is added to the sum as a threshold for the output, as described in equation (1). The result is applied to the activation of the neuron φ(), which specifies its output range, as described in equation (2) [34].

$$u = \sum_{i=0}^{i} x_i \cdot \omega_i + b \tag{1}$$

$$y = \phi(u) \tag{2}$$

There are many different possible activation functions, like e.g., the Sigmoid or the Rectified Linear Unit (ReLU), which are shown in Figure 3. The softmax activation function is an abbreviation of the traditional sigmoid activation function, with the difference that the sum of all its' outputs is always equal to one [37].

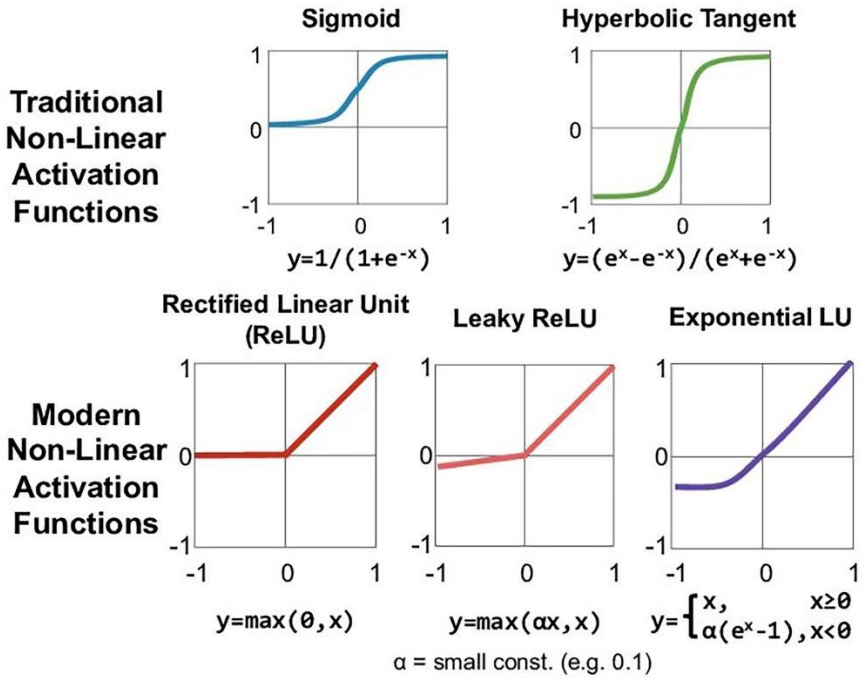


Figure 3. Five different activation functions, which all define different ranges of output values for a neuron. [38]

Multiple concatenated artificial neurons form a layer of an ANN. A model consisting of only an input and an output layer is called single-layer feedforward architecture. A model consisting of an input, an output layer, and at least one hidden layer in between is called multi-layer feedforward architecture [34]. The neurons in hidden layers are responsible for retrieving complex information and patterns out of the input data. An essential characteristic of a multi-layer feedforward architecture is that the output of every neuron from a previous layer is connected with every neuron of the following layer [34]. Figure 4 shows the structure of an ANN with one input layer, two hidden layers, and one output layer. For a classification task, the number of neurons in the output layers usually corresponds to the number of possible classes, so that every neuron outputs the likelihood of its specific class.

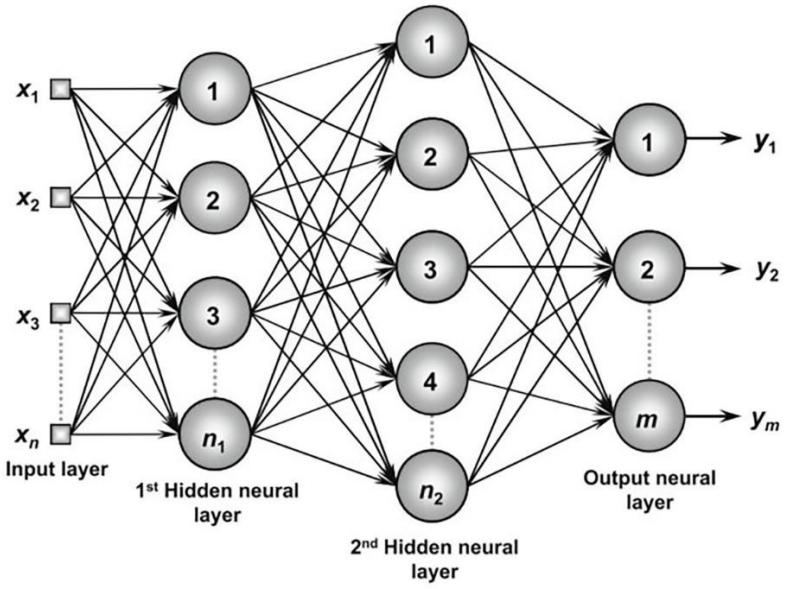


Figure 4. The architecture of a feedforward ANN with all connections between the layers. Each node represents a neuron as shown in Figure 2. [34]

3.2 Training an artificial neural network

For the training of an ANN, labeled data is necessary. One way to label data is the so-called “One Hot Encoding”. With this method, the label for each class is a vector with the same length as the number of classes available in the dataset. So in the example of classifying a dataset of cats, dogs, and hamsters, the vector would have three digits. Each digit represents a class, e.g., the first digit represents the cat, the second represents the dog, and the third represents the hamster.

$$\begin{matrix} 1 & 0 & 0 \\ cat: 0\ ; & dog: 1\ ; & hamster: 0 \\ 0 & 0 & 1 \end{matrix}$$

The ANN to this task would have three output neurons, that predict the likelihood of its corresponding class. Processing the output of an NN is called forward propagation. The resulting vector of the forward propagation of an image of a cat could look like this:

$$\begin{matrix} 0.8 \\ 0.1 \\ 0.1 \end{matrix}$$

The loss function of an ANN calculates the individual error for each output node. E.g., the Squared Error loss function calculates the error in the following way:

$$\begin{pmatrix} 1 & 0.8 \\ 0 - & 0.1 \\ 0 & 0.1 \end{pmatrix}^2 = \begin{matrix} 0.04 \\ 0.01 \\ 0.01 \end{matrix}$$

The loss values are passed to the optimizer of a NN, which uses backpropagation, to determine the amount in which the weights of the networked have to be adapted [32]. Some state of the art optimizers are, for example, Adam [39], Root Mean Square Propagation (RMSprop) [40], or Stochastic Gradient Descent (SGD) [41]. This way, the weights of the ANN can adapt to a specific task like classifying images of cats, dogs, and hamsters.

3.3 Convolutional neural networks

When it comes to the task of computer vision, ANNs with a feedforward architecture face a few problems: Due to all neurons of one layer being connected to all neurons in the next layer, there are too many parameters in the network, which would require an extensive amount of training data [17]. For example, an ANN that processes an image of the size 30 x 30 pixels would require a total number of 900 nodes in the input layer. When containing e.g., 500 nodes in the hidden layer, there are already $900 \cdot 500 + 500 = 450\,500$ parameters (number of input nodes · number of neurons + number of biases). With 10 output nodes, the network contains an additional $500 \cdot 10 + 10 = 5\,010$, and in total $450500 + 5010 = 455\,510$ parameters. Additionally, modern neural networks are deeper than three layers because it has been found that increasing the size of a neural network generally increases its performance [19]. So for larger images and deeper networks, the feedforward (fully-connected) architecture is very inefficient, and therefore not applicable. Another disadvantage of the fully-connected architecture is that it ignores the spatial structure of an image so that it cannot receive any information about the correlation between local pixels [17].

The solution to these problems is the architecture of a Convolutional Neural Network. The basic idea is to mimic the organization of receptive fields in the visual cortex, which was first described by Hubel and Wiesel in 1962. They state that the visual cortex of a cat is divided into multiple regions, which have specific neurons only connected to that single region [17]. So, a CNN mimics the visual cortex by connecting only units from one layer to units in adjacent layers with the same local position. As a result, the network can detect basic visual shapes, like edges or corners, and pass them through the network without losing the information about the location of those features [17].

3.3.1 The convolutional layer

The way a CNN achieves this local connection is through the mathematical operation of convolution. Figure 5 shows an example of this operation on an image. The black numbers represent the pixel values of the image, and the red numbers represent the values of the 3 x 3 convolutional kernel. The output value is calculated by building the sum of each multiplication of pixel value with overlapping kernel value. This calculation is repeated for every pixel in the image, thus resulting in a new image with the output values at the corresponding position, where the kernel was positioned, calculating them [42].

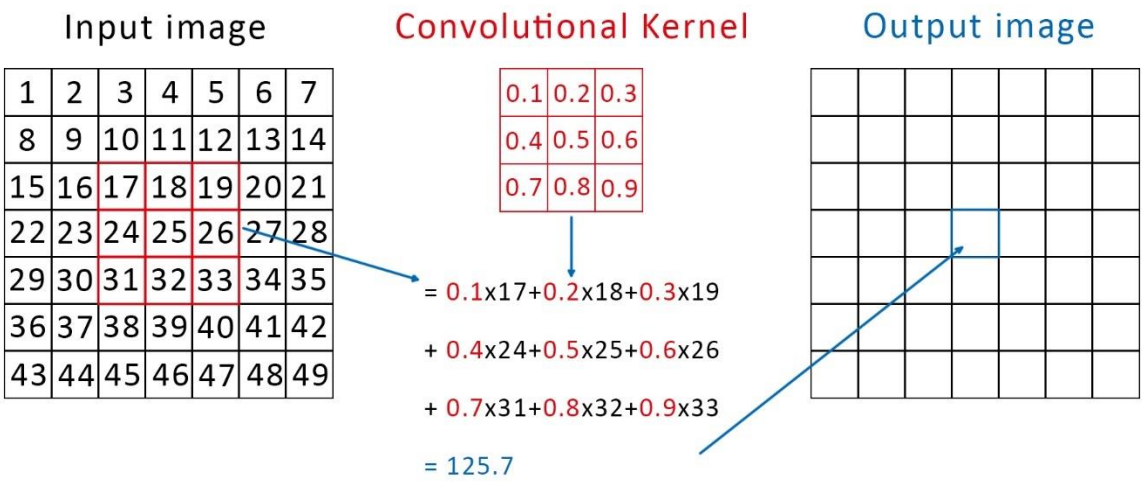


Figure 5. An example of a convolution operation. The kernel slides over the input image to produce an output image. The numbers of the input image represent the greyscale values of the respective pixel. The numbers in the red convolutional kernel represent the weights of the kernel. At each position, the result of the convolution is calculated (125.7 in this example).

The core structure of a CNN is the convolutional layer, which consists of one or multiple convolutional kernels. It takes a batch of images or features as input and calculates another batch of features by applying the convolution operation. It serves a similar function as the hidden layer of a feedforward network, with the convolutional kernel replacing the weights. A 2D convolutional layer generally has four dimensions: *[height, width, n_input_features, n_output_features]*. *Height* and *width* define the dimensions of the convolutional kernel. The term *n_input_features* stands for the number of input features that are forwarded to the convolutional layer. The term *n_output_features* describes the number of output features, the layer produces. This is defined by the number of convolutional kernels the layer consists of [32]. For example, the input to a convolutional layer has 32 features. When the layer should consist of 3x3 convolutional kernels and produce 64 output features, its' dimensions must be: *[3, 3, 32, 64]*. Similar to the neurons of a feedforward neural network, the outputs of the

convolutional layers are passed to an activation function before being forwarded to the next layer [32]. The weights of the convolutional kernels are the parameters of a CNN that can be changed during back propagation. So a CNN learns by adapting the convolutional kernels that can best detect the individual features of the input images [32]. In the earlier layers of a CNN, general features like e.g., edges or circles are detected, and in the deeper layers, more features that are more specific to the input data are detected.

CNNs can also consist of other layers, like a max pooling layer, which looks at a patch from the input feature map, takes the maximum value out of the patch, and discards the other values. Afterward, the image size is significantly reduced, as shown in Figure 6. The size reduction is essential for lowering the computational power required for processing a CNN [43].

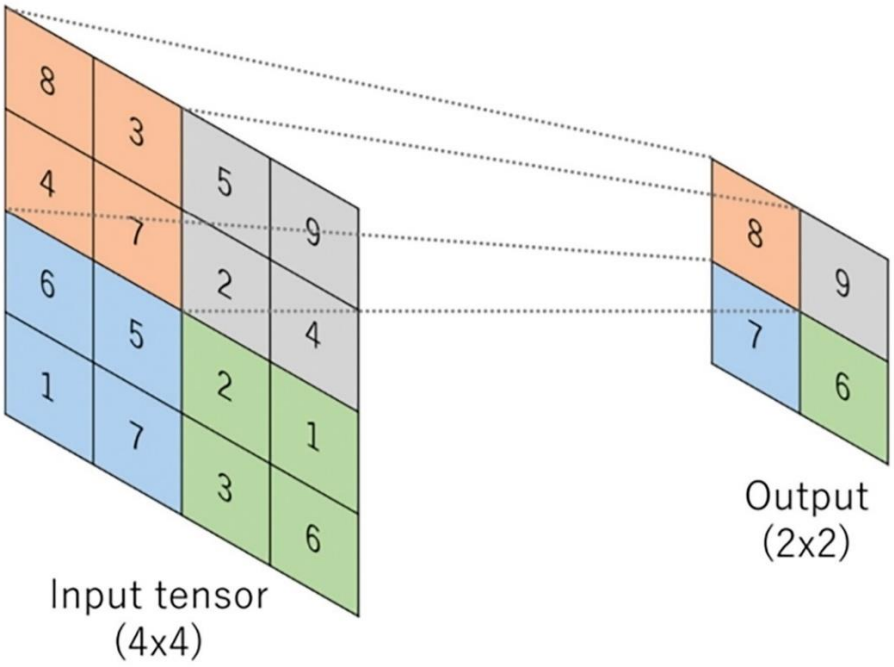


Figure 6. The size reduction of an image by a max pooling layer [32].

In summary, the CNN architecture is an adaption of the ANN architecture, with fewer parameters and additional information about the positions of pixels in the input image. The convolutional layer of the CNN resembles the hidden layer of an ANN with the convolutional kernels replacing the artificial neurons.

A CNN additionally consists of fully connected (FC) layers at the end, which are of the same structure as an ANN and therefore require a 1D vector as input. As a result, the output of the last convolutional layer has to be reshaped to consist of one dimension. In a CNN constructed for a classification task, the last FC layer usually consists of the same number of neurons as there are classes in the classification task. The last layers' output, represents the output or prediction of the whole CNN. As shown in Figure 7, the same principles of forward- and back propagation of a feedforward ANN, apply to the CNN.

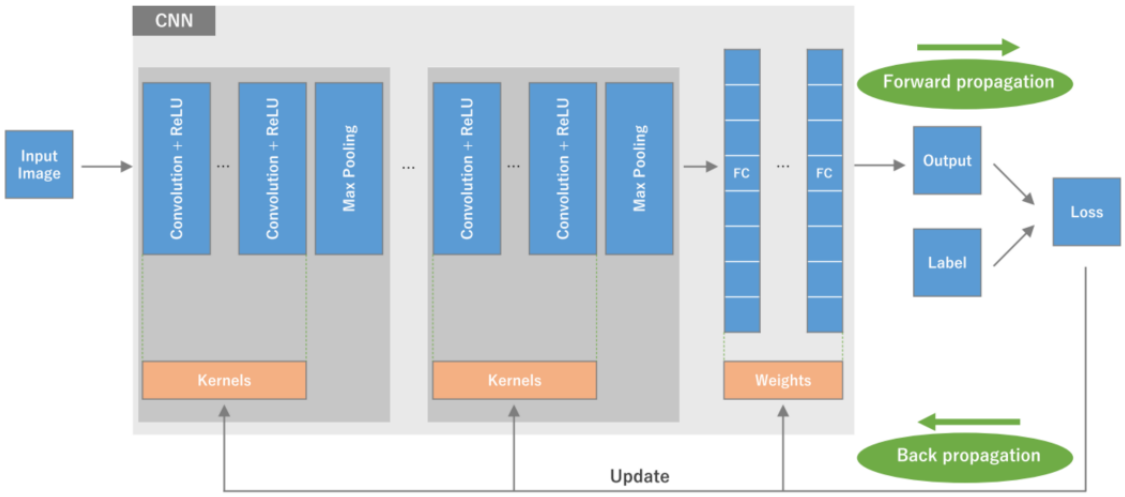


Figure 7. The complete structure of a CNN with the path of forward- and back propagation. As explained earlier, the kernels of the convolutional layer are updated during back propagation. The loss of a CNN is calculated in the same way as for the ANN [32].

3.3.2 Hyperparameters

Hyperparameters are characteristics of a CNN that influence its general architecture or functionality. They are set before training and can have a big influence on the models' performance [32]. A few examples of important hyperparameters are:

- Batch size: Number of data that are parallelly processed by the network (Also influences the calculation of the loss)
- Epochs: Number of iterations, the complete training data is passed through the network
- Loss function: The function, that computes the loss (error) of a batch
- Optimizer: The optimizer algorithm minimizes the error of the loss function
- Learning rate: The rate in which weights are updated in the optimizer
- Activation function: The layer-specific activation function, that determines the value range of the output feature maps (See Figure 3)

Every neural network needs individual optimization for its specific hyperparameters in order to achieve the best possible performance. A kind of brute-force method for doing so is called grid search. With it, a specific value range is specified for every hyperparameter that has to be optimized. E.g. *learning-rate: [0.001, 0.01, 0.1]* and *batch-size: [8, 16, 32]*. After that, the models' performance with every hyperparameter configuration is evaluated with performing a complete training session on it. So, in this case, the total number of training sessions, the grid search algorithm has to perform, would be $3 \cdot 3 = 9$. Afterward, the best performing configuration is evaluated on chosen criteria and the process can be repeated with new value ranges [44]. Normally, the results of every configuration are validated with cross-validation, to reduce the impact of statistical outliers. Cross-validation trains the neural network with multiple iterations, each time choosing a different portion of the whole dataset to be training and validation data (Figure 8). The outcome of cross-validation is calculated by taking the mean of each training result [45].

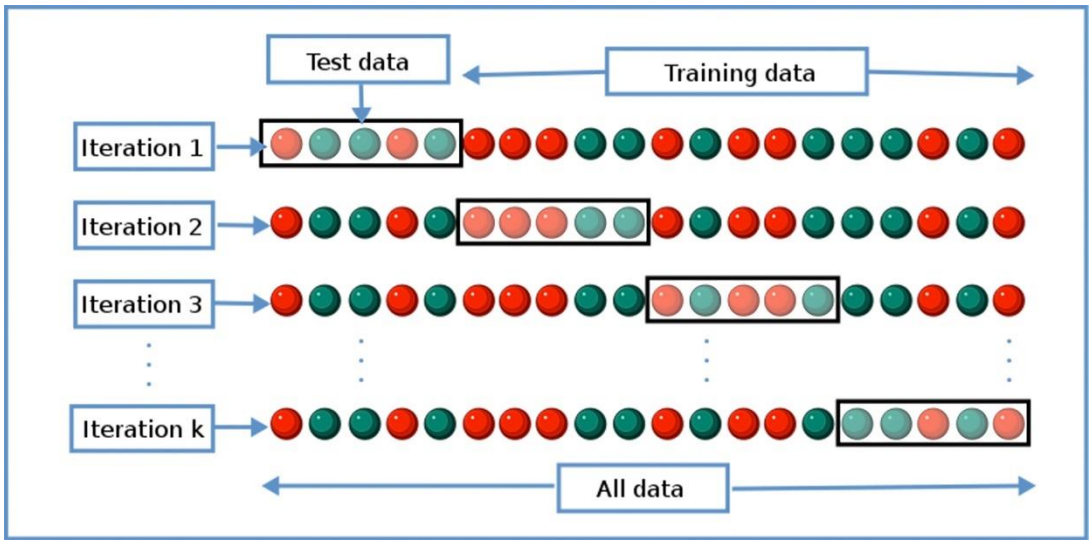


Figure 8. The different distributions of training- and test data over multiple iterations (or folds) for cross-validation.

