Budapest University of Technology and Economics

Measurement of typical MR imaging artifacts with convolutional methods

Master's thesis

Anna Siroki (L3C651)
Supervisor: Dr. Regina Deák-Meszlényi
Co-supervisor: Dr. Dávid Legrády

18. 01. 2019.

Contents

Contents **1**

1 Introduction **3**
1.1 Motivation . . . 3
1.2 MRIQC:Advancing the automatic prediction of image quality in MRI from unseen sites . . . 5

2 MRI Artifacts **7**
2.1 Motion-related artifacts . . . 7
2.2 Aliasing . . . 10
2.3 Moiré fringes (zebra artifact) . . . 10
2.4 Gibbs Artifact . . . 11

3 Convolutional Neural Networks **13**
3.1 Introduction . . . 13
3.1.1 Linear Classifier . . . 13
3.1.2 Neural Networks . . . 13
3.1.3 Convolution kernels . . . 15
3.2 Training convolutional neural networks . . . 15
3.2.1 Image preprocessing . . . 16
3.2.2 Convolutional neural network architecture . . . 17
3.2.3 Model training . . . 20
3.2.4 Model evaluation . . . 21

4 MRIQC with Convolutional Neural Networks **25**
4.1 Data . . . 25
4.1.1 The structure of the data: BIDS format . . . 26
4.2 Data preprocessing . . . 28
4.2.1 Rescale . . . 28
4.2.2 Crop to desired image dimensions . . . 28
4.2.3 Standardize . . . 29
4.2.4 Augmentation . . . 29
4.3 Models . . . 29
4.4 Best performing model evaluation on test . . . 33
4.4.1 Comparison of convolutional and random forests classifier based MRIQC tool . . . 33
4.4.2 Investigation of convolution kernels in CNN . . . 34
4.4.3 Activation after convolution kernels on a classified true positive image . . . 34

4.4.4 Activation of convolutional layers on images with artifacts . . 35
4.4.5 Orientation issue 44
4.5 Software support 46

5 Conclusion **47**
5.1 Error analysis 47
5.2 Summary 48
5.3 Acknowledgement 49

Bibliography **50**

Chapter 1

Introduction

1.1 Motivation

MR (Magnetic resonance) imaging artifacts can cause huge problems in the medical procedure. On the one hand the doctor can hardly or misdiagnose the disease of the patient. On the other hand it is terribly inconvenient for the patients. Most of us are familiar with the difficulties of being in a hospital: waiting, one is sent from one doctor to another, from one examination room to another. Repeating an examination can cause more of these undiserable experiences and stress. Despite of this, it still happens quite often. It would be great to have a software which is able to identify immediately (or with minimal preprocessing time) the artifacts and to suggests a modification of scanning parameters or environmental changes to get the best image possible.

This thesis would like to solve the problem above. My task was to create a convolutional neural network model to evaluate the 3D MRI images. The work contained study of literature, I understood the basics of deep learning algorithms and learned how to build different architectures and to evaluate them. I acquired experience in differentiating MRI artifacts that were shown on the images from the public datasets and understood their physical background. I gained knowledge in the process of building three dimensional convolutional networks. In every step I introduced new techniques, that could optimize my algorithm. Finally, I examined the artifacts, which were recognized and not recognized by the deep neural network.

The thesis starts with an introduction, which contains a brief review of the most important publication [1] that inspired the topic of this thesis. In the next chapter, I sum up the most common artifacts in MR imaging. Then I write about Convolutional Neural Networks (CNNs) in general. My algorithm is presented under the title "MRIQC with Convolutional Neural Networks". It contains information

about the data, data handling, my model and results. The latter contains numerical measures of the performance of the CNN. I compare the MRIQC tool of [1] and my quality control tool to evaluate their performance. I analize the output channels of deeper convolutional layers and their connections with the artifacts. In the end I give a conclusion and outlook.

1.2 MRIQC:Advancing the automatic prediction of image quality in MRI from unseen sites

The following section is based on Ref. [1]

An automated quality control of MR images is required because the current method of quality control is not the most efficient and also not the most advantageous. On the one hand there are some problems with quality assessment that appear because the work is done by people who have different educational background and mistakes are also made because of the lack of concentration due to stress and tiredness. These effects cause inter-rater variability. On the other hand quality control procedure has a perspective which can not be done by humans. For example, the appropriate acquisition parameters are not detected, and the different quality issues caused by the variability of sites are also not easily noticeable. This work would like to improve the machine learning based automatic prediction of the image quality on unseen sites.

MRI Quality Control tool makes the following steps:

1. Collects, organizes, labels the data
2. Creates an image preprocessing workflow
3. Calculates the needed Image Quality Metrics
4. Creates visual assessment
5. Trains and evaluates more models
6. Tests the best performing model on a held-out dataset (with unseen sites)

ABIDE (Autism Brain Imaging Data Exchange) dataset was used for training and cross-validating. DS000030 dataset was used as a held-out dataset to test on unseen sites. The data was organized into BIDS (Brain Imaging Data Structure) file system. The labelling protocol was done by two experts. In the training set there were 100 (out of 1102) images that were labelled by both, the inter-rater variability was investigated based on these. [2] Possible label options were: "doubtful", "accept" and "exclude" (later it was binarized into two classes: "accept" and "exclude").

The preprocessing workflow can be seen on Figure 1.1. The preprocessing also contains site-wise normalization (including scaling and centering) due to the batch effect that was caused by inter-site variability.

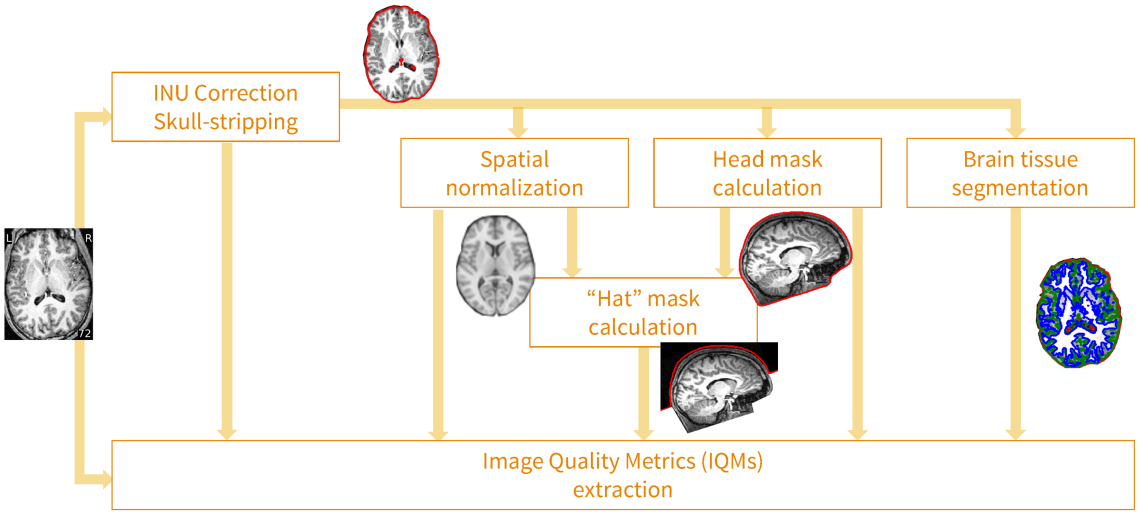


Figure 1.1: MRIQC's processing data flow to compute IQMs. [1] The input images were centered and scaled site-wise before the preprocessing workflow.

Afterwards Image Quality Metrics (IQMs) were calculated. The model's prediction is based on 64 IQMs as features. These 64 quality metrics were selected based on previous studies [3] [4].

The IQMs were used as an input to a supervised machine learning classification algorithm. Two models were evaluated: support-vector machine classifier (SVC) and random forest classifier (RFC). The metrics of the evaluation was an objective function called "area under the precision-recall curve" (PR-AUC) and F1-score respect for the imbalanced labels. The scientists used leave-one-site-out (LoSo) cross-validation for model selection. With this technique, they left one site out as a validation set at each cross-validation fold. The classifier was tested on a held-out dataset (on an unseen site).

The results of the study is presented in Chapter 4 with the evaluation of the convolutional method.

Chapter 2

MRI Artifacts

In this chapter I present the most common artifacts in brain MR images. They are the followings: motion-related artifacts, moiré fringes (zebra), aliasing, Gibbs artifact. The example images are from the datasets I used for the convolutional deep learning model.

We can categorize the MRI artifacts based on their source. In this aspect we get three groups: hardware-related (e.g. zebra artifact, zippers), sequence-related (e.g. aliasing, Gibbs) and patient-related artifacts (e.g. motion, susceptibility).

2.1 Motion-related artifacts

In case of motion artifact the source of the error is the patient, the periodic (e.g. cardiac, respiratory) or nonperiodic (e.g. eye motion, head) motion of the patient. It is produced because the position of the signal changes. This artifact mainly effects the phase-encoding direction. [5]

Most common issues that specify the effect of the motion on the image: the k-space ordering and the type of the motion. A continuous slow motion simulation can be seen on Figure 2.1. Sudden orientation change simulations can be observed on Figure 2.2. In latter case, the different rows correspond to different k-space ordering with different amount of inconsistent data. [6]

On these simulations it can be clearly seen that k-space ordering influences the most the appearance of the motion artifact. We can also conclude that depending on the k-space ordering, the type of the motion is also decisive.

An example of motion artifact from ABIDE dataset is seen on Figure 2.3.

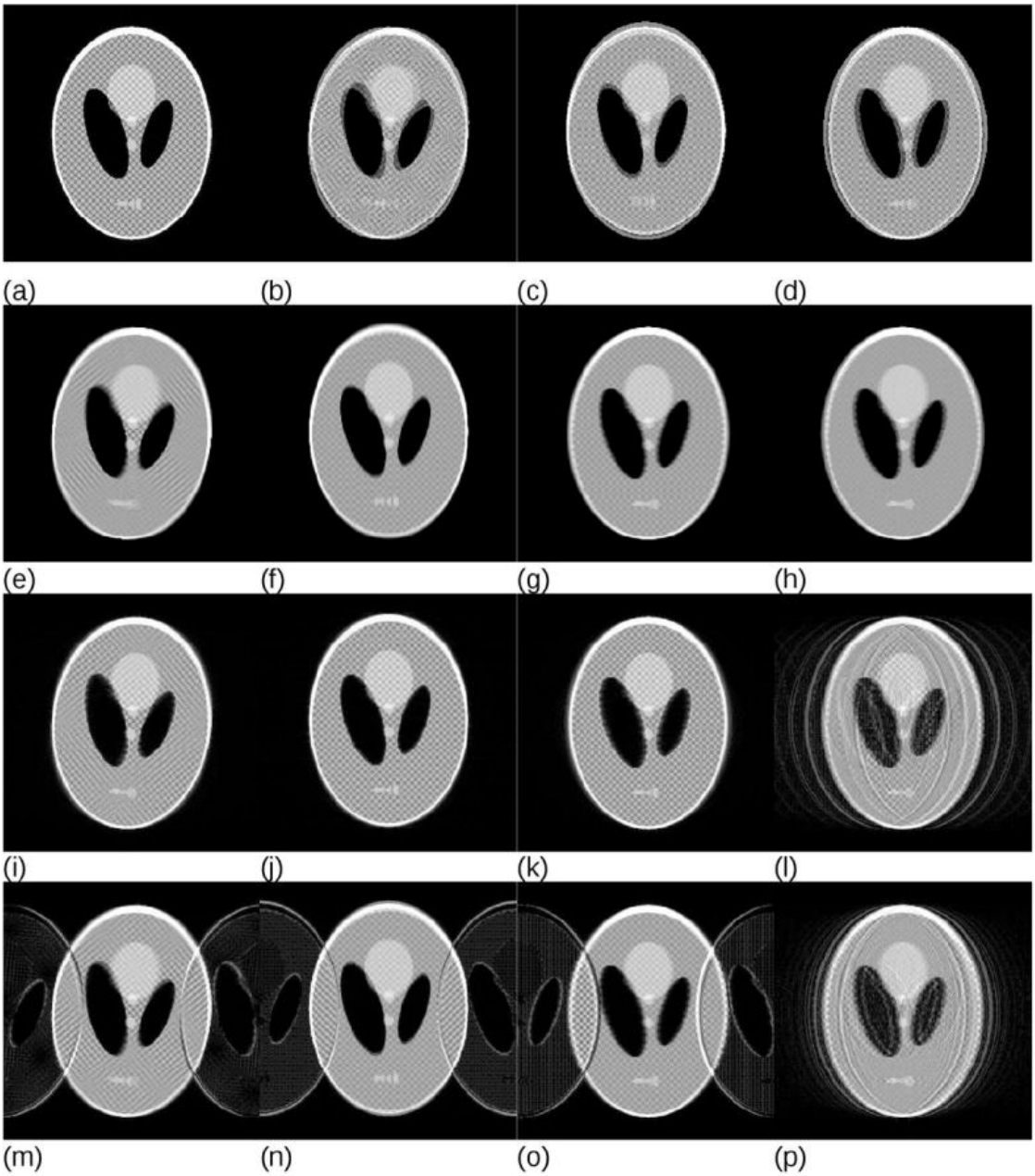


Figure 2.1: (a) the original reconstruction without motion, (b-d) sum of first and last image of the motion-simulation: (b) rotation, (c) vertical translation, (d) horizontal translation, (e-h) averaging the motion in image-space (photography-like images with equivalently long exposure), (i-l) MRI acquisitions with linear k-space ordering, (m-p) MRI acquisitions with (two-shot) interleaved k-space ordering [6]

