Cerebral Aneurysm Detection in 3D Medical Images Applying Convolutional Neural Networks

Jonathan Kloss (Computer Science), Ulrike Schnaithmann (Data and Knowledge Engineering)
Supervisors: Dr.-Ing. Sylvia Saalfeld, Dr. Sandy Engelhardt, Prof. Dr.-Ing. habil. Bernhard Preim

Otto-von-Guericke-Universität, Magdeburg

Abstract

Within the scope of this project we conducted research into applied deep learning techniques for medical image analysis specifically for the detection of cerebral aneurysms. We implemented a software which handles the end-to-end detection process. The program includes preprocessing and augmentation of DICOM format images and training of a 3D Unet on the respective data. The predicted output of the trained model is evaluated and further visualized in a user-friendly and inituitive way.

Contents

Index of Abbreviations

CT - Computerized Tomography
CNN - Convolutional Neural Network
DICOM - Digital Imaging and Communications in Medicine
MRI - Magnetic Resonance Imaging
ReLU - Rectified Linear Unit
TP - True Positive
TN - True Negative
FP - False Positive
FN - False Negative
SGD - Stochastic Gradient Descent
SVM - Support Vector Machines

1. State-of-the-Art in Medical Image Analysis

1.1. Motivation for Deep Learning in Medical Imaging

With the discovery of x-radiation in 1895 and the possibility to produce cross-sectional pictures of internal organs and bone structures the field of medical imaging was created.
The initial restriction of x-rays to 2-dimensional images was removed with the introduction of computerized tomography (CT). This technique generates a series of 2D x-ray scans from different angles which can be further processed into a 3-dimensional image. Magnetic Resonance Imaging (MRI) takes a different approach as it does not use ionized radiation like CT. It rather utilizes powerful magnetic fields to excite hydrogen in human tissue generating a signal which can be spatially encoded to be transformed into images. Similar to CT, conventional MR imaging processes 2D slices into a 3D stack. Newer approaches also adopted volumetric measurement to capture body structures in voxels rather than pixels and slices [1].

In recent years the amount of medical image data taken increased significantly. The conventional evaluation of these images is time-consuming and expensive as it requires an expert performing the manual search. To make this process more time-efficient and less prone to errors, it is highly relevant to establish a routine of running automated detection software across all types of brain scans independent of their original objective. A system which scans every medical image for any type of irregularities would greatly support radiologists and prevent overlooking conditions not showing symptoms. Especially smaller aneurysms are often overlooked, even when the medical image is examined by an expert. Additionally, a lot of patients have multiple aneurysms which remain undetected despite already being diagnosed with this condition.

MRI generally produces the best soft-tissue contrast and high spatial resolution which makes it most suitable for detecting abnormal structures like tumors, lesions or aneurysms [2]. Correct diagnoses, surgical planning or clinical trials require careful analysis of the scans, i.e. labeling of pixels or voxels by an expert. Manual execution of these processes is not only tedious and expensive but also prone to human error. As taking medical images of different modalities is becoming a routine in most diagnostic processes and

technological advances are increasing the size and complexity of scans, it is of critical importance to develop means for automated analysis.

1.2. Cerebral Aneurysms

Cerebral aneurysms are abnormal, weak or thin spots on arteries in the brain causing blood streams to widen an outward bulge into the form of a bubble or balloon. They tend to be located at the junctions between the main arteries supplying blood to the brain and surrounding structures that make up the Circle of Willis. The cause often stems from disease, injury or hereditary conditions [3]. Aneurysm types include fusiform and the more common saccular shape, with the latter being the focus of this project.

Aneurysms are not necessarily life-threatening. A lot of patients will never be affected if their aneurysm is stable and intervention may only introduce unnecessary complications. However, undetected conditions always pose a risk of rupture. The resulting spill of blood into the surrounding tissue may lead to serious complications including stroke, permanent nerve damage, or death. Most symptoms only emerge when the aneurysm ruptures and are rarely noticed in advance. Accordingly, most diagnoses happen coincidentally when brain scans are analyzed for a different condition. The type of intervention is chosen based on the patient's age, medical condition, size and location of the aneurysm and many more factors. The two options available are endovascular coiling and surgical clipping. The latter is an invasive method requiring a temporal removal of part of the skull and placement of a metal clip at the neck of the aneurysm which prevents further blood flow into the bulge. Endovascular coiling does not involve an incision in the skull but rather inserts platinum coils into the location of interest, causing a local blood clot on the inside of the aneurysm and therefore preventing rupture. While experts argue which treatment is less harmful and more effective in the long-term [4] [5], they agree that both pose a high risk either during the intervention or of a rebleeding at a later time.

1.3. Deep Learning Models

Deep Learning refers to the class of optimization methods which pass data through multiple layers of nonlinear transformations before returning an output. By repeatedly nesting simple mathematical functions, a complex representation of the original input is achieved [6]. Conventional machine

learning models like support vector machines (SVMs) or regression trees require hand-crafted features as input. Manual construction of these features to distinguish abnormal from normal structures based on shapes or locations is highly time-consuming and was seen as a major bottleneck in machine learning [7][8]. Therefore, the advances in medical image analysis were mainly driven by the introduction of Deep Learning algorithms as they are able to automatically extract complex representative features from the data. A few known examples include boltzmann machines, auto-encoders and convolutional neural networks (CNNs). Greenspan et. al [9] analyze recent studies in medical image analysis tasks and attribute the most promising results to CNNs, including detections of cerebral microbleeds [10] or segmentations of brain structures and lesions [11]. CNNs employ a hierarchical architecture of large amounts of stacked layers which convolve the input into an output volume. During this process low level features are extracted first and later combined into higher level features in the deeper layers. Therefore, it is not necessary to rely on handcrafted features, which in return greatly reduces the cost for obtaining training data.
The concept of deep neural networks was conceptually and structurally inspired by some of the actions in the visual cortex. Human visual neurons in the early layers are most sensitive to local information and respond to more complex receptive fields in the higher layers.

1.4. Convolutional Neural Networks

CNNs typically consist of repeating convolution, activation and pooling (sub-sampling) layers (see Fig. 1). During the convolution a filter kernel is moved across the entire input resulting in a smaller activation map. The activation indicates the spatial response of the image to the filter. In this way, local features such as an edge or corner can be detected multiple times in several locations by convolving a set of shared connection weights (i.e. a filter or kernel) across the entire image. The number of used filters determines the amount of produced activation maps and the size of each map depends on the stride at which the filter is moved. Afterwards, an activation function is applied which decides whether the threshold of activating the neuron for a specific feature was surpassed. Different functions can be used, but most commonly Rectified Linear Unit (ReLU) is chosen. ReLU leaves positive input values unchanged and raises negative ones to zero before passing them on as output. This step does not alter the size of the activation map and is

usually followed by a pooling layer which gradually reduces the size of the output. The objective of this layer is to prevent overfitting and reduce the complexity and size of each output slice by computing a maximum or average value for a specified size of neighborhood activations. This effect can also be achieved by introducing stride variables larger than 1 for the convolution and leaving out the pooling layer. To avoid full connections between nodes of subsequent layers every neuron is only connected to a subregion of the input image. This largely reduces the amount of weight parameters in need of training compared to a regular neural network with full connections between layers. After several repetitions of these layer combinations, class scores are predicted in the fully connected layer with each neuron having a connection to all neurons of the previous layer.

