Automated segmentation of lacunes of presumed vascular origin in brain MRI scans

Master Thesis

by

E. Ooms

to obtain the degree of Master of Science in Applied Mathematics
at the Delft University of Technology,
to be defended publicly on Monday May 17, 2021 at 03:00 PM.

Student number:	4749448	
Project duration:	March 24, 2020 – May 17, 2021	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, supervisor
	Prof. dr. ir. A. W. Heemink,	TU Delft
	Prof. dr. ir. M. de Bruijne,	Erasmus MC, supervisor
	Ir. K. M. H. van Wijnen,	Erasmus MC, supervisor

Abstract

Lacunes of presumed vascular origin (lacunes) are small lesions in the brain and are an important indicator of cerebral small vessel disease (cSVD). To gain more insight in this disease, obtaining more information about the shape, size and location of lacunes is essential. However, manual segmentation (the voxel-wise labeling of lacunes in a scan) can be very time-consuming.

In this thesis, we optimize convolutional neural networks using different loss functions to automate the process of segmenting lacunes in full brain MRI scans. A set of 111 scans was used for development and a separate test set of 111 scans was used for evaluation. As lacunes are small, they generally occupy less than 0.02% of an MRI scan. This leads to an extreme data imbalance between lacune voxels and background voxels, which complicates the optimization process. We trained networks with the binary cross-entropy (BCE) loss, the weighted binary cross-entropy (WBCE) loss and the Dice loss. Additionally, we trained networks with two proposed adaptations of the Dice loss: Dice-ReLU loss and constrained Dice-ReLU (CDR) loss.

Our experiments show that all losses except the BCE loss are able to cope with the data imbalance and learn to segment lacunes. The Dice-ReLU loss performs best on detection with a sensitivity of 0.79, but produces an excessive amount of false positives (FPs) with on average 26 per image. Adding a size constraint (the constrained Dice-ReLU loss) improves the number of FPs considerably to 11.52 FPs per image with a sensitivity of 0.70. The Dice loss has just 1.93 FPs per image, but reaches a detection sensitivity of only 0.43. However, the Dice loss obtains the best segmentation performance of the true positive (TP) elements with a Dice similarity coefficient score of 0.47. Compared to the Dice loss, the WBCE loss performs slightly less on FPs per image and TP-element-wise segmentation, but slightly better on sensitivity.

We developed methods that were able to learn to segment lacunes. However, the data imbalance still influences the optimization process considerably, leading to methods that either focus too much on the foreground or too much on the background. Further work on developing a loss function that can cope better with this data imbalance could greatly improve lacune segmentation performance.

Acknowledgements

This thesis came about in cooperation with the Delft University of Technology and the Erasmus Medical Center Rotterdam. As many people played a role in this, I would like to take this opportunity to express my gratitude for their contributions.

First of all, I would like to thank all of my supervisors for their time. I was privileged with so many moments of contact with all of you. I would like to thank my supervisor Kees Vuik for our weekly meetings. I really appreciated the face-to-face feedback. The ”live” contact, albeit via Skype, made it much easier for me to understand the comments and questions on my thesis and to recognize which parts of my thesis were not clearly described yet. I would also like to thank Arnold Heemink for taking place into my thesis committee and taking the effort to read and evaluate my thesis. Next, I would like to thank Marleen de Bruijne for giving me the opportunity to do an internship at the research group MBM and to work on my thesis under her supervision. Thank you for all the meetings and for helping me to think more critically. Your questions greatly helped in making me better understand the research topics. Last but not least, I would like to thank Kim van Wijnen for everything she did for me, and that was a lot! I know supervising a student was new to you, but I really could not have hoped for anyone better. You were an absolutely incredible help to me. Looking back at the beginning of my thesis and internship, I have learned so much! Thank you for all the discussions, feedback and for your patience when I was stressing out again.

I would also like to thank Wiro Niessen for giving me the opportunity to do an internship at BIGR. I am really grateful that I was able to choose from four different interesting projects. In addition, I would like to thank Niels van der Werf for bringing me into contact with Wiro. Because of you I was able to familiarize myself with the field of deep learning and get the Erasmus MC-experience.

I would like to thank Marleen, Kim, Gerda, Robin, Antonio, Shuai, Shengnan, Deep, Ivan, Oliver, Natalie, Luc, Alexis, Thomas, Emmanuel and Hugues from the MBM for the warm welcome they gave me. Although due to Corona most of you I have never spoken to in real-life, you really gave me the feeling that I as a student was also a part of the group.

Furthermore, I would like to thank Florian Dubost for letting me use his code and Meike Vernooij for providing me with the dataset of the Rotterdam scan study. Thank you to Tavia Evans for letting me use the scans that she manually segmented. It was great that we had so many segmented scans to work with. It must have taken a lot of time and effort to go through all those scans. I would also like to thank the GPU cluster maintenance group for letting me use the cluster and especially Karin van Garderen for fixing the nodes almost immediately when a node was reported to be down. Thank you to Bo Li for the brainstorm sessions we had on lacunes.

Lastly, I would like to thank Jasper for everything he did to make my life easier during my thesis.

Contents

1 Introduction

Cerebral small vessel disease (cSVD) is a disease that can result in different types of brain lesions, including lacunes of presumed vascular origin, perivascular spaces, white matter hyperintensities and microbleeds. It has an effect on the physical and psychological abilities of especially the elderly. However, the mechanisms causing the disease are not fully known. cSVD itself is hard to detect with neuroimaging techniques, but the lesions resulting from cSVD are better to capture using neuroimaging. Therefore, the disease is detected by inspecting brain MRI images for these lesions. Currently, this task is mostly performed manually by a radiologist and can take up a lot of time. Additionally, as the interpretation is executed by a human being, it is subject to bias and variations across interpreters.

In order to assist the radiologist in making more accurate decisions and save time, during the past couple of years, researchers came up with several artificial intelligence methods to automate the manual detection process. For lacunes of presumed vascular origin (lacunes) these methods were mostly detection based, which means that the method detects the lesion with drawing a box around it. However, to get more precise information about the size, shape and location, instance segmentation is needed as this procedure provides the exact outline of the lesion. Only a few methods were designed to segment lacunes. All detection and segmentation methods developed thus far, use either 2D images or 3D sub-images for this. As a consequence, to evaluate entire 3D images with these methods, extra computational costs, time or manual labour is needed.