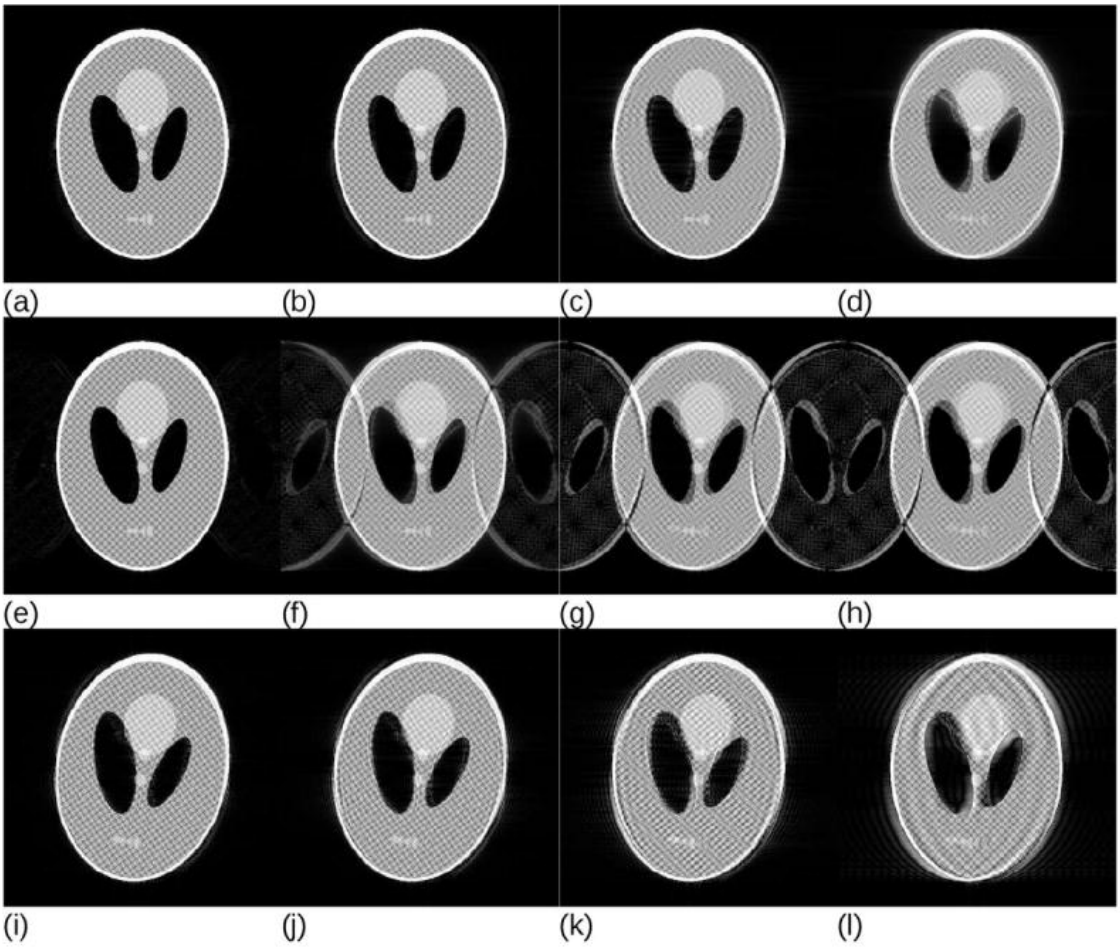


Figure 2.2: Sudden orientation change. (a-d) linear, (e-h) (two-shot) interleaved, (i-l) centric k-space ordering. The amount of inconsistent data are the following: (a)(e) 12.5%, (b)(f) 25%, (c)(g) 37.5%, (d)(h)(i) 50%, (j) 62.5%, (k) 75%, (l) 87.5% [6]

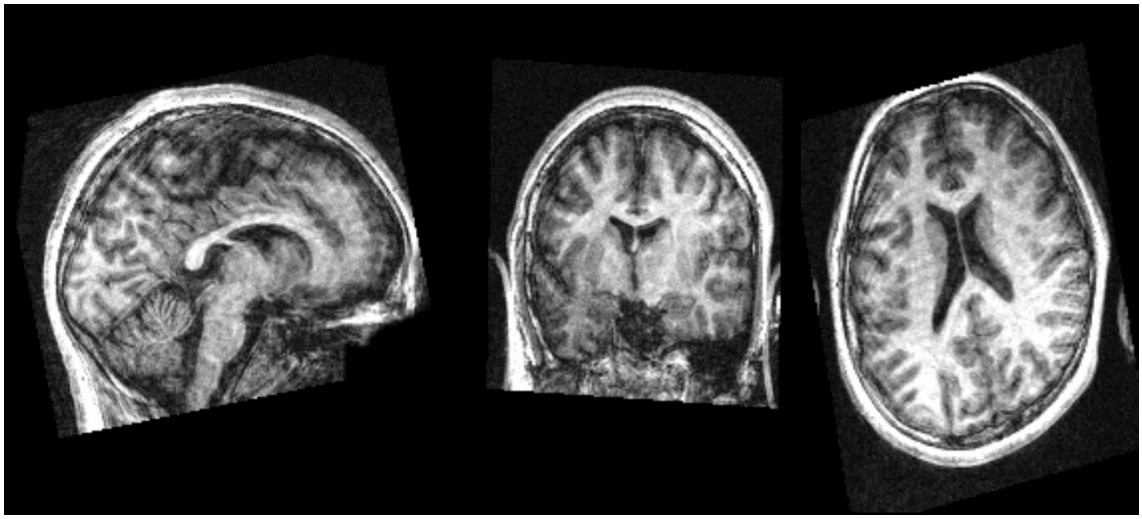


Figure 2.3: Motion artifact from ABIDE dataset. [7]

2.2 Aliasing

Due to discrete and finite sampling we have a criterion between the sample size (A) and the k-space step (Δk), this is the so-called *Nyquist Criterion*:

$$L < \frac{1}{\Delta k}. \tag{2.1}$$

We define the field-of-view (FOV) as:

$$\mathrm{FOV} \coloneqq \frac{1}{\Delta k}. \tag{2.2}$$

In case of aliasing artifact, an inadequate L is selected. This cause overlapping in image-space. The measurement of the signal is in k-space. In this space the measurement of physical density is repeated with the periodicity of $x + L$. When Δk spacing is not small enough, thus the FOV is not big enough, the object overlaps with itself and appears on the other side of the image. An example is shown on Figure 2.4. [8]

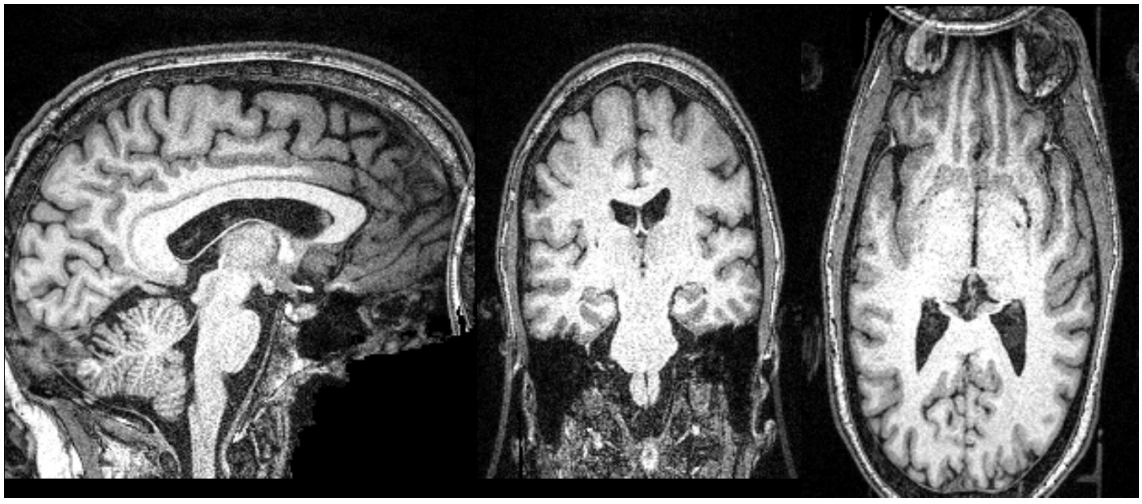


Figure 2.4: Aliasing artifact from DS000030 dataset. [9]

2.3 Moiré fringes (zebra artifact)

Moiré fringes or zebra artifact is present because of the interference of aliasing artifact and field inhomogeneity in gradient echo imaging. This type of MR imaging is very sensitive to phase shift. When a coil is not perfectly centered, these zebra stripes appear. The signal from different voxels alternately add and cancel, as a result aliasing bands show up. [10] Figure 2.5. shows an example.

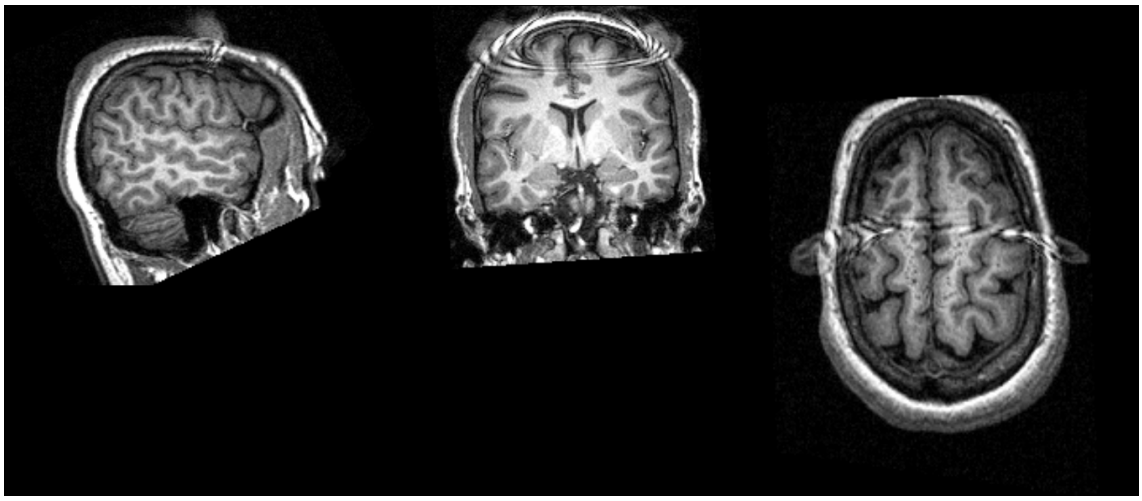


Figure 2.5: Moiré fringes artifact from ABIDE dataset. [7]

2.4 Gibbs Artifact

Gibbs artifact (also known as ringing artifact) is an alternation of brighter and darker bands, which are present on high-contrast surfaces. It is formed as a consequence of the Fourier transformation and discrete sampling.

MR imaging collects signal in k-space, sampling with finite number of spatial frequencies. During the calculation of Fourier transform, we assume that the measurement took infinite time, of course we are not able to do that, so a rectangular function is convolved on the image and it cuts off the high frequency data. The Fourier transform of the rectangular function is an oscillating and slowly decaying function. This oscillation appears on image-space as a ringing artifact. An example and an explanation image is shown on Figure 2.6. [8]

Gibbs artifact is visible on some images of the dataset, though it is rated as "accept", because this small error not affects greatly the diagnostic procedure.

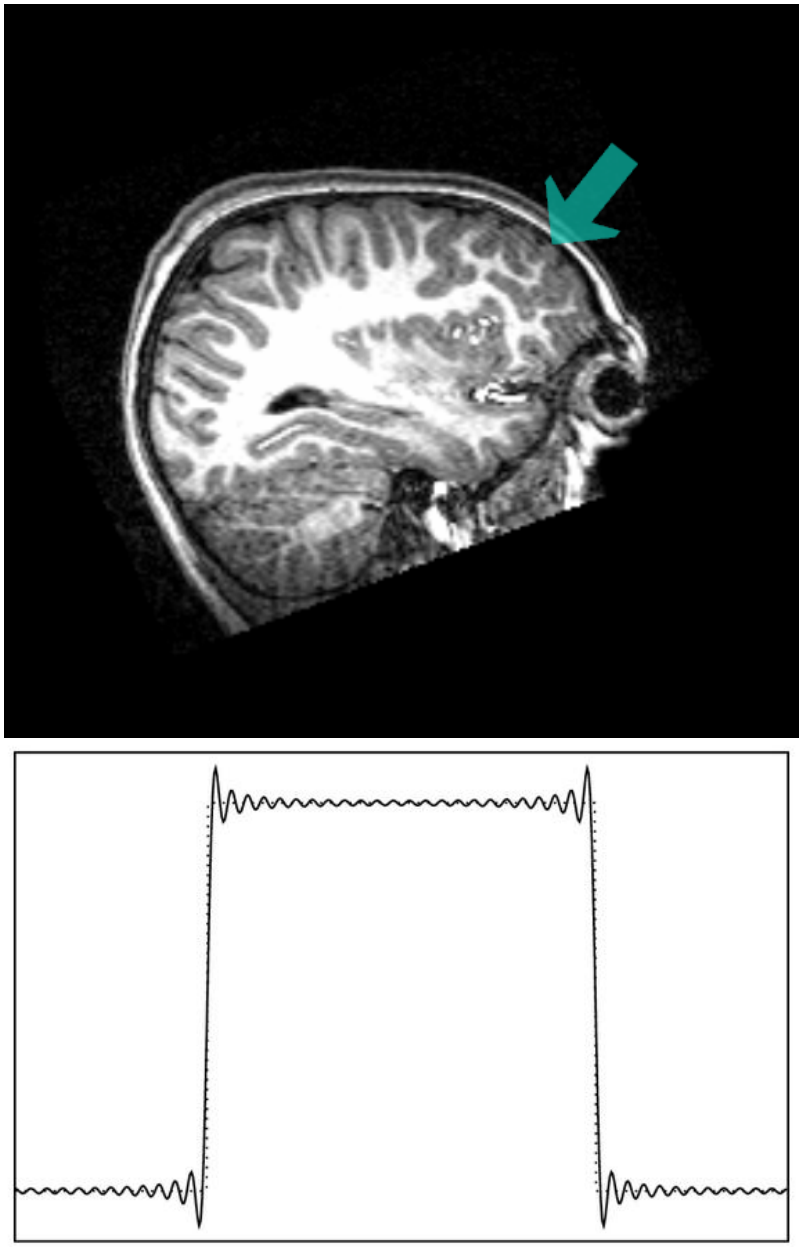


Figure 2.6: Example of MR image with Gibbs artefact above from ABIDE dataset. [7] Below a Fourier reconstruction of a rectangle wave from 96 samples. The real signal should be flat, but the small difference is because of finite sampling. The same periodic alteration can be observed in this one-dimensional example, as in case of ringing artefact. [11]

Chapter 3

Convolutional Neural Networks

3.1 Introduction

In this section I go through the basic definitions and behaviour of neural networks and I write about a classical image procession technique: convolution kernels.

3.1.1 Linear Classifier

A linear classifier (e.g. SVM - Support Vector Machine) classifies using the function below:

$$f(x_i, \mathrm{W}, b) = \mathrm{W} \cdot x_i + b, \tag{3.1}$$

where x_i are the input values (x can be a vector or a matrix - in case of images), W is the weight matrix which has a dimension of $x \times$ class labels (when x is a matrix, it should be flattened to a vector to get the dimensions well). b means the bias, and it has the size of class labels. [12]

3.1.2 Neural Networks

Neural networks are nonlinear classifiers. The first applied step is similar to linear classifier, but after calculating the function value, neural networks apply a nonlinearity. Figure 3.1. shows us how a neuron works.

As in linear classifiers x_i means the input values. The input value is multiplied by weights w_i and the neuron sums all the multiplied values that comes from the inputs. Beside this sum, every neuron has a bias b that is added to the sum. Until this point, the neuron works as a linear classifier. After the sum a nonlinear activation function is applied, thereby nonlinearity is brought to the system. The output is the value, which the neuron gives after the activation function.