3.3.3 The 3D CNN

The input to 2D CNN is limited to the dimensions of an image. When it comes to 3D medical scans, it is desirable to process the whole 3D volume. Therefore 3D CNNs have to be used for medical scan classification. Although the general architecture of both network types is the same, 3D CNNs have few differences. The input to a 3D CNN is a five-dimensional tensor (or matrix), e.g. of the shape: *[batch size, height, width, depth, features]*. *Batch size* is a placeholder for the number of parallelly processed data by the CNN. The convolutional layer that processes the input volume also must contain five dimensions. So in general, a convolutional layer of a 3D CNN consists of the five dimensions *[height, width, depth, n_input_features, n_output_features]* (see section 3.3.1) [46]. Figure 9 shows the 3D convolutional operation on a 3D input volume.

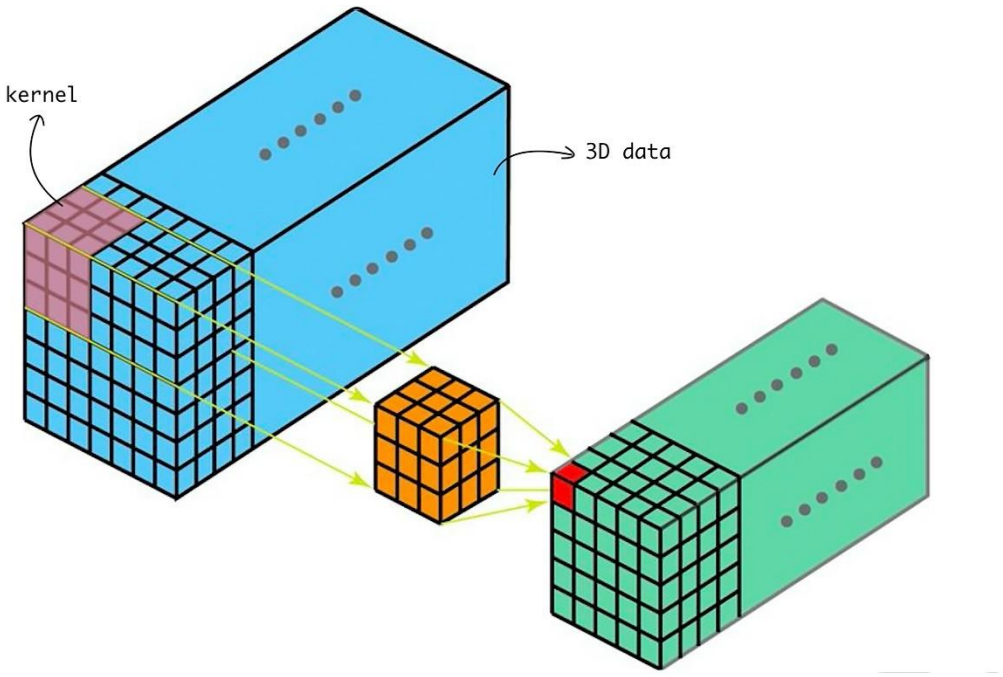


Figure 9. A 3D convolution operation with the input volume (blue), the convolutional kernel (orange) and the output volume (green). The input volume in this example consists of only one channel (feature). The 3D convolutional kernel slides across all three dimensions of the input volume to produce the output volume. [47]

3.4 Transfer learning as a solution for data shortage

The underlying supposition to transfer learning is that every object consists of fundamental features, like edges or circles, which describe, e.g., dogs as well as human brains. Like explained earlier, every CNN extracts those simple features in the first convolutional layers and detects more complex structures in the last layers. Therefore, the first convolutional layers should be similar for every CNN with a computer vision task. Transfer learning makes use of this assumption by taking pretrained CNNs (usually pre-trained on large image datasets like ImageNet) and transforming them to fit a new task [48]. The process discards the fully connected layers at the end of the original network so that only the convolutional layers (convolutional base) are left. After that, a new feature extractor is constructed at the end of the new network that outputs the result for the new classification task. The original network will be referred to as "base model" in this work. There are two different approaches to training the resulting network, which can be seen in Figure 10. The first is to freeze the complete convolutional base and only train the new fully-connected layers. This approach does not apply to medical image data because the features of medical images are too diverse from typical images. The second method also finetunes the convolutional base of the model [32]. An important decision to make with this method is how many layers of the convolutional base should be frozen and how many should be finetuned. Yosinski et al. have shown that finetuning the complete convolutional base results in better accuracy compared to freezing convolutional layers. The accuracy gap increases when more layers of the convolutional base are frozen [29]. The authors claim that the reason is "related to splitting networks in the middle of fragilely co-adapted layers" [29]. Kruithof et al. have shown that the effect of freezing layers is mostly dependent on the number of images per class in the dataset for finetuning. For one to ten images per class, the accuracy of the model stays the same or slightly increases with the number of frozen layers. For 50 or more images per class, the accuracy sharply decreases with the number of frozen layers [49]. This result supports the conclusion of Yosinski et al.

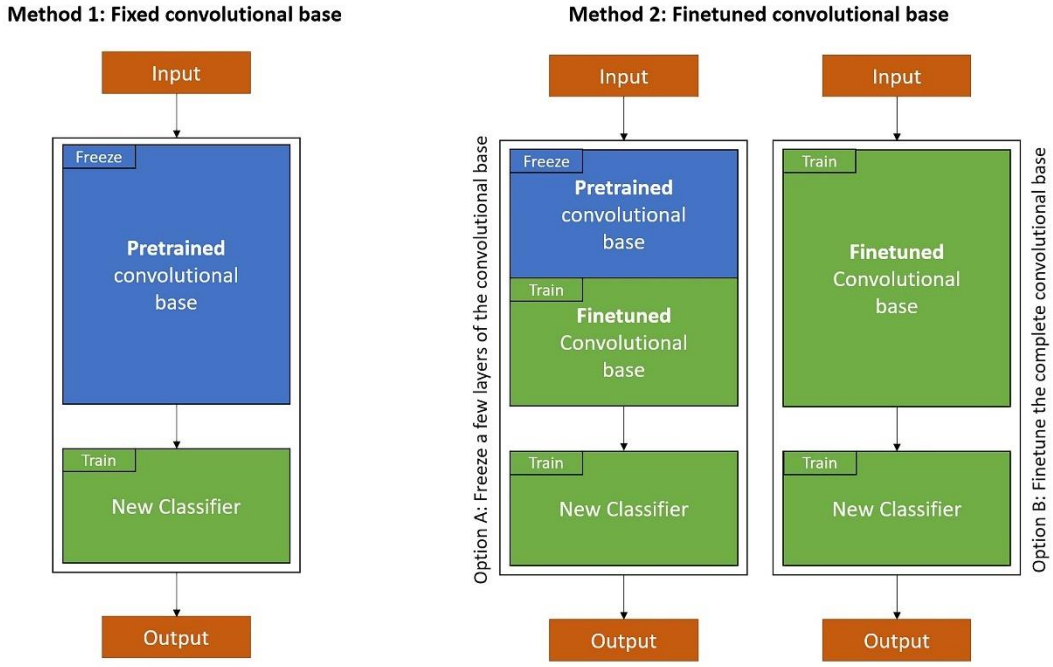


Figure 10. The different methods of transfer learning. Not trainable layers are depicted in blue, and trainable layers are depicted in green. The approaches differ with respect to the number of frozen layers in the new constructed CNN.

3.5 The Inception architecture

The inception architecture was developed by Google DeepMind and published in 2014. The underline argument to the authors was that “the most straightforward way of improving the performance of deep neural networks is by increasing their size” [19]. However, the adverse effects of doing so are the need for a larger dataset and more computational resources. So in order to minimize those negative effects while still taking the benefit of a deeper network, the authors developed the inception module, which can be seen in Figure 11. It consists of four different branches: A convolutional layer with a kernel size of 1x1, a 3x3 convolution, a 5x5 convolution, and a 3x3 max pooling layer. The convolutional layers with sizes larger than 1x1 are serially connected with a 1x1 convolutional layer, to reduce the size of the input features, in order to lessen the computational cost. All four branches are concatenated at the end, to form a collective output for the inception module. This output is then passed on to the next inception module. The whole network was first called “GoogLeNet”, but is now commonly referred to as “Inception-V1”. It consists of 9 inception modules stacked on top of each other, followed by fully connected layers, that result in the classification output [19]. A newer version, called “Inception-V3”, updated the architecture of the old inception model, to further decrease computational cost, by replacing the 5x5 convolutional layer with two 3x3 layers. The reason for that decision was that a 5x5 convolution is 2.78 times more computationally expensive than a 3x3 convolution. Another improvement was to replace some nxn convolutions with an nx1 followed by a 1xn convolution, which further reduced the computational cost. With some additional changes, like using a different optimizer, the model managed to outperform its predecessor in the same classification task [50].

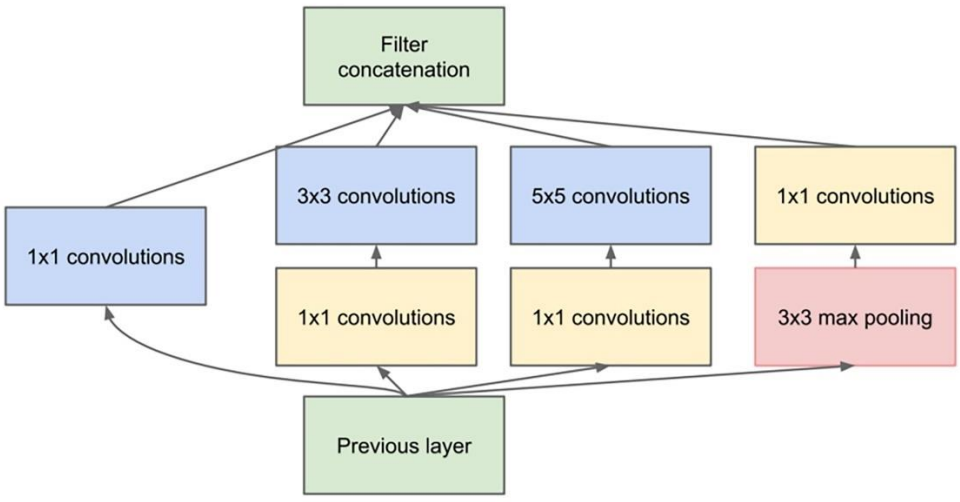


Figure 11. The architecture of the inception module. The blue squares mark the layers responsible for feature extraction, and the yellow layers are responsible for dimension reduction.

4 Method

4.1 Criteria and objectives of the comparison

The focus of this work is to create a comparison of two different architectures of CNNs that utilize transfer learning. The comparison evaluates, which of the two architectures is better suited for the classification of 3D MRI head scans. Therefore, other criteria, in addition to the accuracy of the CNN, are compared. The criteria serve to illustrate two important characteristics of CNNs for medical imaging tasks – The quality and the applicability (Figure 12).

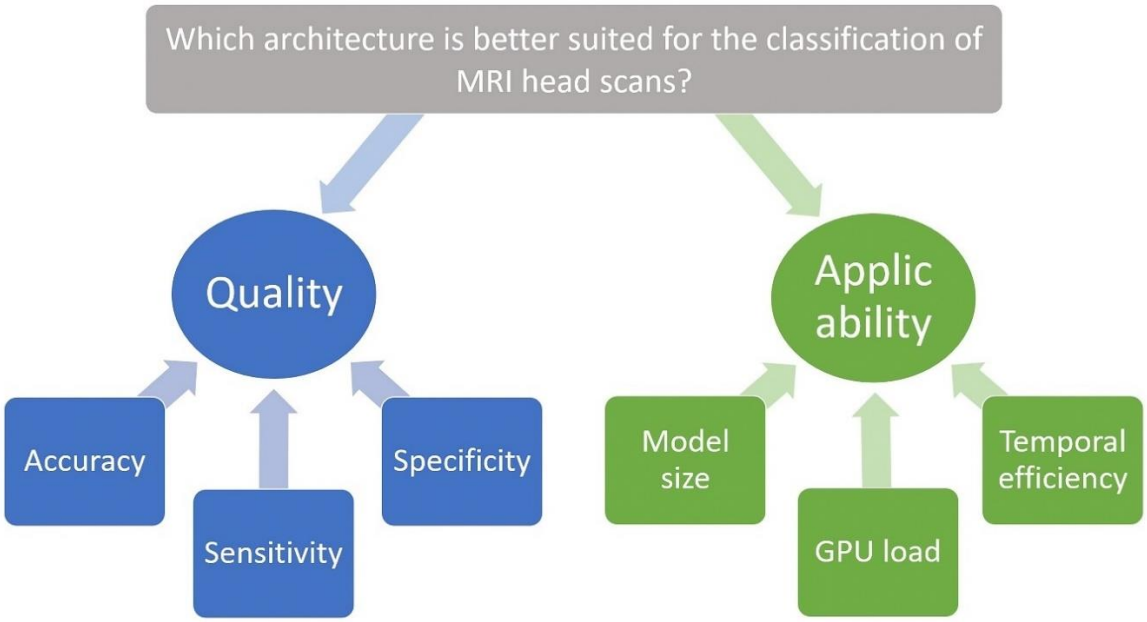


Figure 12. The criteria for the comparison of this work.

The relevance of all criteria are described in the following:

Accuracy, Specificity and Sensitivity (Quality): These criteria describe the statistical metrics of the architecture. A model with better statistical metrics is more valuable as a second opinion for a radiologist due to its lower error rates.

Model size (Applicability): This criterion looks at the memory usage of the model in bytes. The model size is important for the applicability of the architecture due to smaller models being integrated more easily in software that is meant for distribution. Additionally, models with larger size allocate more memory of the GPU during training.

Temporal efficiency (Applicability): The criterion describes the duration for training the model and the duration for the classification of one MRI head scan. Long training durations are disadvantageous because they limit the possible options for hyperparameter optimization . Additionally, the duration for one classification has to be low enough that it does not delay the workflow of a radiologist.

GPU load (Applicability): The criterion evaluates the size of the input data the model can process before an Out of Memory (OOM) error occurs in the GPU. A model with a lower GPU load can be

executed on more systems with less computational power, which facilitates the distribution. Additionally, the size of the input data can be larger for models with less GPU load.

In addition to the criteria mentioned above, the adaptability of the convolutional kernels of each model, with transfer learning is evaluated. This is not included in the comparison, because no conclusion can be made, whether much adaption has a positive effect on the performance of a model. Instead, the adaptability serves as a visual confirmation, whether transfer learning leads to significant adaptions in the kernels of the models.

4.2 The architectures of two different approaches to transfer learning

4.2.1 The inflation of a 2D CNN to a 3D CNN

The underlying principle for the inflation method is to use the filters of a 2D CNN and to scale them into the third dimension. In this way, the resulting 3D CNN can adapt the architecture and the pretrained convolutional base of the 2D CNN. A two-dimensional convolutional kernel of the dimension N x N is simply stacked on top of itself to have the dimensions N x N x N. The already kind of pretrained filters reduce the expense for training the 3D network, which leads to an increase in accuracy compared to the same model with randomly initialized 3D convolutional filtekernelsrs [25].

The inflated 3D model used in this work was developed by João Carreira and Andrew Zisserman from Google DeepMind and is called “Inflated 3D ConvNet” (I3D). It uses the architecture, the filters and the pooling layers of the Inception-V1 CNN (section 3.5), but inflates them into the third dimension. Additionally, the I3D model is additionally trained on the Kinetics-400 dataset to convert its inflated 2D filters into 3D filters with structures [25].

The complete architecture of the implemented I3D model for this work can be seen in Figure 13.

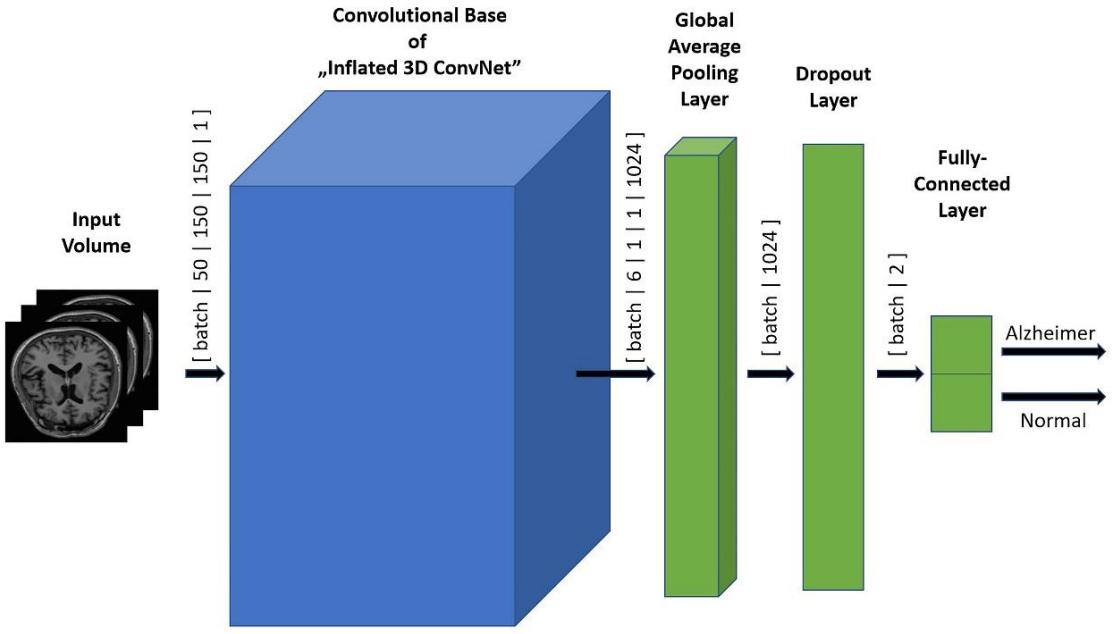


Figure 13. The 3D Inception model for this work. The dimensions of the input tensors leading into an element are displayed on top of the arrows (batch is the placeholder for the hyperparameter batch size). The layers that are changed from the original architecture are depicted in green and the convolutional base is depicted in blue.