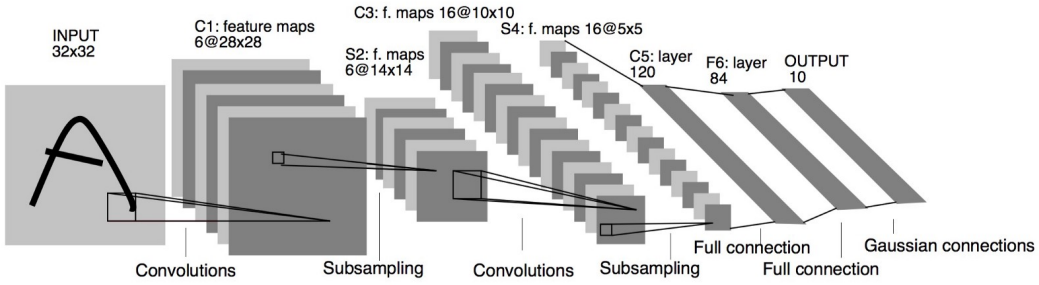

Figure 1: CNN Block Diagram [12]

Training. CNNs use backpropagation as an efficient training method minimizing a loss function between the true and predicted label. The weights are randomly initialized and incrementally improved using an optimization algorithm. Oftentimes, stochastic gradient descent (SGD) is chosen for finding the minimum of the loss function. This approach implements an online version of the gradient descent method calculating the derivative of the error function and therefore the direction of the gradient. A chosen learning rate determines how many steps the weights are adjusted per training sample towards the direction of the estimated minimum. In one iteration the input values are passed forward through the network and then passed backward by recursively computing the gradients and adapting the weights accordingly. A deep model with a large number of layers is usually confronted with the

problem of vanishing gradients. This is caused by passing the error function to update the weights backwards in the form of a partial derivative. Most activation functions are bound within [0,1] and these ranges only decrease with repeatedly deriving the function. Ultimately this can result in values so small that the change of weights barely has an effect anymore and the learning process in the front layers becomes increasingly slow. This problem is often handled by applying different activation functions, such as ReLU, as the value of its derivative for exceeding the neuron threshold is a constant.
Due to the large number of parameters and small amount of training examples the training might converge to an undesirable local minimum. This problem could be counteracted by fine-tuning, i.e. using weights as a starting point which were pre-trained on a different, but similar task. For this reason, many medical deep learning platforms, such as NiftyNet [13] or V-net [14] keep public databases of available pre-trained models for reuse. The more similar the images used for training the models in regard of the modality used for scanning, pictured body structure and tissue type, the more similar the low-level features, which could potentially be reused.

1.5. Deep Learning Tasks in Medical Imaging

Automated medical image analysis pursues many different tasks with the most common ones being classification, detection and segmentation. Classification takes an image as input and categorizes it among a finite set of medical variables such as disease present or absent. The label prediction is derived from the most dominant class score. For example, Anthimopoulos et. al [15] used a CNN to classify patient CT images into one of 7 different Interstitial Lung Disease pattern categories, including one for a healthy state.
Taking this a step further, object detection finds all objects of given classes and their location. A potential application would be to find all lesions of a certain type marked by their bounding box. More specifically, in [16] the authors also employ deep learning algorithms to locate the central pixel of cell nuclei in histopathological images.
Moreover, segmentation divides the image into parts belonging together, for example outlining different organs or tissue types. This can be done by training a CNN on patches of the original image to classify and afterwards combine into a segmentation [17][18] or by using a fully connected or dense architecture to make predictions for each pixel of the entire image [19].

2. Detecting Cerebral Aneurysms using Deep Learning

2.1. Task Description

The task for this project was to build and train a convolutional neural network that can be used for an automated detection of cerebral aneurysms on MRI data. The given database included 35 patient brain scans and expert information on size and location of the aneurysm to be used for building a ground truth. The software should recognize an aneurysm and output its predicted location and shape. The results are to be visualized in a user-friendly and inituitive way.

2.2. Related Work

Cerebral aneurysm detection is an important research topic. Earlier approaches used SVMs [20], spatial filtering [21] and a baysian classifier [22] with mostly promising results.
More recent studies implemented deep convolutional neural networks [23] to segment blood vessels, whereas work using this technique on the actual detection of the aneurysms is rare. Comparing results is difficult due to the different databases, metric definitions and overall objective. For their intracranial detection framework Malik et. al [24] applied a multi-layer perceptron approach on predetermined Harlack texture features and achieved very high evaluation measures (sensitivity of 0.97 and specificity of 0.99) on their data. Some works use computer-aided diagnosis (CAD) algorithms which employ segmentation techniques as one of the steps. In [25] the authors applied a CAD algorithm on a comparably small dataset of 48 patients with rather small aneurysms and achieved a recall value of 0.76. Another promising approach to detection was introduced by Hentschke et. al [26] who integrated aneurysm characteristics into their feature computation and experimented with different non-linear classification algorithms.
As mentioned earlier, the detection of aneurysms is closely related to the estimation of the risk of rupture. For this objective image analysis techniques were developed [27][28] which separate the aneurysm from the vessel to determine morphological features (such as angle or size). These features were then used to build classifiers. Another approach included the analysis of simulated blood flow and identification of typical patterns [29]. For cases when the aneurysm condition requires an intervention, research was done on predicting the success of treatment[30]. The system to be implemented for this

project could be embedded as an early base to ensure reliable identification of aneurysms for further analysis in more detail.

2.3. Employed Hardware

The implementation was done on a computer running Ubuntu 18.04 with an AMD six-core-processor (3.6 GHz) and a GeForce GTX 1080 Ti (16 GB) in the *Forschungscampus STIMULATE*. The repository containing our code can be found on Github[1].

2.4. Dataset Description

We used a dataset of 35 MRI scans acquired at the university hospital of Magdeburg of patients who were diagnosed with cerebral aneurysms. The data was supplied in 'Digital Imaging and Communications in Medicine' (DICOM) format which is the standard for managing and sharing medical images [31]. The data included expert information on the size and location of each aneurysm in tabular format.

The scans were conducted using different imaging techniques, precisely contrast-enhanced MRA (CE-MRA) and time-of-flight MRA (TOF-MRA). CE-MRA uses a contrast agent to enhance the visibility of vascular structure, which is especially useful to detect tumors, vascular lesions [32] or aneurysms, respectively. The main objective of TOF imaging is to visualize flow within vessels, without the need to administer contrast agent. This is helpful for patients who cannot tolerate the agent but might result in poorer image quality. As the TOF approach largely depends on a certain flow of blood, which is rather slow in aneurysms, it might lead to artefacts in the image. Therefore, CE should be preferred for our task, but due to the lack of data TOF scans were also used in the training process. On average the aneurysms had grown to a diameter of 7.86 mm, with a standard deviation of 5.85 mm, minimum size of 2.6 mm and maximum size of 25.6 mm.

2.5. Preprocessing

Preprocessing aims to enforce a certain baseline of standardization upon the data to ensure the model is able to distinguish between relevant and irrelevant features of the dataset. Most of the used data was recorded within a tolerable range of resolution giving a pixel spacing between 0.4 x 0.4 mm and

[1] https://github.com/john-kloss/aneurysmDetection

0.6 x 0.6 mm. The few scans which exceeded these restrictions were resampled to the closest boundary using MeVisLab's[2] Resample3D module. This was followed by a recalculation of the world coordinates for the aneurysm locations which we updated using the WorldVoxelConvert module.

As the implementation was done using Python 3.6.5, all the DICOM files were imported with pydicom[3] to access the image data. In the next step the location of the aneurysm(s) was/were added to each resulting object. It is important to note that the original dimensions were not consistent with the imported ones and thus required additional adaptations.