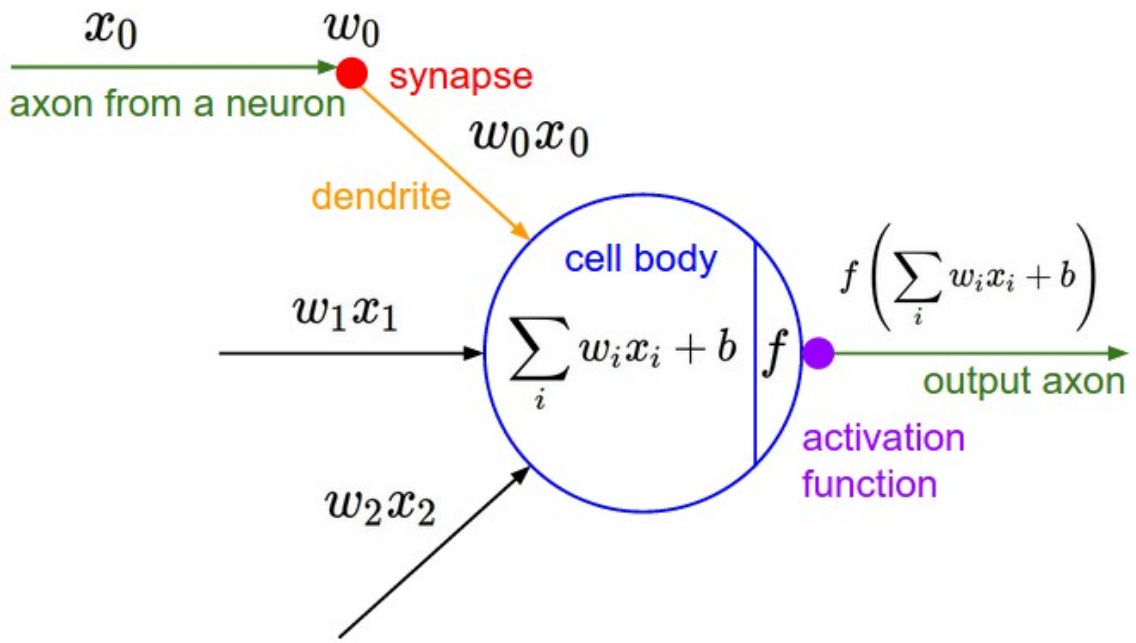


Figure 3.1: The mathematical model of an artificial neuron. [13] The colour signs describe parts of a biological neuron to explain the identification.

We can apply as many neurons as we want to solve a classification problem. It is also possible to use the output of a neuron as an input for another neuron and this is a new layer of the neural network. The neurons that get the input from another neuron called "hidden neurons" the layer that is built called "hidden layer". A network architecture can be seen on Figure 3.2. The network is called "deep" when two or more hidden layers are in the architecture. [13]

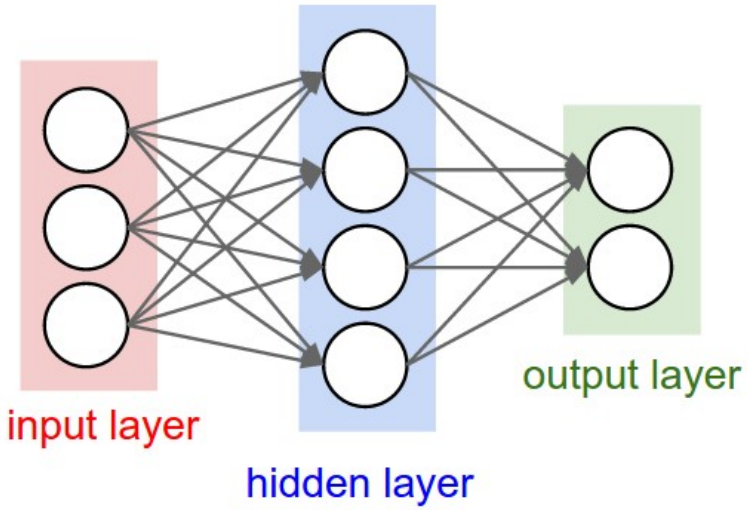


Figure 3.2: Neural network architecture. [13]

3.1.3 Convolution kernels

In this subsection I present a widely known image processing technique: convolution kernels. [14]

Convolution kernels are matrices that are applied to make visible changes on images. The aim of these changes can be: noise reduction, edge detection, sharpen etc. The main idea is to apply a kernel (matrix), multiply by a part of the original image, and it gives an output number, which can be considered as a pixel value. One can go through the whole original image to calculate these values and get another image. I show an example of the calculation on Figure 3.3. The applied filter's matrix is the following:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.2}$$

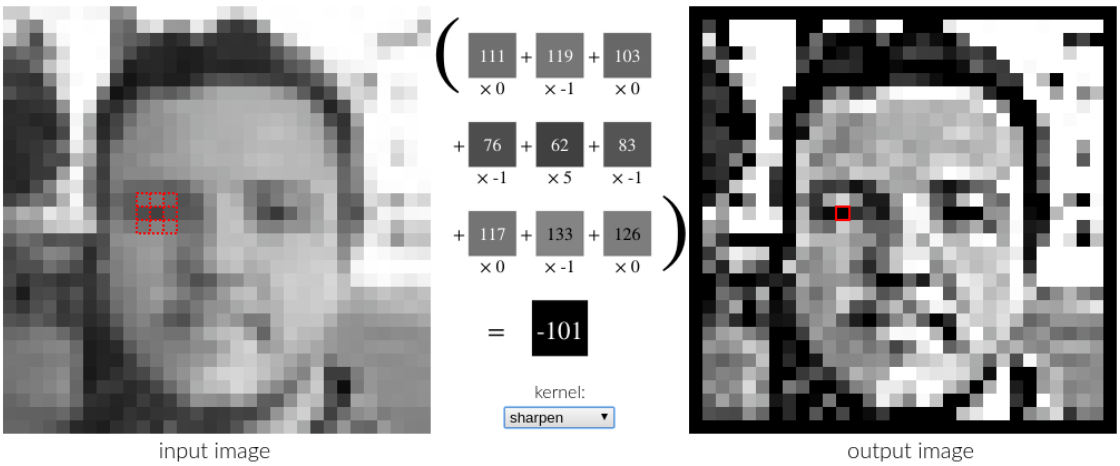


Figure 3.3: Sharpening kernel used on an example image to show how convolution kernels work [14]

In Chapter 4 I analyze the convolutional network weights, which can also be considered as convolution kernels.

3.2 Training convolutional neural networks

In the following section I go through the steps of training a convolutional neural network for image classification. This contains the followings:

1. Image preprocessing
2. Convolutional neural network architecture

3. Model training

4. Model evaluation

3.2.1 Image preprocessing

Data split

Data split is one of the heart processing in machine learning. When we have enough "good" data, we should split it into three parts: training, validation and test set. Training set is used for training, the model is fitted on this dataset. Validation set is used to provide evaluation during training, when tuning the model's hyper-parameters. The test dataset is used to evaluate the performance of the final model and to compare it with other algorithms. [15]

Data standardization

A data standardization method in neaural networks is needed to speed up the learning phase. [16] The first step of standardization is to zero-center the data. Zero-centering is implemented by subtracting the mean pixel value from each pixel value. The second step of standardization is normalization. After the data is zero-centered, it is desirable to bring all the data to the same scale. It is implemented in two ways: one divides the values by the standard deviation, the other divides by the maximum absolute pixel value. A process explanation with plots can be observed on Figure 3.4.

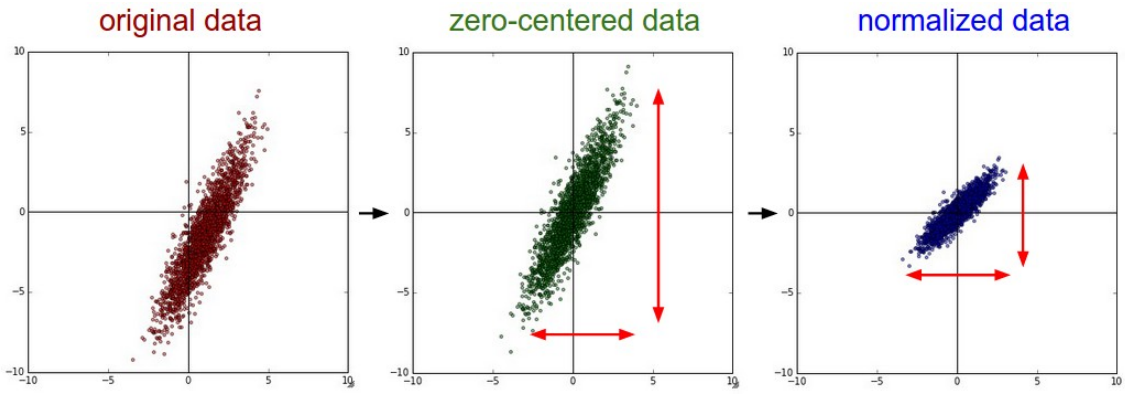


Figure 3.4: Data standardization. On the first image, we see the original 2-dimensional data. On the second, the zero-centered data. On the third image the data is also normalized, each value is divided by its standard deviation. [17]

In image classification problems each image is an individual sequence of data. Thus it is possible to calculate mean and maximum absolute pixel value for each image separately. [18]

Data augmentation

Data augmentation is widely used to improve performance and to prevent overfitting in deep learning models. Augmentation techniques are applied only on training dataset.

There are various types of data augmentation techniques, nevertheless in the following, I focus on classical augmentation on images. [19] This technique involves image rotating, horizontal/vertical shifting, zooming, horizontal/vertical flipping. [20] In brain images rotating, zooming and shifting are convenient transformations. Flipping the brain does not come out with the same result, because the difference between the two cerebral hemispheres are not interchangeable.

Another way to augment mostly imbalanced data (unequal number of data with different labels in the dataset) is oversampling. This means for example that we have a minority class of 0 labels, we double this data, in this way the dataset could become balanced. [21]

3.2.2 Convolutional neural network architecture

Convolutional Neural Networks (CNNs) are widely used for image processing. A schematic architecture is illustrated on Figure 3.5.

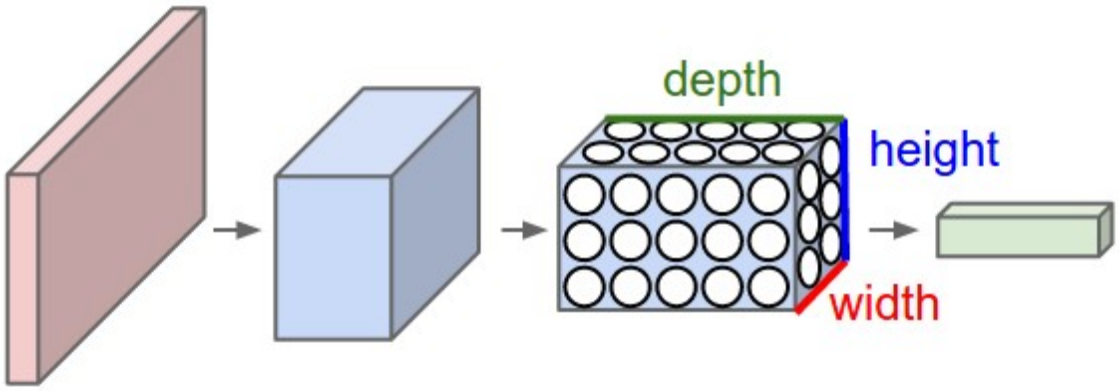


Figure 3.5: Schematic architecture of a convolutional neural network. [22] At the input layer height and width means the resolution of the image. Depth represents the different channels that are created with the convolution.

Three different layers are widely used in a CNN: convolutional layer, pooling layer and fully connected layer. I describe these architectural elements below.

Convolutional layer creates local connectivity, which is done by applying different filters on the image and that gives us another image. The same happens with images after applying convolution kernels. The values of the convolutional kernel are considered as weights and the parameter update changes the effect of the filter as

it changes the weights. In each layer we apply more filters to save the image information in a different form. We denote the number of resulting images with *channel* number.

The operation of a convolutional layer can be seen on Figure 3.6. and described with Equation (3.1). After the operation showed on the picture an activation layer is applied.

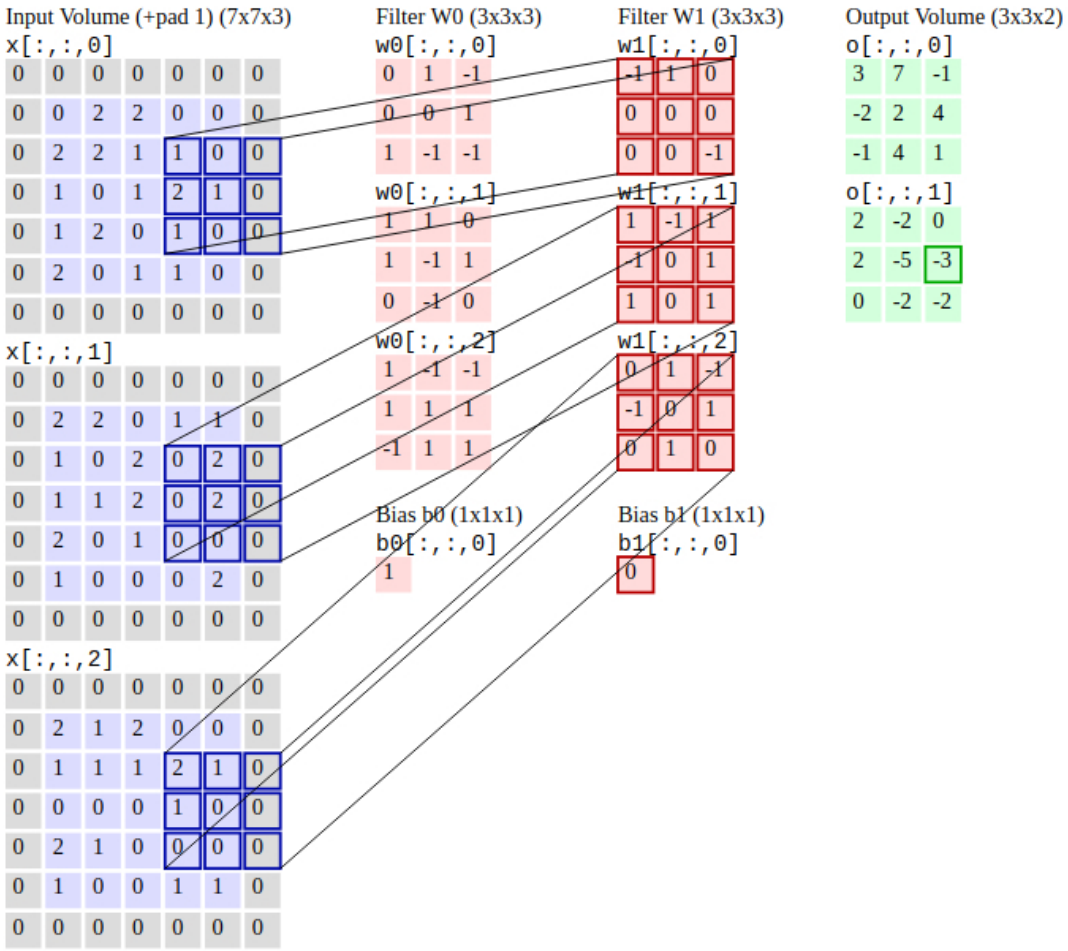


Figure 3.6: Operation of convolutional layers [22]. The input volume represents a 2D image with the resolution of (5x5) with three-colour channel and it uses two filters for each images.