The Keras implementation of the Inflated 3D ConvNet is provided on GitHub by the user “dplbc” [51]. The method of transfer learning is used in order to adapt the I3D model to the classification of MRI head scans. Therefore, the fully-connected layers of the model are replaced. The output of the convolutional base is passed into a global average pooling layer, which produces a one-dimensional vector. A dropout layer is attached to it, and after that, the two artificial neurons of the final fully-connected layer, produce the classification output of the neural network. All layers of the convolutional base are set to be trainable with transfer learning.

4.2.2 The multi-view CNN

An entirely different approach to processing 3D volumes with a CNN is to use multiple parallel 2D CNNs. An architecture based on this idea was developed by Su et al. [52] in order to challenge the conventional 3D CNN architecture. The authors' method was to create multiple views of a 3D volume by rendering their shape from different angles. The resulting images each serve as input to a 2D CNN, as shown in Figure 14. All parallel 2D CNNs compute an output, which is then summed up, to result in the output of the complete Multi-View CNN [52]. The 2D CNN Su et al. use in their work is called VGG-M [53].

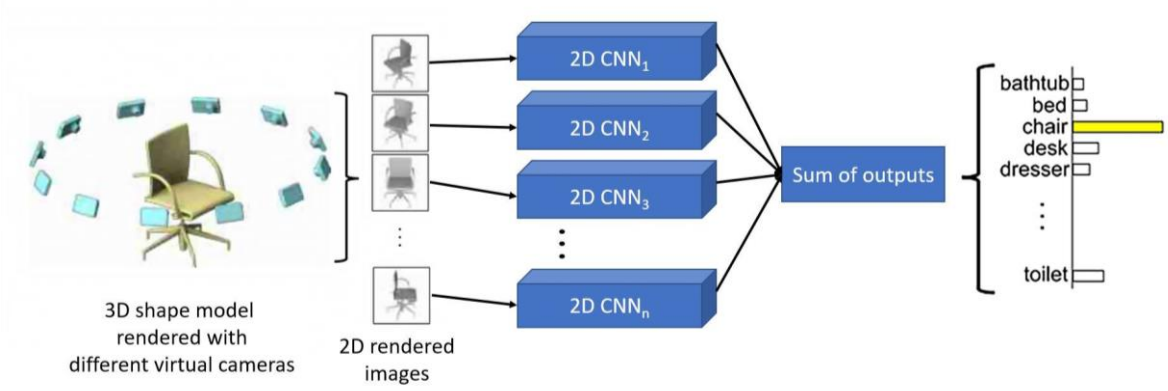


Figure 14. The architecture of the Multi-View CNN from Su et al. All 2D CNNs receive a rendered image of the 3D model as input. The outputs of the 2D CNNs are added up to produce the output of the complete MV CNN on the right. This figure is based on a figure from [52].

In order to use the Multi-View CNN for MRI image classification, the workflow of the model is different from Su et al. architecture. Instead of rendering multiple views of a 3D Volume, the individual slices of the MRI scan data are used as input to the 2D CNNs. To reduce the complexity of the approach, only eight slices are selected from each MRI head scan. In this way, the convolutional kernel of each 2D model can adapt to the individual features of its’ corresponding slice. The distance between the slices is variable due to incoherent MRI sequence parameters (section 4.6). As shown in Figure 15, all eight Inception-V3 models produce an output vector of length two, which all are concatenated to a vector of length 16. This concatenation layer is fully-connected to the last layer, which produces the classification output of the Multi-View CNN.

The base 2D CNN, used for the MV CNN in this work, is the “Inception-V3” model, pretrained on Imagenet, which is explained in section 3.5. The Inception-V3 model is a subsequent version of the Inception-V3 model, on which the I3D architecture is based on (section 4.2.1). The implementation of the model is provided by the Keras library. For the method of transfer learning, the fully connected layers at the end of the model are replaced, and a new classifier is constructed with the architecture

shown in Figure 16. The output of the Inception-V3 model's convolutional base is passed to a 1x1 convolutional layer, to decrease the number of its features. After that, the output is forwarded to a global average pooling layer, which transforms the four-dimensional tensor to a vector. This vector serves as input to a dropout layer whose output is passed to the final fully-connected classification layer with two output nodes. All layers of the convolutional base are set to be trainable with transfer learning. All eight 2D CNNs, in Figure 15, are constructed with this architecture.

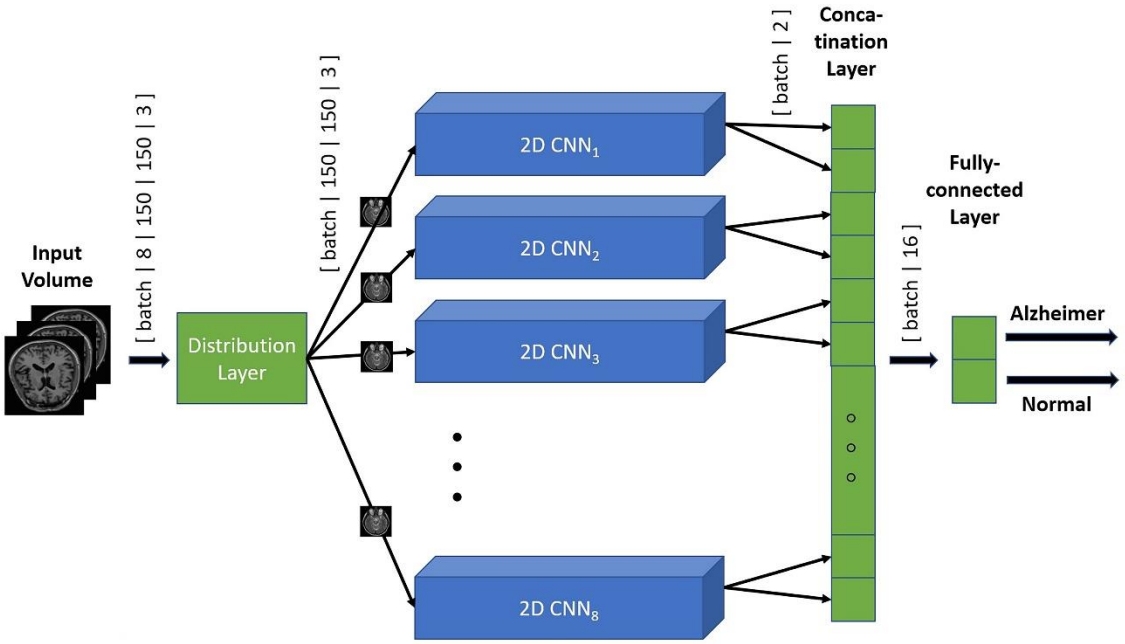


Figure 15 The architecture of the Multi-View CNN for this work. It utilizes eight parallel 2D CNNs (blue). Their architecture is described in Figure 16. The dimension of the input tensors leading into an element are displayed on top of the arrows (batch is the placeholder for the hyperparameter batch size). The layers for the distribution of the input data and the concatenation of the output data are depicted in green. The 2D CNNs are depicted in blue.

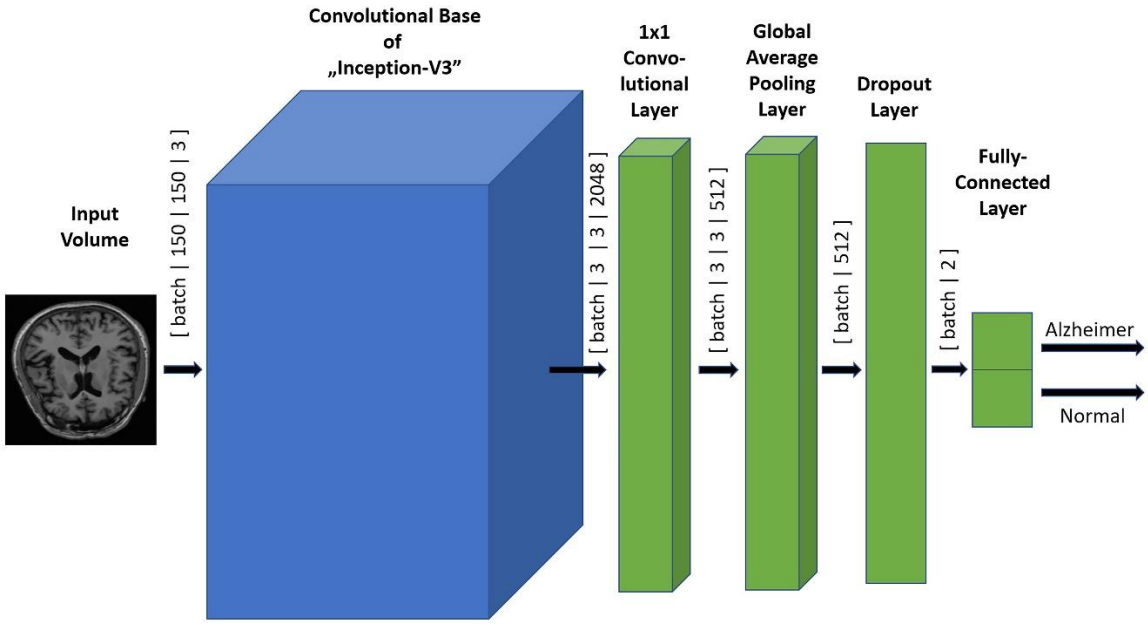


Figure 16. The 2D Inception-V3 model used in the MV CNN of this work. The dimensions of the input tensors leading into an element are displayed on top of the arrows (batch is the placeholder for the hyperparameter batch size). The layers that are changed from the original architecture are depicted in green and the convolutional base is depicted in blue. All eight 2D CNNs, in Figure 15, are constructed with this architecture.

4.3 Hyperparameter optimization

In order to provide a fair comparison, hyperparameter optimization is performed for both models. To roughly narrow down the range of effective hyperparameters, grid-search, in combination with manual presets, is applied. Due to the limited availability of computational power, resulting in longer training periods, only a single training instead of cross-validation is performed for each hyperparameter configuration.

The following hyperparameters are pre-defined for both architectures, and therefore are not optimized with the grid search method:

- Number of epochs: 100
- Batch size: 8
- Activation function for the fully-connected layer: ReLU
- Activation function for the last layer: Softmax
- Loss function: Categorical Cross-Entropy

The hyperparameters learning rate, optimizer and dropout rate are optimized with the grid search method. Their value ranges are the same for both models:

- Learning rate: [10^{-6}, $5 \cdot 10^{-6}$, 10^{-5}]
- Optimizer: [Adam, RMSprop]
- Dropout rate: [0.3, 0.5, 0.7]

The grid search for the optimization of the MV CNN is performed with only one 2D Inception-V3 model, to reduce the duration of the method.

In summary, eighteen different iterations of training are performed for both architectures. The configuration with the best accuracy is utilized for training the respective model for the comparison.

4.4 Tests for the comparison and evaluation of criteria

The comparison is constructed in a way that six criteria of the models are compared with each other. Therefore, the criteria have to be evaluated with different tests. The structure of the tests is described in this section.

Test 1 – The statistical metrics (Quality):

The statistical metrics accuracy, specificity, and sensitivity are calculated by cross-validation with ten folds of the dataset, which results in a split of 90% training data and 10% test data. Training of the model is only performed with five of the ten folds due to long training times. The equations (3), (4) and (5) are used to calculate the statistical metrics of a CNN. True positive (TP) is the number of AD data classified as AD, false positive (FP) is the number of normal data classified as AD, true negative (TN) is the number of normal data classified as normal, and false negative (FN) is the number of AD data classified as normal [54].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

Test 2 – The model size (Applicability):

The model size is determined by the size of the respective models' save file in bytes.

Test 3 – The temporal efficiency (Applicability):

The duration for training the model is calculated by taking the average of the five training sessions that are performed for the cross-validation in test 1. The duration for the classification of one MRI head scan is calculated by measuring the time the respective models take for producing a classification output for one MRI head scan. Five different durations for five different scans are measured for each model, and the average value of them is calculated. The durations are measured with pre-loaded models so that the loading times do not affect the outcome.

Test 4 – The GPU load (Applicability):

The GPU load of a model is defined by the maximum possible input size, the model can process with the Nvidia Geforce GTX 1080ti GPU. Multiple sessions of training are performed with varying input sizes. The respective input volume is calculated by the following equation:

$$Input\ volume = batch\ size \cdot slices \cdot width \cdot height \tag{6}$$

It is recorded whether an OOM error occurred during training. As a result, the largest input volume, that does not cause an OOM error is an estimation of the maximum input volume for a model on the Nvidia Geforce GTX 1080ti GPU. The model with a larger maximum input volume, therefore, causes less GPU load.

Test 5 – The adaptability with transfer learning:

In order to evaluate a models' adaptability with transfer learning, the difference of the convolutional kernels before and after finetuning is compared. The difference is calculated with the equations (7) and (8), using following parameters:

- di: The difference between the kernels of two CNNs.
- n: The number of convolutional layers in the CNN.
- k_i and $k_{i,n}$: The tensors of the nth convolutional kernel of the finetuned model and the base model, respectively.
- a_i, b_i, c_i, d_i, e_i: The dimensions of the tensor of the nth convolutional layer.

$$di_{2D} = \frac{1}{n}\sum_{i=0}^{n} \frac{1}{a_i \cdot b_i \cdot c_i \cdot d_i} \sum_{a=0}^{a_i}\sum_{b=0}^{b_i}\sum_{c=0}^{c_i}\sum_{d=0}^{d_i} \left| k_i(a,b,c,d) - k_{0,i}(a,b,c,d) \right| \tag{7}$$

$$di_{3D} = \frac{1}{n}\sum_{i=0}^{n}\frac{1}{a_i \cdot b_i \cdot c_i \cdot d_i \cdot e_i}\sum_{a=0}^{a_i}\sum_{b=0}^{b_i}\sum_{c=0}^{c_i}\sum_{d=0}^{d_i}\sum_{e=0}^{e_i}\left|k_i(a,b,c,d,e) - k_{0,i}(a,b,c,d,e)\right| \quad (8)$$

1x1, 1xn, and nx1 convolutional layers are not included in the calculation.

The convolutional kernels and the activation maps of the base models and the finetuned models are visualized, in order to provide a better visual comparison and understanding of the effects of transfer learning on the two CNNs. The method used for the visualization is explained in the following:

The weights of the convolutional kernels are fetched from the model and visualized on a graph. The activation maps are the output of a convolutional layer. In order to receive the values of them, a new CNN is built. Its input is the same as the original model, but its output is the output of the chosen convolutional layer. With that, the new model outputs the activation maps of the convolutional layer, which then can be visualized on a graph. The method for the visualization can be seen in Figure 17. Its implementation is inspired by a Keras blog post [55].

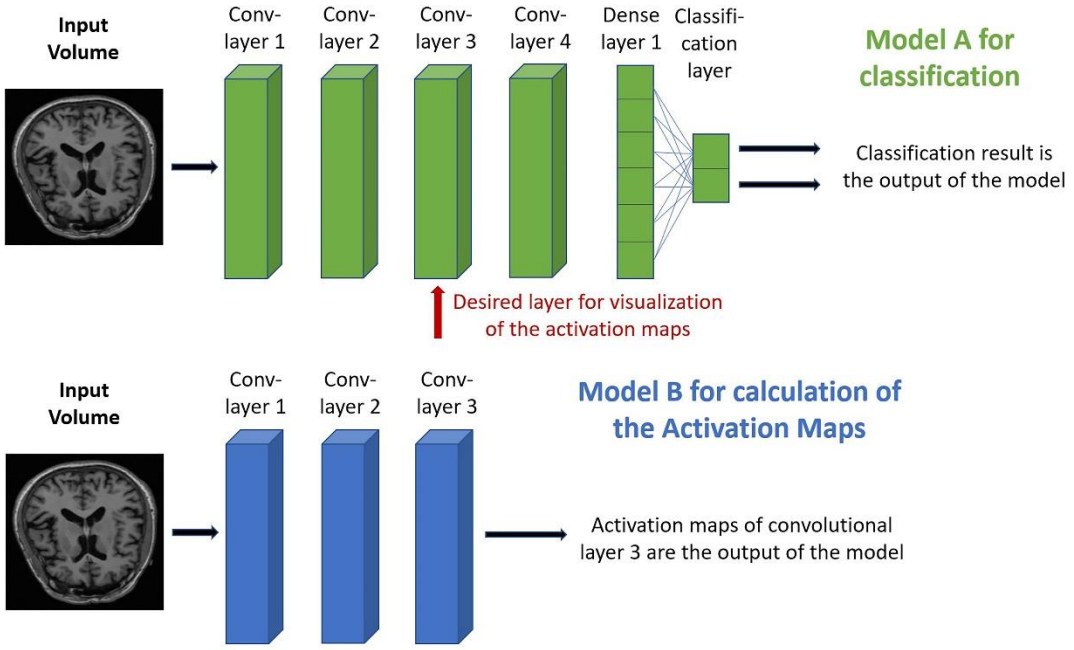


Figure 17. The method for visualizing the activation maps of a model. Model A is the original model for classification and Model B is the new model that calculates the activation maps.

4.5 The value benefit analysis for evaluating the test results

To determine which model is better suited for the classification of MRI head scans, the value benefit analysis is performed [56]. An example of the workflow of the method is shown in Figure 18. The first step is to define the weights for every criterion of the comparison. With the results of the five tests, the level of achievement is evaluated for each criterion. The level can range from one to three, with one being the worst and three being the best level of achievement. After that, the sum of all weights multiplied with the respective levels is taken. As a result, the model with the highest sum is better suited for this task, and therefore the winner of the comparison.

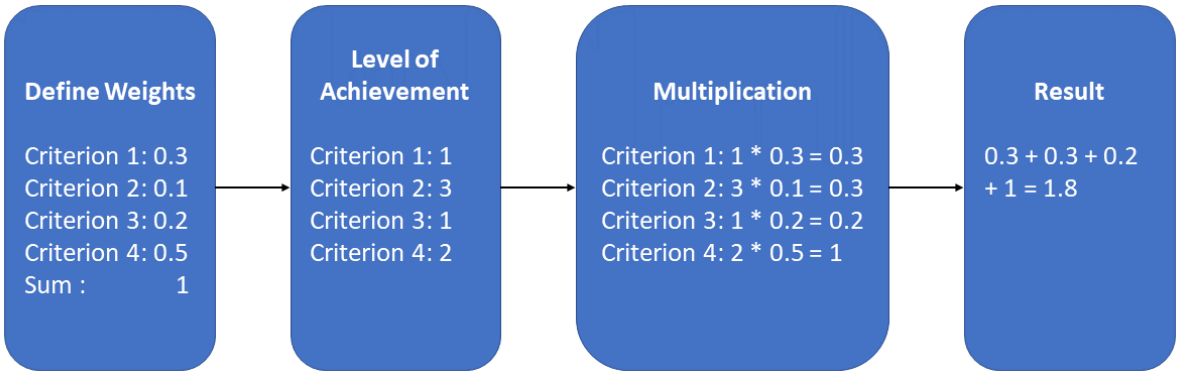


Figure 18. An example of the Value Benefit Analysis.