2.6. Mask Creation

The patients had aneurysms of different locations and sizes. In order to identify this area as distinctly as possible, a mask was created. The dataset included a list specifying coordinates and diameter of each aneurysm diagnosed for the patient. As shown in Figure 2, we constructed a 3D mask of the same shape as the input image. In order to avoid manually labeling each aneurysm, we assigned ones signaling the presence of the aneurysm by laying a bounding sphere of the given diameter on top of the mask. Ones signaled the presence of an aneurysm and zeros the absence.

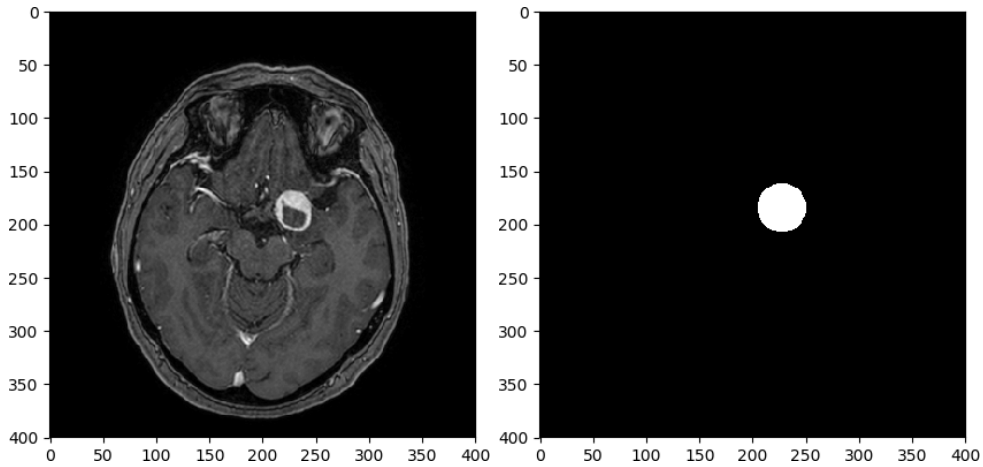

Figure 2: Example of brain scan (left) and corresponding created mask (right).

[2]https://www.mevislab.de/
[3]https://pypi.org/project/pydicom/

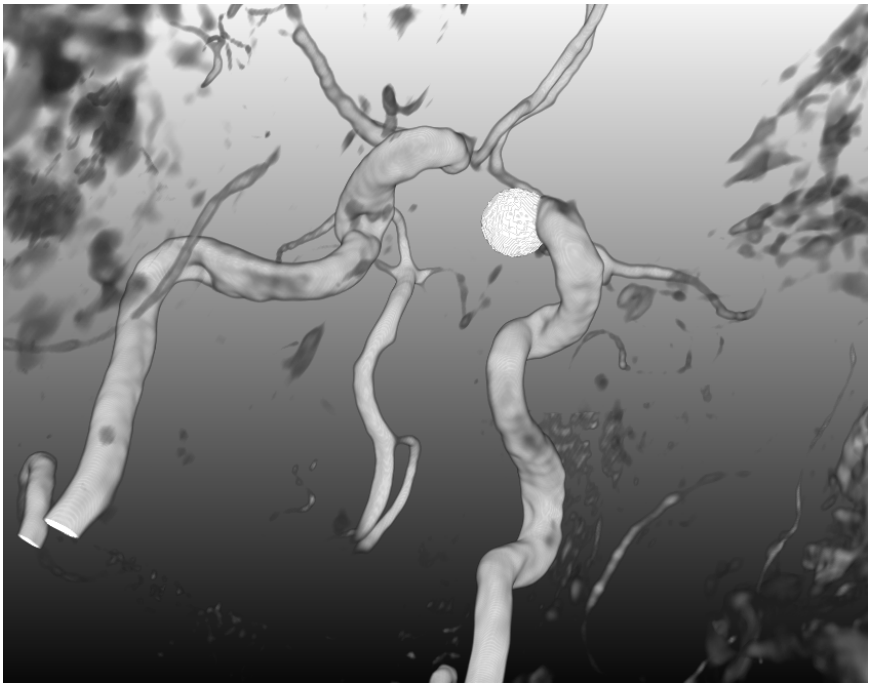

Figure 3: Subvolume image and mask plotted together in 3D MeVisLab volume rendering view.

2.7. Data Augmentation

Due to large amounts of weights which have to be learned during the training of a CNN (see Section 2.9) a high number of training samples are required to sufficiently train the network and simultaneously prevent overfitting. This problem intensifies in medical image analysis research because of the cost-intensive nature of image recordings and consultation of experts for a ground truth. While generating natural images is comparably cheap and datasets are abundantly available, medical datasets are usually restricted for distribution.

These limitations are met with different data augmentation techniques which artificially increase the size of the dataset. Common approaches for grayscale images mostly include affine transformations like rotation or scaling of the images. In [33], different strategies were compared with respect to accuracy in medical image classification tasks. The augmentation types which yielded the highest accuracy scores were adopted for our dataset and

built into our own pipeline of transformation. For the 3D image preprocessing steps we made use of the extension packages skimage [4] and transforms3d[5] which break the task down to matrix transformations. Within specified ranges we randomly chose transformation parameters. A total of 9 transformations per patient were performed on each medical image and mask. As some transformations also slightly changed the numerical values of the masks, additional integer rounding was applied after each augmentation to preserve the original ground truth labels of 1 and 0.

Rotations. We kept the z-axis fixed and rotated around the x-y plane within a range of [-15,15] degrees.

Shearing. Shearing was randomly applied to the scans within a very small range of [0, 0.05] to keep the brain structures recognizable.

Scaling. The size of the image was scaled up or down by a random factor of up to 0.5. As the scaling distorts the size of the voxel spacing only mildly, we considered it to be in the acceptable boundaries without any resampling.

Grayscale Normalization. After each augmentation the images were normalized to be within the range of [0,1] and guarantee equal activation values for similar transitions in brain structures.

We constructed a flow chart to visualize the steps in our data preprocessing routine to be seen in Figure 4.

2.8. Dataset Generation

Choosing the size of input for the CNN is critical. While it is extremely important to generate subvolumes which are large enough for the network to recognize complex features of the image and guarantee a high level of abstraction, the trade-off of computational expenses cannot be neglected. We ultimately settled for a subvolume size for the input data of (64,64,64). As part of the preprocessing pipeline the patient data was cropped into training samples by generating coordinates for the subvolume centroids scattered around the aneurysm. These locations are randomly drawn out of a truncated gaussian distribution to ensure some random variety in the patches

[4] https://scikit-image.org/

[5] https://github.com/matthew-brett/transforms3d

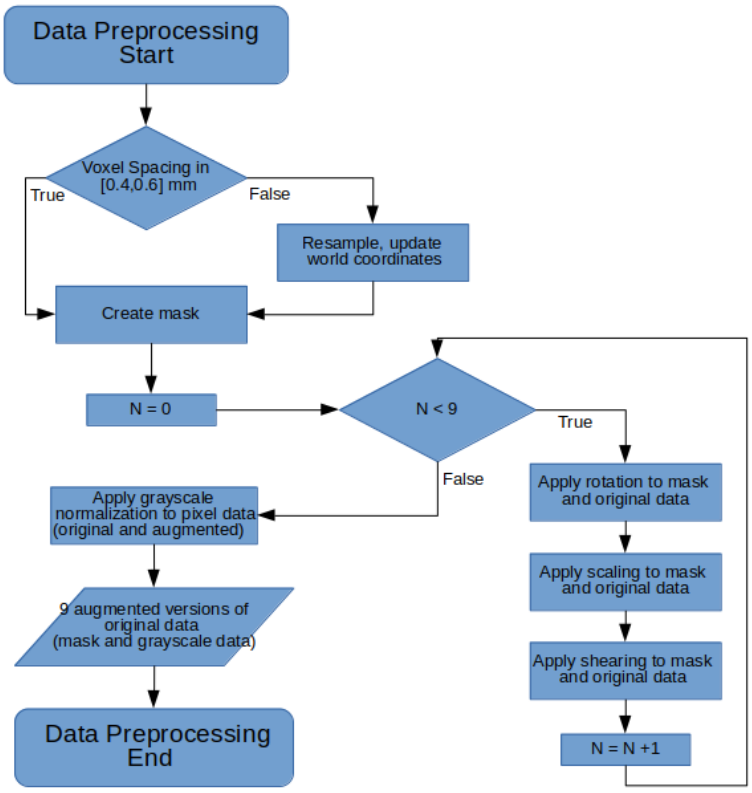

Figure 4: Flowchart of the applied preprocessing steps.