Separable convolution implements convolutional kernel with more one-dimensional kernels. In case of three-dimensional images three different kernels are used to reduce the original kernel-size. An example: [3x3x3] kernel size was applied on the input image, but with spatial separable convolution we apply a [3x1x1], a [1x3x1] and a [1x1x3] kernel. We have to train 27 weights with classical convolutional method, but with the latter technique just 9 trainable parameters remain. Previous works shows, that separable convolution achieves similar or better performance in image classification. [23] [24]

Pooling layer is applied mostly to reduce the resolution of the images after convolution. We apply the filters to get different features from the images, but at the mean time we have to reduce the spatial size. With these layers, we gain to converge to the size of the output volume. The pooling layer could be average and maximal. Maxpooling layer is widely used in literature, because of its fast convergence and select better the superior invariant features. [25] [26] [27] [28]

After all convolutional and pooling layers we have to apply a *fully connected layer*, which makes a one dimensional layer. This is one standard procedure that can be used when the output of the classification is not an image. (For example when we would like to make a segmentation we do not have to apply a fully connected layer because the output is also two-dimensional.) In practice this means, that we have to flatten all the weights to get one vector. This is what the green volume symbolize on Figure 3.5.

There are several tools in deep learning to prevent overfitting on the train dataset. One, which is used in model architecture, is *dropout*. In implementation it means, that we "drop out" the output of randomly selected neurons in a neural network model. This is a commonly used regularization technique, because of its computational effectiveness. With dropout technically we try large amount of models without tuning all of them. [29]

Spatial dropout is also a regularization technique in image classification. In contrast with standard dropout, spatial dropout do not choose to drop out the neurons output independently, it drops out full channel planes. [30]

We apply *activation functions* after each layer to insure the nonlinearity of the model, as I wrote about in Section 3.1.2.

Sigmoid activation function applies the following function:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{3.3}$$

This is a frequently used activation function. [13]

Relu (Rectified Linear Units) is an activation that applies the function below:

$$f(x) = \max(0, x). \tag{3.4}$$

The function is plotted on Figure 3.7. This function is widely used in deep convolutional neural networks because of it's low computing capacity and fast convergence. [19]

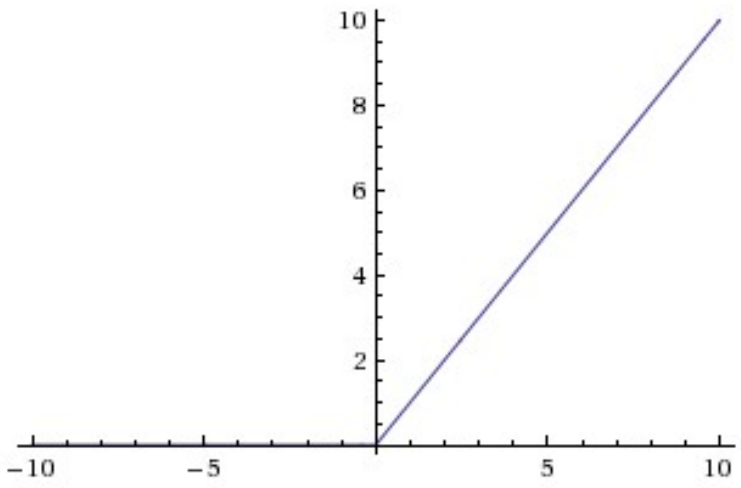


Figure 3.7: Relu function [13]

Softmax function is used after the logit (not normalized scores) in a deep learning model. It is calculated with the following formula:

$$f(x_i) = \frac{e_i^x}{\sum_j e^{x_j}}, \tag{3.5}$$

where x_i is one individual logit value. This activation function is widely used in the last layer, because its output from the different neurons sums to one. In this way the higher activation gives us the class, that the network predicts. [31]

3.2.3 Model training

In this section I will go through the process of training a neural network model. I present weight initialization and the optimization of weight updates.

The first step of training is weight initialization. The default initializer of convolutional and fully connected layers is *Glorot initializer*. With this technique the initial weights are 'normalized' and they are set with Gaussian distribution with the variance based on the input and the output units of the weight tensor. [32] The formula is below:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right], \tag{3.6}$$

where n_j is the number of input units, n_{j+1} is the number of output units. [33]

The main aim of the training is to get as many labels predicted the same value as the true value. We have to introduce a metric, that refers to the difference between predictions and true labels, and this is *loss function* (also called *cost function*). Categorical cross-entropy is widely used in classification problems, its calculation is

below:

$$\mathcal{L} = \frac{1}{N}\sum_{i}\mathcal{L}_i = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log \hat{y_{i,j}}, \tag{3.7}$$

where $\mathcal{L}$ is the categorical cross-entropy for all the data, N is the number of datapoints, $\mathcal{L}_i$ is the loss for a datapoint,i corresponds to sample points, C is the number of classes, j corresponds to the class, $y_{i,j}$ is the true label of sample and $\hat{y_{i,j}}$ is the predicted label. [34]

Our aim during the training is to minimize the loss function, and a common practice to do so in numerical methods, is gradient descent. In neural networks, this method is called *backpropagation*. This algorithm uses the derivative of the cost function to update the weights. For the update, we have to give a hyperparameter, the so-called *learning rate*. A simple way to calculate the new weights:

$$\omega^{t+1} = \omega t - \lambda \cdot \delta\omega, \tag{3.8}$$

where ω is a weight, t corresponds to the current value, $t+1$ is the value in the next iteration, λ is the learning rate and $\delta\omega$ is the derivative of the cost function, that is calculated with backpropagation.

There are different optimization algorithms for weight updates that are used in deep learning. *Adam algorithm* is beneficial, because of its efficiency in computation, fast convergence, with noisy or sparse gradients and with non-stationary problems. Adam classifier calculates learning rates individually for each parameters, counting in the average of gradient (first momentum) and the squared gradient (second momentum). [35]

3.2.4 Model evaluation

A criterion to train a network is to have enough data. We need to have a huge amount of samples to induce better performance. This is why we always go through the data more times. This hyperparameter called *epoch* shows us how many times we went through the training set.

During training, one has to be cautious of *overfitting*. As all samples contain noise, we should train our network to not to learn the noise. Overfitting occurs when the model is fitted also to the noise and its predictive power decreases. We use different techniques to prevent that. One is that I mentioned before, dropout. Another is to validate the model after each epoch. When we use particular training and validation set, we monitor the model performance on the validation.

An other way to reduce overfitting is to set *early stopping* in the model. This

function decides, if the model performs better on the validation set or not. If it does, the new weights are saved, if not, we can decide how "patient" we are. Patience is a measure that indicates how many more epochs should the network still learn without a better evaluation score. The background thought is, that we can overfit on the training set, but that causes worse performance on the validation set. Patience is needed, because it can happen, that the model finds a local minimum on the surface of cost function, but we would like to find the global minimum. The loss function of an overfitted model can be observed on Figure 3.8.

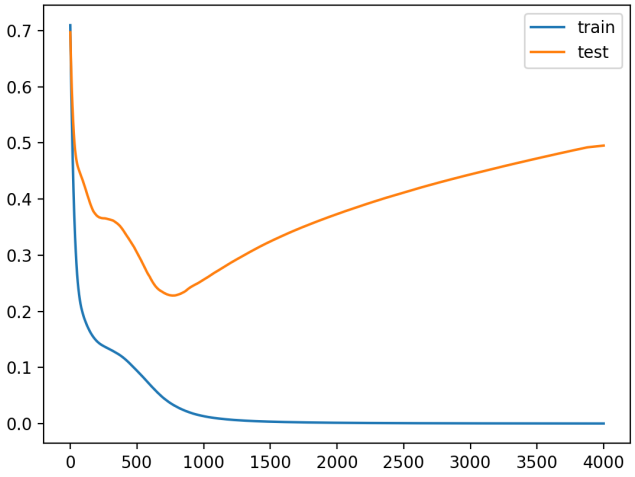


Figure 3.8: Plot of train and test dataset during training on an overfitted model. X-axis corresponds to the epochs, y-axis to the loss function. [36]

The evaluation of the training is done on the validation dataset. During training, the best model is selected based on its performance on the validation set. We can choose different metrics to select the best model. I present the evaluation metrics below.

Evaluation metrics

ACC: Accuracy is calculated with the following expression:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.9}$$

where TP means true positive (predicted and labelled as positive), TN means true negative (predicted and labelled as negative), FP means false positive (predicted as positive, but the true label was negative) and FN means false negative (predicted as negative, true labelled as positive) samples. Basically it calculates the sum of true positive and negative samples over all datapoints.

Loss function: It is the same, that I defined before, with Eq. 3.7.

Precision: the definition of precision is the following:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{3.10}$$

where TP means true positive, FP means false positive samples on the evaluated database. [37]

Recall: (also called sensitivity or true positive rate) the definition can be described with the following equation:

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3.11}$$

where TP means true positive, FN means the false negative samples on the evaluated database. [37]

F1 score: This measure takes both precision and recall into account. The definition of F-score is below:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.12}$$

It calculates the harmonic mean of precision and recall. [37]

PR-AUC: (Area Under Precision-Recall Curve) We can plot a precision-recall curve which shows us the proportion of true positive samples to the false negatives and false positives. The model is better if the area under the curve is maximal. A precision-recall curve is shown on Figure 3.9. [37]

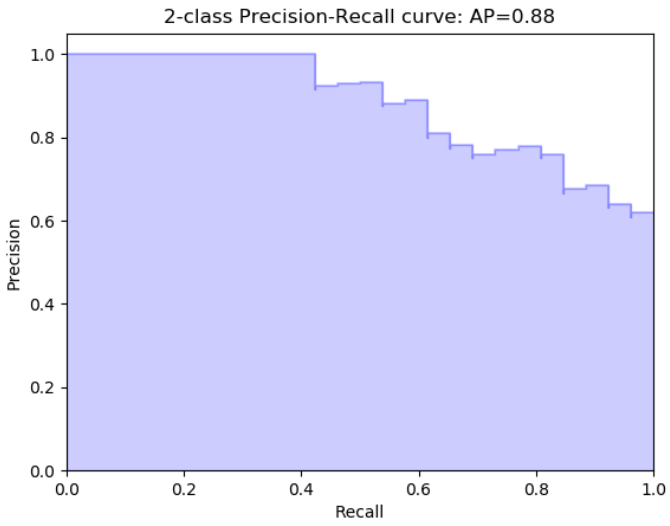


Figure 3.9: A precision-recall curve. [37]

Confusion matrix: The plot shows the amount of true predicted labels and false labels implied by the classes. The diagonal contains those elements, where the predicted label equals to the true label. An example is on Figure 3.10. [38]

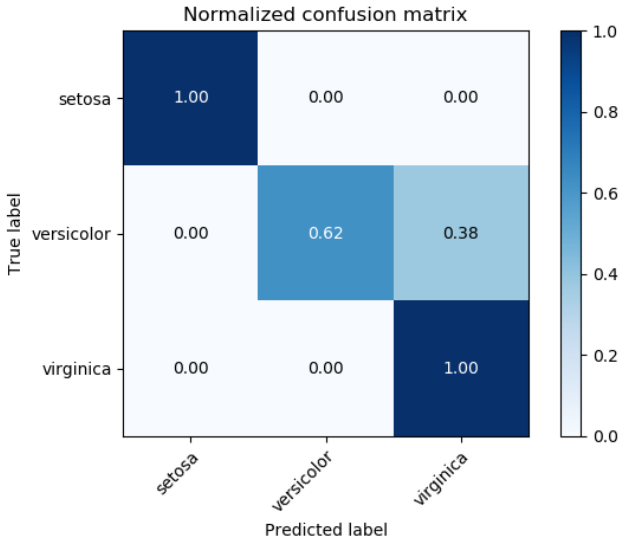


Figure 3.10: Confusion matrix of iris dataset. [38]

Chapter 4

MRIQC with Convolutional Neural Networks

4.1 Data

Two datasets were used from different sources. One of it is called "ABIDE" (Autism Brain Imaging Data Exchange). [7] A held-out dataset, that was only used as validation and test set, was the so-called: "DS000030". [9] The details of these datasets are collected in Table 4.1.

	ABIDE	DS000030
Number of participants	1099	265
Number of exclusions	342	73
Number of acceptions	757	192

Table 4.1: Details of datasets

In the labelling protocol three possible classes could be chosen: "accept", "exclude" and "maybe". The authors of the MRIQC tool [1] classified it to binary classes, simply with labelling all "maybe" images as "accept". In the ABIDE dataset 100 images were labelled by two experts. Figure 4.1. contains the final true labels of the convolutional model for those images which were labelled by both experts.

The two datasets were splitted into three groups. One part of it was used only for training the model. This group was selected only from the ABIDE dataset and it contained 954 images. The second group was the validation set, it contained 205 images from both ABIDE and DS000030 datasets. The third group is the test set, it was not used during the training and as the validation set, it contained 205 images from both ABIDE and DS000030 datasets. The purpose of this kind of data splitting was to have three independent datasets. The training set contained images from

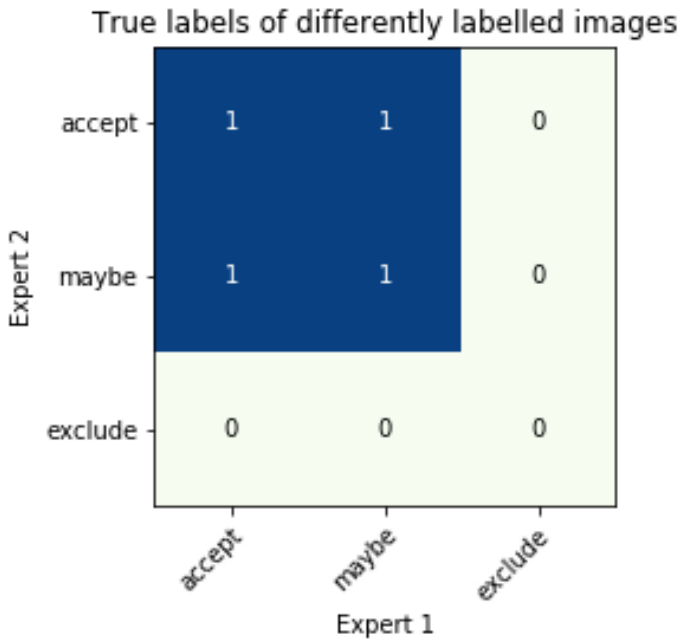


Figure 4.1: Labelling protocol when images were labelled by two experts. The Figure shows the true labels in different cases. Label 1 corresponds to accepted images, label 0 to excluded images.

one source (ABIDE), the validation set also involves subjects from the DS000030 dataset, and the test set has the same combination of data as the validation. Table 4.2. contains the label of participants according to the splitted data.

Set	accept	exclude	sum
Train	626	328	954
Validation	163	42	205
Test	160	45	205

Table 4.2: Number of different labels after data splitting.