Therefore, this thesis proposes a method that is able to detect and segment lacunes in 3D images at once. It is hypothesized that the deep learning U-net architecture [38] is suitable for this task, as it has proven to successfully segment other lesions before [11, 13, 15, 25, 48, 56]. However, developing a 3D method comes with a few challenges. Lacunes are tiny structures that generally occupy only less than 0.02% of a 3D image. This leads to an extreme data imbalance between lacune voxels (the equivalent of a pixel) and background voxels (non-lacune voxels), which may complicate the optimization process. It is hypothesized that the challenge of the data imbalance can be tackled by placing more emphasis on the lacune voxels, as this will lead to a better balance between the lacune and background voxels. This will be addressed in the loss function. Three existing loss functions as well as an adaptation to one of them are tested in this thesis. Additionally, previous work on lacunes already showed that several similarly looking structures can be incorrectly identified as lacunes. It is expected that if the method will be punished for incorrectly classifying too many background voxels as a lacune, it will be forced to learn how to differentiate between a lacune and a similar looking structure. Therefore, this challenge might be addressed by putting a constraint in the loss function on the volume of the background voxels.

The remainder of this thesis is structured as follows: chapter 2 provides a medical background on lacunes of presumed vascular origin and cerebral small vessel disease. An introduction on convolutional neural networks is given in chapter 3, after which the previously developed lacune detection and segmentation methods are described in chapter 4. Information of the data that is used for optimizing the network can be found in chapter 5. Chapter 6 describes the elements and procedures needed for training and testing the network, which includes the proposed loss functions. An exact description of the performed experiments is provided in chapter 7, which is followed by the results in chapter 8. Chapter 9 discusses these results and gives recommendations for future work. A final conclusion is given in chapter 10.

2 Medical Background

2.1. Cerebral small vessel disease

The term cerebral small vessel disease (cSVD) refers to changes in the small arteries, arterioles, venules and capillaries in the brain. All together these blood vessels are called the small vessels. As a consequence of these changes, lesions can occur in the brain, such as white matter hyperintensities, lacunes of presumed vascular origin, perivascular spaces and microbleeds. However, the mechanisms causing the changes that lead to lesions are heterogeneous and not fully understood. Since alterations in small vessels are hard to detect with neuroimaging, whereas the resulting lesions are better to image, these lesions are often used as biomarkers for cerebral small vessel disease. That is, the presence of the lesions is determinative for diagnosing the disease. As a result of this, the term cerebral small vessel disease often refers to its lesions rather than the changes in the small vessels [35].

There are several types of small vessel disease which can be subdivided based on its pathology into the amyloidal form and the non-amyloidal form. This separation is based on whether aggregates of proteins, called amyloids, play a role in the disease or not. That is, the types in the amyloidal group are related to the deposition of these amyloids. Within this amyloidal form cerebral amyloid angiopathy (CAA) is a very common subtype [9]. CAA is a chronic degenerative disease which is predominantly associated with advanced age [5]. It is caused by progressive accumulation of β-amyloid in the walls of small arteries in the brain. The deposition of the β-amyloid leads to occlusion and rupture of the vessels and eventually to brain injuries [7, 39]. The non-amyloidal group, on the other hand, is not related to the deposition of amyloids, but represents itself by an increase of the vessel wall, narrowed interior of the vessels, the occurrence of dilated and elongated vessels or the buildup of fats, cholesterol and other substances on the vessel walls. As this non-amyloidal form is often associated to ageing, diabetes and hypertension, it is also known as age-related and vascular risk-factor-related small vessel disease [35].

Although the exact link between cSVD and changes to the small vessels in the brain is still unclear, it is hypothesized that damage to the blood-brain barrier might be a possible cause of some cSVD types [50]. The blood-brain barrier (BBB) is a semipermeable barrier that separates the cardiovascular system from the fluid of the central nervous system. It prevents the pathogens and circulating immune cells from passively entering and as a result damaging the brain [1, 9]. In several studies it was found that in patients with cSVD, BBB leakage was present [42, 51, 55, 59]. The hypothesis is that with increasing age the permeability of the BBB increases and this damage process is enhanced by several risk factors such as hypertension, diabetes and inflammation. If the BBB is weakened, pathogens and immune cells can invade into the brain and induce lesions [9, 50, 55].

The consequences of small vessel disease are diverse. Elderly patients with cSVD experience cognitive decline [47], depressive symptoms [19] and physical disabilities such as gait disturbances [10] and urinary problems [37]. Furthermore, the disease is the cause of 25% of all ischaemic strokes [3], contributes to up to 45% of dementias [52] and appears to be associated with Alzheimer's Disease and Parkinson's Disease [9]. However, the exact relation of cSVD with these neurodegenerative diseases is still unclear.

In order to detect small vessel disease, neuroimaging and especially magnetic resonance imaging

(MRI) is used to visualize the different brain lesions. To assess the severity of the disease a total cSVD score is proposed, which takes several of these imaged biomarkers into account instead of evaluating them separately [28, 54]. Besides this, neuroimaging biomarkers can also be used to gain insight in the cause of the disease and its connection to other neurodegenerative diseases. It is shown that dysfunction of the blood-brain barrier is associated with at least some of the cSVD related lesions [55, 59]. However, if the exact relation and process between the leaking BBB and the disease can be unraveled, it might help to prevent small vessel disease from originating.

The brain lesions that are commonly used as cSVD biomarkers are recent small subcortical infarcts, lacunes of presumed vascular origin, white matter hyperintensities of presumed vascular origin, perivascular spaces, cerebral microbleeds and brain atrophy. These lesions vary in size, shape and location in the brain. Furthermore, their intensities on an MRI scan may vary as well from having a low intensity (white) to having a high intensity (dark gray/black) [53]. It is beyond the scope of this thesis to discuss all of the lesions here. However, since the goal of this thesis is to automate the segmentation of lacunes of presumed vascular origin, this lesion type will be discussed in more detail in the next section, as well as other lesions that are relevant due to their similarities in appearance.