while still maintaining the entire aneurysm within the subvolume. Specifically, the distribution was defined for each dimension of the aneurysm coordinates with the coordinate as the central point and a cutoff to each side to keep within image bounds. We repeated this process for each of the voxel coordinates, thus generating sufficient variety within the chosen subvolume centroids. The standard deviation of the distribution was set to be so small, that the aneurysm was always fully covered in the subvolume. The described centroid generation procedure is visualized in Figure 7. We produced 15 patches for each of the nine augmented versions and 90 (15 * 6) for the original image to keep a balance between the amount of subvolumes from original and augmented data, leading to a total of 225 input training samples per patient. For patients diagnosed with multiple aneurysms the cropping process was executed for each aneurysm, respectively. However, the amount of

subvolumes was divided by the number of aneurysms to keep the coverage a patient sample received during training at a steady ratio. Figure 5 represents a visualization of the cropping and flowchart 6 shows the sequence of applied actions.

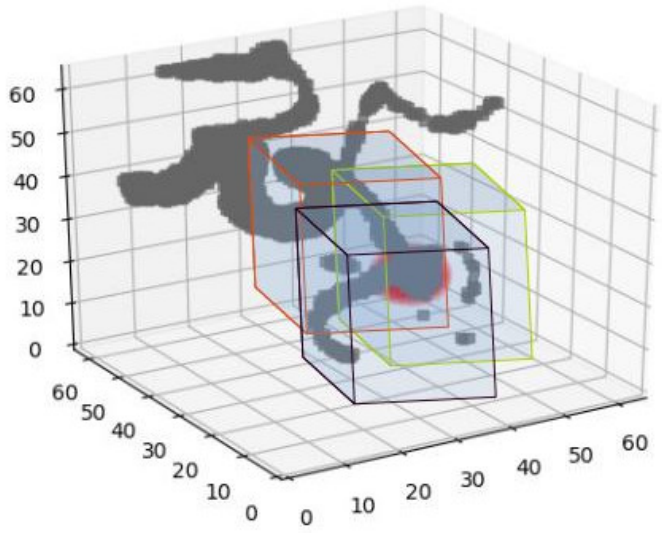

Figure 5: Visualization of the cropping process.

Dataset split. As suggested by experienced researchers within the *Forschungscampus STIMULATE* we chose a 80-10-10 split for our data, dividing it into 80% training, 10% validation and 10% test set on patient level. As our dataset could not be evenly partitioned we opted for 3 validation and 4 test patients. Thus the network was trained on different patients than it was tested on to prevent overfitting.

2.9. Model Architecture

The CNN model architecture should be chosen according to the task. We selected the U-net architecture [34] as it has proven to work well on large medical datasets (see [35], [36] and [37]). Figure 8 displays the default network architecture. On the left side the input image is fed into the network. Afterwards the data is propagated through the paths, producing a heatmap

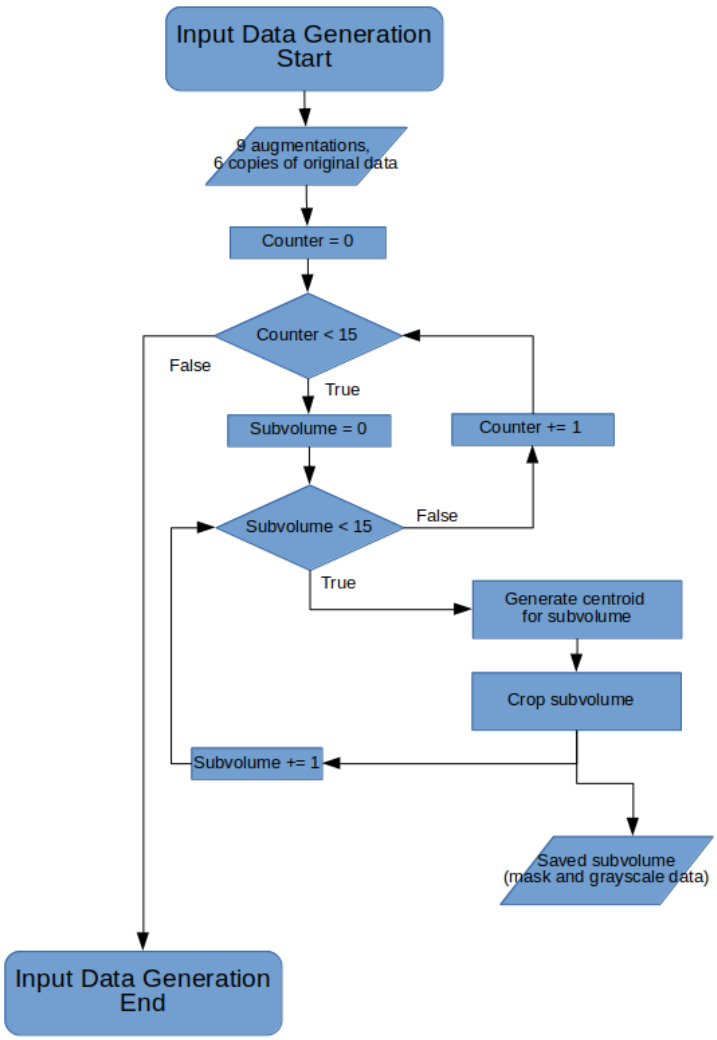

Figure 6: Flowchart of dataset generation process which is repeated for every patient dataset. The action of generating a centroid is explained in more detail in Figure 7.

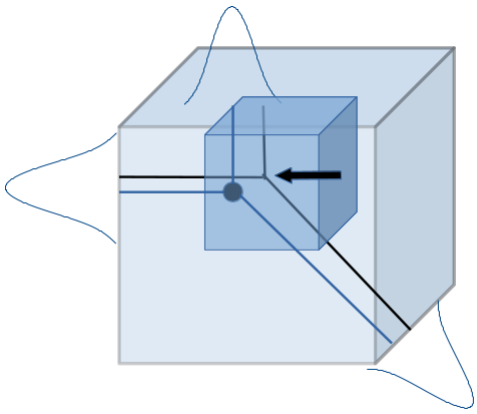

Figure 7: Visualization of the subvolume centroid generation process. The arrow indicates the new centroid drawn from three distributions centered around the aneurysm coordinates. By keeping the standard deviation small this method introduces some variety while keeping the aneurysm fully covered within each subvolume.

per channel of the same size as input.
The network is divided into a contracting and an expansive path with the first one performing down-convolution, pooling and activation operations like any regular CNN. The resulting low-resolution representation of the image is then up-sampled in the second part of the network. De-convolutions are followed by concatenations of correspondingly cropped features from the first path. In our single channel task the output is one 3D heatmap of the same size as the input image with each voxel value representing the predicted label. For the final outcome the subvolume outputs were reconstructed to the size of the original image. Values higher than 0.7 were considered as confident predictions.