4.1.1 The structure of the data: BIDS format

The MRIQC tool [1] requires BIDS (Brain Imaging Data Structure) formatted images. The advantage of this data structure is its ease of use, potential to cooperate with other researchers and compatibility with data analysis softwares. In this chapter I describe the specifications based on the scientific paper [39] that introduced BIDS formatted images.

Before BIDS formatted data there was no widely used standard of organizing MRI images. As the number of neuroimaging studies growed significantly in the past decades, it became important to create a standard data format, that makes

it easy to use neuroimaging data within and between labs. It allows the reuse of research data by sharing it in public databases (e.g. OpenfMRI).

The specification of BIDS contains conventions for filenames, filepaths and file-formats. The structure shown on Figure 4.2.

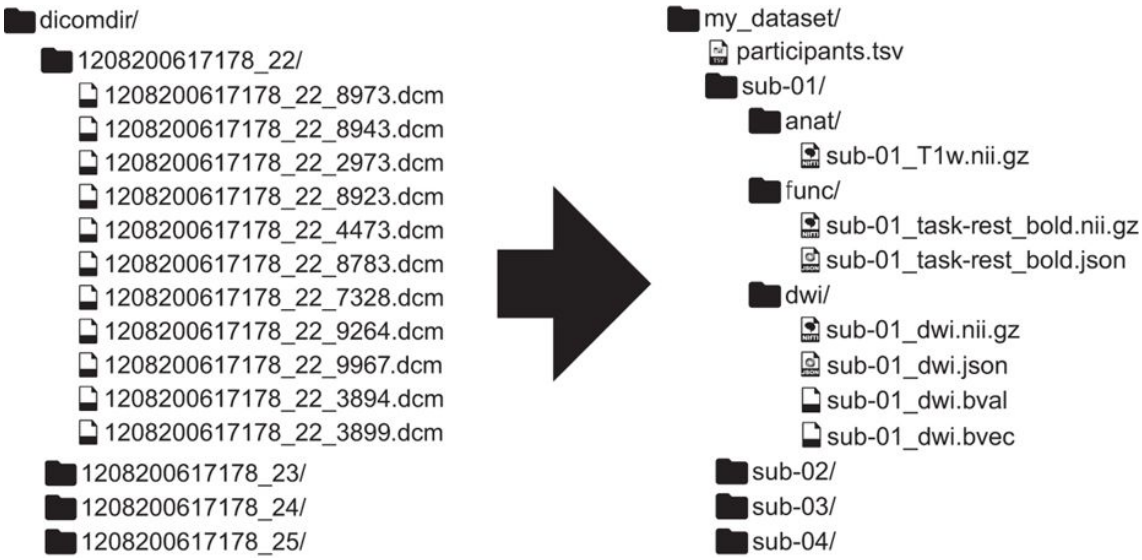


Figure 4.2: "Illustration of a BIDS structured dataset." [39] DICOM structured data shown on the left, whereas BIDS is on the right.

The images and image metadata are stored in a compressed version of NIfTI (Neuroimaging Informatics Technology Initiative) file format. [40] The metadata of the examination is stored in JSON (JavaScript Object Notation). The authors of [39] provided tools to convert images from DICOM to BIDS.

BIDS data structuring in neural network based quality control

The downloaded DS000030 dataset was in BIDS format. On the other hand ABIDE (Autism Brain Imaging Data Exchange) was not in BIDS format. It contained seven scan folders which had folders of different sites with different research tasks. I was not able to use the recommended DICOM to BIDS converter tool because the downloaded data was in NIfTI file format (which is neither DICOM nor BIDS). The paths of the data differed from the specification given in BIDS and the JSON files were missing.

I built the given path structure, renamed the files and completed them with the missing JSON files. To describe the dataset, I created a TSV (Tab Separated Values) file where I collected the participant IDs. The last step was the validation with BIDS-validator [41].

After I created BIDS formatted data, I could run the MRIQC tool to verify the performance claimed in [1]. My model only used the anatomical (T1-weighted) im-

ages. The reason is, because this thesis focuses only on quality control of anatomical images. Further research can investigate targeting functional images.

4.2 Data preprocessing

Data preprocessing was necessary as in every deep learning task. I collected the steps of the procedure below and I give the details afterwards:

1. Rescale
2. Crop to desired image dimensions
3. Standardize
4. Augmentation

4.2.1 Rescale

The size of the voxels differed on the images of the used datasets. I inspected the images on both datasets and identified, that a 1.5 mm^3 voxel size is appropriate (as this was the largest in the datasets). I used the *SciPy* ecosystem's *Multi-dimensional image processing* [42] package to zoom the images to desired voxel size. This function implements all the necessary algorithms for interpolation.

4.2.2 Crop to desired image dimensions

Different input images were used as samples in the model training and evaluation. Both image dimensions and orientation of the images were diverse.

Firstly, I set all the images to the same RAS orientation (as this is the canonical orientation in brain images), which was easy to implement, as orientation information is in the metadata of the images. NiBabel package [43] has a function which implements the setting of canonical orientation.

I also cropped all the images to have the same image dimension in every orientation. The cropping was done in two steps. First I saved the center of the images with the size of [140x180x180]. When the image was smaller in one direction I used zero padding. The second step was done during training, it was also used as an augmentation tool, thus details are below in Section 4.2.4.

4.2.3 Standardize

Standardization has two steps: zero-centering and normalization. Substracting the mean value of the image gives zero-centered data. Normalization is done by dividing the image with the maxiumum absolute value. These processes should be done color-channelwise, but since I have greyscaled images I do not have to take colors into account. The formula of the standardization is written below:

$$X' = \frac{X - \overline{X}}{max\left(\mid X \mid\right)}, \tag{4.1}$$

where X is the original image voxel, $\overline{X}$ is the mean value of the image array, $max\left(\mid X \mid\right)$ is the maximum absolute value of the image array and X' is the standardized image voxel. The implementation of the previous steps results an image with zero mean and maximum absolute value of 1.

4.2.4 Augmentation

The MRI images are high resolution 3D images. This is why I used 'online' (real-time) augmentation during training.

The full image I read in to train the network, is always transformed differently. I crop the images to the smaller resolution of [128x164x164]. This cropping algorithm is done by a self-written Datagenerator class. The randomized crop function decides, which part of the image is going to remain, in a way that the middle of the image is always kept (as the most important information can be found here). The image boundaries are shifted in a range of [12x16x16] voxel size.

Random rotation is also an appropriate augmentation technique in brain images. I implemented a function which rotates the images with the probability of 0.5. The rotation angle is chosen from a Gaussian distribution with zero degree mean and standard deviation of three degrees.

4.3 Models

The purpose of this thesis is to build a three-dimensional convolutional neural network which is able to classify MRI images, whether their quality is acceptable for diagnosis or not. A recent publication [44] (published in November 2018.) has the same goal, but they achieved this aim with two-dimensional deep CNN.

I trained and evaluated more than 40 different models. I selected 5 of them to present their results in Table 4.3.

	Validation				Test			
Model	loss	accuracy	"F1-0"	"F1-1"	loss	accuracy	"F1-0"	"F1-1"
1	0.39	0.86	0.13	0.93	0.6	0.72	0.0	0.84
2	0.39	0.86	0.13	0.93	0.84	0.59	0.24	0.73
3	0.4	0.84	0.15	0.91	0.56	0.73	0.36	0.82
4	0.56	0.74	0.17	0.86	0.57	0.71	0.32	0.84
5	0.6	0.69	**0.4**	0.83	0.64	0.62	**0.4**	0.76

Table 4.3: Results of 5 different models."F1-0" means the F1 score on excluded images and "F1-1" is the F1 score on accepted images.

The input of the models is a three dimensional image with the resolution of [128x164x164]. All models consist of four three-dimensional convolutional layers and four three-dimensional max pooling layers. The activation function of the convolution layers are relu type. The kernel size in the first convolutional layer is [5x5x5], in the second, third and fourth [3x3x3]. The max pooling layers have a pool size of [2x2x2] in each model thereby, after the max pooling operation, the dimensions of the image decreases to the half of the previous size. After flattening the image, a fully connected layer is applied with the size of 512 (in 1,2 models) or 128 (in 3,4,5). This fully connected layer has an activation function of sigmoid type. It is connected to an output fully connected layer which has a softmax activation and it contains two output neurons. The model architecture is shown on Figure 4.3.

I used Adam optimizer with a learning rate of 0.0001. (The 40 models also contained trainings with higher learning rates, but with worse performance as the selected ones.) Models use categorical cross entropy to calculate the loss, this measure is also used for weight updates and to measure the performance of the models. When the loss decreases during validation, the algorithm updates the saved best performing model. I also set early stopping with patience between 15 and 25.

The main differences between models came from the use of different techniques that are recommended to use becuse of imbalanced data or faster and better convergence of the models. In the following paragraphs, I present the motivation of the techniques and I interpret the results.

Model "1" contains four three-dimensional convolution layers with three-dimensional kernels. After the maxpooling layers, a simple dropout is applied with a rate of 0.25. After the 512 long fully connected layer the dropout rate is 0.5. The validation accuracy is high in this simple model (0.86), although, in case of imbalanced data, the F1 score is a better metric to evaluate the performance. The F1 score on the "exclude" class is really low (0.13) and it is observable on the test set that the model predicts only the "accept" class. This causes high accuracy also on

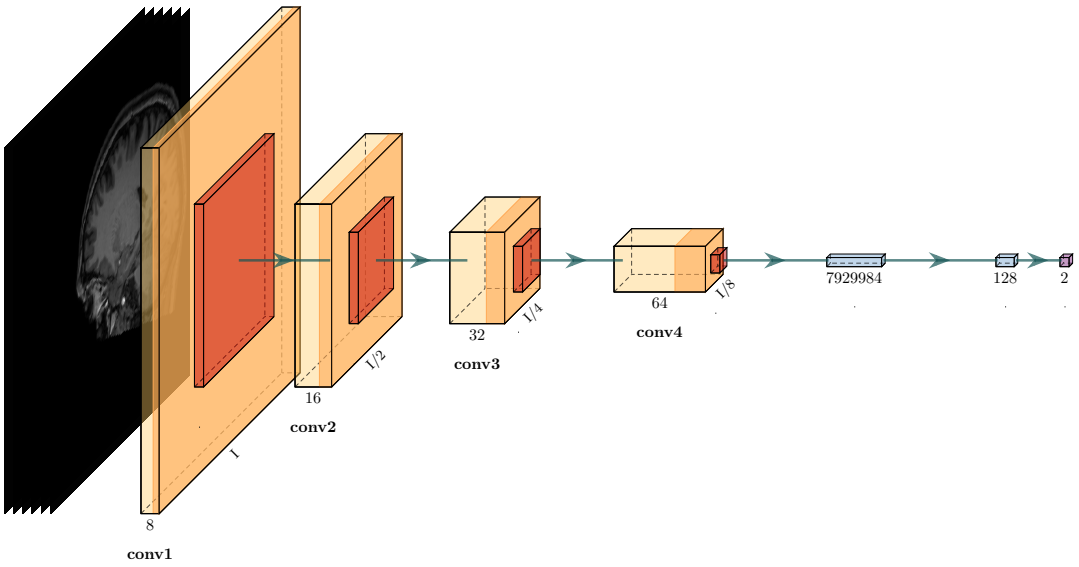


Figure 4.3: The used architecture. This image shows slices of the MRI image indicating that the input is a 3D image. I used 3D convolution, instead of using convolution layers slicewise. The reason I display one slice's convolution is that I also wanted to show the number of filters applied (indicated by the thickness of layers and the number underneath). Every layer contains a 3D convolution, activation function, a max pooling operation and dropout. At the end I flatten the 3D network to get a one dimensional output. The lighter orange boxes indicate the convolution layers, the darker orange marks the "relu" activation, the smaller darkest orange box is the maxpooling layer. Fully connected layers are shown as blue boxes. The output is red on the right side of the image.

test set (caused by the majority class of accept label), but 0.0 F1 score on exclude class.

In model "2" I used separable convolution. Instead of one three-dimensional kernel, I applied three one-dimensional kernels in each layer. This technique allows to train less weights and it is efficient computationally. [45] It performs identically on the validation set compared to model "1". On the test set the accuracy is way lower (0.59), though the F1 score on excluded images is much better (0.24).

To handle the imbalanced data, I applied a new augmentation technique: the upsampling of minority class. [46] In model "3", the images with label "exclude" were doubled, this way the ratio of label 1 and 0 was 626:656. This model also contains separable convolution layers. On the one hand, validation accuracy and loss were a bit worse than in the previously presented models, only the F1 score on 0 labelled data was slightly better. This better performance is also present in the test evaluation. The F1 score of excluded images was 0.36 and compared to model "2" the accuracy was also higher and the loss lower.

In image classification, instead of normal dropout spatial dropout is more appropriate to use [30], thus model "4" operates with spatial dropout with a rate of 0.5 after each separable convolution layer. The performance on the validation set was not satisfactory, although the F1 score is slightly better, than on the previous models (0.17), the accuracy and the loss were insufficient. The F1 score on excluded images in the test set was almost as high as in model "3" (0.32), the accuracy and the loss were also similar.

The best performing model in the sense of F1 score on excluded images, is model "5" with 0.4. This score also remains on the test set. This model contains random rotation [47] (with probability of 0.5, in the range of [-3,3] degrees with Gaussian distribution), separable convolution layers with spatial dropout (with rate 0.25). In the sense of accuracy and loss this model achieves a lower rank, but as I mentioned before, in case of imbalanced data F1 score metric is more significant, thereby I consider this model as the best performing one. A further research can investigate the tuning of hyperparameters. I assume that with a deeper network, a further trained model with lower learning rate and more data, the performance could be enhanced, but this hyperparameter-tuning was not the target of this thesis.

4.4 Best performing model evaluation on test

As we saw in Section 4.3. the 5^{th} model performs the best on the validation set and the same performance can be observed also on the test dataset. In the following section I compare this performance with the MRIQC tool from [1]. Later, I inspect the outputs of different channels in the layers of the best performing model, thus I present the appearance of different artifacts in deeper layers.

4.4.1 Comparison of convolutional and random forests classifier based MRIQC tool

Table 4.4. contains the evaluated metrics on test sets to compare the convolutional quality control method (CNN) and the MRIQC tool from Ref. [1] which is based on a random forests classifier (RFC). The test datasets are not completely the same, as I used same data combination as in valiadation. The confusion matrices of both classifiers are on Figure 4.4.

Model	accuracy	"F1-0"	"F1-1"	average F1 score
RFC	0.76	0.4	0.85	0.72
CNN	0.62	0.4	0.76	0.57

Table 4.4: Comparison of convolutional method and RFC based MRIQC tool."F1-0" means the F1 score on excluded images and "F1-1" is the F1 score on accepted images.