2.2. Lacunes of presumed vascular origin

Lacunes of presumed vascular origin are small cavities filled with fluid and are mainly located in the deeper parts of the brain [16]. They are presumed to be a result of subcortical infarcts, which on their turn are expected to be caused by occlusion of small arteries [16, 53]. However, the lesions may also be caused by intracerebral haemorrhage as it was found to be associated with lacunes of presumed vascular origin [36, 40].

Since across papers there was little consistency in terminology and definitions for the biomarkers of small vessel disease, Wardlaw et al. proposed a consensus term and definition for each biomarker. In this thesis the recommended standards for lacunes of presumed vascular origin as stated by Wardlaw et al. [53] will be used. Therefore, a lacune of presumed vascular origin is defined as ”a round or ovoid, subcortical, fluid-filled cavity of between 3 mm and about 15 mm in diameter, consistent with a previous acute small deep brain infarct or haemorrhage in the territory of one perforating arteriole”. Lesions can be recognized on MRI sequences, which are combinations of particular settings of radiofrequency pulses and field gradients that influence the appearance of an image. On all MRI sequences the lacunes of presumed vascular origin can be recognized by its intensity, that is, hypointense (dark gray/black) or hyperintense (white) depending on the sequence used. In addition, images of the fluid-attenuated inversion recovery sequence can in some cases also display a hyperintense rim around the central hypointensity. However, this hyperintensity is not always present and in this case the lacunes of presumed vascular origin appear entirely hypointense [53]. An example of a lacune imaged with a fluid-attenuated inversion recovery (FLAIR) sequence is depicted in figure 2.1.

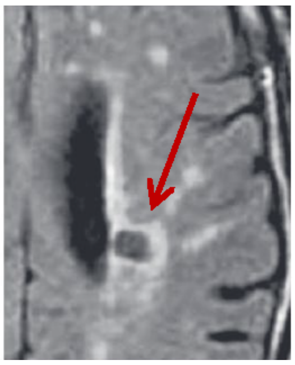


Figure 2.1: A lacune on a FLAIR image having a hyperintense rim around a center of hypointensity [53].

Lacunes of presumed vascular origin need to be distinguished from perivascular spaces as they can look very similar. According to Wardlaw et al. [53], perivascular spaces are defined as ”fluid-filled spaces that follow the typical course of a vessel as it goes through grey or white matter”. On all MRI sequences these lesions have an intensity that is similar to the appearance of lacunes of presumed vascular origin. They can have an elongated shape if imaged parallel to the course of the vessel, but they can also appear round or ovoid in shape if imaged perpendicular to the course of the vessel. When they pass through an area of white matter hyperintensity, they can even mimic the hyperintense rim of a lacune of presumed vascular origin on FLAIR images [53]. Several studies differentiate between these two lesions based on the diameter size and the presence of the hyperintense rim [4, 17, 29–31, 36]. In these articles, FLAIR imaged hypointense lesions with a diameter between 3 mm and 15 mm with a hyperintense rim are classified as lacunes of presumed vascular origin. Moreover, when the lesion is not round or ovoid in shape but somewhat elongated, it is classified as a perivascular space.

In the remainder of this thesis, the term lacunes of presumed vascular origin will be abbreviated to

’lacunes’. However, in all cases this refers to the consensus term and definition of lacunes of presumed vascular origin.

3 Convolutional neural networks

Artificial intelligence gained a lot of attention in the last couple of years. It can be defined as ”the effort to automate intellectual tasks normally performed by humans” [8]. A part of the field of artificial intelligence consists of non-learning approaches, which means that with these approaches programmers manually have to define a large set of explicit rules to automate a process. However, although this appears to be suitable for simple well-defined and logical problems, these approaches fail to solve the more complex problems.

A subfield of artificial intelligence that is able to address those more complex problems, such as language translation, speech recognition and image classification, is called machine learning. Instead of being manually specified, in machine learning the set of rules are learned by a computer when looking at a set of examples. When a machine learning system is provided with many examples, which include both the problems and their answers, and with optionally certain features it needs to pay attention to, the system can learn a pattern and can come up with rules to automate the task. These learned rules can then be applied to unseen examples of the same task to predict the answer of the new examples. For example, if we want to let a machine learning system learn to classify whether a picture contains a cat or not, we have to feed the system with a lot of examples of pictures with cats and pictures without cats. Additionally, as the system needs to know how a cat can be recognized, we also need to tell the system which features are characteristic for a cat, such as a tail, ears, eyes and whiskers. With these example pictures and features, it defines rules to automate the classification process such that when the learned system is provided with a new image it can categorize it as a cat or a non-cat image.

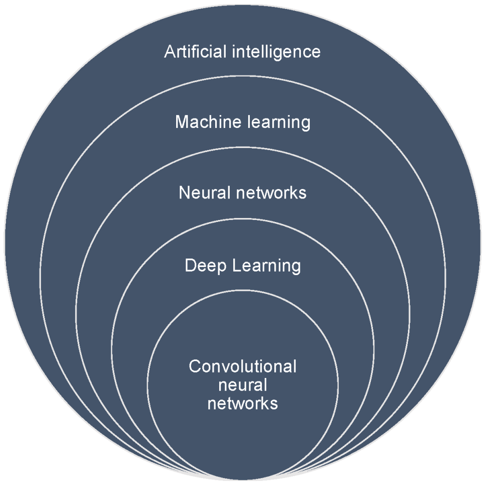
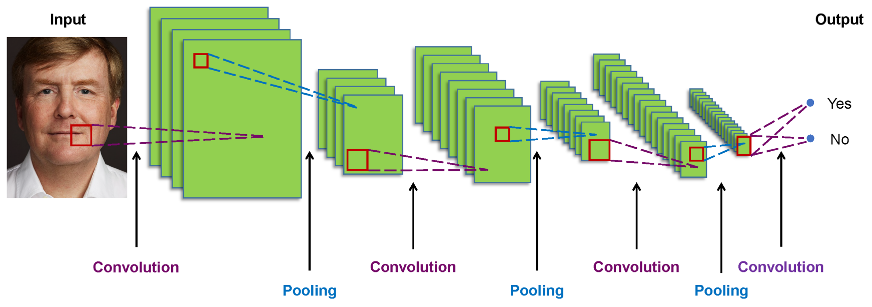


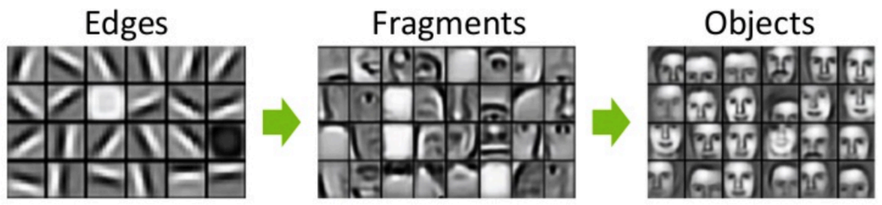
Figure 3.1: Graphical explanation of the relationships between artificial intelligence, machine learning, deep learning, neural networks and convolutional neural networks.