For the implementation of the model we chose Keras[6] with Tensorflow as backend and adjusted publicly available code[7] for the architecture to fit our task. We experimented with different parameter settings in rather short training sessions and based on the development of the training losses ultimately settled for the following settings:

[6]https://keras.io/
[7]https://github.com/ellisdg/3DUnetCNN

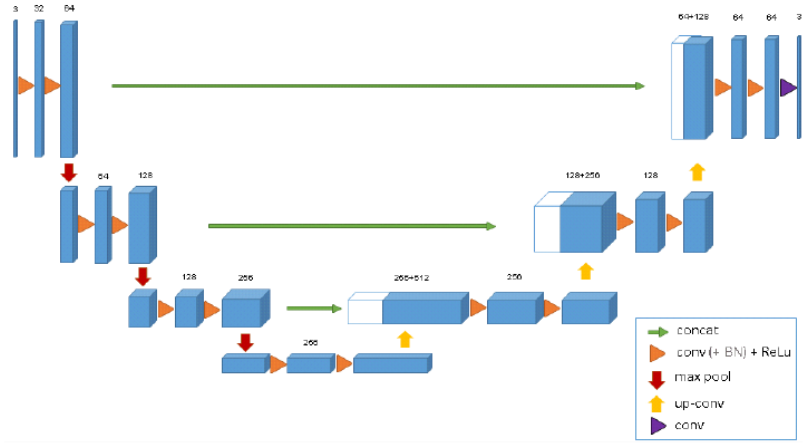

Figure 8: U-net architecture as presented in [38].

- depth 3
- 11 convolutional and activation layers
- 2 pooling layers of size 4
- 2 up-sampling and concatenation layers
- 32 input kernels
- stride 1 during convolution

This produced a total of 3,928,001 trainable parameters. A summary of our architecture can be seen in Appendix A. We chose ReLU as an activation function because of its relative robustness to vanishing gradients (see Section 1.4) and general applicability.

2.10. Training

2.10.1. General Training Settings

The training process is divided into epochs and iterations. One epoch consisted of one forward and backward pass of the entire dataset and is followed by a short validation phase. The batch size determines the number of iterations during an epoch and therefore the amount of weight updates. We fixed the batch size at one, turning the training into online learning as the

weight parameters were adjusted after each training sample was processed. In [39], it was pointed out that this stochastic learning approach is much faster than batch learning, especially when the dataset is composed of multiple versions of the same images like it is the case here.

The network was trained on 35 patient scans with different settings for 35 epochs each, summing up to a total training time of approximately 10 hours. The number of epochs were chosen purely based on the processing time, however, a higher number of epochs is desirable. Included in one patient dataset were a total of 225 patches out of which 135 originated from different augmentations and 90 were cropped from the original. The input consisted of the generated subvolumes (as described in Section 2.8) and the corresponding masks as ground truth labels (outlined in Section 2.6). The validation set included three patients and was handed to the model separately to make sure the dataset split stayed consistent.

2.10.2. Metrics and Loss Function

A loss function represents the compatibility between our ground truth and the prediction. During training the objective function is minimized and the weights are adjusted accordingly. Therefore, the improvement of the model is largely dependant on the loss function as a representative of the quality of the prediction. We experimented with the two most common loss functions in image segmentation tasks:

- pixel-wise cross-entropy
- dice coefficient

The cross-entropy evaluates predictions for each voxel individually and strongly penalizes misclassification. It always returns a positive value and converges towards zero as the confidence of the prediction increases. False predictions with a high confidence suffer from the largest penalties. Cross-entropy enables fast computations, because of the straightforward derivation of logarithmic functions. The formula is shown in Equation 1 where p is the predicted probability and y the true label.

$$Loss = -(y \cdot log(p) + (1 - y)log(1 - p)) \tag{1}$$

Since class imbalance is not considered, it is necessary to introduce a weighting factor for highly skewed datasets. As the aneurysms account for only a minimal fraction of the input volume, we tried different weight settings and ultimately chose 0.95 and 0.05 for true and false, respectively. To integrate these weights we implemented a wrapper function that multiplies the loss with a weight vector as seen in Equations 2 and 3.

$$Loss_{weighted} = V_{weight} * Loss \tag{2}$$

$$V_{weight} = y * 0.95 + (1 - y) * 0.05 \tag{3}$$

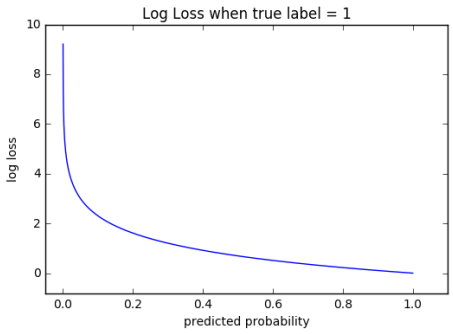

Figure 9: Range of possible loss values [40]

As it was shown in Table 1 we also tested a loss function based on the dice coefficient. This metric directly computes an overlap between ground truth and prediction. Theoretically it performs better with imbalanced datasets, which should be the case for our data.

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{4}$$

We took both objective functions into account and compared the results. The outcome revealed to be more successful for training with a cross-entropy loss function and was therefore used in the further process.

For the final evaluation we additionally contemplated recall and precision. Recall measures the amount of detected instances among all relevant ones and precision expresses the fraction of found aneurysms in relation to the total number of structures classified as aneurysms. The relevance of these metrics greatly differs with the purpose of the application. The cost of missing an

aneurysm is significantly higher than a false positive which can be discarded by an expert afterwards. Therefore, a high recall value is desirable. By itself it can misleading since declaring everything an aneurysm would achieve a perfect recall value but the application would have no practical use. As a result, precision is also monitored to ensure an appropriate ratio.

$$Precison = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

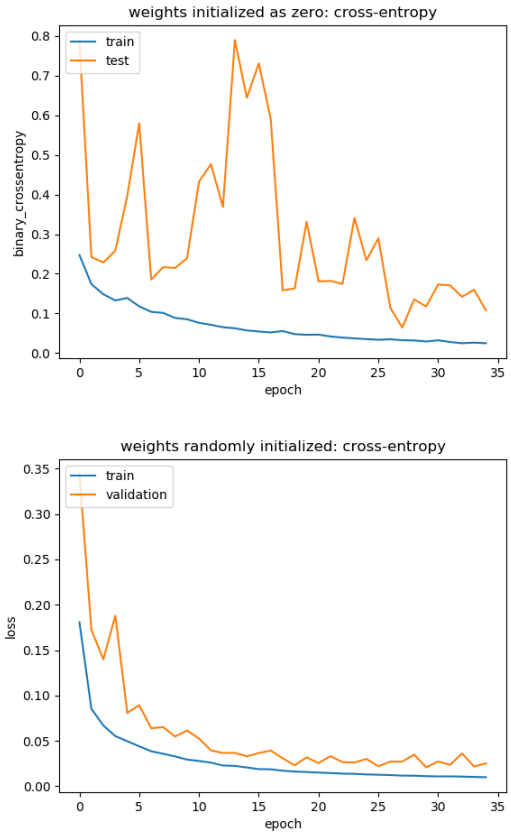

Figure 10: Comparison of loss function for training initialized with zero and random weights.