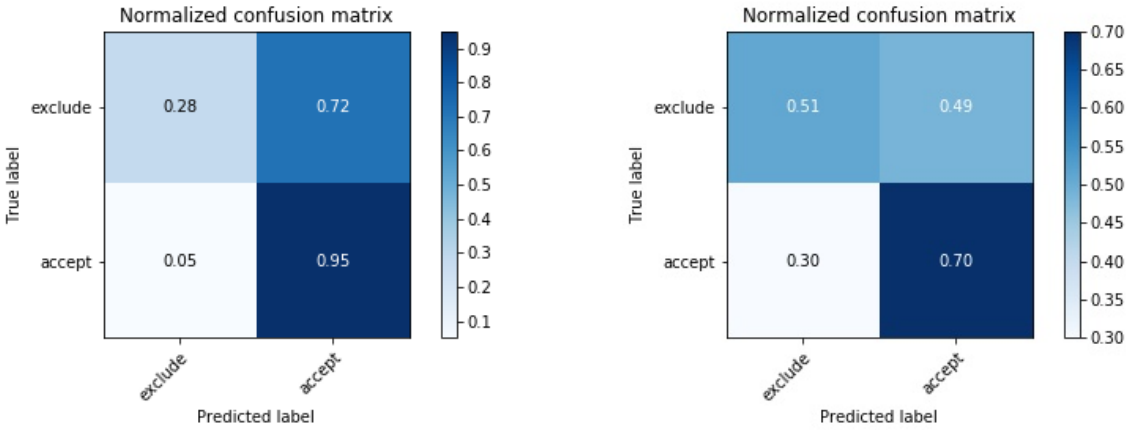


Figure 4.4: Confusion matrices of the classifiers on test dataset, (left) shows the RFCs, (right) shows the CNNs confusion matrix.

The two models have the same 0.4 F1 score on excluded images. The convolutional method has worse metrics in the aspect of accuracy (just 0.62 compared to

0.76) and F1 score on accepted images (0.76 in comparison with 0.85) as it does not classify well the class with label 1. This causes the great difference between the average F1 scores also (0.72 versus 0.57).

On the confusion matrices in Figure 4.4. it is clearly visible, that on the one hand, the CNN classifier performs better in the classification of excluded images. The true negative score is 0.51 with CNN classifier which is much higher than the true negatives of RFC (0.28). On the other hand, the classification of accepted images are better in RFC with 0.95 in comparison with 0.7.

It is undoubtedly extremely important to correctly classify true negative samples in medical images. Erroneous positive labelling can have severe consequences, such as misdiagnosis or unnecessary inconvenience for the patient. This is the reason why we should be cautious with the evaluation of the models. The model I built might have lower average score than the RFC model, however it performs good in this crucial aspect.

4.4.2 Investigation of convolution kernels in CNN

In the following section, I present the output activations of different convolutional layers for different types of artifacts and other imaging errors that cause the exclusion of images. I show the activation after each layer (until the third layer) meaning that the output of one convolutional layer is after the last separated convolution.

Figure 4.5. is an example for convolutional kernels. On this image 24 different kernels can be seen. As the best performing model contains separable convolution, the dimension of kernels are [5x1x1] in each direction (the image is three-dimensional thus we have [x,y,z] kernels). The first layer contains 8 different filters, so the rows (from 0 to 7) on the figure corresponds to different convolution kernels.

4.4.3 Activation after convolution kernels on a classified true positive image

Figure 4.6. is an example of a true positive classified image. The upper image shows the original slice of the MR image. The figure also contains the last activations of convolutional layers in the depth of the first, second and third convolution. It can be seen that mostly the inner part of the brain activates in deeper layers (e.g. first, second and forth image on 4.6c., second, third image on 4.6d.). What is also interesting, that only the background activates with the kernel on the third image of 4.6d. We should make an other observation, because it is going to be important later, regarding the activation of the skull. In the outer and also in deeper layers

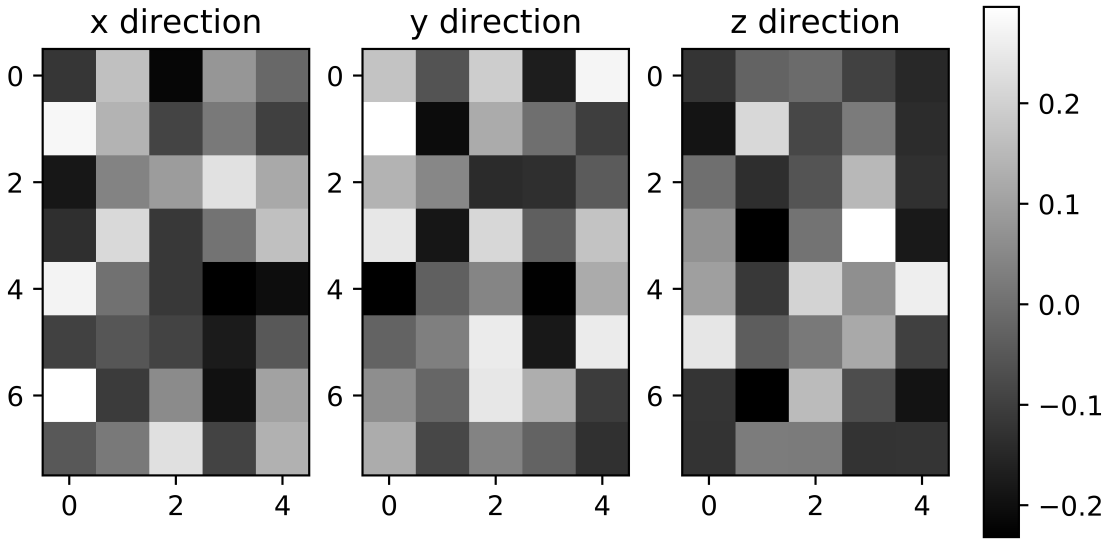


Figure 4.5: The convolution kernels of the first layer in the best performing CNN model.

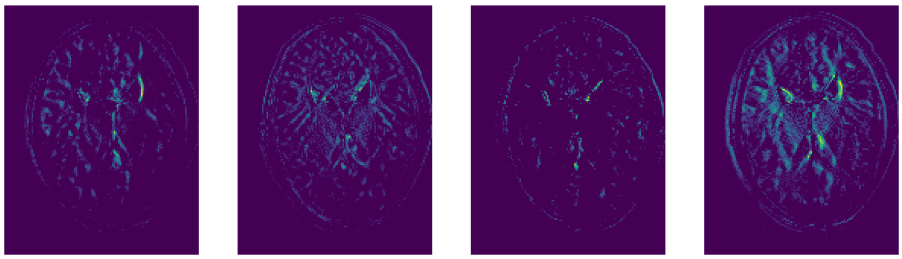
it can be seen, that the contour of the skull is not highly activated. This kind of output is usual on images with label "accept". The minority of the selected channels works as an "inverse" compared to others (the third image of the second layer looks like it is an inverse of the second image of the same layer).

4.4.4 Activation of convolutional layers on images with artifacts

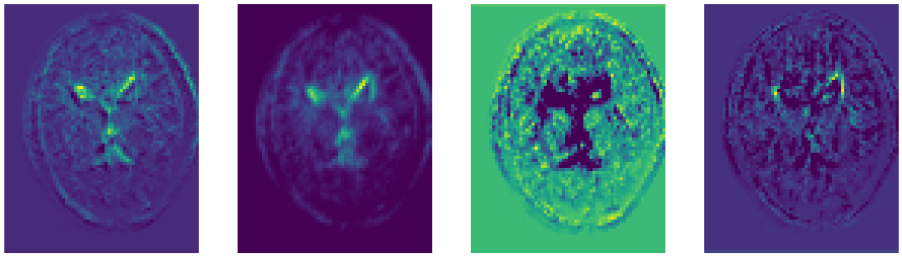
Moiré fringes artifact with convolutional kernels

On Figure 4.7. a moiré fringes artifact is observable. This image is an example of true negative sample which means the predicted label was "exclude", the same as the true label. On the upper image the blue arrows show the critical areas that help to detect the artifact for us. If we carefully look at the output of the first layer, the contours of the artifact are clearly visible. In deeper layers, the activation is really high at the arrow pointed part of the brain. The increased activation of the skull is a characteristic feature of artifact images. In case of moiré fringes artifact, the zebra stripes are also quite activated areas on the output image (e.g. on the second, third, fourth image of the third convolutional layer). This artifact is easily classified by the CNN model, despite the fact that it is not really frequent in the databases.

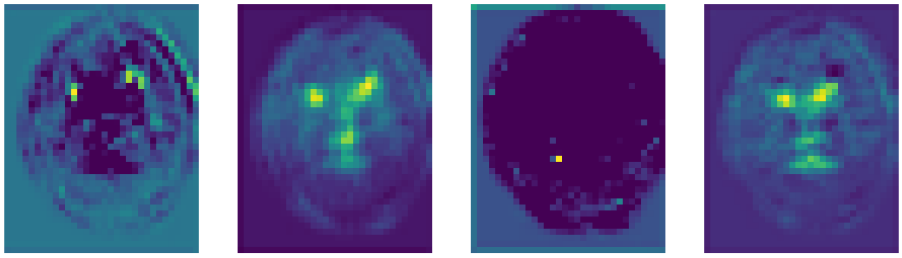
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



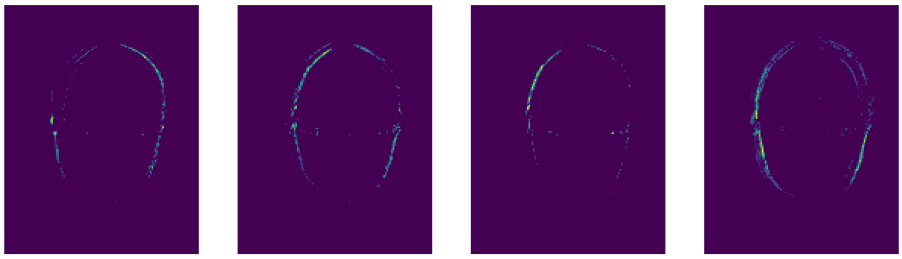
(c) After the activation of the second convolutional layer



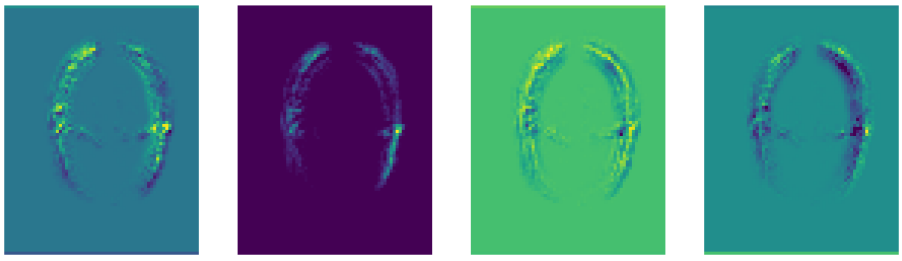
(d) After the activation of the third convolutional layer

Figure 4.6: The activations in case of a classified true positive image.

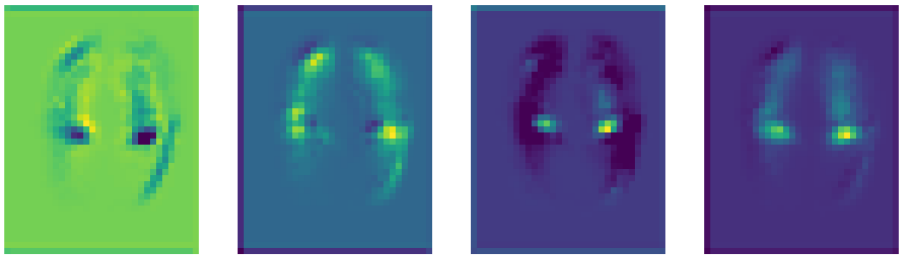
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



(c) After the activation of the second convolutional layer



(d) After the activation of the third convolutional layer

Figure 4.7: Activation in deeper layers in case of moire fringes artifact, classified as true negative.

Motion artifact with convolutional kernels

On figure 4.8. the upper region is highly activated on the deeper layers' outputs (e.g. first, second and third images on 4.8d). This high output activation pattern was also seen before on the image with moiré fringes, thus the predicted label is "exclude".

Figure 4.9. shows a false positive sample for motion artifact. On the upper image, it is observable that the motion stripes are inside the white and grey matter. In this case the model predicts that the image is good, and the patterns in deeper layers' outputs are more similar to true positive images (e.g. on the first and second image of second convolutional layer, the second and fourth image of the third convolutional layer the middle part of the brain is slightly more activated on the output).

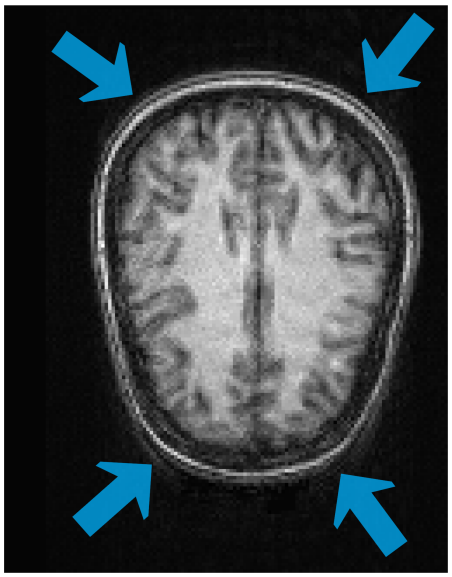
Aliasing artifact with convolutional kernels

The model can hardly classify the images with aliasing. As I checked all the validation and test MRI images, I discovered that the number of aliasing artifacts in the group of false negatives is also quite high. Those images that contained aliasing, but the artifact did not appear in the white or grey matter, were labelled as "accept" by the experts. I inspected these images and concluded, that it is even hard for human observer to understand the logic behind the labelling.

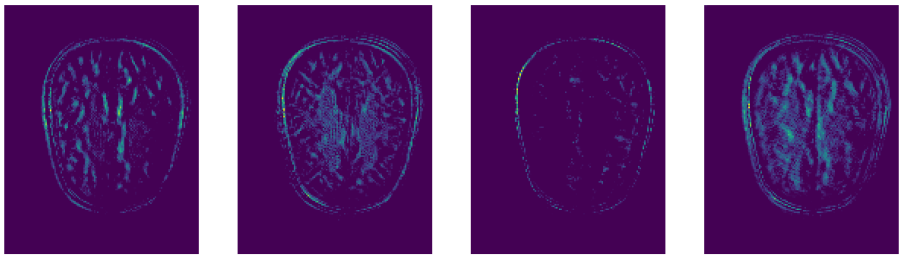
On Figure 4.10. an example of true negative image is shown. As I stated before in Section 4.4.4., images with artifacts tend to activate (or deactivate on images with "inverse" filters) the skull in deeper regions. This effect can clearly be seen on the second and third layers' output channels.

Figure 4.11. shows an example that the model was not able to well classify. This image is also an aliasing image, labelled as "exclude", but the CNN predicted "accept". In the first layer the marked areas are also activated as on the true negative image, although this activity decreases in the deeper regions (e.g. on the fourth image of 4.11d). On the third image of the third layer the contrast is as high, as in true positive images and not as in well-classified aliasing. On the other hand, the other images look more like the output in case of excluded images: the output activation of the skull is increased and the inner part of the brain is not as active. All these taken into account, the class of the image could be both "accept" and "exclude".