Specifically for image analysis and image classification, deep learning is a subgroup of the field of neural networks, which is on its turn a subfield of machine learning (see figure 3.1). The main difference between neural network and deep learning techniques and other techniques within the machine learning field, is that the neural network and deep learning techniques learn to identify characteristic features of an object automatically, while for other techniques these features have to be manually specified. For the cat classification problem, this means that the deep learning technique is able to learn that cats can be recognized by their eyes, ears and whiskers. Because of this automated feature extraction, more complex and an increased number of features can be extracted which leads to better performance on many problems [8, 18].

A convolutional neural network (CNN) is a deep learning algorithm that is mostly used for analyzing images and image classification problems. A general CNN architecture consists of several layers. Often used layers are convolution layers, activation functions, pooling layers and an end activation function. The order of these layers is not fixed, but can differ across architectures. The convolution layers are specifically used in CNNs and are the discriminating factor between CNNs and other types of neural networks. An example of a CNN architecture is given in figure 3.2a. In this example the CNN is provided with an image of Willem-Alexander, King of the Netherlands. The information of this image will be passed on to the first convolution layer, which transforms this input with a so called kernel and outputs the new information to a pooling layer. The kernel acts as a kind of filter that is able to detect patterns. As can be seen in figure 3.2b, in the first layers these patterns are simple features, like edges [8]. The kernels in later layers can detect more sophisticated features such as eyes, noses and mouths. In the deepest layers, even entire objects can be detected using these kernels. With this information the network is able to indicate whether King Willem-Alexander is portrayed on the input image or not (figure 3.2a). A more detailed explanation of the components of a CNN follows later in this chapter.



(a) Graphical explanation of the process within a convolutional neural network. An image goes through several operation layers after which a prediction of the output is given.



(b) Examples of features through a neural network, starting with simple edges and ending with entire objects.

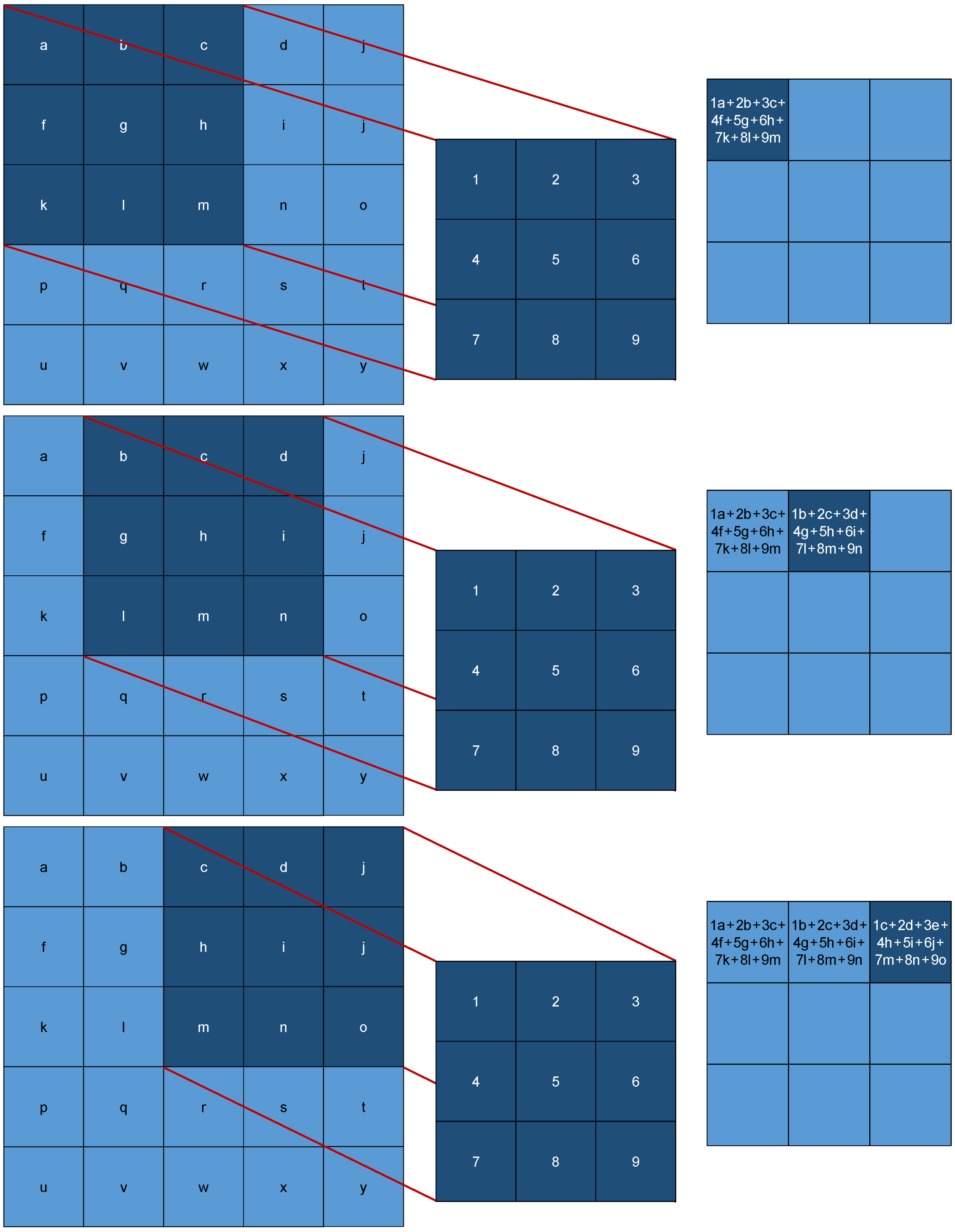
Figure 3.2: An overview of a convolution neural network (CNN).