The level of achievement in a criterion is defined in the following way:

- The level of achievement is 3 for the architecture with a better result and 2 or 1 for the architecture with the worse result
- If higher values are better in the criterion, the level of achievement for the worse (lower) value is 2, if the worse value is above 70% of the better value, and 1, if it is below 70%
- If lower values are better in the criterion, the level of achievement for the worse (higher) value is 2, if the worse value is below 140% of the better value, and 1, if it is above 140%
- Both values are treated as equal if the difference is smaller than the standard deviation of both values. The level of achievement, in this case, is defined as 2 for both architectures

For example, if the accuracy values are 90% for the I3D and 80% for the MV CNN, the level would be 3 for the I3D, and 2 for the MV CNN, because 80% is higher than $0.7 \cdot 90\% = 63\%$.

The weights for the Value Benefit Analysis are defined with the method of Pairwise Comparison, as shown in Table 1 [57].

Table 1. The pairwise comparison for the criteria of the comparison in this work. The weights of the respective criteria are marked green.

	Accuracy	Specificity	Sensitivity	Model size	Temporal efficiency	GPU load	Sum	Weights [%]
Accuracy		2	2	2	2	2	10	33
Specificity	0		1	2	2	1	6	20
Sensitivity	0	1		2	2	1	6	20
Model size	0	0	0		1	0	1	3.3
Temporal efficiency	0	0	0	1		0	1	3.3
GPU load	0	1	1	2	2		6	20
Sum							30	100

Each criterion is compared to the other criteria, and points are distributed between them. Two points indicate that the criterion is more important than the other criteria, one point indicates that they are equally important, and zero points indicate that it is less important. The sum of the points for each criterion is taken. The weight is determined by calculating the percentage of the individual sum [57].

4.6 The medical data

The medical scans used for this comparison are provided by the Alzheimer Disease Neuroimaging Initiative (ADNI). Their database consists of MRI- and positron emission tomography (PET) -head scans from a control group and people with different stages of Alzheimer's Disease (AD) [27]. The data used for training the CNNs contain only MRI head scans from the control group (CN) with no medical findings and scans from patients with AD. They were recorded with the Magnetization Prepared Rapid Gradient Echo (MP-RAGE), which is a 3D T1 weighted MRI sequence [58]. There are dissimilarities of the MRI sequence parameters between different scans, which makes the dataset inhomogeneous. The volumes of the dimensions of the volumes are also incoherent. They range from 146 to 184 pixels in height, from 192 to 256 pixels in width and 192 to 256 slices in depth. ADNI provides processed images with gradient warp, if necessary B1 correction, and scaling of the volume [59]. The complete dataset for the comparison consists of 849 CN scans and 552 AD scans (1401 scans in total). For the I3D architecture, 50 slices of the original volume are taken, as shown in Figure 19, and for the MV CNN architecture, 8 slices of the original volume are taken, as shown in Figure 20. For both architectures, all slices are resized with image interpolation [60], to have the dimensions 150 x 150. As a result, the input volumes to the I3D model and the MV CNN model consist of the dimensions [50, 150, 150] and [8, 150, 150], respectively. The data is labeled with one-hot encoding, so the label for AD is the vector $\begin{bmatrix}1\\0\end{bmatrix}$, and for CN, it is $\begin{bmatrix}0\\1\end{bmatrix}$.

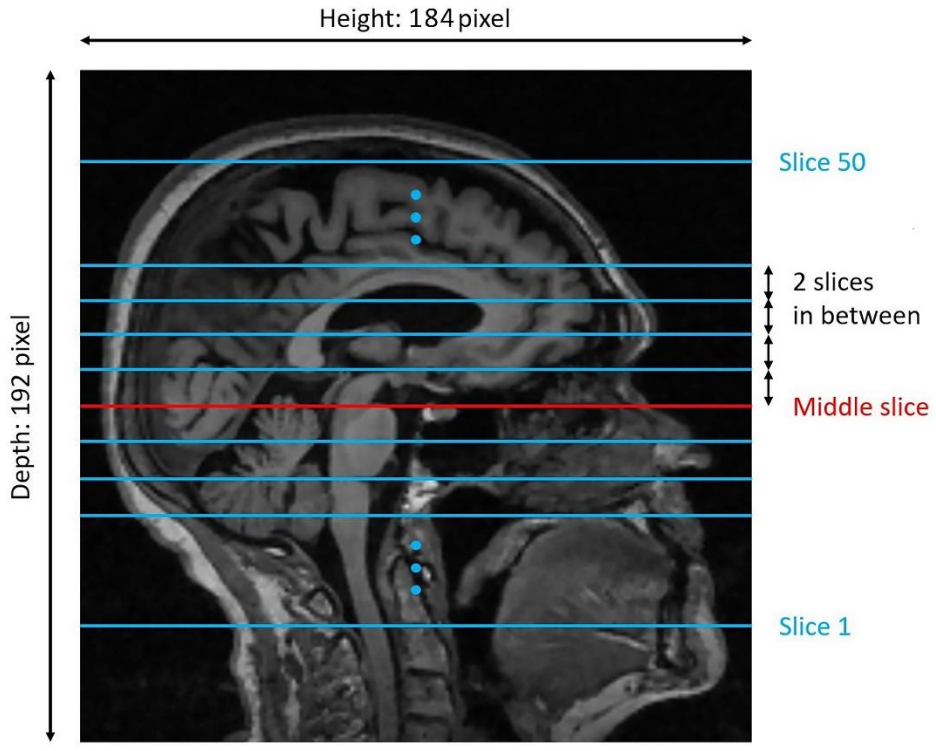


Figure 19. The method for choosing the 50 slices for the input data of the I3D architecture. The image shown is from an MRI scan of the ADNI dataset.

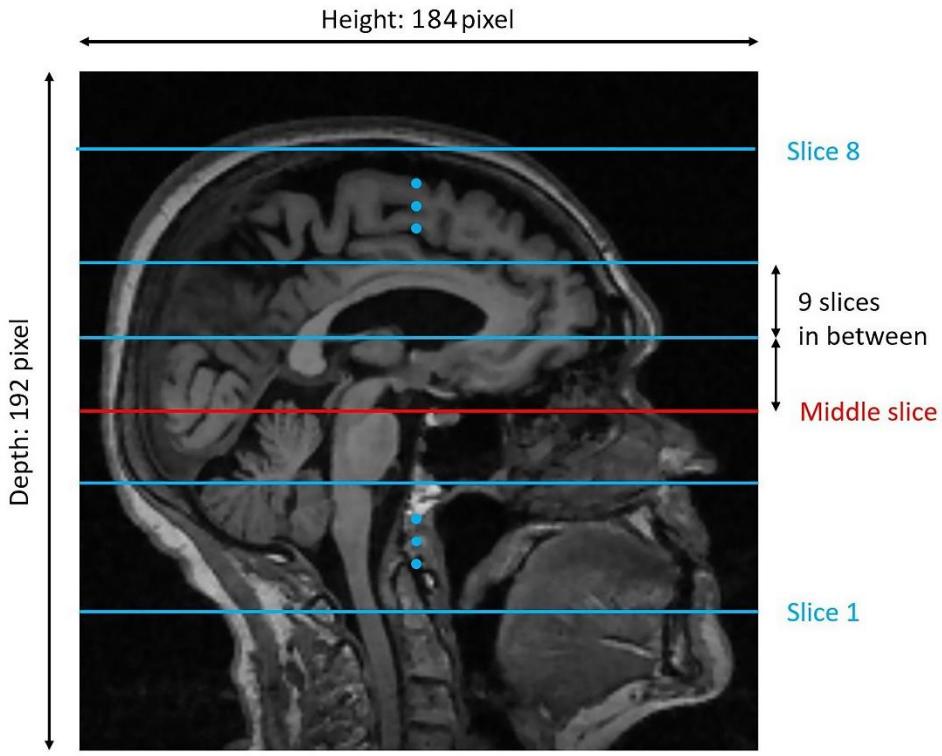


Figure 20. The method for choosing the 8 slices for the input data of the MV CNN architecture. The image shown is from an MRI scan of the ADNI dataset.

4.7 Materials

All models, functions, and scripts are implemented in the programming language Python, version 3.7.4 [61]. Anacondas' virtual environments are used to manage external libraries and their versions [62]. The deep learning library Keras (Version 2.1.5) [63] with Tensorflow (Version 1.14.0) [64] as the backend is used for the implementation, training, and validation of both CNNs. Following additional libraries are used in this work:

- **Nibabel**: Version 2.5.0 [65]
- **Numpy**: Version 1.16.4 [66]
- **Scikit-learn**: Version 0.21.3 [67]
- **Matplotlib**: Version 3.1.3 [68]
- **OpenCV**: Version 4.1.1.2.6 [69]
- **tqdm**: Version 4.36.0 [70]

The CNNs are processed on the Nvidia Geforce GTX 1080ti GPU.

5 Results

5.1 Results of the hyperparameter optimization

The results of the hyperparameter optimization for both architectures of this work are presented in this section. Table 2 shows the results of the grid search method for the I3D architecture. The mean accuracy value is 87.8% and the highest accuracy value is 93.2%. The model achieves it with a learning rate of 10^{-5}, a dropout rate of 0.5, and Adam optimizer. Therefore, this hyperparameter configuration is chosen for the cross-validation of the model. Out of all three hyperparameters, the learning rate has the most significant impact on the accuracy of the model. Figure 21 shows the validation accuracy over all 100 epochs for three models from the grid search. The graph depicts the models with the best, worst, and medium accuracy.

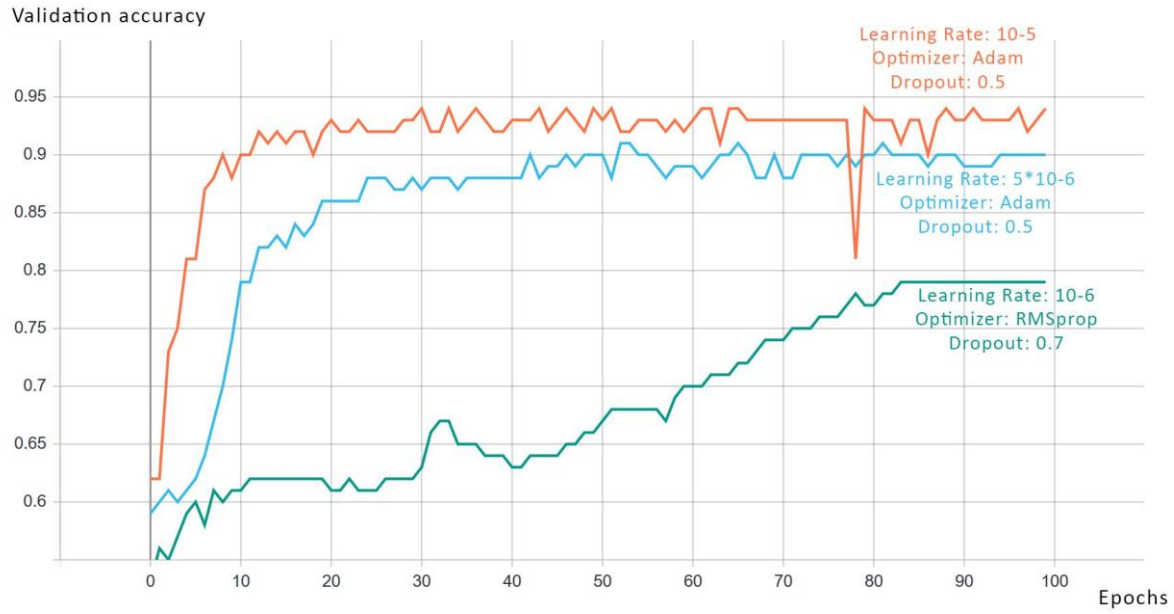


Figure 21. The validation accuracy over 100 epochs of three models from the grid search of the Inflated 3D ConvNet. The orange line depicts the model with the best accuracy, the green line depicts the model wit the worst accuracy and the blue line depicts the model with the medium accuracy.

Table 2. The results of the grid search method for the 3D ConvNet. The values of the hyperparameters are shown on the left, and the corresponding accuracy is shown on the right.

Configuration			
Learning rate	**Optimizer**	**Dropout rate**	**Accuracy**
10^{-6}	Adam	0.3	0.809
10^{-6}	Adam	0.5	0.829
10^{-6}	Adam	0.7	0.794
10^{-6}	RMSprop	0.3	0.849
10^{-6}	RMSprop	0.5	0.831
10^{-6}	RMSprop	0.7	0.79
$5 \cdot 10^{-5}$	Adam	0.3	0.89
$5 \cdot 10^{-5}$	Adam	0.5	0.891
$5 \cdot 10^{-5}$	Adam	0.7	0.896
$5 \cdot 10^{-5}$	RMSprop	0.3	0.922
$5 \cdot 10^{-5}$	RMSprop	0.5	0.918

$5 \cdot 10^{-5}$	RMSprop	0.7	0.889
10^{-5}	Adam	0.3	0.908
10^{-5}	**Adam**	**0.5**	**0.932**
10^{-5}	Adam	0.7	0.914
10^{-5}	RMSprop	0.3	0.899
10^{-5}	RMSprop	0.5	0.921
10^{-5}	RMSprop	0.7	0.923

The results of the grid search for the 2D Inception-V3 can be seen in Table 3. The mean accuracy value is 79.96% and the highest accuracy value is 84.2%. The model achieves it with a learning rate of 10^{-5}, a dropout rate of 0.7 and an RMSprop optimizer. Therefore, this hyperparameter configuration is chosen for all 2D CNNs in the MV CNN, when performing the cross-validation. Out of all three hyperparameters, the learning rate has the most significant impact on the accuracy of the model. Figure 22 shows the validation accuracy over all 100 epochs for three models from the grid search. The graph depicts the models with the best, worst, and medium accuracy.

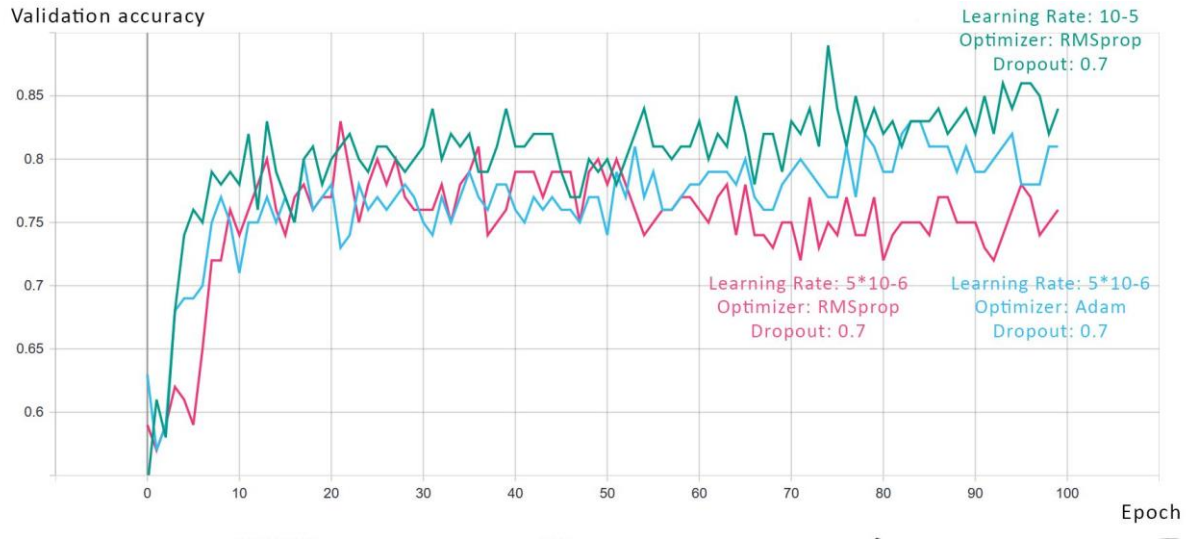


Figure 22. The validation accuracy over 100 epochs of three models from the grid search of the 2D Inception-V3. The green line depicts the model with the best accuracy, the red line depicts the model with the worst accuracy and the blue line depicts the model with the medium accuracy.

Table 3. The results of the grid search method for the 2D Inception-V3. The values of the hyperparameters are shown on the left, and the corresponding accuracy is shown on the right.

Configuration			
Learning rate	**Optimizer**	**Dropout rate**	**Accuracy**
10^{-6}	Adam	0.3	0.777
10^{-6}	Adam	0.5	0.768
10^{-6}	Adam	0.7	0.771
10^{-6}	RMSprop	0.3	0.762
10^{-6}	RMSprop	0.5	0.781
10^{-6}	RMSprop	0.7	0.771
$5 \cdot 10^{-5}$	Adam	0.3	0.836
$5 \cdot 10^{-5}$	Adam	0.5	0.807
$5 \cdot 10^{-5}$	Adam	0.7	0.797

$5 \cdot 10^{-5}$	RMSprop	0.3	0.82
$5 \cdot 10^{-5}$	RMSprop	0.5	0.799
$5 \cdot 10^{-5}$	RMSprop	0.7	0.75
10^{-5}	Adam	0.3	0.795
10^{-5}	Adam	0.5	0.824
10^{-5}	Adam	0.7	0.84
10^{-5}	RMSprop	0.3	0.83
10^{-5}	RMSprop	0.5	0.823
10^{-5}	**RMSprop**	**0.7**	**0.842**

The accuracies of the grid-search algorithm with models, whose layers are frozen up to the seventh inception module are displayed in Table 4. The best performing model in this grid-search achieves an accuracy of 67.4%. The mean value of the accuracies is 50.53%.

Table 4. The results of the grid search method for the partially frozen 2D Inception-V3. The values of the hyperparameters are shown on the left, and the corresponding accuracy is shown on the right.

Configuration			
Learning rate	**Optimizer**	**Dropout rate**	**Accuracy**
10^{-6}	Adam	0.3	0.558
10^{-6}	Adam	0.5	0.559
10^{-6}	Adam	0.7	0.573
10^{-6}	RMSprop	0.3	0.548
10^{-6}	RMSprop	0.5	0.539
10^{-6}	RMSprop	0.7	0.59
$5 \cdot 10^{-5}$	**Adam**	**0.3**	**0.674**
$5 \cdot 10^{-5}$	Adam	0.5	0.582
$5 \cdot 10^{-5}$	Adam	0.7	0.542
$5 \cdot 10^{-5}$	RMSprop	0.3	0.416
$5 \cdot 10^{-5}$	RMSprop	0.5	0.417
$5 \cdot 10^{-5}$	RMSprop	0.7	0.39
10^{-5}	Adam	0.3	0.624
10^{-5}	Adam	0.5	0.484
10^{-5}	Adam	0.7	0.427
10^{-5}	RMSprop	0.3	0.391
10^{-5}	RMSprop	0.5	0.392
10^{-5}	RMSprop	0.7	0.39

5.2 Results for the quality of the architectures

The results of cross-validation for both models can be seen in Table 5. The I3D model achieves a mean accuracy of 92.99%, with its specificity and sensitivity being at similar values. The mean accuracy of the MV CNN is at 80.66% noticeably lower than the accuracy of the I3D. The sensitivity of the MV CNN is at 93.75% noticeably higher than its specificity at 73.28% and at a similar value as the sensitivity of the I3D. The standard deviation of the I3D is lower than the MV CNN for accuracy and specificity but slightly higher for sensitivity.

Table 5. The results of the cross-validation for the Inflated 3D ConvNet and the Multi-View CNN. The mean value of accuracy specificity and sensitivity can be seen in the bottom row.