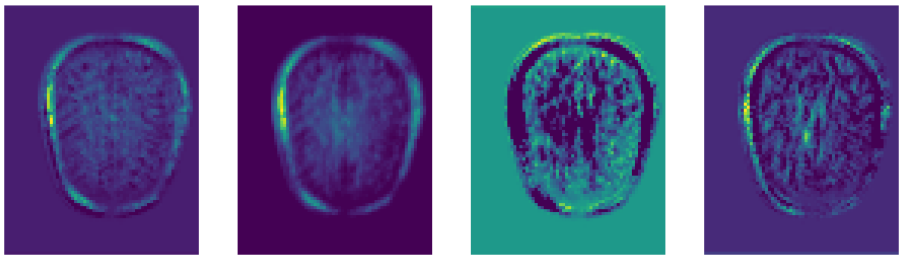
A false negative example can be observed on Figure 4.12. The aliasing is in the white matter, however the true label is "accept". The output channels in deeper layers are indeed similar to an artifacted image: the activation around the skull is increased (or decreased on "inverse" images).



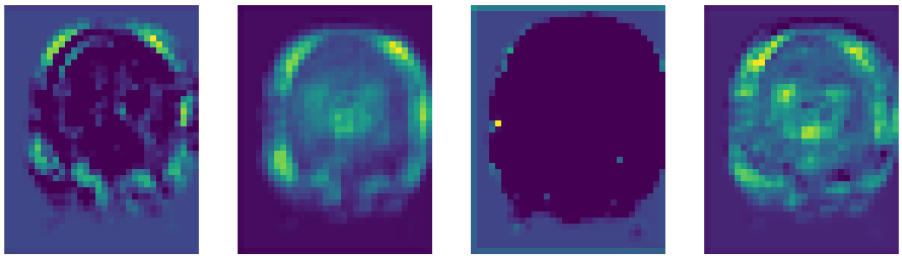
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



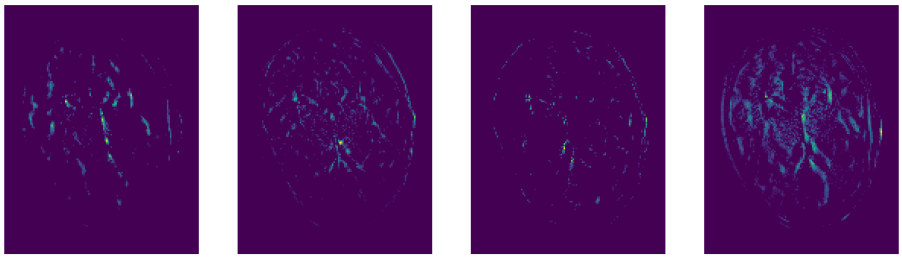
(c) After the activation of the second convolutional layer



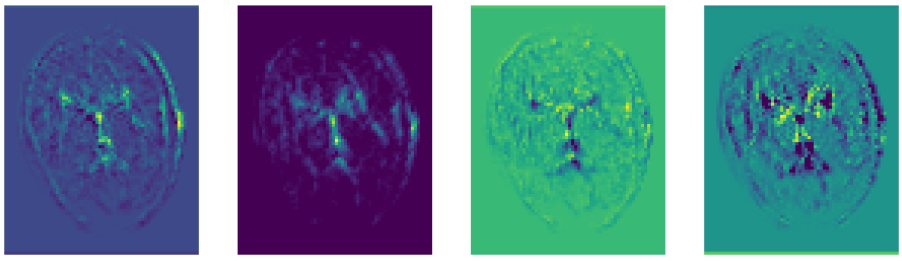
(d) After the activation of the third convolutional layer

Figure 4.8: Activation in deeper layers in case of motion artifact, classified as true negative.

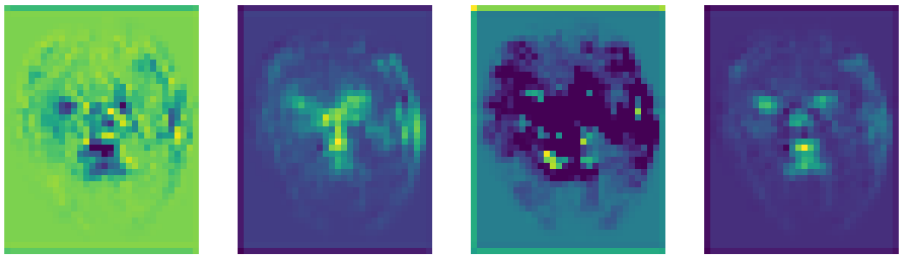
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



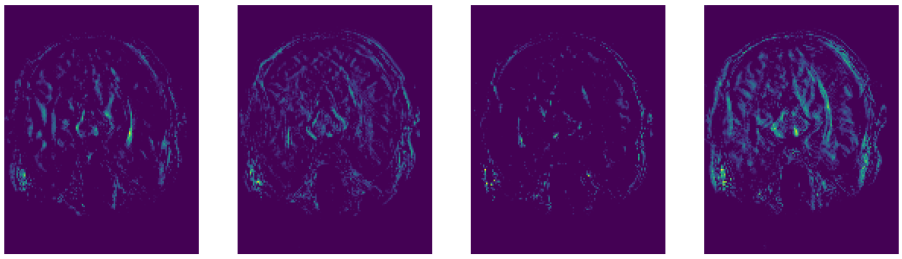
(c) After the activation of the second convolutional layer



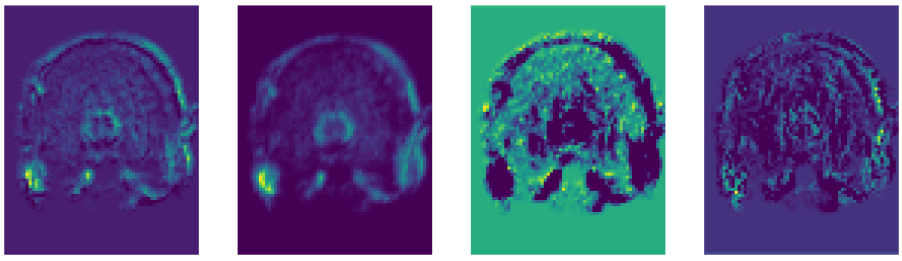
(d) After the activation of the third convolutional layer

Figure 4.9: Activation in deeper layers in case of motion artifact, classified as false positive.

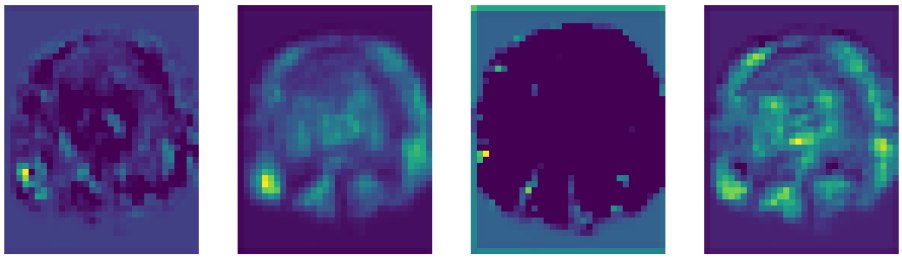
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



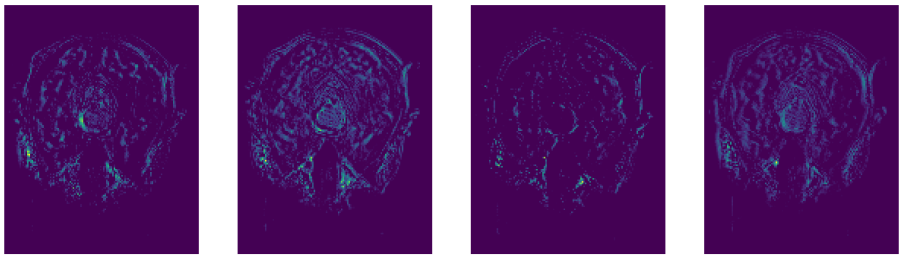
(c) After the activation of the second convolutional layer



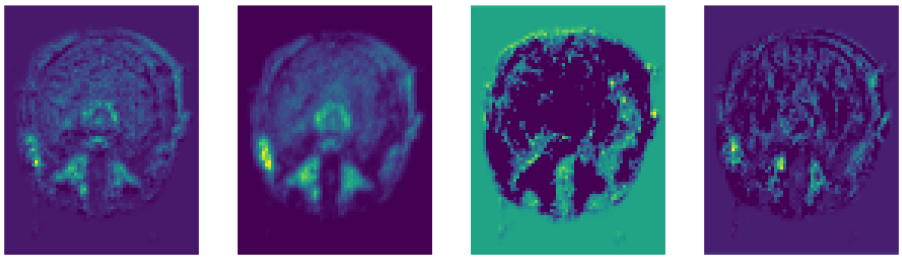
(d) After the activation of the third convolutional layer

Figure 4.10: Activations after convolutional layers on an image with aliasing artifact. The model classified the image as "exclude" identical to the true label.

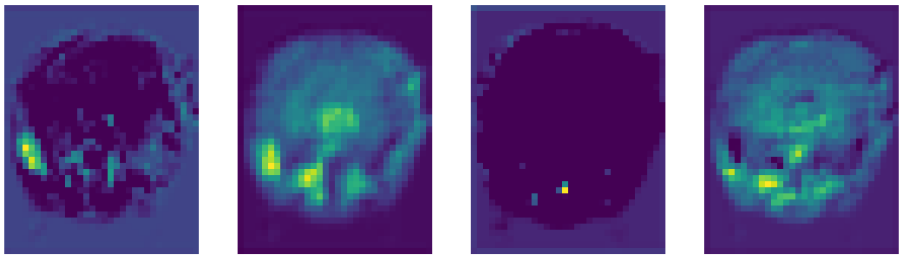
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



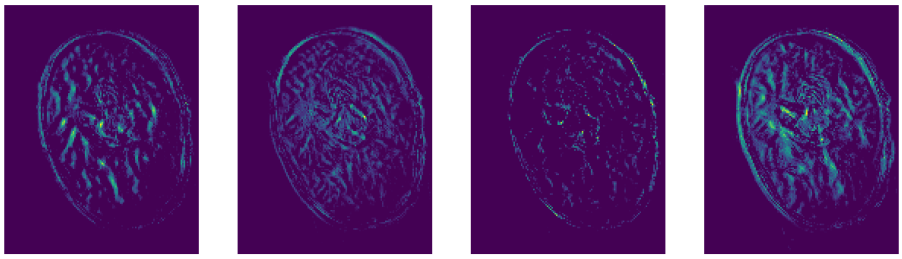
(c) After the activation of the second convolutional layer



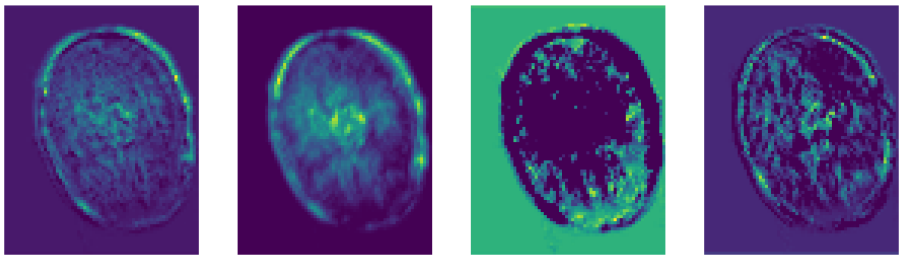
(d) After the activation of the third convolutional layer

Figure 4.11: Activations after convolutional layers on an image with aliasing artifact. The model classified the image as "accept" but the true label is "exclude".

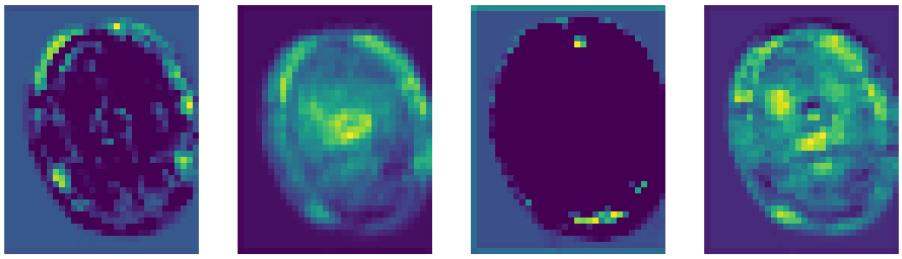
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



(c) After the activation of the second convolutional layer



(d) After the activation of the third convolutional layer

Figure 4.12: Activations after convolutional layers on an image with aliasing artifact. The model classified the image as "exclude" although the true label is "accept".

4.4.5 Orientation issue

Orientation issue is not an artifact, but it is also a typical cause of excluded images in the used datasets. The orientation of an image can be found in metadata of MR images. In case of the *NIFTI* file format the default orientation is "RAS". "RAS" means that the data in the first dimension ("R") is filled from left to right, in the second dimension ("A") is filled from posterior to anterior, and in the third ("S") is filled from inferior to superior direction. [40] A few examples of the default orientation have been shown in Chapter 2. in Figure 2.3., 2.4., 2.5. In the images, that have been excluded because of orientation issue, the metadata of the images are not consistent with the anatomical orientation. A comparison image is shown on Figure 4.13.