3.1. Components of a convolutional neural network

3.1.1. Convolution layer

In a convolution layer, one or a multiple of a so called feature map are given as input and transformed using a kernel into one new feature map. The input feature maps, output feature maps and kernels are usually multi-dimensional arrays. In the first convolution layer, the input feature map can be an image, but in later layers these input maps are the output feature maps of previous layers. A kernel is a matrix with weights that is able to find one specific pattern, for example a horizontal edge. It often has the size of $3x3$ or $5x5$. By applying the specific kernel to an entire input image or feature map, it can extract all cases of a certain feature in this image, such as a horizontal edge. If we want the network to find even more features, such as vertical and diagonal edges, other kernels need to be applied to extract these patterns as well [8, 14, 18].

A convolution is executed by sliding the kernel over the input feature map, stopping at all possible positions. At each location, every entry of the kernel is multiplied by its corresponding overlapping entry of the input feature map. The results of all kernel entry multiplications are summed to obtain the value of the output feature map at that position [8, 14]. Figure 3.4 shows how the process works for a 2-dimensional problem. In the top of the figure we see that each entry of the $3x3$ kernel is multiplied by the corresponding value of the upper left $3x3$ window that is part of the $5x5$ input feature map.


a b c d j
f g h i j
k l m n o
p q r s t
u v w x y
1 2 3
4 5 6
7 8 9
1a+2b+3c+4f+5g+6h+7k+8l+9m
a b c d j
f g h i j
k l m n o
p q r s t
u v w x y
1 2 3
4 5 6
7 8 9
1a+2b+3c+4f+5g+6h+7k+8l+9m
1b+2c+3d+4g+5h+6i+7l+8m+9n
a b c d j
f g h i j
k l m n o
p q r s t
u v w x y
1 2 3
4 5 6
7 8 9
1a+2b+3c+4f+5g+6h+7k+8l+9m
1b+2c+3d+4g+5h+6i+7l+8m+9n
1c+2d+3e+4h+5i+6j+7m+8n+9o

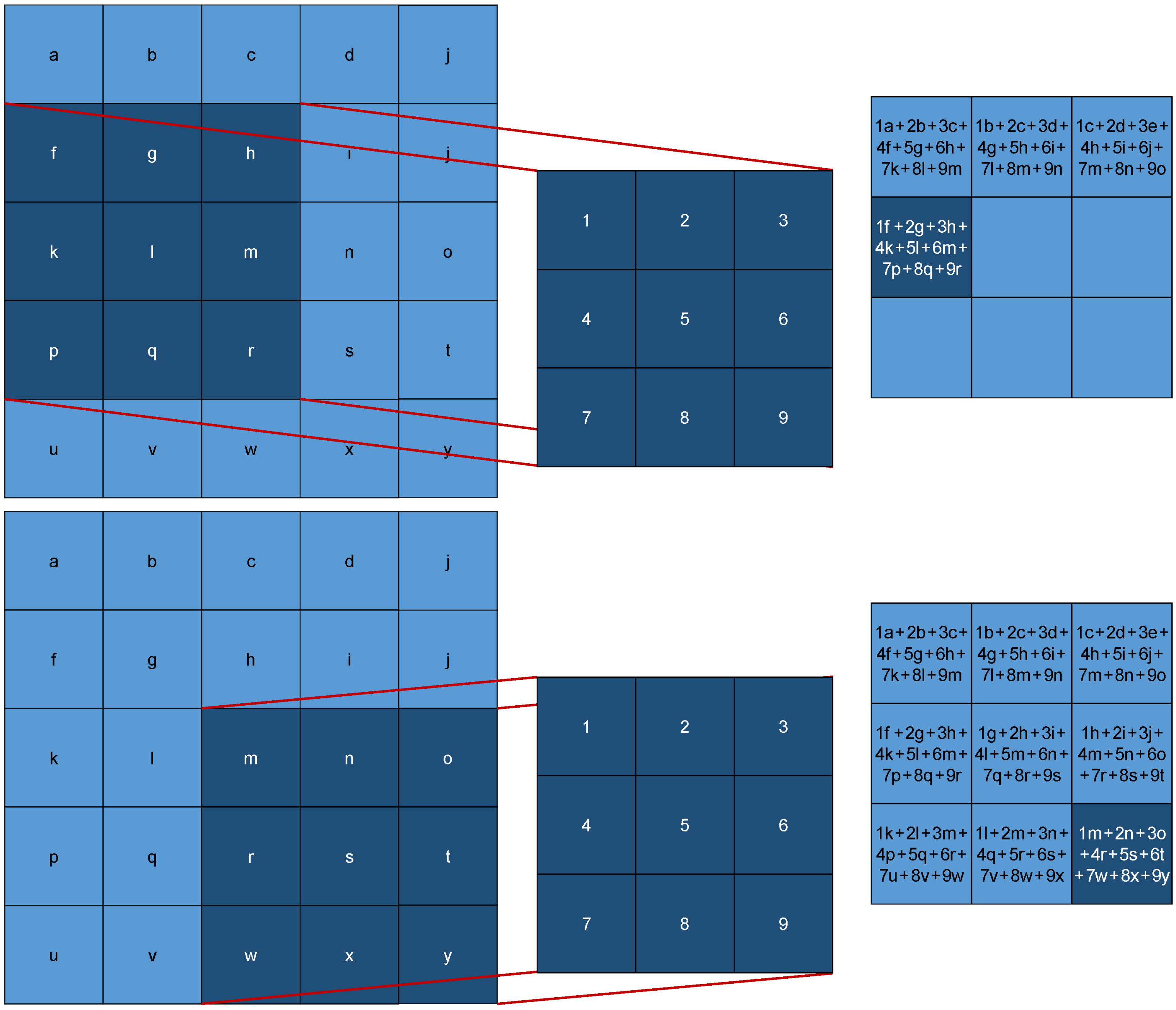

a
b
c
d
j
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
1
2
3
4
5
6
7
8
9
1a+2b+3c+ 4f+5g+6h+ 7k+8l+9m
1b+2c+3d+ 4g+5h+6i+ 7l+8m+9n
1c+2d+3e+ 4h+5i+6j+ 7m+8n+9o
1f +2g+3h+ 4k+5l+6m+ 7p+8q+9r
1g+2h+3i+ 4l+5m+6n+ 7q+8r+9s
1h+2i+3j+ 4m+5n+6o +7r+8s+9t
1k+2l+3m+ 4p+5q+6r+ 7u+8v+9w
1l+2m+3n+ 4q+5r+6s+ 7v+8w+9x
1m+2n+3o +4r+5s+6t +7w+8x+9y