	Inflated 3D ConvNet			Multi-View CNN		
	Accuracy	**Specificity**	**Sensitivity**	**Accuracy**	**Specificity**	**Sensitivity**
Iteration 1	0.9231	0.9254	0.9200	0.8416	0.7879	0.9429
Iteration 2	0.9231	0.9091	0.9412	0.7900	0.7031	0.9444
Iteration 3	0.9487	0.9545	0.9412	0.8000	0.7018	0.9302
Iteration 4	0.8974	0.9178	0.8636	0.8614	0.8333	0.9024
Iteration 5	0.9573	0.9620	0.9474	0.7400	0.6377	0.9677
Mean	**0.9299**	**0.9338**	**0.9227**	**0.8066**	**0.7328**	**0.9375**
Standard Deviation	0.0212	0.0208	0.0309	0.0424	0.0693	0.0213

5.3 Results for the applicability of the architectures

The results of the tests on the applicability of the models are presented in the following.

Model size:

The total size of the I3D model for this work is 96.025 Mb. The total size of the MV CNN model is 1431.675 Mb.

Temporal efficiency:

The mean duration of training the I3D with 100 epochs is 2 hours and 59 minutes, and the mean duration of training the MV CNN is 10 hours and 33 minutes, as shown in Table 6. The mean time for the classification of one image is 56 milliseconds for the I3D and 542 milliseconds for the MV CNN. The standard deviation of the I3D architecture is for both durations lower compared to the MV CNN.

Table 6. The duration of training the model with 100 epochs and classification of one input volume. The values of the Inflated 3D ConvNet are on the left side, and the values of the Multi-View CNN are on the right side.

	Inflated 3D ConvNet		Multi-View CNN	
	100 epochs [h]	**One volume [s]**	**100 epochs [h]**	**One volume [s]**
Iteration 1	3:00	0.056	10:39	0.567
Iteration 2	3:00	0.055	10:06	0.531
Iteration 3	2:59	0.048	10:57	0.550
Iteration 4	3:00	0.053	10:22	0.516
Iteration 5	2:59	0.048	10:42	0.547
Mean	**2:59**	**0.05**	**10:33**	**0.542**
Standard Deviation	00:00:29	0.003	00:17:34	0.017

GPU load

The results for the approximation of the maximum input volume of the I3D architecture are shown in Table 7. All input volumes greater than or equal to 16 000 000 data points produced an OOM Error during training. The largest input volume that did not produce an error consisted of 13 500 000 data points. Consequently, the maximum possible input value for the I3D must be between 13 500 000 and 16 000 000 data points.

Table 7.The input volume size and related out of memory error events that occurred during training of the I3D. The input volume is calculated with batch size · slices · width · height. The listings are sorted ascending by the input volume.

Inflated 3D ConvNet					
Batch size	**Slices**	**Width**	**Height**	**Input Volume**	**Out of Memory Error**
8	32	32	32	262 144	No
8	50	100	100	4 000 000	No
16	50	100	100	8 000 000	No
8	50	150	150	9 000 000	No
12	50	150	150	13 500 000	No
32	**50**	**100**	**100**	**16 000 000**	**Yes**
16	**50**	**150**	**150**	**18 000 000**	**Yes**
32	**50**	**150**	**150**	**36 000 000**	**Yes**

The results for the approximation of the maximum input volume of the MV CNN architecture are shown in Table 8. All input volumes greater than or equal to 5 040 000 data points produced an OOM Error during training. The largest input volume that did not produce an error consisted of 4 320 000 data points. Consequently, the maximum possible input value for the MV CNN must be between 5 040 000 and 4 320 000 data points.

Table 8. The input volume size and related out of memory error events that occurred during training of the MV CN. The input volume is calculated with batch size · slices · width · height. The listings are sorted ascending by the input volume.

Multi-View CNN					
Batch size	**Slices**	**Width**	**Height**	**Input Volume**	**Out of Memory Error**
8	8	150	150	1 440 000	No
16	8	150	150	2 880 000	No
20	8	150	150	3 600 000	No
24	8	150	150	4 320 000	No
28	**8**	**150**	**150**	**5 040 000**	**Yes**
32	**8**	**150**	**150**	**5 760 000**	**Yes**
64	**8**	**150**	**150**	**11 520 000**	**Yes**
128	**8**	**150**	**150**	**23 040 000**	**Yes**

5.4 Results of the comparison with the value component analysis

Table 9 shows the level of achievement for each criterion for both architectures. The levels are defined with the method described in section 4.5.

The results of the value component analysis can be seen in Table 10. The I3D architecture achieves a higher score (2.788) than the MV CNN architecture (1.726).

Table 9. The test results and the corresponding level of achievement for each criterion. The better level of achievement is depicted in green. The sensitivity values of both models are regarded as equal due to the standard deviation being higher than the difference (Table 5).

Criterion	I3D		MV CNN	
	Value	Level of achievement	Value	Level of achievement
Accuracy (higher is better)	92.99%	3	80.66%	2
Specificity (higher is better)	93.38%	3	73.28%	2
Sensitivity (higher is better)	92.27%	2	93.75%	2
Model size (lower is better)	65.025 Mb	3	1431.675 Mb	1
Temporal efficiency (lower is better)	2:59 h	3	10:33 h	1
GPU load (higher is better)	13 500 000	3	4 032 000	1

Table 10. The result of the value component analysis. The achieved score of both architectures is depicted in purple.

Criterion	I3D			MV CNN		
	Level of achievement	Weight	Result	Level of achievement	Weight	Result
Accuracy	3	0.330	**0.990**	2	0.330	**0.660**
Specificity	3	0.200	**0.600**	2	0.200	**0.400**
Sensitivity	2	0.200	**0.400**	2	0.200	**0.400**
Model size	3	0.033	**0.099**	1	0.033	**0.033**
Temporal efficiency	3	0.033	**0.099**	1	0.033	**0.033**
GPU load	3	0.200	**0.600**	1	0.200	**0.200**
		Sum	2.788		Sum	1.726

5.5 The adaptability of the architectures with transfer learning

The mean difference in kernel weights between the two finetuned models and their corresponding base models can be seen in Figure 23.

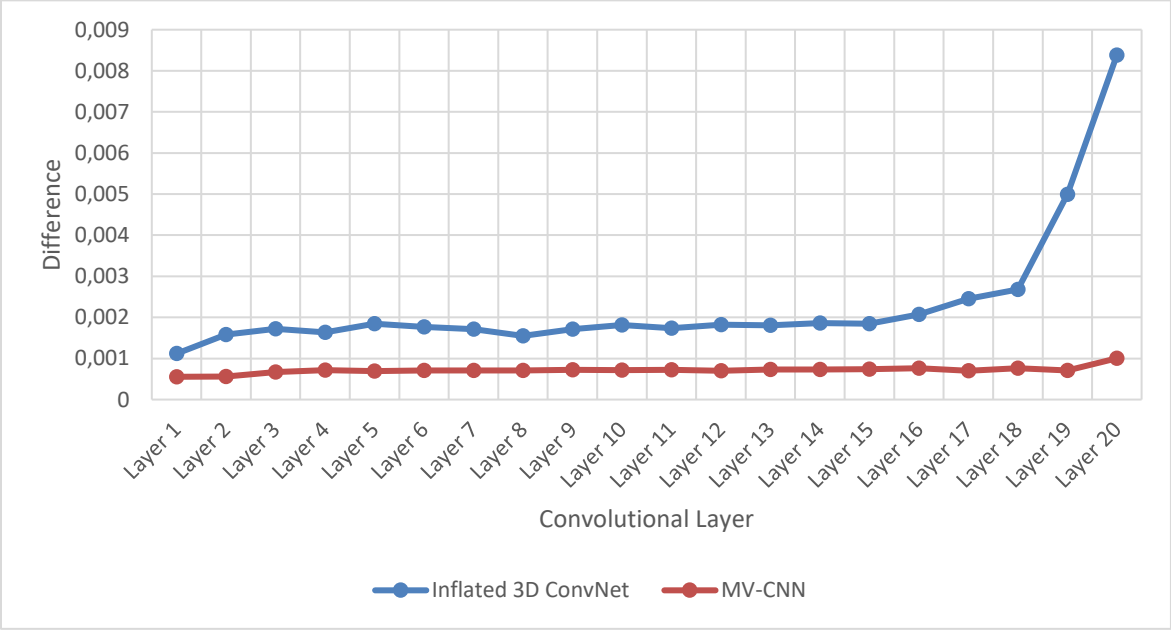


Figure 23. The difference between filters of the finetuned models and the base models. The blue and red line depict the difference for the Inflated 3D ConvNet and the MV-CNN, respectively.

The difference increases throughout the layers of the network for both architectures. For the Inflated 3D ConvNet, the difference is highest at the last layer, with a value of 0.008379, and the mean difference is 0.002306. The maximum value for the MV CNN at the last layer is 0.001004, and the mean difference is 0.000717.

5.5.1 The visualized convolutional kernels

The following four figures visualize the weight difference between convolutional kernels of the base model and the finetuned model, for both architectures. Figure 24 and Figure 25 serve as a visual comparison of the first layer of the I3D architecture and the last layer, respectively. Although there are differences between the convolutional kernels of the first layer, as shown in Figure 23, they are not noticeable in Figure 24. Apparently, the finetuning has little effect on the kernels of the first convolutional layer. Figure 25 shows more apparent differences, but still no qualitative changes to the kernel structure.

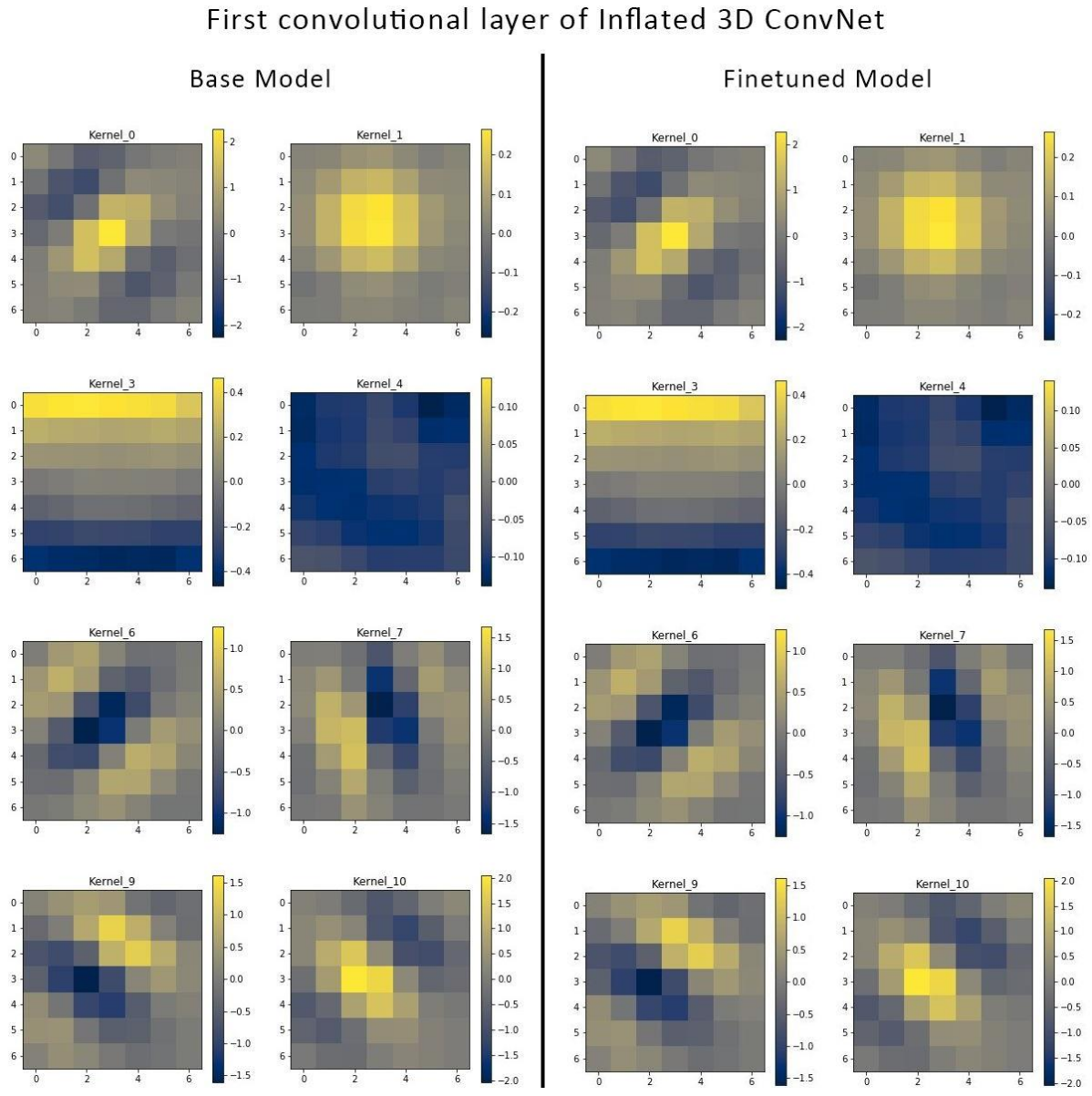


Figure 24. Slices of the 3D 7x7 convolutional kernels of the first convolutional layer of two Inflated 3D ConvNets. The kernels of the base model are on the left side and the kernels of the finetuned model are on the right side. Next to each kernel is a colorbar that defines the numerical value for each color.

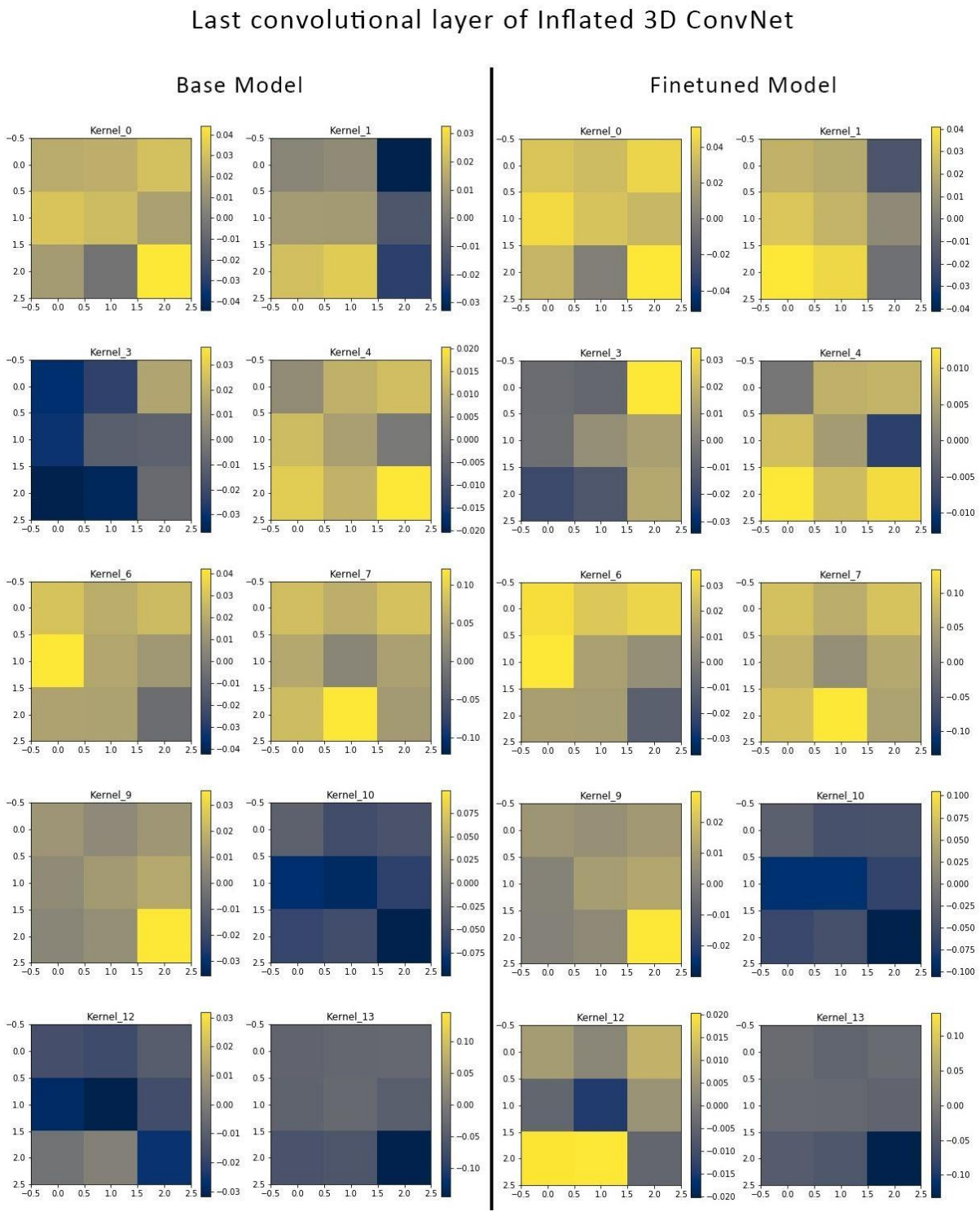


Figure 25. Slices of the 3D 3x3 convolutional kernels of the last convolutional layer of two Inflated 3D ConvNets. The kernels of the base model are on the left side and the kernels of the finetuned model are on the right side. Next to each kernel is a colorbar that defines the numerical value for each color.

Figure 26 and Figure 27 serve as a visual comparison of the first layer of the MV CNN architecture and the last layer respectively. Same as for the I3D architecture, there are no noticeable differences between the first layer in Figure 26, and small differences between the last layer in Figure 27. However, there are also no qualitative changes to the kernel structure.