2.10.3. Experimental Training Parameters

The model trained using stochastic gradient descent (see Section 1.4) as implemented in the Keras optimizer Adam[8]. As stated in [41] this method is "computationally efficient, has little memory requirements [...] and is well suited for problems that are large in terms of data and/or parameters." Additionally, Adam handles learning rate optimization by computing adaptive learning rates. Therefore, we did not experiment with different settings for the learning rate and fixed it at the value of 0.00001, which was recommended as default in the mentioned code repository[9] to be low enough for the network to converge to an optimum but still high enough to keep the training time within acceptable terms. As it is difficult to choose the optimal parameters from all available network settings, we ran multiple trainings and saved the weights to compare the resulting predictions afterwards. In Table 1 the different parameter settings are presented.

Name	Epochs	Weighting	Loss function	Bias Initializer	Depth
Default	35	0.95/0.05	Cross-entropy	zeros	3
Dice	35	-	Dice Coefficient	random	3
Weighting	35	0.9/0.1	Cross-entropy	random	3
Depth	35	0.9/0.1	Cross-entropy	random	4
Regularized	35	0.95/0.05	Cross-entropy	random	3

Table 1: Different training setups

While the initial training used a zero bias initializer for the starting weight, in [42] it is suggested to use a random distribution. This change accelerated the search for the correct weights and rightaway skipped to a more fluent start (see Fig. 10). Therefore the starting weights for the subsequent runs were normally distributed.
Furthermore, the author advises to shuffle the input data during the training which we adopted as well.

[8]https://keras.io/optimizers/adam
[9]https://github.com/ellisdg/3DUnetCN

Moreover, the increase of the depth of the network seemed to be a viable option to enhance the predictions. A comparison of the results showed no distinct improvements which is why the depth remained at 3.

2.11. Evaluation

As it was stated earlier, 10% of the patients were reserved for the test set. It was generated by cutting the central 80% of the DICOM data into smaller images of size (64x64x64) while ensuring to preserve the whole aneurysm in one of these subvolumes. Finally, the network with the previously trained weights was used to predict our test data.

During the evaluation each subvolume was processed individually. If it (as indicated by the ground truth) contained an aneurysm and the dice coefficient or cross-entropy was sufficiently high, the image was counted as a true positive. We specifically used the dice coefficient for evaluation despite not using it in the loss function because it is a good measure for overlap. However, if a prediction of correct size and shape is located just a little to the side of the actual aneurysm the dice coefficient falls to zero, although the prediction still hints at the correct area. Therefore, we also took cross-entropy into account. In case of an underprediction, it was classified as a false positive.
In the event that there was no aneurysm, the confidence of the network was checked globally (by the number of as true predicted voxels) and by local proximity of activations. The local criteria was based on the fact that only close voxels can represent a confidently predicted aneurysm shape and activations scattered across the image were disregarded as random noise. Given both metrics reached an adequate level, the prediction was counted as a false positive. All other cases were treated as true negatives.

2.12. Results

The 'default' setting performed best during the prediction. All subsequent assumptions are based on this specific setting.
We achieved a recall of 0.7 and precision of 0.2. The low precision value stems from the higher penalty assigned to missing an aneurysm versus generating false positives. In order to put each predicted subvolume into the relation of the entire brain we plotted the most confident labels on top of the original image and saved it back into a DICOM format for better visualization

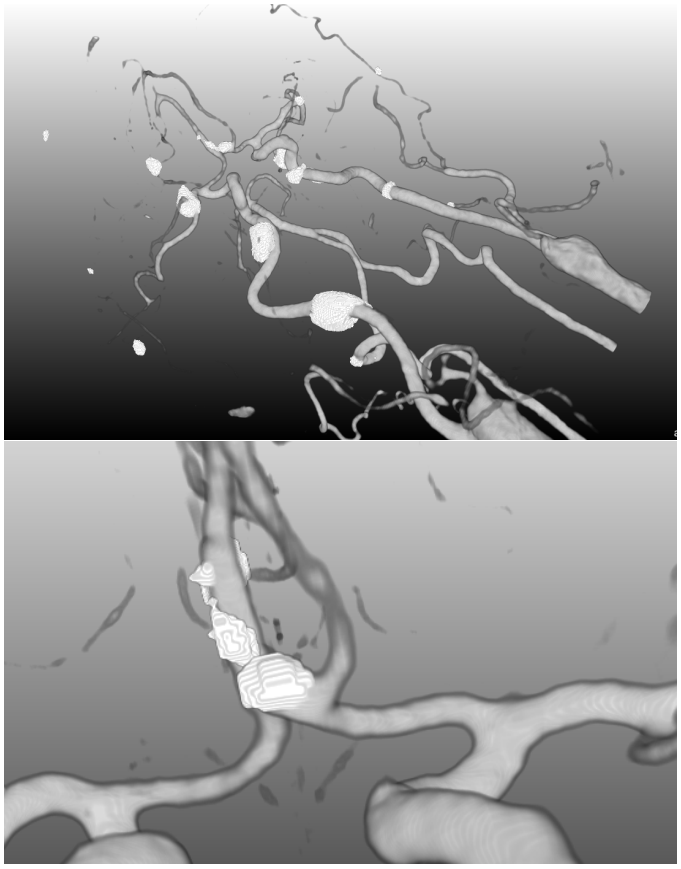

Figure 11: Visualization of predictions plotted on top of original image. The whitest areas represent the predictions. Top - full image, bottom - close up of aneurysm

(see Fig. 11). The visualization reflects the high number of false positives, however, the aneurysm is also detected in most cases, which is of higher significance. As stated earlier, (see Section 2.10.2) it is more important to generate a high recall value since the system acts as a support tool for the radiologist for whom it is more efficient to discard false positives rather than search for missing aneurysms.
It is hard to compare these values to similar works, as there are few studies using the same types of deep learning algorithms on cerebral aneurysms. As outlined in Section 2.2 the few that could be compared achieved mostly higher metrics.

As it was shown earlier the loss decreased constantly during the training

although the predictions on the test set were not satisfactory. This was most likely caused by overfitting onto the training and validation data. This effect could be prevented by using more data but also reducing the amount of training samples per patient. We initally generated around 1000 subvolumes per patient but then chose a lower number which, however, did not change the results too much.

3. Limitations

A major problem during the training was the limited amount of available data. Compared to [26] who used a database three times the size of ours, the results are less accurate.
Although the number of training samples was increased by augmentation, it is hard to judge whether the variety of the resulting images is sufficient or not. The network might overfit onto the data during the training and is not be able to generalize on the learned shapes to predict different images. On the other hand, too many alteration might diverge the images too far from the original which may cause the network to learn from unrealistic data.

Another limiting factor was the time constraint and access to a high-end computer equipped with a GPU, as these resources are in high demand and could only be used in a restricted time period. We assume that a deeper and therefore more complex model requires more time to sufficiently train the increased number of parameters. Due to a lack of time we were not able to execute trainings for a much higher number of epochs which is why we did not proceed with a model of higher complexity.

In the visualizations of the test predictions we witnessed a tendency towards large false positives while the subvolumes which contained an aneurysm were predicted with approximately the correct size. The patients assigned to the test set were chosen randomly but post-analysis revealed that the average aneurysm size in the training and validation set was at about 8,045 mm while the average aneurysm in the training set had a diameter of 6,18 mm. However, a t-test showed that the difference was not significant ($p = 0.5$) which lead us to reject this hypothesis and believe that this could not have been a major artifact.

4. Future Outlook

Temporal and performance aspects The trend of the loss function in the Figure 9 shows that the training has not yet converged and more epochs will result in smaller loss and therefore better results. Furthermore, additional data would help to prevent overfitting on the currently very small number of different aneurysm shapes, structures and locations.