Figure 3.4: Example of a $3x3x3$ convolution operation. The input feature map is given on the left, containing entries from a to y. The kernel is placed in the middle and has e.g. values from 1 to 9. The resulting output feature map is depicted on the right. A kernel of $3x3$ slides over the input feature map of $5x5$. At each position it calculates the sum of the multiplications of each kernel entry with their corresponding overlapping window entry. The result of the calculation is positioned in the output feature map at the same location. The kernel slides with one stride (stepsize) to the right and below. Only the first four and the last windows and their calculations are given in this figure [14, 18].

The result of the first window appears in the output feature map at the same location. In the second step, the kernel has slid one stride, which is a stepsize, to the right to apply the same transformation to a new window. After three horizontal slides, the next window is located one stride lower, at the left. This procedure of sliding and calculating continues until the kernel transformed every value of the input feature map into a new value in the next convolution layer.

The entire process can be repeated by applying as many different kernels as needed. However, when more than one kernels are used, their resulting output feature maps will be stacked onto each other which leads to an extra dimension. As a consequence of this, cuboid, 3-dimensional, kernels should be applied to slide over the width, height and feature map dimension of the input in the next convolution layer [14].

The convolution arithmetic from figure 3.4 can also be represented by one generalized formula. Let $I \in \mathbb{R}^{K \times L}$ be the input feature map, $K \in \mathbb{R}^{M \times N}$ be the kernel and the output feature map S with dimensions of $\left(\frac{K-M}{\text{stride}} + 1\right) \times \left(\frac{L-N}{\text{stride}} + 1\right)$. An entry of the output map can then be calculated as follows [18],

$$S(i,j) = (K * I)(i,j) = \sum_{m}^{M} \sum_{n}^{N} I(i+m, j+n) K(m,n). \tag{3.1}$$

3.1.2. Activation function

In the convolution layer, only linear operations are used. This means that the network would only be able to learn the linear transformations of the input. If we want the model to learn complex mapping functions as well, a non-linearity, also referred to as activation function, is needed [8]. An activation function is applied elementwise such that for $X \in \mathbb{R}^{m \times n}, f : \mathbb{R}^{m \times n} \longrightarrow \mathbb{R}^{m \times n}$ [20]

$$(f(x))_{ij} = f(x_{ij}). \tag{3.2}$$

The most common activation function for neural networks is the rectified linear unit (ReLU) [18]. If the input is positive it ”activates”, if the input is negative it will become 0 [23, 34]:

$$f(x_{ij}) = \begin{cases} 0 & \text{for } x_{ij} \leq 0, \\ x_{ij} & \text{for } x_{ij} > 0. \end{cases} \tag{3.3}$$

3.1.3. Pooling layer

After the application of an activation function, a pooling layer can be applied. The output feature map is downsampled by a pooling function, which maps parts of the output feature map to one summary statistic. This is often done by taking the maximum value from a certain window of the output feature map, which is called max pooling [20]. Similar to the window sliding procedure in the convolution layer, the windows in the pooling layer are also chosen by sliding a square over the feature map. However, this pooling square is mostly smaller, usually $2x2$, than the convolutional squared kernel. Additionally, a stride of two is often used, which means that the window slides with two steps across the input. As a result of the stride of two, the feature map is downsampled with a factor of 2 [8]. In figure 3.5 the mapping of the max pooling is shown.