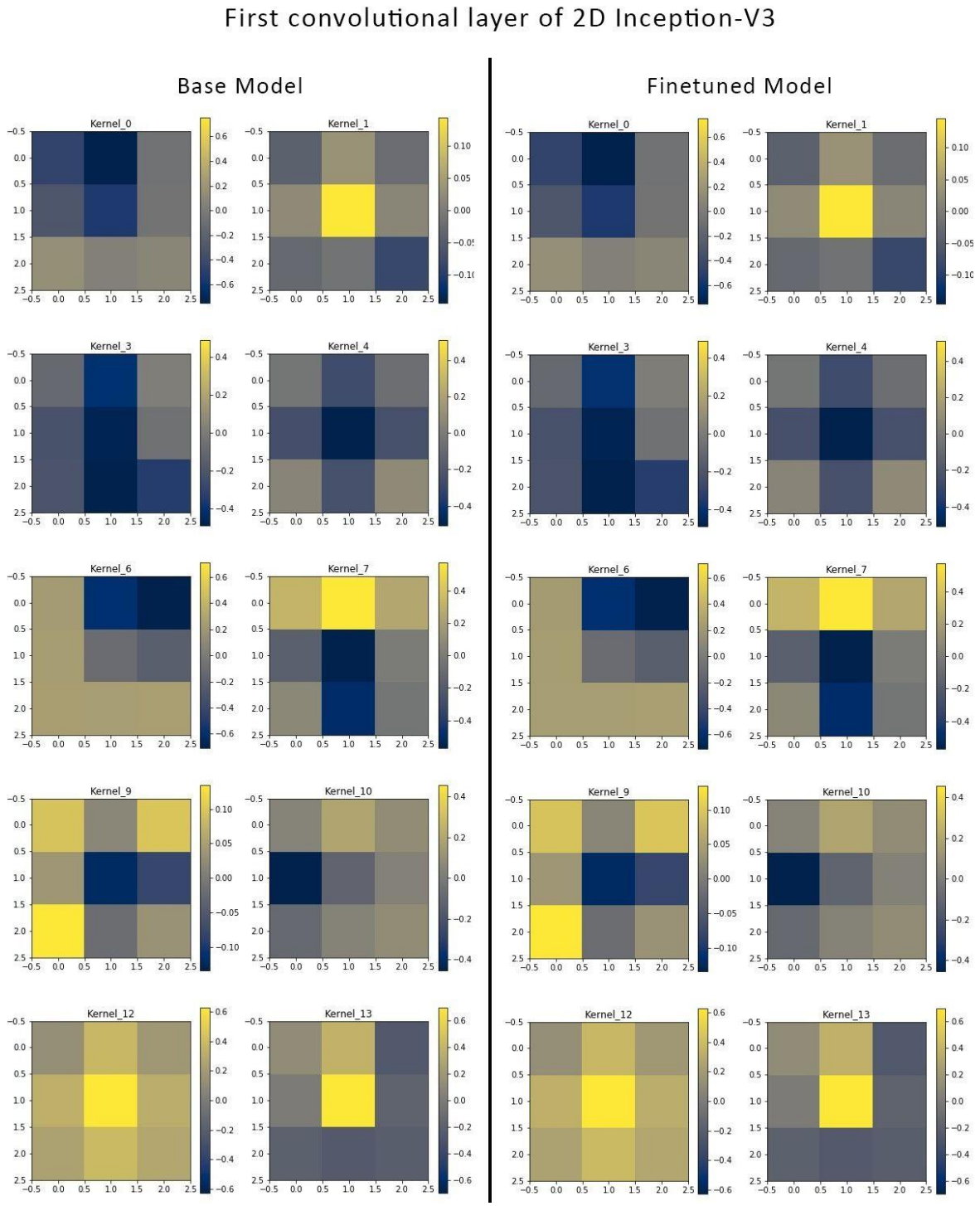


Figure 26. The 3x3 convolutional kernels of the first convolutional layer of two 2D Inception-V3 models. The kernels of the base model are on the left side and the kernels of the finetuned model are on the right side. Next to each kernel is a colorbar that defines the numerical value for each color.

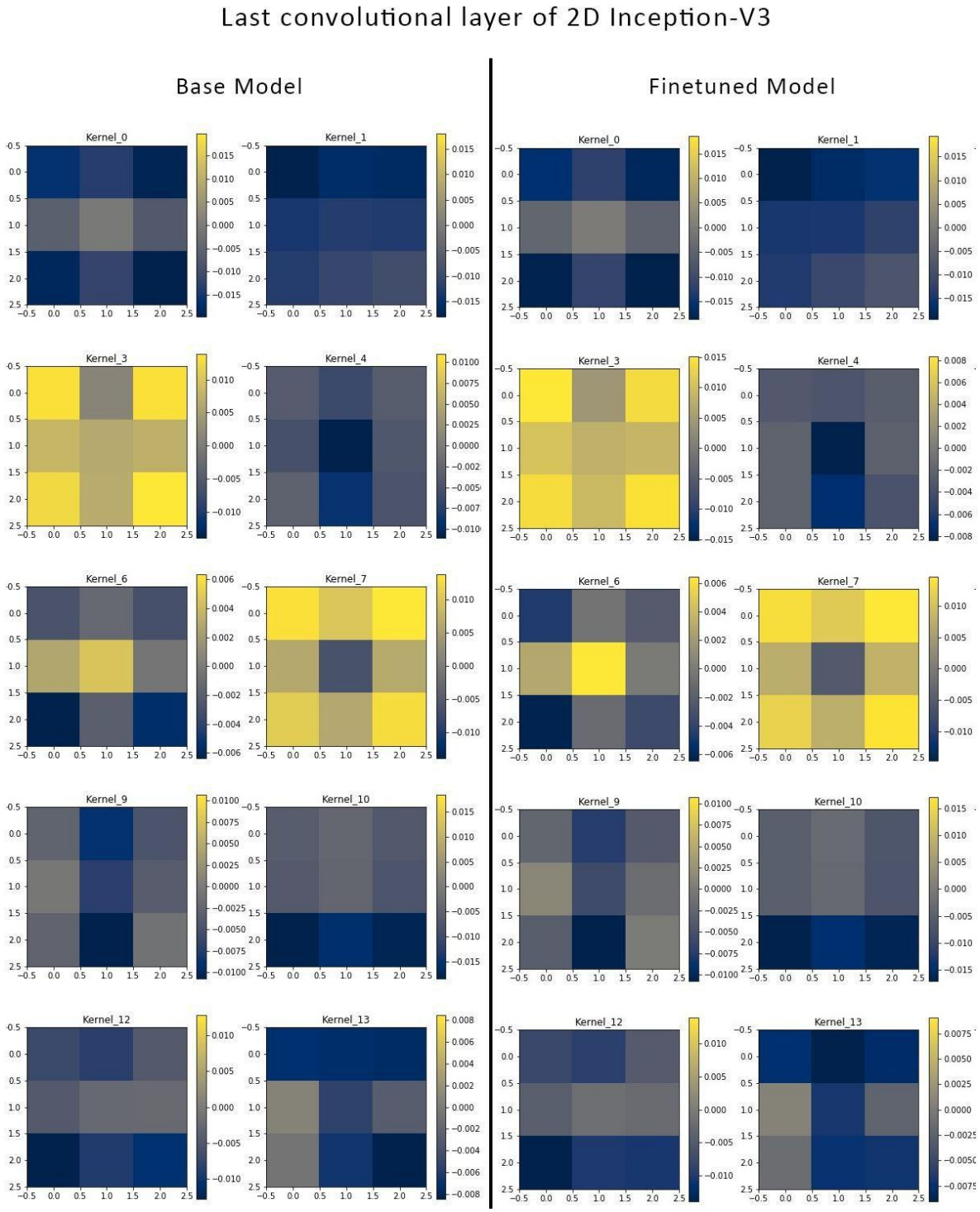


Figure 27. The 3x3 convolutional kernels of the last convolutional layer of two 2D Inception-V3 models. The kernels of the base model are on the left side and the kernels of the finetuned model are on the right side. Next to each kernel is a colorbar that defines the numerical value for each color.

5.5.2 The visualized activation maps

Figure 28 and Figure 29 show an activation map from every convolutional layer of the base I3D model and the finetuned I3D model, respectively (except for 1x1 convolutional layers). Figure 29 shows the same for the finetuned I3D model. There are apparent differences between the activation maps of the two models. It is noticeable that in the last two layers of the finetuned I3D in Figure 29, the higher values (positive and negative) are centered in the activation maps.

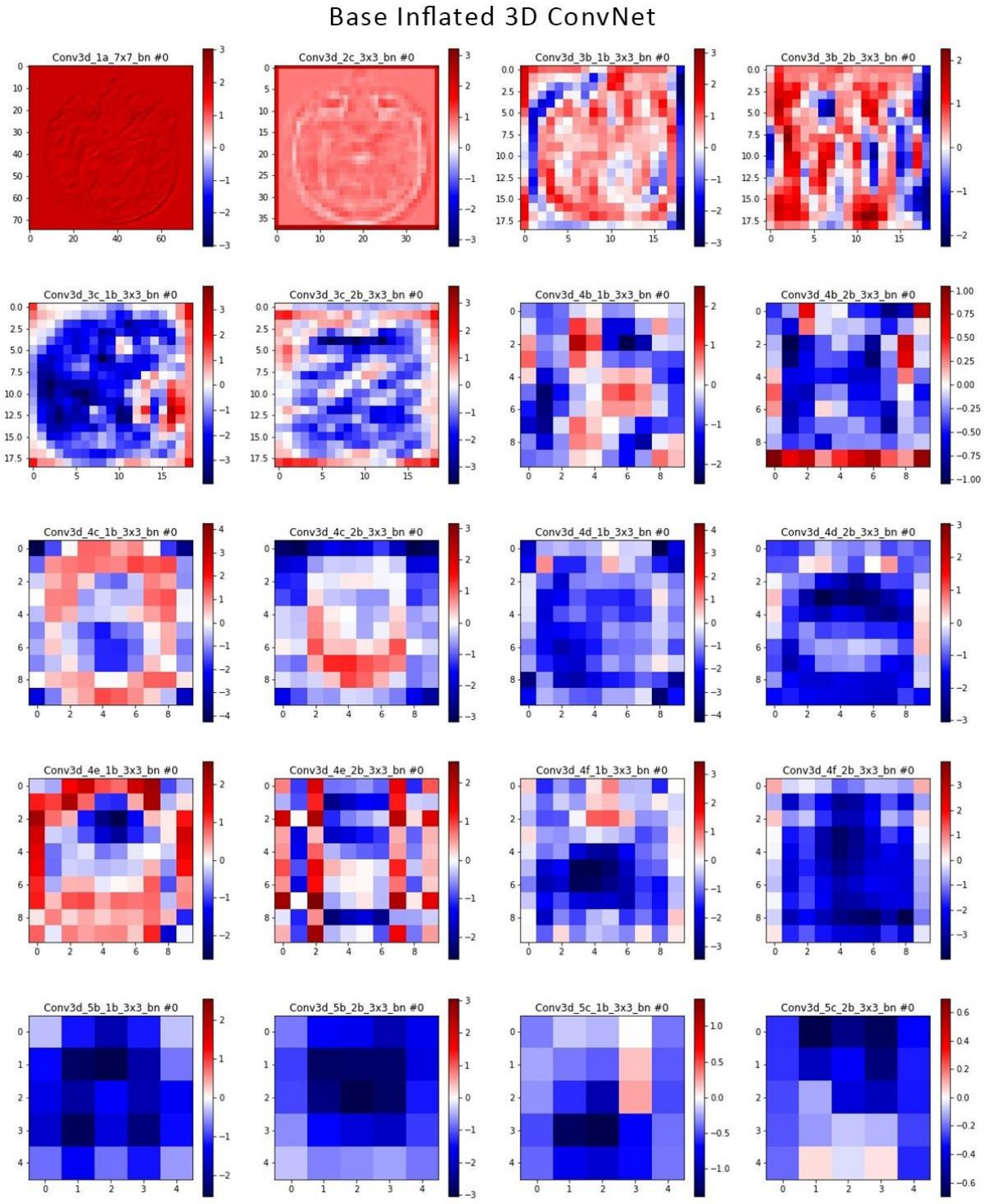


Figure 28. The slices of 20 activation maps from the base Inflated 3D ConvNet. They are ordered chronologically, with Conv_3d_1a being an activation map of the first convolutional layer, and Conv3d_5c being an activation map of the last convolutional layer. The dimensions of all activation maps from the respective convolutional layer are depicted in square brackets. Next to each activation map is a colorbar that defines the numerical value for each color.

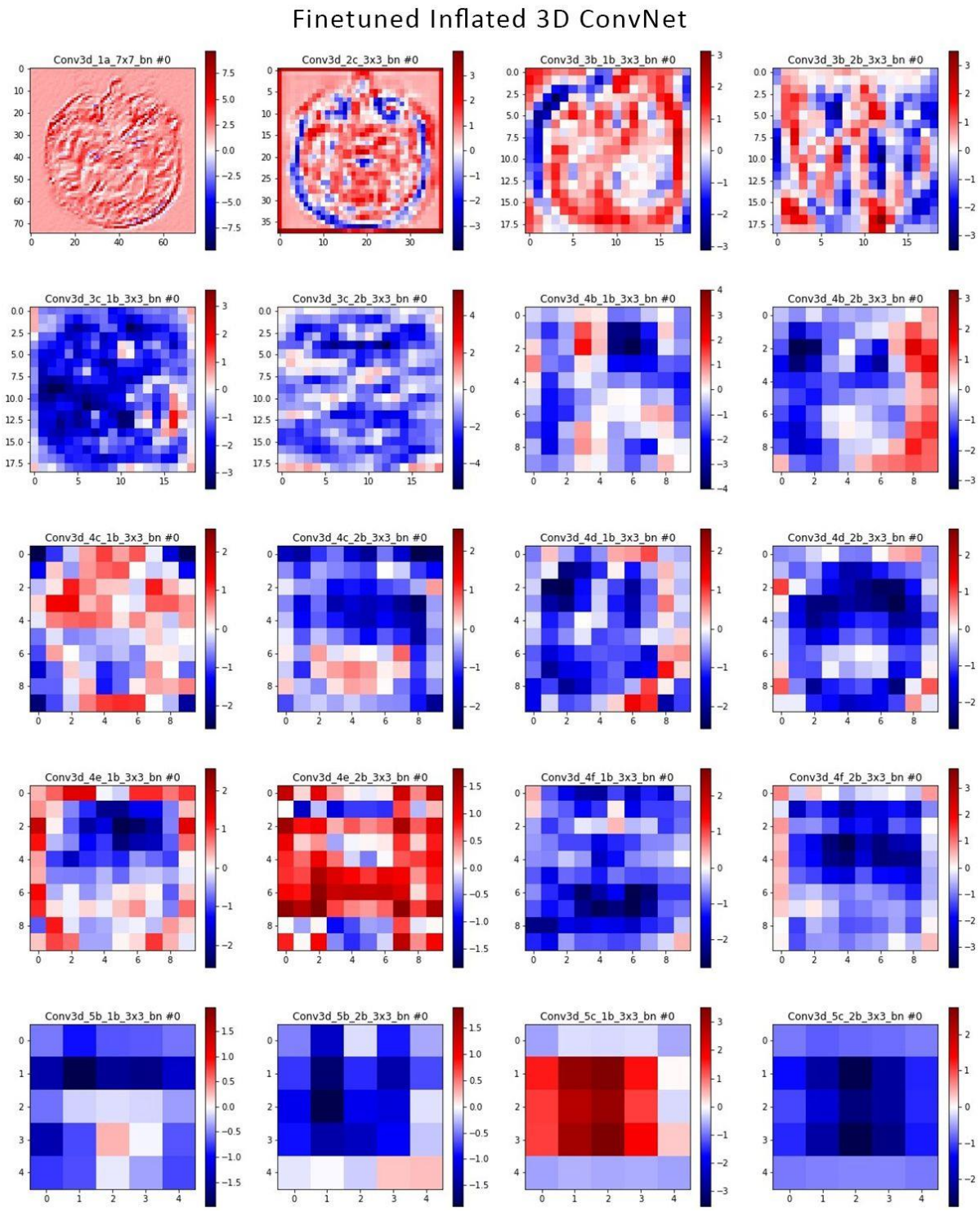


Figure 29. The slices of 20 activation maps from the finetuned Inflated 3D ConvNet. They are ordered chronologically, with Conv_3d_1a being an activation map of the first convolutional layer, and Conv3d_5c being an activation map from the last convolutional layer. The dimensions of all activation maps from the respective convolutional layer are depicted in square brackets. Next to each activation map is a colorbar that defines the numerical value for each color.

Figure 30 and Figure 31 show an activation map from every convolutional layer of the base 2D Inception-V3 model and the finetuned 2D Inception-V3 model, respectively (except for 1x1 convolutional layers). There are noticeable differences between the activation maps of the two models. The maps decrease in size throughout the network.

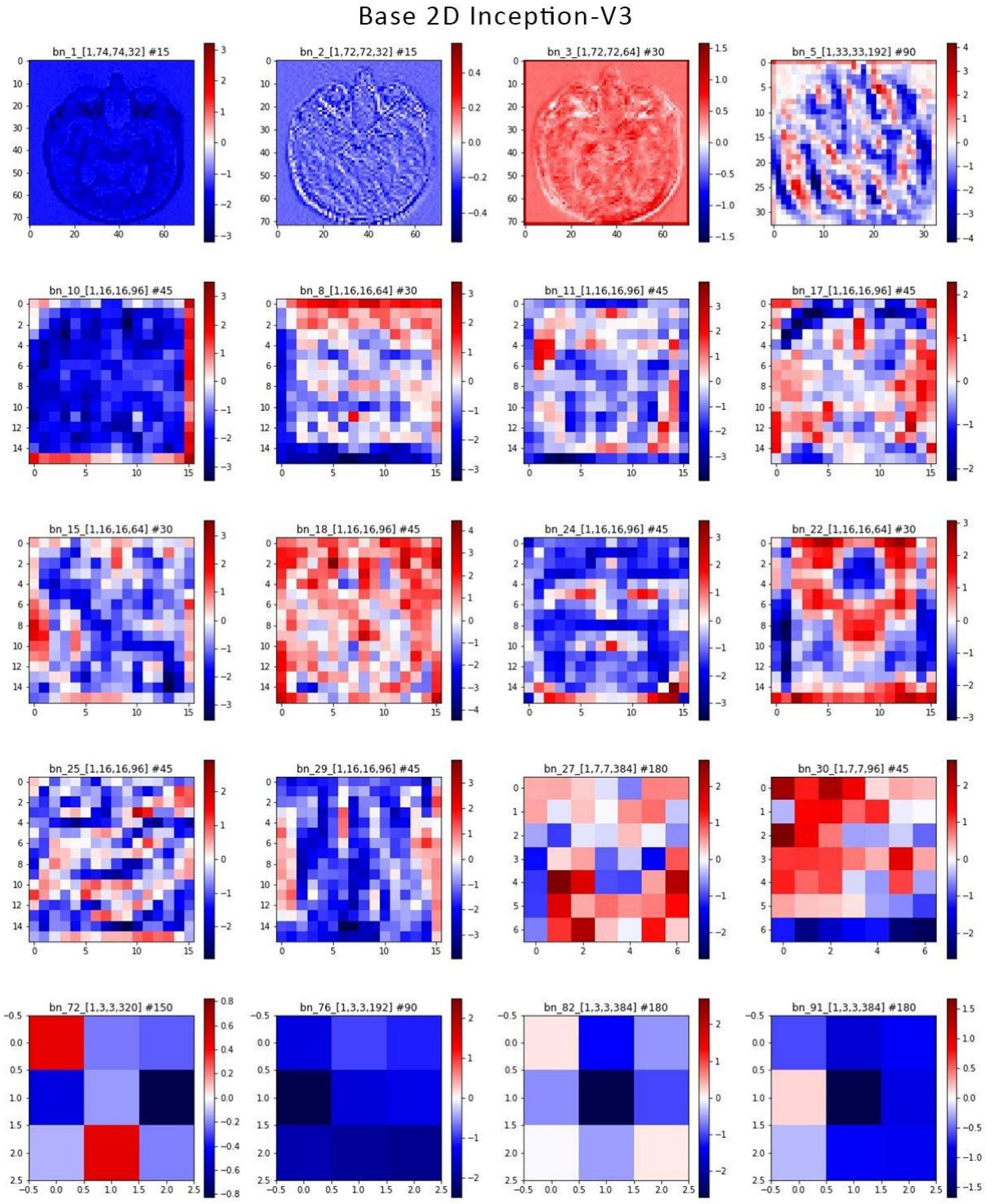


Figure 30. Twenty activation maps from the base 2D Inception-V3 model. They are ordered chronologically, with bn_1 being an activation map of the first convolutional layer, and bn_91 being an activation map from the last convolutional layer. The dimensions of all activation maps from the respective convolutional layer are depicted in square brackets. Next to each activation map is a colorbar that defines the numerical value for each color.