Additionally, there are pre-trained models available which have been used for other medical image detections or segmentations as outlined in section 1.4. These weights might help skip the inital slow period and further speed up the training process.

Implementing data augmentation in real-time would address our problem of limited disk space and accelerate the process due to the elimination of reading from disk.

Training parameter optimizations Selecting the correct parameter values is of critical significance in the entire process as potential for optimization is rooted in each step. Choices for the preprocessing step include transformation ranges, amount of generated subvolumes or augmentations, size of the patches and dataset split. An even greater number of settings are to be configured during the construction of the model: depth, padding, max pool size, learning rate, learning rate decay, number of epochs, activation function, number of filters etc. Experimenting with different loss functions could have also yielded better results. As described in [43] there are many different options available.

Future research should test a sufficient amount of possible combinations of settings. A potential approach could involve a randomized grid search across different parameter settings for each stage of the process.

Further improvement is achieveable by cropping larger subvolumes of (128x128x128). The relatively small size of 64 might have been a limiting factor, because the model was not able to properly generalize across global features for larger aneurysms which cover a large quantity of the subvolume space.

During the convolution our implementation evenly padded its output images with zeros to maintain the same size. When using subvolumes as input data to the network this is critical, because it artificially creates grayscale value transitions at the borders. A better approach would be to use no

padding to prevent false predictions at the bounds of a subvolume. Furthermore, experimenting with the number of filters can yield improvement as this implies a different number of generated feature maps. We kept this value at 32 for the entire training.

Real-world application In a real-world application, the data input would be quite easy. The DICOM file format is widely used and can be fed into the program. The time to create a prediction for the full scan on an average computer is approximately five minutes. The computation should be integrated into the medical imaging process and automatically launched at the end, for the radiologist to consult at a later time. The output can be defined to be at certain level of detail and information as the system produces both, prediction per subvolume, as well as a global prediction. The global version includes an image in DICOM format with all predicted aneurysms plotted on top of the original data. This output format can be easily visualized and quickly analyzed by the radiologist. Another feature to consider for future usage would be the manual inclusion and exclusion of brain parts, as e.g. the outer skull or border area of the image is rather irrelevant. Additionally, it could be useful to integrate a transparency parameter for the predicted aneurysms to toggle the visibility of the actual blood vessels underneath the prediction.

5. Conclusion

We implemented a system that takes medical data as input and outputs the same image with marked areas of possible aneurysms. The predictions can be viewed as areas of high likelihood and included almost all present aneurysms. Thus, our system supports a medical doctor who can check the specified regions instead of evaluating the entire dataset. This is of high importance as it is easier to check these specific regions manually compared to the whole MRI scan. This feature could be extended by displaying the regions predicted to contain an aneurysm ranked by their confidence. The chance of missing an aneurysm especially in case of multiple occurrences decreases heavily. However, the final metrics for recall of 0.7 and precision of 0.2 are yet to be improved. In future works the network should complete more training epochs and process more data during the training. This will achieve more accurate results, as the trend of the loss function is indicating. Especially the widely standardized data formats for input and visualized

output are beneficial for end-users. We consider this system to be a useful tool to be refined for support in the diagnostic process.

Bibliography

[1] R. Tongdee, V. Narra, E. Oliveira, W. Chapman, K. Elsayes, and J. Brown, "Utility of 3d magnetic resonance imaging in preoperative evaluation of hepatobiliary diseases," *HPB*, vol. 8, no. 4, pp. 311–317, 2006.

[2] Z. Akkus, A. Galimzianova, A. Hoogi, D. L. Rubin, and B. J. Erickson, "Deep learning for brain MRI segmentation: State of the art and future directions," *Journal of Digital Imaging*, vol. 30, no. 4, pp. 449–459, 2017.

[3] J. Novitzke, "The basics of brain aneurysms: A guide for patients," *Journal of Vascular and Interventional Neurology*, vol. 1, no. 3, pp. 89–90, 2008.

[4] L. N. Williams and R. D. Brown, "Management of unruptured intracranial aneurysms," *Neurology: Clinical Practice*, vol. 3, no. 2, pp. 99–108, 2013.

[5] P. V. Raja, J. Huang, A. V. Germanwala, P. Gailloud, K. P. Murphy, and R. J. Tamargo, "Microsurgical Clipping and Endovascular Coiling of intracranial aneurysms, journal = Neurosurgery," vol. 62, no. 6, pp. 1187–1203, 2008.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016. http://www.deeplearningbook.org.

[7] R. M. Summers, "Deep learning and computer-aided diagnosis for medical image processing: A personal perspective," in *Deep Learning and Convolutional Neural Networks for Medical Image Computing*, pp. 3–10, 2017.

[8] G. Carneiro, Y. Zheng, F. Xing, and L. Yang, "Review of deep learning methods in mammography, cardiovascular, and microscopy image analysis," in *Deep Learning and Convolutional Neural Networks for Medical Image Computing*, pp. 11–32, 2017.

[9] H. Greenspan, B. van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[10] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P.-A. Heng, "Automatic detection of cerebral microbleeds from MR images via 3d convolutional neural networks," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1182–1195, 2016.

[11] T. Brosch, L. Y. W. Tang, Y. Yoo, D. K. B. Li, A. Traboulsee, and R. Tam, "Deep 3d convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation," *IEEE Transactions on Medical Imaging*, vol. 35, pp. 1229–1239, May 2016.

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

[13] E. Gibson, W. Li, C. Sudre, L. Fidon, D. I. Shakir, G. Wang, Z. Eaton-Rosen, R. Gray, T. Doel, Y. Hu, T. Whyntie, P. Nachev, M. Modat, D. C. Barratt, S. Ourselin, M. J. Cardoso, and T. Vercauteren, "NiftyNet: a deep-learning platform for medical imaging," *Computer Methods and Programs in Biomedicine*, vol. 158, pp. 113–122, 2018.

[14] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016.

[15] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1207–1216, 2016.

[16] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. J. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[17] K. Kamnitsas, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient multi-scale 3d CNN with fully connected CRF for accurate brain lesion segmentation," *Medical Image Analysis*, vol. 36, pp. 61–78, 2017.

[18] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain tumor segmentation using convolutional neural networks in MRI images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1240–1251, 2016.

[19] D. Nie, L. Wang, Y. Gao, and D. Sken, "Fully convolutional networks for multi-modality isointense infant brain image segmentation," in *IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 2016.

[20] S. Hanaoka, Y. Nomura, M. Nemoto, S. Miki, T. Yoshikawa, N. Hayashi, K. Ohtomo, Y. Masutani, and A. Shimizu, "HoTPiG: A novel geometrical feature for vessel morphometry and its application to cerebral aneurysm detection," in *MICCAI 2015. Lecture Notes in Computer Science, vol 9350*, pp. 103–110, 2015.

[21] J. Mitra and A. Chandra, "Detection of cerebral aneurysm by performing thresholding-spatial filtering-thresholding operations on digital subtraction angiogram," in *Advances in Computing and Information Technology*, pp. 915–921, 2013.

[22] J. Wu, G. Zhang, Y. Cao, and Z. Cui, "Research on cerebral aneurysm image recognition method using bayesian classification," *Proceedings of the 2009 International Symposium on Information Processing (ISIP'09)*, vol. 21-23, pp. 58–62, 2018.

[23] R. Phellan, A. Peixinho, A. Falcão, and N. D. Forkert, "Vascular segmentation in TOF MRA images of the brain using a deep convolutional neural network," in *LABELS 2017, STENT 2017, CVII 2017 Lecture Notes in Computer Science*, pp. 39–46, 2017.