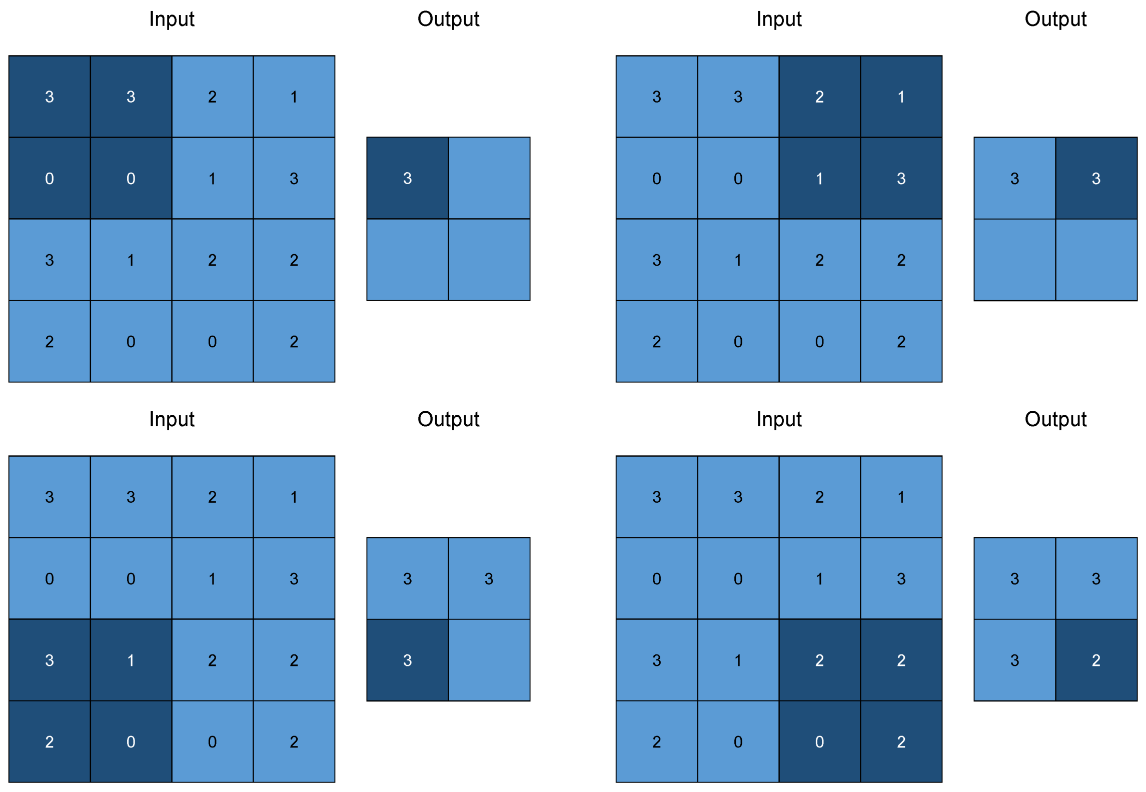


Figure 3.5: Example of max pooling. A window of $2x2$ slides over the input feature map of $4x4$ with a stride of 2. At each position it takes the maximum of the entries within that window. The result is positioned in the output at the same location [14].

A consequence of reducing the size of the feature maps is that there are less parameters to process for the next convolution layer, which will make it more computational efficient. Furthermore, downsizing will also make later convolution layers look at larger windows. That is, when more pooling layers are applied, a bigger window of the input image is used to predict one pixel of the output image. As a result of this, the network will be able to recognize more sophisticated features, such as eyes, ears or even entire faces [8]. Finally, because the results of the convolution layer will be positioned at the same location in the output feature map as their input features, it can be sensitive to translations in the input. For example, if the object in the input image would be slightly shifted, the output feature maps created by later convolution layers would look different. By using a pooling function, the output layers become more invariant to these small movements [18].

3.1.4. Final layer with end activation function

In order to give a classification to the input image, in the last part of the architecture a final layer is applied followed by an end activation function. This final layer can be a last convolution where the number of kernels should be equal to the preferred number of outputs. Finally, to output a probability that gives an indication whether the image belongs to a certain class or not, a final activation function is applied after the last layer. For this final transformation it is common to use a sigmoid function or a softmax function. Both of these end activation functions output values between 0 and 1. The sigmoid function is suitable for binary classification tasks (for example, is there a cat in the input image ”yes” or ”no”) and if we let $\mathbf{x} \in \mathbb{R}^2$ it can be written as follows [18],

$$f(x_i) = \frac{1}{1 + e^{-x_i}}. \tag{3.4}$$

For a multiclass classification task (for example, is there a cat, a dog, a cow, a pig, a goat or a goose in the input image), the softmax function is used. In this function, the exponent of every entry is taken and divided by the sum of all exponentiated entries. Letting $\mathbf{x} \in \mathbb{R}^c$, where c represents the number of classes, the softmax function is given by [18],

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}}. \tag{3.5}$$

3.2. Training a convolutional neural network

The goal of training a network is to obtain an optimized network that is able to predict the manually segmented label as accurate as possible. This can be accomplished by updating parameters in the network, such that the discrepancy between the predicted outcomes and the manually segmented labels will be minimized. A loss function can be used to calculate this discrepancy between predictions and their corresponding actual outcomes. The result of the loss function, which is also called the loss, can be minimized using an optimizer which updates the weights from the network.

3.2.1. Loss function

The objective of a loss function is to quantify the difference between the predicted outcomes and the manually segmented labels. There are several functions that can be applied for this task, but the most commonly used loss function for classification problems is the cross-entropy loss. Where the binary cross-entropy loss is most often used for binary classification tasks, and the categorical cross-entropy loss is most often used for tackling multiclass classification problems [8]. Suppose we want to train a dataset of m examples $\{X^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{V\times W}$ with manually segmented outputs $\{Y^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{V\times W}$ and the predictions computed by a network $\{\hat{P}(X^{(i)};\Theta)\}_{i=1}^{m} \in \mathbb{R}^{V\times W}$ where Θ are the weights in the network, then the categorical cross-entropy loss is given by,

$$L(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} Y_k^{(i)} \log\left(\hat{P}_k(X^{(i)};\Theta)\right), \tag{3.6}$$

where K is the total number of classes. If we take $K = 2$, the binary cross-entropy loss follows from this:

$$L(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} Y^{(i)} \log\left(\hat{P}(X^{(i)};\Theta)\right) + \left(1 - Y^{(i)}\right)\log\left(1 - \hat{P}(X^{(i)};\Theta)\right). \tag{3.7}$$

3.2.2. Optimizer

To decrease the discrepancy between the predicted output and the manually segmented output, the loss can be minimized using an optimizer by adjusting the weights of the network. At the start of the training, the weights of the network are initialized with random values. As a consequence, the predictions will be very far from what the actual output should be and therefore the value of the loss function will be high. To decrease this value, the optimizer updates the weights in the convolution and fully connected layers using backpropagation. This backpropagation operates from the end of the network, the loss function, through all of the layers to the beginning of the network, the first convolution layer. It computes the contribution of every weight to the loss function by applying the chain rule. Each time an example is processed by the network, the weights will be updated in the right direction and the predictions will slowly become more accurate [8]. When every example of a dataset passed forward and backward through the network once, it is called an epoch. For the minimization of the loss function,

$$\underset{\Theta}{\text{minimize}} \quad L\left(\hat{P}(X^{(i)};\Theta), Y^{(i)}\right), \tag{3.8}$$

stochastic gradient descent (SGD) and its variants are the most common optimizers used in deep learning [18].