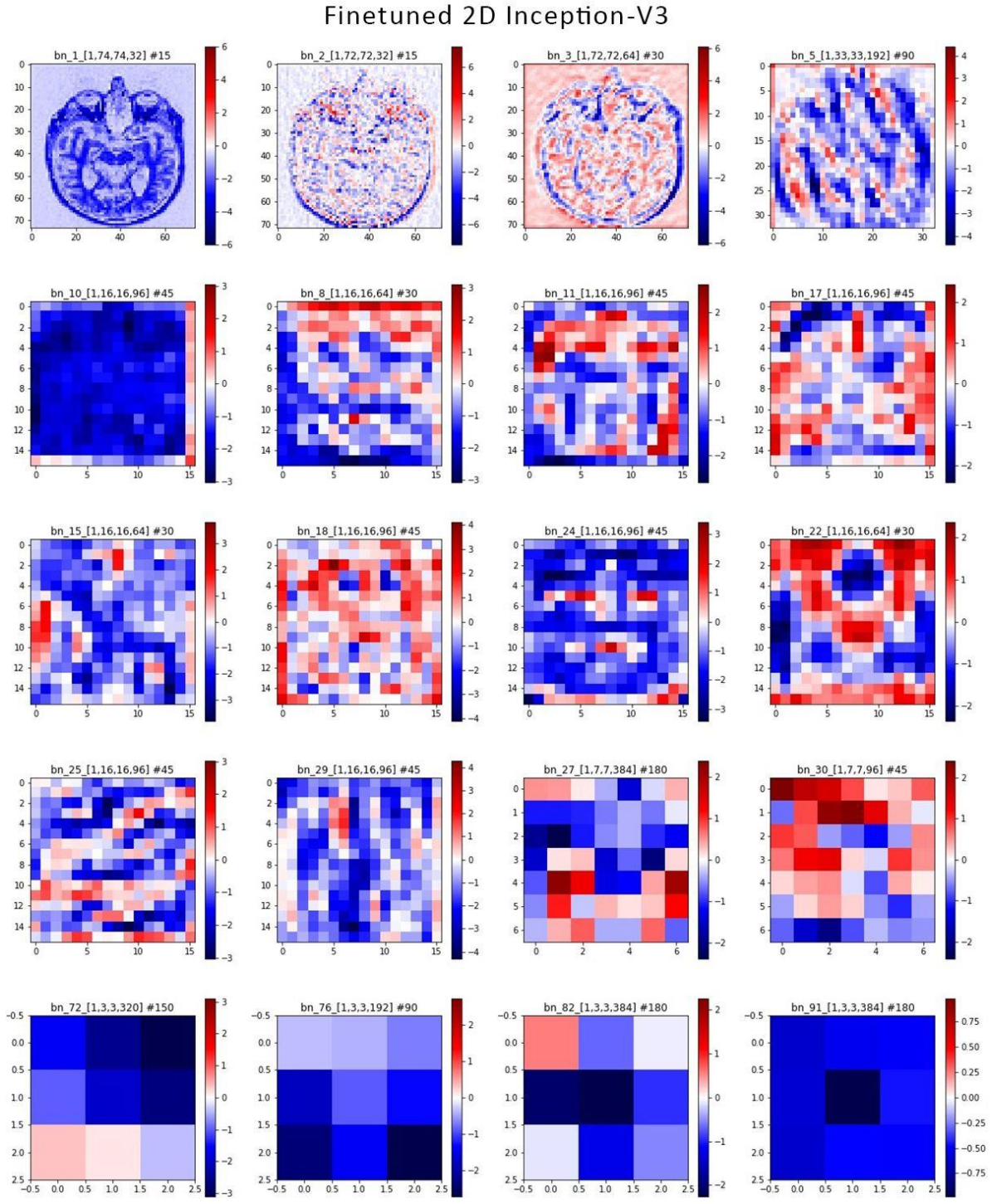


Figure 31. Twenty activation maps from the finetuned 2D Inception-V3 model. They are ordered chronologically, with bn_1 being an activation map of the first convolutional layer, and bn_91 being an activation map from the last convolutional layer. The dimensions of all activation maps of the respective convolutional layer are depicted in square brackets. Next to each activation map is a colorbar that describes the numerical value for each color.

In summary, finetuning the CNN with the MRI data has more impact on the I3D architecture than on the MV CNN architecture. The difference is notable through calculation and visual comparison. However, there are no qualitative changes noticeable between the base models and finetuned models for both architectures.

6 Discussion

The Inflated 3D ConvNet from Carreira and Zisserman [25] was chosen as an architecture for the comparison, due to it being a 3D CNN based on the established Inception-V1 [19] architecture, with publically available pretrained weights. Since the publication of the Inception-V1 architecture in 2014, models with better performance on benchmark datasets have been developed, like e.g., the Inception-V3 architecture [50]. However, no 3D Versions of those 2D models with pretrained weights have yet been published. Therefore, the MV CNN architecture of this work was developed. It allows the classification of 3D medical imaging data with recently published, state of the art 2D CNNs. The Inception-V3 architecture was chosen as 2D CNN for the MV CNN because it is the predecessor of the Inception-V1 architecture, with better performance on the ImageNet benchmark dataset.

Before the Inflated 3D Inception ConvNet could be compared with the Multi-View CNN, both models' hyperparameters had to be optimized in order to provide a fair comparison ground. Although Bergstra and Bengio [71] show that random search is more efficient, grid search was chosen as the method for hyperparameter optimization. The reason for this choice was that the number of training sessions that could be performed was limited due to the long training times shown in section 5.3. The long training durations were also the reasons why no cross-validation was performed for every hyperparameter configuration. As a result, grid search was used to narrow down the area of viable hyperparameters for efficient training of the models. The results of the grid search method show that the values of the hyperparameters have a significant impact on the performance of the CNNs. Nevertheless, in order to find the actually best performing hyperparameter configuration for each model, way more iterations of grid search or random search must be performed. This task was impossible to execute due to hardware limitations and will, therefore, be left for future work.

Another important measure for increasing the performance of a CNN is the augmentation of the training dataset. This includes, e.g., image flipping, image cropping, image rotation, or noise injection [72]. The augmentation of the ADNI dataset was not important for the comparison of the two models and will therefore be left for future work, that focuses on further optimization of the proposed I3D and MV CNN architecture.

The split into training and test data, in this work, has two flaws: The first one is that no test data in addition to the training and validation data was separated from the complete ADNI dataset. This could lead to a biased hyperparameter optimization. The second flaw emerges due to the ADNI dataset containing multiple MRI head scans of some patients. The split into training and test dataset did not occur at the patient-level, but at the image-level, which can result in the appearance of data from the same subject in both sets. According to Wen et al. [73], this data leakage can lead to an increase in the accuracy of a model compared to the same model with no prevalent data leakage. Both flaws impair the comparison of the models' statistical metrics to other state of the art CNNs.

The results of transfer learning with a partially frozen convolutional base (Table 4) show that freezing no layers led to better accuracy of the MV CNN. This consistent with the results of Kruithof et al. and Yosinski et al. that were shown in section 3.4. Therefore, all layers in both architectures were set to be trainable (not frozen) during training and evaluation (Option B in Figure 10).

When reflecting on the comparison of both models, it is noticeable that they were not trained with the same input size (50 x 150 x 150 for the I3D and 8 x 150 x 150 for the MV CNN). This was due to the

limited hardware available. A MV CNN consisting of 50 2D Inception-V3 models constantly produced an OOM error during training on the Nvidia Geforce GTX 1080ti GPU. As a result, the number of input slices for the MV CNN was set to 8 instead of 50. Fewer input slices lead to less information of the MRI head scan for the MV CNN, which should result in lower statistical metrics of the model. Due to AD being a neurodegenerative disease that affects the whole brain tissue [74], it is possible that the decrease in accuracy is small-scale. Contrary, when analyzing MRI head scans from patients with a brain tumor, fewer input slices would result in a significant decrease of the MV CNN models' accuracy because the likelihood of the tumor being visible in one of the slices would be lower. The input volume for the I3D was not lowered to also consist of eight slices, because a goal of this work was to achieve optimal performance of both architectures, with the available resources.

6.1 The quality of the architecture

The above-mentioned difference in the number of input slices is probably an important reason for the worse accuracy of the MV CNN architecture compared to the I3D architecture (Table 5). However, there is an additional reason that could explain the difference. In contrast to the 2D convolutional kernels of the MV CNN, the 3D kernels of the I3D architecture also move along the longitudinal axis of the head. This enables the I3D model to compensate for A), the varying slice thicknesses and B), different positions of the brain, across scans from the ADNI dataset.

In contrast to the accuracy values, the sensitivity values of both models are the same (Table 5). The reason for the relatively high sensitivity of the MV CNN compared to its accuracy could be the structure of its feature extraction at the end. Each 2D model produces a classification result, which are then combined to produce the final output of the model. As a result, a single input slice being classified as abnormal can lead to the final output being abnormal, as well. Thus the MRI head scan is more often classified as abnormal or AD, which leads to the high sensitivity of the MV CNN. The high sensitivity shows promising potential for the use of the MV CNN as a second opinion for radiologists because it could assist them in reducing the number of false-negative diagnostics.

6.2 The applicability of the architecture

The MV CNN has a substantially larger size (1431.675 Mb) than the I3D model (96.025 Mb). A possible explanation for that difference could be the additional parameters of a convolutional layer in a CNN. Convolutional layers of a CNN consist of additional information alongside their weights, like activation function, bias, etc. (section 3.3.1), which also need to be stored in the models' save-file. Therefore, it is more memory efficient to store one 3D 3x3x3 convolutional layer, than storing three 2D 3x3 convolutional layers, although both variants are containing $3 \cdot 3 \cdot 3 = 27$ values. As a result, it is also more memory efficient to store a single 3D CNN than multiple 2D CNNs combined in one model. In addition to that, the I3D architecture consists of only 12 252 690 parameters, compared to the 182 823 218 parameters of the MV CNN. Even a single 2D Inception-V3 model contains 22 852 898 parameters, which is more than the complete I3D model. A reason for the large number of parameters is that the MV CNN contains only 2D pooling layers in contrast to the 3D pooling layers of the I3D architecture (section 3.3.1). As a result, the MV CNN cannot reduce the dimensions of the input volume in the slice direction.

The average time for training the MV CNN with 100 epochs (10 hours and 33 minutes) is noticeably higher than training the I3D (2 hours and 59 minutes), which is shown in Table 6. The reason for that difference is the architecture of the MV CNN. During forward propagation, the results of all eight 2D

CNNs is calculated serially instead of parallel. On the one hand, this leads to a lower GPU load, but on the other hand, this increases the training time. However, during a test, where the batch size was doubled to 16 (which also doubles the amount of data that is processed parallelly), the training time decreased to 5 hours and 47 minutes. This shows that the MV CNN could have been optimized further to achieve better temporal efficiency. The average time for classification of one input image is 542 milliseconds for the MV CNN and 52 milliseconds for the I3D architecture. Although the MV CNN is slower than the I3D, the difference is not severe, because both times would not delay the workflow of a radiologist.

6.3 The visual comparison

The figures from section 5.5.1 and section 5.5.2 demonstrate the difficulty of visually interpreting the convolutional kernels and activation maps of a CNN. However, the calculated differences between the kernels of the respective base model and the finetuned model in Figure 23 show the expected results. The difference increases throughout the layers of both architectures, which is to be expected due to the backpropagation algorithm (section 3.2). The mean and maximum difference is higher for the I3D architecture, which shows that the adaption of its weights with transfer learning is higher than the adaption of the MV CNN. This is also expected due to the weights of larger CNNs, with more parameters being less affected by backpropagation.

The convolutional kernels of the base model in Figure 24 show a similar structure to convolutional operators used in hand-crafted digital image processing algorithms. For example, kernel_3 resembles the structure of the Prewitt operator, and kernel_0 resembles the structure of a Laplacian filter [75]. The fact that those convolutional kernels hardly are changed during finetuning shows that the chosen pretrained CNNs are applicable to medical tasks with transfer learning. However, there are no actual qualitative changes to the structure of the convolutional kernels in both architectures. This indicates that transfer learning does not result in the models adapting their kernels to extract new, important features of the brain in the ADNI dataset. Instead, the models still extract features that are important for objects of the ImageNet dataset but learn to combine these features in order to classify MRI head scans. Raghu, Zhang et al. have shown similar results when analyzing the effects of transfer learning for medical imaging on CNNs with different sizes. Their explanation for the minor changes in the convolutional kernels was also based on the overparameterization of the large CNNs that were developed for classification of the ImageNet dataset [76].

Figure 29 shows that the activation maps of the last two convolutional layers of the finetuned I3D model, have high values concentrated in the center. This structure is not visible in the activation maps of the base I3D model in Figure 28. Due to the high values being at similar respective positions as the brain of the MRI head scans, the structure could be a result of the minor adaption to the ADNI dataset.

6.4 Recommendation on an architecture

In summary, the I3D architecture is better than the MV CNN architecture in every criterion of the comparison, besides the sensitivity. Accordingly, the result of the Value Benefit Analysis in Table 10 also shows the I3D architecture as a winner for the comparison with 2.788 points to 1.726 points. Although the MV CNN shows potentially good results in terms of its' quality, the poor applicability of the architecture makes it difficult to embed it into mobile CAD software. Therefore, the I3D architecture is recommended over the MV CNN architecture for the task of classifying MRI head scans and the usage in CAD software.

6.5 Comparison to other state of the art CNNs

Due to the earlier mentioned data leakage, a fair comparison of the proposed I3D models' accuracy with other state of the art CNNs is hardly possible. However, Wen et al. have published an extensive overview of CNNs that were constructed for the classification of Alzheimer's Disease. In their work, the authors also compared studies with clear and unclear occurrences of data leakage [73]. A list of those models can be seen in Table 11.

Table 11. The accuracy of different CNNs performing classification of Alzheimer's disease. Studies with clear and unclear occurrence of data leakage are marked in red and orange, respectively. The accuracies for the binary classification of AD and CN are listed in the second to left column. MCI, sMCI and pMCI stand for different stages of Alzheimer's disease. [73]

Study	Performance					Approach	Data leakage (type)
	AD vs CN	sMCI vs pMCI	MCI vs CN	AD vs MCI	Multi-class		
(Aderghal et al., 2017a)	ACC=0.91	--	ACC=0.66	ACC=0.70	--	ROI-based	Unclear (b,c)
(Basaia et al., 2019)	BA=0.99	BA=0.75	--	--	--	3D subject-level	Unclear (b)
(Hon and Khan, 2017)	ACC=0.96	--	--	--	--	2D slice-level	Unclear (a,c)
(Hosseini Asl et al., 2018)	ACC=0.99	--	ACC=0.94	ACC=1.00	ACC=0.95[2]	3D subject-level	Unclear (a)
(Islam and Zhang, 2017)	--	--	--	--	ACC=0.74[1]†	2D slice-level	Unclear (b,c)
(Lin et al., 2018)	ACC=0.89	ACC=0.73	--	--	--	ROI-based	Unclear (b)
(Manhua Liu et al., 2018)	ACC=0.85	ACC=0.74	--	--	--	3D patch-level	Unclear (d)
(Taqi et al., 2018)	ACC=1.00	--	--	--	--	2D slice-level	Unclear (b)
(Vu et al., 2017)	ACC=0.85	--	--	--	--	3D subject-level	Unclear (a)
(S.-H. Wang et al., 2018)	ACC=0.98	--	--	--	--	2D slice-level	Unclear (b)
(Bäckström et al., 2018)*	ACC=0.99	--	--	--	--	3D subject-level	Clear (a)
(Farooq et al., 2017)	--	--	--	--	ACC=0.99[3]†	2D slice-level	Clear (a,c)
(Gunawardena et al., 2017)	--	--	--	--	ACC=0.96[2]	3D subject-level	Clear (a,b)
(Vu et al., 2018)	ACC=0.86	--	ACC=0.86	ACC=0.77	ACC=0.80[2]	3D subject-level	Clear (a,c)
(Wang et al., 2017)	--	--	ACC=0.91	--	--	2D slice-level	Clear (a,c)
(Wang et al., 2019)	ACC=0.99	--	ACC=0.98	ACC=0.94	ACC=0.97[2]	3D subject-level	Clear (b)
(Wu et al., 2018)	--	--	--	--	0.95[4]†	2D slice-level	Clear (a,b)

Comparing the accuracy of the proposed architecture to the studies in Table 11, the I3D architecture ranks somewhere in the middle with 92.99%. When comparing the accuracy to studies in Table 12, with no detected data leakage, the model would have achieved better than state of the art accuracy, which is highly unlikely. This further demonstrates the falsification of accuracy due to the flaws of the data split in this work (data leakage).

Table 12. The accuracy of different CNNs performing classification of Alzheimer's disease, with no occurring data leakage, detected. The accuracies for the binary classification of AD and CN are listed in the second to left column. MCI, sMCI and pMCI stand for different stages of Alzheimer's disease. [73]

Study	Performance					Approach	Data leakage
	AD vs CN	sMCI vs pMCI	MCI vs CN	AD vs MCI	Multi-class		
(Aderghal et al., 2017b)	ACC=0.84	--	ACC=0.65	ACC=0.67†	--	ROI-based	None detected
(Aderghal et al., 2018)	BA=0.90	--	BA=0.73	BA=0.83	--	ROI-based	None detected
(Bäckström et al., 2018) *	ACC=0.90	--	--	--	--	3D subject-level	None detected
(Cheng et al., 2017)	ACC=0.87	--	--	--	--	3D patch-level	None detected
(Cheng and Liu, 2017)	ACC=0.85	--	--	--	--	3D subject-level	None detected
(Islam and Zhang, 2018)	--	--	--	--	ACC=0.93[1]†	2D slice-level	None detected
(Korolev et al., 2017)	ACC=0.80	--	--	--	--	3D subject-level	None detected
(Li et al., 2017)	ACC=0.88	--	--	--	--	3D subject-level	None detected
(Li et al., 2018)	ACC=0.90	--	ACC=0.74†	--	--	3D patch-level	None detected
(Lian et al., 2018)	ACC=0.90	ACC=0.80†	--	--	--	3D patch-level	None detected
(Mingxia Liu et al., 2018a)	ACC=0.91	ACC=0.78†	--	--	--	3D patch-level	None detected
(Mingxia Liu et al., 2018c)	ACC=0.91	--	--	--	--	3D patch-level	None detected
(Qiu et al., 2018)	--	ACC=0.83†	--	--	--	2D slice-level	None detected
(Senanayake et al., 2018)	ACC=0.76	--	ACC=0.75	ACC=0.76	--	3D subject-level	None detected
(Shmulev et al., 2018)	--	ACC=0.62	--	--	--	3D subject-level	None detected
(Valliani and Soni, 2017)	ACC=0.81	--	--	--	ACC=0.57[2]	2D slice-level	None detected

The performance of the I3D model was additionally tested on a smaller dataset with 335 MRI head scans, that consisted of more pathologies in addition to AD. For the classification into normal (healthy) and abnormal (pathological), the model achieved an accuracy of 83.9%. This indicates good applicability of the I3D architecture for different classification tasks of medical imaging scans.