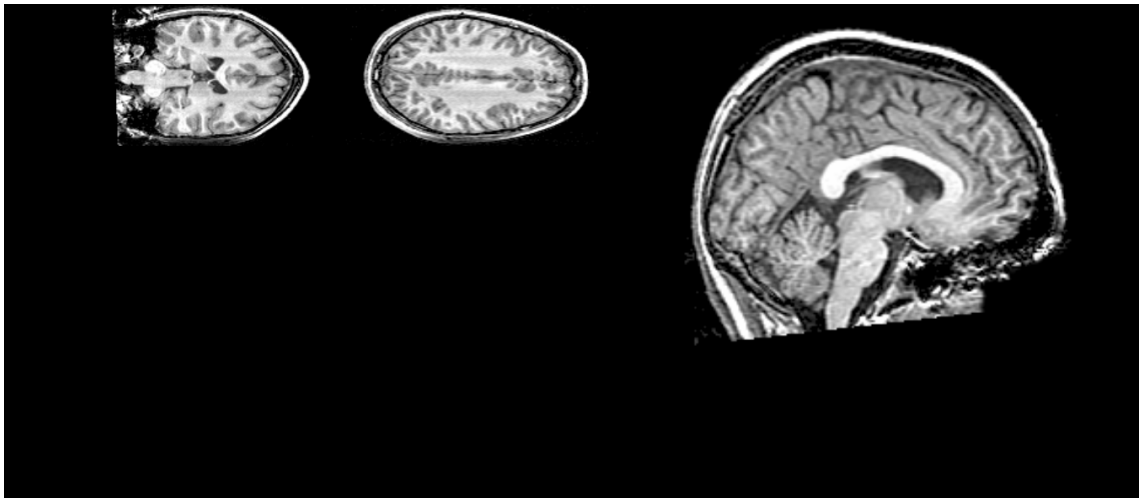
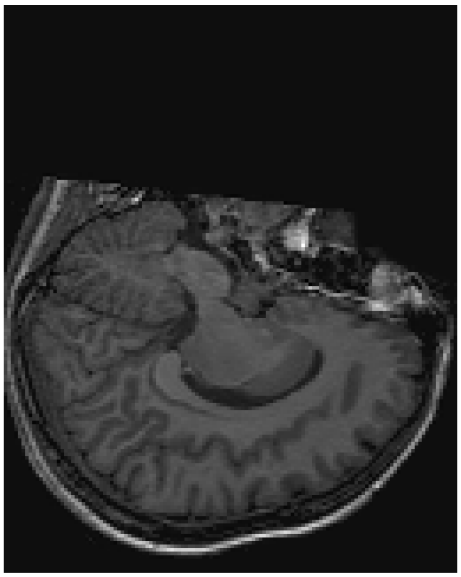


Figure 4.13: Orientation problem from ABIDE dataset. [7]

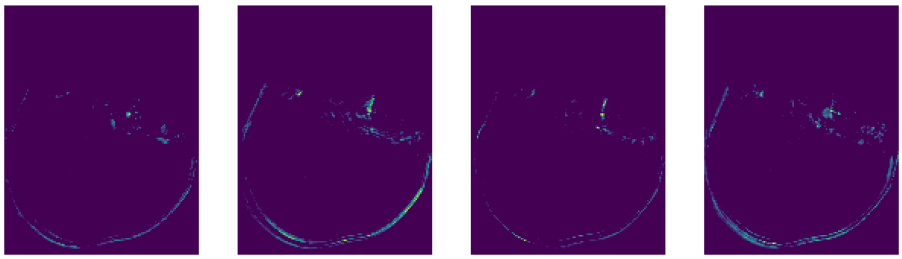
Orientation issue with convolutional kernels

The convoluted images after the activations can be seen on Figure 4.14. One can observe that the brain image is not at the same orientation as before (e.g. in case of motion artifact on Figure 4.8.).

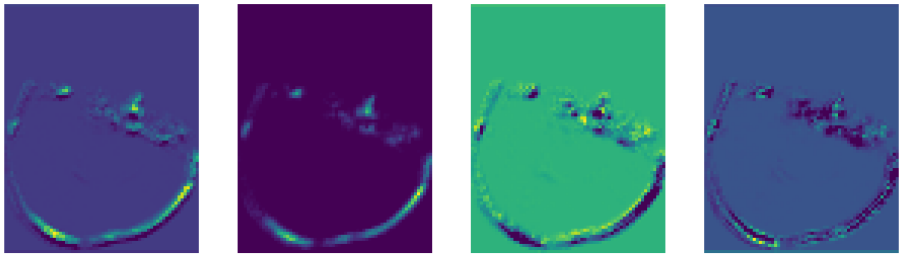
If we look at the output of deeper convolutional layers, it can be seen that the upper part of the skull (it is on the bottom of the image on this orientation) is highly activated (e.g. on the second image of Figure 4.14d). This kind of activation has not been seen before, therefore it was easy to learn for the model, despite the low number of samples which contained orientation problems.



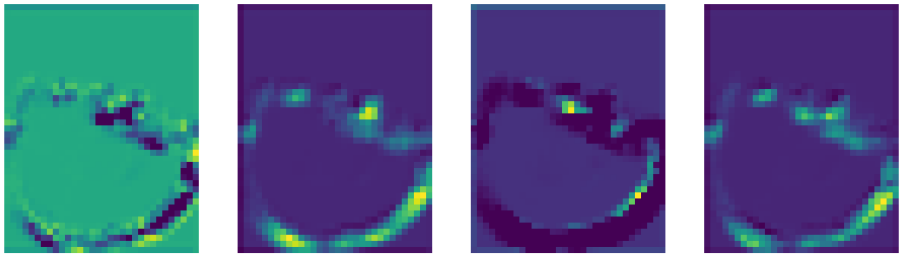
(a) Original slice of image



(b) The same slice after the activation of the first convolutional layer



(c) After the activation of the second convolutional layer



(d) After the activation of the third convolutional layer

Figure 4.14: Orientation issue in deeper layers.

4.5 Software support

I wrote the CNN based MRI quality control tool in Python, in the Anaconda environment [48] and ran it in Jupyter Notebook [49]. In image preprocessing I used the support of NiBabel [43] and SciPy [42] packages. The tools of Keras library [50] were used to build, train, validate and test the convolutional neural network. The evaluation was done with the toolbox of Scikit-learn [51]. The plots are made with the help of Matplotlib library [52]. During the visual inspection of the MR brain images I worked with MRIcron [53].

Chapter 5

Conclusion

5.1 Error analysis

In Section 4.4.4., I presented the output channels of deeper layers on artifacted images. On the one hand moiré fringes was easily classified by the best performing convolutional model. On the other hand there were some problems in the classification of images with motion artifact, and the hardest was to classify aliasing. In the true labels of aliasing artifact, I showed some inconsistency: namely, that even when the aliasing artifact is visible inside the brain, the true label was "accept", although the quality of this image is clearly not acceptable.

Based on these findings, I suggest to examine the contradiction in the labelling protocol as the labels are sometimes inadequate. The labelling protocol contained an investigation of inter-rater variability. [1] An image from the publication can be seen on Figure 5.1. This image shows the results of the 100 images that were rated by two experts. This heatmap shows us, that only 57% of the images were labelled with the same label by both experts. After the labels were binarized the rate increased to 78%.

Because of the imbalanced data an other measure is more appropriate to investigate: Cohen's kappa index. This score describes the agreement of two raters, not only with accuracy, but also taking into account the chance of the possibility of agreement. The coefficient of complete agreement is $\kappa = 1$. [54] The Cohen's kappa can be calculated based on the heatmap. Its value for the previous labelling protocol is $\kappa = 0.39$ which is according to *Viera et al.* [55] "fair". In this article the agreements are grouped into 5 categories from which 'fair" is the second to last. When the mapping of the labels from three groups to two is done, the index increases to $\kappa = 0.51$ which is "moderate" in the scale of Viera. "Moderate" is still just the middle of the scale, which can not be considered high. This examination shows us,

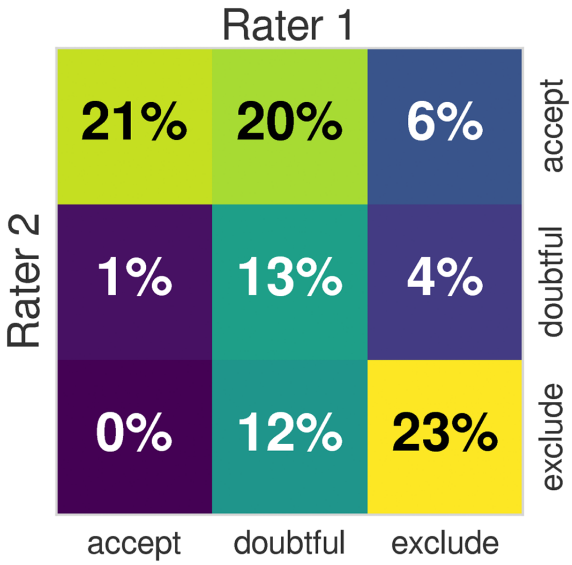


Figure 5.1: Inter-rater variability from [1].

that further investigation of labelling should be done. Two experts whose job is to evaluate these images rated them differently, thus the model is not able to learn it. More statistical information is needed to create a better, consistent labelling.

5.2 Summary

The aim of this thesis was to build an MR image quality control based on deep convolutional neural networks and to analize the output channels in case of artifacted images. I presented the physical background of common artifacts: motion, aliasing, moiré fringes and Gibbs. I went through the relevant scientific articles of the field and presented the behaviour of CNNs.

I examined more than 40 different models from which I detailed the results of five. I compared the performance of my best model and the MRIQC tool [1]. On the one hand, the F1 score on excluded images were the same on the test dataset, and my model was even better in the prediction of negative samples. On the other hand, the accuracy was lower, further hyperparameter tuning could be done to achieve better performance.

Lastly, I examined the output channels on images with different artifacts. I concluded that the activation of the skull area of the head is determinative in classifying the artifacted MRI images. Error analysis shows us that a labelling protocol including more raters is needed, to statistically decrease the inconsistency of labelling.

5.3 Acknowledgement

I am extremely grateful to my supervisor Dr. Regina Deák-Meszlényi for the professional advices, guidance, flexibility and fast replies.

I would like to thank Dr. Dávid Légrády his patience, valuable insights and his help with administrative tasks.

I would also like to express my gratitude to the Brain Structure and Dynamics Research Group at the Hungarian Academy of Sciences for the provided infrastructure and technical support (especially to Dr. Pál Vakli and Virág Darányi).

I am also thankful to my family and friends for supporting me.

Bibliography

[1] O. Esteban, D. Birman, M. Schaer, O. O. Koyejo, R. A. Poldrack, and K. J. Gorgolewski, "Mriqc: Advancing the automatic prediction of image quality in mri from unseen sites," *PloS one* **12** no. 9, (2017) e0184661.

[2] "Mriqc supporting information." `https://journals.plos.org/plosone/article/file?type=supplementary&id=info:doi/10.1371/journal.pone.0184661.s001`. Accessed:2018-10-01.

[3] J. P. Woodard and M. P. Carley-Spencer, "No-reference image quality metrics for structural mri," *Neuroinformatics* **4** no. 3, (2006) 243–262.

[4] B. Mortamet, M. A. Bernstein, C. R. Jack Jr, J. L. Gunter, C. Ward, P. J. Britson, R. Meuli, J.-P. Thiran, and G. Krueger, "Automatic quality assessment in structural brain magnetic resonance imaging," *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine* **62** no. 2, (2009) 365–372.

[5] F. Godenschweger, U. Kägebein, D. Stucht, U. Yarach, A. Sciarra, R. Yakupov, F. Lüsebrink, P. Schulze, and O. Speck, "Motion correction in mri of the brain," *Physics in Medicine & Biology* **61** no. 5, (2016) R32.

[6] M. Zaitsev, J. Maclaren, and M. Herbst, "Motion artifacts in mri: a complex problem with many partial solutions," *Journal of Magnetic Resonance Imaging* **42** no. 4, (2015) 887–901.

[7] C. Informatics and N. Suite, "All pre-approved abide data," 2013. ABIDE: `http://fcon_1000.projects.nitrc.org/indi/abide/`, COINS: `https://coins.mrn.org/`.

[8] R. W. Brown, E. M. Haacke, Y.-C. N. Cheng, M. R. Thompson, and R. Venkatesan, *Magnetic resonance imaging: physical principles and sequence design*. John Wiley & Sons, 2014.

[9] N. R. for Medical Research grants, "Ds000030," 2017. This data was obtained from the OpenfMRI database. Its accession number is ds000030 `https://legacy.openfmri.org/dataset/ds000030/`.

[10] A. Stadler, W. Schima, A. Ba-Ssalamah, J. Kettenbach, and E. Eisenhuber, "Artifacts in body mr imaging: their appearance and how to eliminate them," *European radiology* **17** no. 5, (2007) 1242–1255.

[11] K. T. Block, M. Uecker, and J. Frahm, "Suppression of mri truncation artifacts using total variation constrained data extrapolation," *International journal of biomedical imaging* **2008** (2008) .

[12] "Linear classification." `http://cs231n.github.io/linear-classify/`.

[13] "Neural networks." `http://cs231n.github.io/neural-networks-1/`.

[14] "Image kernels." `http://setosa.io/ev/image-kernels/`.

[15] M. Kuhn and K. Johnson, *Applied predictive modeling*, vol. 26. Springer, 2013.

[16] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *Journal of Computer Science* **2** no. 9, (2006) 735–739.

[17] "Data standardization." `http://cs231n.github.io/neural-networks-2/`.

[18] "Per image normalization vs overall dataset normalization." `https://stats.stackexchange.com/questions/322802/per-image-normalization-vs-overall-dataset-normalization`.

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

[20] "Keras imagedatagenerator class." `https://keras.io/preprocessing/image/#imagedatagenerator-class`.

[21] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. of the Int'l Conf. on Artificial Intelligence*. 2000.

[22] "Convolutional neural network." `http://cs231n.github.io/convolutional-networks/`.

[23] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," *PhD thesis, Ph. D. thesis* **1** (2014) 3.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861* (2017) .

[25] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, pp. 92–101, Springer. 2010.

[26] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[27] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification.," in *IJCNN*, pp. 1918–1921. 2011.

[28] J. Masci, A. Giusti, D. Ciresan, G. Fricout, and J. Schmidhuber, "A fast learning algorithm for image segmentation with max-pooling convolutional networks," in *2013 IEEE International Conference on Image Processing*, pp. 2713–2717, IEEE. 2013.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research* **15** no. 1, (2014) 1929–1958.

[30] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656. 2015.

[31] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378* (2018) .

[32] "Keras glorot initializery." `https://keras.io/initializers/,`.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. 2010.

[34] "Formula of categorical crossentropy." `https://lasagne.readthedocs.io/en/latest/modules/objectives.html#lasagne.objectives.categorical_crossentropy`.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980* (2014) .

[36] "Overfit model figure." `https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stop`

[37] "Precision-recall." `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html`.

[38] "Confusion matrix." `https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py`.

[39] K. J. Gorgolewski, T. Auer, V. D. Calhoun, R. C. Craddock, S. Das, E. P. Duff, G. Flandin, S. S. Ghosh, T. Glatard, Y. O. Halchenko, *et al.*, "The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments," *Scientific Data* **3** (2016) 160044.

[40] "Nifti file format." `https://brainder.org/2012/09/23/the-nifti-file-format/`.

[41] "Bids-validator." `https://github.com/bids-standard/bids-validator`.

[42] "Scipy ndimage.". `https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.zoom.html`.

[43] "Nibabel." `https://nipy.org/nibabel/`.

[44] S. Sujit, R. Gabr, I. Coronado, M. Robinson, S. Datta, and P. Narayana, "Automated image quality evaluation of structural brain magnetic resonance images using deep convolutional neural networks." Easychair preprint no. 650, EasyChair, 2018.

[45] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2754–2761. 2013.

[46] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter* **6** no. 1, (2004) 20–29.

[47] P. Lakhani, D. L. Gray, C. R. Pett, P. Nagy, and G. Shih, "Hello world deep learning in medical imaging," *Journal of digital imaging* **31** no. 3, (2018) 283–289.

[48] "Anaconda environment." `https://docs.conda.io/projects/conda/en/latest/index.html`.

[49] "Jupyter notebook." `https://jupyter.org/`.

[50] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[51] "Scikit-learn." `https://scikit-learn.org/stable/`.

[52] "Matplotlib." `https://matplotlib.org/`.

[53] "Mricron." `https://www.nitrc.org/projects/mricron`.

[54] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement* **20** no. 1, (1960) 37–46.

[55] A. J. Viera, J. M. Garrett, *et al.*, "Understanding interobserver agreement: the kappa statistic," *Fam med* **37** no. 5, (2005) 360–363.