[24] K. M. Malik, S. M. Anjum, H. Soltanian-Zadeh, H. Malik, and G. M. Malik, "A framework for intracranial saccular aneurysm detection and quantification using morphological analysis of cerebral angiograms," *IEEE Access*, vol. 6, pp. 7970–7986, 2018.

[25] H. E. Hamdaoui, M. Maaroufi, B. Alami, N. E. Chaoui, and S. Boujraf, "Computer-aided diagnosis systems for detecting intracranial aneurysms using 3d angiographic data sets: Review," in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2017.

[26] C. M. Hentschke, O. Beuing, H. Paukisch, C. Scherlach, M. Skalej, and K. D. Tönnies, "A system to detect cerebral aneurysms in multimodality angiographic data sets," *Medical Physics*, vol. 41, no. 9, p. 091904, 2014.

[27] U. Niemann, P. Berg, A. Niemann, O. Beuing, B. Preim, M. Spiliopoulou, and S. Saalfeld, "Rupture status classification of intracranial aneurysms using morphological parameters," in *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 48–53, IEEE, 2018.

[28] S. Saalfeld, P. Berg, A. Niemann, M. Luz, B. Preim, and O. Beuing, "Semiautomatic neck curve reconstruction for intracranial aneurysm rupture risk assessment based on morphological parameters," *International journal of computer assisted radiology and surgery*, pp. 1–13, 2018.

[29] M. Meuschke, S. Voß, B. Preim, and K. Lawonn, "Exploration of blood flow patterns in cerebral aneurysms during the cardiac cycle," *Computers & Graphics*, vol. 72, pp. 12–25, 2018.

[30] P. Berg, S. Saalfeld, G. Janiga, O. Brina, N. M. Cancelliere, P. Machi, and V. M. Pereira, "Virtual stenting of intracranial aneurysms: A pilot study for the prediction of treatment success based on hemodynamic simulations," *The International journal of artificial organs*, p. 0391398818775521, 2018.

[31] "Dicom standard." https://www.dicomstandard.org/, 2018. Accessed: 2018-09-24.

[32] A. Staub, J. Schappler, S. Rudaz, and J.-L. Veuthey, "CE-TOF/MS: Fundamental concepts, instrumental considerations and applications," *Electrophoresis*, vol. 30, no. 10, pp. 1610–1623, 2009.

[33] Z. Hussain, F. Gimenez, D. Yi, and D. Rubin, "Differential data augmentation techniques for medical imaging classification tasks," *AMIA Annual Symposium Proceedings*, vol. 2017, pp. 979–984, 2017.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Bildverarbeitung für die Medizin 2017. Lecture Notes in Computer Science*, pp. 234–241, 2015.

[35] F. Isensee, P. F. Jaeger, P. M. Full, I. Wolf, S. Engelhardt, and K. H. Maier-Hein, "Automatic cardiac disease assessment on cine-MRI via time-series segmentation and domain specific features," in *ACDC and MMWHS Challenges. STACOM 2017. Lecture Notes in Computer Science*, pp. 120–129, 2018.

[36] F. Isensee, P. Kickingereder, D. Bonekamp, M. Bendszus, W. Wick, H.-P. Schlemmer, and K. Maier-Hein, "Brain tumor segmentation using large receptive field deep convolutional neural networks," in *Bildverarbeitung für die Medizin 2017. Informatik aktuell*, pp. 86–91, 2017.

[37] C. Zotti, Z. Luo, O. Humbert, A. Lalande, and P.-M. Jodoin, "GridNet with automatic shape prior registration for automatic MRI cardiac segmentation," in *ACDC and MMWHS Challenges. STACOM 2017. Lecture Notes in Computer Science*, pp. 73–81, 2018.

[38] Özgün Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention. MICCAI 2016*, pp. 424–432, 2016.

[39] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, pp. 9–50, 1998.

[40] "Ml cheatsheet." https://ml-cheatsheet.readthedocs.io/en/latest/lossfunctions.html, 2018. Accessed: 2018-09-04.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing Research Repository*, vol. abs/1412.6980, 2014.

[42] N. Ketkar, "Introduction to tensorflow," in *Deep Learning with Python*, pp. 159–194, 2017.

[43] "Keras." https://keras.io/optimizers/, 2018. Accessed: 2018-09-11.

Appendix A: Keras model architecture

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1, 64, 64, 64	0	
conv3d_1 (Conv3D)	(None, 32, 64, 64, 6	896	input_1[0][0]
activation_1 (Activation)	(None, 32, 64, 64, 6	0	conv3d_1[0][0]
conv3d_2 (Conv3D)	(None, 64, 64, 64, 6	55360	activation_1[0][0]
activation_2 (Activation)	(None, 64, 64, 64, 6	0	conv3d_2[0][0]
max_pooling3d_1 (MaxPooling3D)	(None, 64, 16, 16, 1	0	activation_2[0][0]
conv3d_3 (Conv3D)	(None, 64, 16, 16, 1	110656	max_pooling3d_1[0][0]
activation_3 (Activation)	(None, 64, 16, 16, 1	0	conv3d_3[0][0]
conv3d_4 (Conv3D)	(None, 128, 16, 16,	221312	activation_3[0][0]
activation_4 (Activation)	(None, 128, 16, 16,	0	conv3d_4[0][0]
max_pooling3d_2 (MaxPooling3D)	(None, 128, 4, 4, 4)	0	activation_4[0][0]
conv3d_5 (Conv3D)	(None, 128, 4, 4, 4)	442496	max_pooling3d_2[0][0]
activation_5 (Activation)	(None, 128, 4, 4, 4)	0	conv3d_5[0][0]
conv3d_6 (Conv3D)	(None, 256, 4, 4, 4)	884992	activation_5[0][0]
activation_6 (Activation)	(None, 256, 4, 4, 4)	0	conv3d_6[0][0]
up_sampling3d_1 (UpSampling3D)	(None, 256, 16, 16,	0	activation_6[0][0]
concatenate_1 (Concatenate)	(None, 384, 16, 16,	0	up_sampling3d_1[0][0] activation_4[0][0]
conv3d_7 (Conv3D)	(None, 128, 16, 16,	1327232	concatenate_1[0][0]
activation_7 (Activation)	(None, 128, 16, 16,	0	conv3d_7[0][0]
conv3d_8 (Conv3D)	(None, 128, 16, 16,	442496	activation_7[0][0]
activation_8 (Activation)	(None, 128, 16, 16,	0	conv3d_8[0][0]
up_sampling3d_2 (UpSampling3D)	(None, 128, 64, 64,	0	activation_8[0][0]
concatenate_2 (Concatenate)	(None, 192, 64, 64,	0	up_sampling3d_2[0][0] activation_2[0][0]
conv3d_9 (Conv3D)	(None, 64, 64, 64, 6	331840	concatenate_2[0][0]
activation_9 (Activation)	(None, 64, 64, 64, 6	0	conv3d_9[0][0]

```
conv3d_10 (Conv3D)              (None, 64, 64, 64, 6 110656      activation_9[0][0]
_________________________________________________________________________________________________
activation_10 (Activation)      (None, 64, 64, 64, 6 0           conv3d_10[0][0]
_________________________________________________________________________________________________
conv3d_11 (Conv3D)              (None, 1, 64, 64, 64 65          activation_10[0][0]
_________________________________________________________________________________________________
activation_11 (Activation)      (None, 1, 64, 64, 64 0           conv3d_11[0][0]
=================================================================================================
Total params: 3,928,001
Trainable params: 3,928,001
Non-trainable params: 0
```