7 Conclusion

In summary, the proposed I3D architecture is better in comparison to the MV CNN architecture, in the classification of MRI head scans, under given circumstances. The performance of the I3D model can be further enhanced with data augmentation and additional hyperparameter optimization, which will be left as a task for future work. The flaws of the data split in this work (data leakage) make the comparison to state of the art CNNs difficult. However, the achieved accuracy of 92.99%, shows promising potential of the architecture, in comparison to other CNNs with data leakage. Therefore, the I3D architecture of this work is recommended for the application of classifying MRI head scans and the usage in CAD software.

8 Acknowledgment

Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012).

I want to thank my family for their constant support throughout my time as a student. Without them, I probably wouldn’t have been able to work on this thesis.

I also would like to thank Prof. Dr. Stefanie Remmele for her excellent guidance and mentoring during the time of this work.

9 References

1 *Attia ZI, Noseworthy PA, Lopez-Jimenez F et al.* An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction. The Lancet 2019; 394: 861 – 867

2 *Kang G, Liu K, Hou B et al.* 3D multi-view convolutional neural networks for lung nodule classification. PloS one 2017; 12: e0188290

3 *Ardila D, Kiraly AP, Bharadwaj S et al.* End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. Nat Med 2019; 25: 954 – 961

4 *Walsh CG, Ribeiro JD, Franklin JC*. Predicting Risk of Suicide Attempts Over Time Through Machine Learning. Clinical Psychological Science 2017; 5: 457 – 469

5 *Smith-Bindman R, Kwan ML, Marlow EC et al.* Trends in Use of Medical Imaging in US Health Care Systems and in Ontario, Canada, 2000-2016. JAMA 2019; 322: 843 – 856

6 *Pitman A, Cowan IA, Floyd RA et al.* Measuring radiologist workload: Progressing from RVUs to study ascribable times. Journal of medical imaging and radiation oncology 2018; 62: 605 – 618

7 *Hanna TN, Lamoureux C, Krupinski EA et al.* Effect of Shift, Schedule, and Volume on Interpretive Accuracy: A Retrospective Analysis of 2.9 Million Radiologic Examinations. Radiology 2018; 287: 205 – 212

8 *Grobe TG, Dörning H, Schwartz FW*. Schwerpunkt: Bildgebende Diagnostik - Computer- und Magnetresonanztomographie: BARMER GEK Arztreport 2011. St. Augustin: Asgard-Verlag, 2011

9 *Gupta V, Khandelwal N, Prabhakar A et al.* Prevalence of normal head CT and positive CT findings in a large cohort of patients with chronic headaches. The Neuroradiology Journal 2015; 28: 421 – 425

10 *Zan E, Yousem DM, Carone M et al.* Second-opinion consultations in neuroradiology. Radiology 2010; 255: 135 – 141

11 *Briggs GM, Flynn PA, Worthington M et al.* The role of specialist neuroradiology second opinion reporting: is there added value? Clinical radiology 2008; 63: 791 – 795

12 *Dong H, Yang G, Liu F, Mo Y, Guo Y*. Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks. Communications in Computer and Information 2017: 506 – 517

13 *Lian C, Liu M, Zhang J et al.* Hierarchical Fully Convolutional Network for Joint Atrophy Localization and Alzheimer's Disease Diagnosis using Structural MRI. IEEE transactions on pattern analysis and machine intelligence 2018: 1

14 *Suk H-I, Lee S-W, Shen D et al.* Deep sparse multi-task learning for feature selection in Alzheimer’s disease diagnosis. Brain Structure and Function 2016; 221: 2569 – 2587

15 *Lin W, Tong T, Gao Q et al.* Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease Prediction From Mild Cognitive Impairment. Frontiers in neuroscience 2018; 12: 777

16 *Khvostikov A, Aderghal K, Benois-Pineau J et al.* 3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies, 2018

17 *Lecun Y, Haffner P, Bottou L, Bengio Y*. Object Recognition with Gradient-Based Learning. In: Forsyth DA (ed). Shape, contour and grouping in computer vision. Berlin [etc.]: Springer, 1999: 319 – 345

18 *Krizhevsky A, Sutskever I, Hinton G*. ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems 2012

19 *C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich*. Going deeper with Convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015: 1 – 9
20 *K. He, X. Zhang, S. Ren, J. Sun*. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 770 – 778
21 *Cho J, Lee K, Shin E et al.* How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?, 2015
22 Health Insurance Portability and Accountability Act of 1996 (HIPAA) | CDC, 2019, https://www.cdc.gov/phlp/publications/topic/hipaa.html; Status: 01.11.2019
23 EUR-Lex - 32016R0679 - EN - EUR-Lex, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC; Status: 01.11.2019
24 *Kay W, Carreira J, Simonyan K et al.* The Kinetics Human Action Video Dataset, 2017
25 *Carreira J, Zisserman A*. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017: 4724 – 4733
26 *LaMontagne PJ, Keefe S, Lauren W et al.* OASIS-3: Longitudinal Neuroimaging, Clinical, and Cognitive Dataset for Normal Aging and Alzheimer Disease. Alzheimer's & Dementia 2018; 14: P138
27 *Petersen RC, Aisen PS, Beckett LA et al.* Alzheimer's Disease Neuroimaging Initiative (ADNI): clinical characterization. Neurology 2010; 74: 201 – 209
28 *Mullin E*. Artificial intelligence pinpoints nine different abnormalities in head scans. Nat Med 2018; 58: 452
29 *Yosinski J, Clune J, Bengio Y, Lipson H*. How Transferable Are Features in Deep Neural Networks? Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. Cambridge, MA, USA: MIT Press, 2014: 3320 – 3328
30 *Kuperman V*. Magnetic resonance imaging: Physical principles and applications. San Diego: Academic Press, 2000
31 *Diba A, Fayyaz M, Sharma V, Arzani MM, Yousefzadeh R, Gall J, van Gool L*. Spatio-temporal Channel Correlation Networks for Action Classification. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds). Computer Vision - ECCV 2018:15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV. Cham: Springer International Publishing; Imprint; Springer, 2018: 299 – 315
32 *Yamashita R, Nishio M, Do RKG et al.* Convolutional neural networks: an overview and application in radiology. Insights into imaging 2018; 9: 611 – 629
33 Applications - Keras Documentation, 2019, https://keras.io/applications/; Status: 09.11.2019
34 *da Silva IN, Hernane Spatti D, Andrade Flauzino R et al.* Artificial Neural Networks: A practical course. Cham: Springer, 2017
35 *ROSENBLATT F*. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 1958; 65: 386 – 408
36 *Ahire JB*. The Artificial Neural Networks Handbook: Part 4, 2018, https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e; Status: 16.12.2019
37 *Nwankpa C, Ijomah W, Gachagan A et al.* Activation Functions: Comparison of trends in Practice and Research for Deep Learning, 2018

38 *Sze V, Chen Y-H, Yang T-J et al.* Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proc. IEEE 2017; 105: 2295 – 2329
39 *Kingma D, Ba J*. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations 2014
40 *Kurbiel T, Khaleghian S*. Training of Deep Neural Networks based on Distance Measures using RMSProp 2017
41 *Duda J*. SGD momentum optimizer with step estimation by online parabola model, 2019
42 *Dumoulin V, Visin F*. A guide to convolution arithmetic for deep learning, 2016
43 *Gu J, Wang Z, Kuen J et al.* Recent Advances in Convolutional Neural Networks. Pattern Recognition 2018: 354 – 377
44 *Bergstra JS, Bardenet R, Bengio Y, Kégl B*. Algorithms for Hyper-Parameter Optimization. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds). Advances in Neural Information Processing Systems 24: Curran Associates, Inc, 2011: 2546 – 2554
45 *Refaeilzadeh P, Tang L, Liu H*. Cross-Validation. In: LIU L, Özsu MT (eds). Encyclopedia of Database Systems. Boston, MA: Springer US, 2009: 532 – 538
46 S. Ji, W. Xu, M. Yang *et al.* 3D Convolutional Neural Networks for Human Action Recognition. IEEE transactions on pattern analysis and machine intelligence 2013; 35: 221 – 231
47 *Verma S*. Understanding 1D and 3D Convolution Neural Network | Keras, 2019, https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610; Status: 20.12.2019
48 *Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C*. A Survey on Deep Transfer Learning: 27th International Conference on Artificial Neural Networks: 270 – 279
49 *Kruithof M, Bouma H, Fischer N, Schutte K*. Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers. In: Burgess D, Owen G, Bouma H, Carlysle-Davies F, Stokes RJ, Yitzhaky Y (eds). Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII: SPIE, 2016 - 2016: 99950J
50 *Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z*. Rethinking the Inception Architecture for Computer Vision. Computer Vision and Pattern Recognition 2016
51 dlpbc/keras-kinetics-i3d, https://github.com/dlpbc/keras-kinetics-i3d; Status: 27.11.2019
52 *Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik G. Learned-Miller*. Multi-view Convolutional Neural Networks for 3D Shape Recognition. Proc. ICCV, 2015
53 *Chatfield K, Simonyan K, Vedaldi A et al.* Return of the Devil in the Details: Delving Deep into Convolutional Nets. BMVC 2014 - Proceedings of the British Machine Vision Conference 2014 2014
54 *Ting KM*. Confusion Matrix. In: Sammut C, Webb GI (eds). Encyclopedia of machine learning and data mining. 2nd ed. New York: Springer, op. 2017: 260
55 How convolutional neural networks see the world, 2018, https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html; Status: 29.11.2019
56 *Niklas C*. Mehr Entscheidungssicherheit mit der Nutzwertanalyse. Projekt Magazin 2014
57 *Krejčí J*. Pairwise Comparison Matrices. In: Krejčí J (ed). Pairwise Comparison Matrices and their Fuzzy Extension. Cham: Springer International Publishing, 2018: 17 – 56
58 *Brant-Zawadzki M, Gillan GD, Nitz WR*. MP RAGE: a three-dimensional, T1-weighted, gradient-echo sequence--initial experience in the brain. Radiology 1992; 182: 769 – 775
59 ADNI | MRI Pre-Processing, http://adni.loni.usc.edu/methods/mri-tool/mri-pre-processing/; Status: 10.11.2019

60 *Parsania PS, Virparia PV*. A Comparative Analysis of Image Interpolation Algorithms. International Journal of Advanced Research in Computer and Communication Engineering 2016; 5: 29 – 34
61 Python Release Python 3.7.4, https://www.python.org/downloads/release/python-374/; Status: 26.11.2019
62 Anaconda: Computer software: Anaconda Software Distribution, 2016
63 *Chollet F, others*. Keras, 2015
64 *Mart\'ın Abadi, Ashish Agarwal, Paul Barham et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015, http://tensorflow.org/
65 *Brett M, Markiewicz CJ, Hanke M et al.* Nibabel: 2.5.0, 2019, https://doi.org/10.5281/zenodo.3360650
66 *van der Walt S, Colbert SC, Varoquaux G*. The NumPy Array: A Structure for Efficient Numerical Computation. Comput. Sci. Eng. 2011; 13: 22 – 30
67 *Pedregosa F, Varoquaux G, Gramfort A et al.* Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 2011: 2825 – 2830
68 *Hunter JD*. Matplotlib: A 2D Graphics Environment. Comput. Sci. Eng. 2007; 9: 90 – 95
69 *Bradski G*. The OpenCV Library. Dr. Dobb's Journal of Software Tools 2000
70 *da Costa-Luis CO*. tqdm: A Fast, Extensible Progress Meter for Python and CLI. JOSS 2019; 4: 1277
71 *Bergstra J, Bengio Y*. Random search for hyper-parameter optimization. Journal of Machine Learning Research 2012; 13: 281 – 305
72 *Shorten C, Khoshgoftaar TM*. A survey on Image Data Augmentation for Deep Learning. J Big Data 2019; 6: 60
73 *Wen J, Thibeau-Sutre E, Diaz-Melo M et al.* Convolutional Neural Networks for Classification of Alzheimer's Disease: Overview and Reproducible Evaluation, 2019
74 *Lau L-F, Brodney MA, Berg S*. Alzheimer's disease. Berlin, Heidelberg: Springer, 2008
75 *Basheer N, Aaref A, Ayyed D*. Digital Image Sobel Edge Detection Using FPGA 2015
76 *Raghu M, Zhang C, Kleinberg J et al.* Transfusion: Understanding Transfer Learning for Medical Imaging, 2019
77 *Regina Felix*. Nicht-Normalbefunds-Warnung in der Radiologie – Konzeptionierung einer technischen Lösung: Hochschule Landshut, 2017

10 Appendix

10.1 Appendix A

Table 4. RANZCR 2016 MR descriptors and SATs

Consensus descriptor	Consensus SAT (mins)	BSF system	Descriptor in BSF used as equivalent	Includes	Excludes	Notes
MR brain	18	CNS/H&N	Brain NOS	Skull, pituitary, cranial nerves, COW MRA if included		
MR face/orbits/sinuses/ temporal bones	18	CNS/H&N	Orbits	Temporal bones, sinuses, facial bones	Skull	
MR internal auditory meati	10	CNS/H&N	Internal auditory meati			
MR pituitary	14	CNS/H&N	Pituitary gland			
MR neck	20		Entire neck	Nasopharynx, oropharynx, tongue, larynx, skull base		
MR cervical spine	18	MSK	Cervical spine	Cervical cord		
MR thoracic spine	15	MSK	Thoracic spine	Thoracic cord		
MR lumbar spine	15	MSK	Lumbar spine with/without sacrum	Cauda equina/lumbar nerve roots		
MR spine any 2 of above 3	18	n/a	n/a			
MR spine all of above 3	24	n/a	n/a			
MR hip	16	MSK	Hip joint			
MR knee	16	MSK	Knee joint			
MR ankle or foot	18	MSK	Ankle joint	Forefoot, hindfoot		
MR shoulder	16	MSK	Shoulder joint			
MR elbow	16	MSK	Elbow joint			
MR wrist/hand	18	MSK	Wrist/distal radioulnar joint	Hand (W or WO fingers)		
MR TMJs	14	CNS/H&N	Temporomandibular joints			
MR brachial plexus	20	CNS/H&N	Brachial plexus			
MR breasts	35	OBS&breast	Breasts	One or both, screening and staging		
MR stereo fiducials	8	CNS/H&N	Stereotactic fiducial target localization for surgical guidance			
MR thoracic body wall	15	MSK	Chest wall	Ribs, sternum, thorax		
MR heart	20	Visceral thorax	Chambers and valves with/ without myocardium		Flow quantitation	
MR heart with flow quantitation	Use own data	Visceral thorax	Chambers and valves with/ without myocardium			
MRCP	12	Visceral abdopelvis	MRCP stand alone		Multiphase liver	
MR upper abdomen	15	Visceral abdopelvis	Upper abdomen	Adrenals, kidneys	Multiphase liver	
MR multiphase liver	26	Visceral abdopelvis	Liver dedicated	Primovist liver, multiphase pancreas		
MR whole abdopelvis	20	Visceral abdopelvis	Abdomen and pelvis		Small bowel	
MR bowel	24	Visceral abdopelvis	Small bowel	Conography dedicated		
MR pelvis	15	Visceral abdopelvis	Whole pelvis nos	Male pelvis, female pelvis	Prostate dedicated	
MR prostate	25	Visceral abdopelvis	Prostate with/out endorectal coil		Pelvis	Without coil set up time
MR rectum	25	Visceral abdopelvis	Rectum and anal canal dedicated	Anal canal for fistula		
MR angiography abdomen	20	Visceral abdopelvis	Renal arteries	Visceral arteries, thoracoabdominal aorta		
MR angiography thorax	15	Visceral thorax	Thoracic aorta	Pulmonary arteries and veins		
MR spectroscopy (any area)	20					
MR NOS, any protocol, one structure	15					

Figure 32. The average time expended by a radiologist for the complete diagnosis of MRI scans by body part [6]

10.2 Appendix B

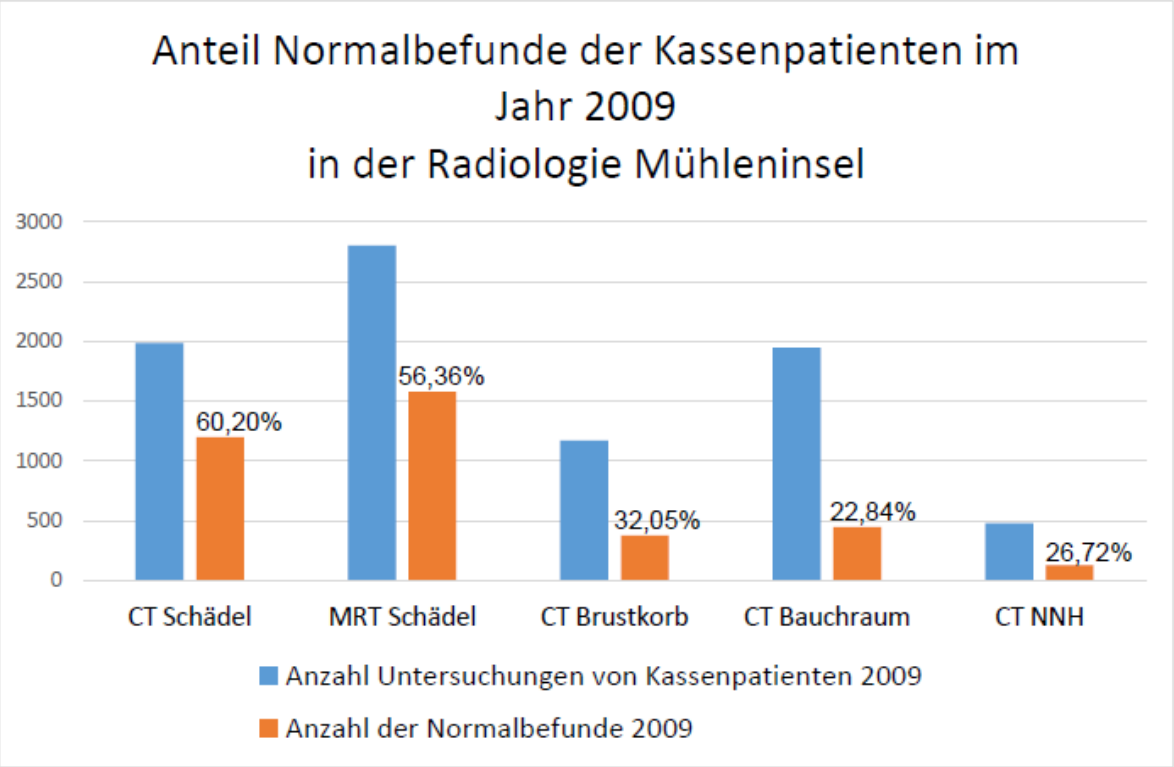


Figure 33. The percentage of normal findings per body part of the Radiologie Mühleninsel in 2009. [77]

10.3 Appendix C: List of Figures

10.4 Appendix D: List of Tables