Stochastic gradient descent, which is comparable to steepest descent, proceeds iteratively, that is, a sequence of matrices is computed with the goal of converging to a matrix, the predicted outcome, that minimizes the loss function. SGD executes this task as follows. In the model the weights are stored in matrices, Θ. However, for ease of the explanation we will assume that the weights that need to be updated are now represented by a vector, $\boldsymbol{\theta} \in \mathbb{R}^R$. Let the update of this vector $\Delta\boldsymbol{\theta}$ be small. Then, when the terms of order $||\Delta\boldsymbol{\theta}||^2$ are left out, the Taylor series expansion can be written as

$$L(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \sum_{r=1}^{R} \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_r} \Delta\theta_r, \tag{3.9}$$

where $\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_r}$ is the partial derivative of the loss function with respect to weight r. We can simplify this equation by using the gradient, $(\nabla L(\theta))_r = \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_r}$, such that

$$L(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \nabla L(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}.$$

Now, to minimize the expression we need to take the step $\Delta\boldsymbol{\theta}$ in such a manner that $\nabla L(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}$ is as negative as possible. From the Cauchy-Schwartz inequality, for any $f, g \in \mathbb{R}^R$, it holds that

$$|f^T g| \leq ||f||_2 ||g||_2.$$

As a consequence, $-||\Delta\boldsymbol{\theta}||_2 ||\nabla L(\boldsymbol{\theta})||_2$ is the most negative as $\nabla L(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}$ can get, which is true if $\Delta\boldsymbol{\theta} = -\nabla L(\boldsymbol{\theta})$. Therefore, the update step will be as follows

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla L(\boldsymbol{\theta}),$$

where ϵ is called the learning rate, which is a small stepsize as we assumed that $\Delta\boldsymbol{\theta}$ should be small [20]. However, when we need to deal with a large number of examples and many weights that need to be updated, updating can become extremely computationally expensive. In these cases it is common to take a subset of the examples for some $n << m$, a so called minibatch. Instead of calculating the mean of gradients over all examples, with a minibatch the mean of gradients will be calculated over only a few examples [20]. If we write the loss function as

$$L(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} C_{X_i}(\boldsymbol{\theta}),$$

where $C_{X_i}(\boldsymbol{\theta})$ represents the loss per example, then the algorithm for SGD can be written as given in algorithm 1. In this algorithm we can see that the learning rate changes with every iteration, k. Generally, it decays linearly until iteration τ after which ϵ stays constant:

$$\epsilon_k = \left(1 - \frac{k}{\tau}\right)\epsilon_0 + \frac{k}{\tau}\epsilon_\tau.$$

Without using the batches of the stochastic gradient descent, the optimizer is referred to as gradient descent which is similar to steepest descent. Variants of the SGD optimizer that are often used are SGD with momentum, RMSProp, RMSProp with momentum, AdaDelta and Adam. However, "the choice of which algorithm to use, at this point, seems to depend largely on the user's familiarity with the algorithm" [18].

Algorithm 1: Stochastic gradient descent

```
Require: Learning rate schedule $\epsilon_1, \epsilon_2, ...$
Require: Initial parameter $\boldsymbol{\theta}$
k ←1;
while stopping criterion not met do
    Sample a subset of $n$ examples from the training set $\{X^{(1)}, ..., X^{(m)}\}$ with corresponding
     labels $Y^{(i)}$;
    Compute gradient estimate: $\hat{g} \leftarrow \frac{1}{n}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n} C_{X_i}(\boldsymbol{\theta})$ ;
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{g}$;
    $k \leftarrow k + 1$;
end
```

3.3. Application to image data

Convolutional neural networks can be applied to data with a grid-like topology. Therefore, as images are 2-dimensional or 3-dimensional arrays, CNNs are often applied to image data [18]. They can be used for various imaging tasks of which image classification, object detection and image segmentation are very common. These three tasks will be explained in more detail below.

3.3.1. Image classification

The goal of image classification is to predict the class of an object in an image. An image that contains an object is given to the network and based on this input image the network outputs the probability of the object belonging to a class. The example with King Willem-Alexander in figure 3.2a from the introduction of this chapter, is an example of image classification. An image with an object is given, King Willem-Alexander, and based on this image the network produces probabilities for the object belonging to each of the two classes, 'King Willem-Alexander' and 'Not King Willem-Alexander'.

3.3.2. Object detection

In object detection, the goal is to localize and classify the objects of an image. For this, an image with one or more objects is given to the network. The network outputs an image in which the objects are marked by a bounding box or a single point. Probabilities of the objects belonging to certain classes are attached to these marks. So, compared to image classification which only outputs a classification, object detection outputs an entire image in which it classifies the object and additionally also localizes it.

3.3.3. Image segmentation

With image segmentation one divides an image into segments. This is done by classifying every pixel, such that pixels belonging to the same class will get the same label and create one segment. So, when an image with one or more objects is provided to the network, it will output an image with a separate segmentation for each object. In comparison to object detection, where an object is localized and classified as a whole, image segmentation classifies each pixel and as a result gives more precise information about the exact location, shape and size of an object.

3.4. U-Net architecture

Within the field of medical image analysis, the U-Net architecture developed by Ronneberger et al. [38] is a well known architecture for segmentation problems [32]. It has been proven to be effective as other researchers have successfully applied this U-Net for their segmentation problems [11, 13, 15, 25, 48, 56]. Rather than just getting one classification output for the input image, this architecture outputs an entire image by giving a classification to every pixel of the image. That is, for every pixel it predicts whether it belongs to the foreground (a specific lesion or organ) or to the background (the rest of the image). As a result of this, the output of the network can have the same shape as the input.

The U-Net consists of a contracting part, in which feature maps are downsampled to analyze the image, and an expanding part, where using upsampling a full-resolution segmentation is produced (see figure 3.6). The contracting part consists of 4 blocks containing two $3x3$ convolutions with stride 1, which are both followed by a ReLU activation function. Each block ends with one $2x2$ max pooling operation with stride 2. During the first block of operations 64 kernels are used and with every following contracting block the number of kernels is doubled.

The expanding part also consists of 4 blocks. These blocks contain convolutions, upsampling operations and copy and crop operations. In an upsampling operation, the output feature map of the last convolution is taken and expanded with a factor of two. This is executed by copying the intensity value of a pixel and paste it to three attached neighboring pixels and repeating this procedure for every pixel of the image. This process is called nearest neighbor upsampling and can be seen in figure 3.7. To reduce the number of feature maps, this result is used in a $2x2$ convolution. The combination of this upsampling operation and $2x2$ convolution operation is called 'up-conv $2x2$' in figure 3.6. Copy and crop operations, sometimes also referred to as skip connections, are used to transfer information between the contracting path and the expanding path. As by downsampling during the pooling operation localization information can get lost, while it is needed for the segmentation task, a part of the feature maps containing high resolution features from the contracting path is copied and concatenated to its

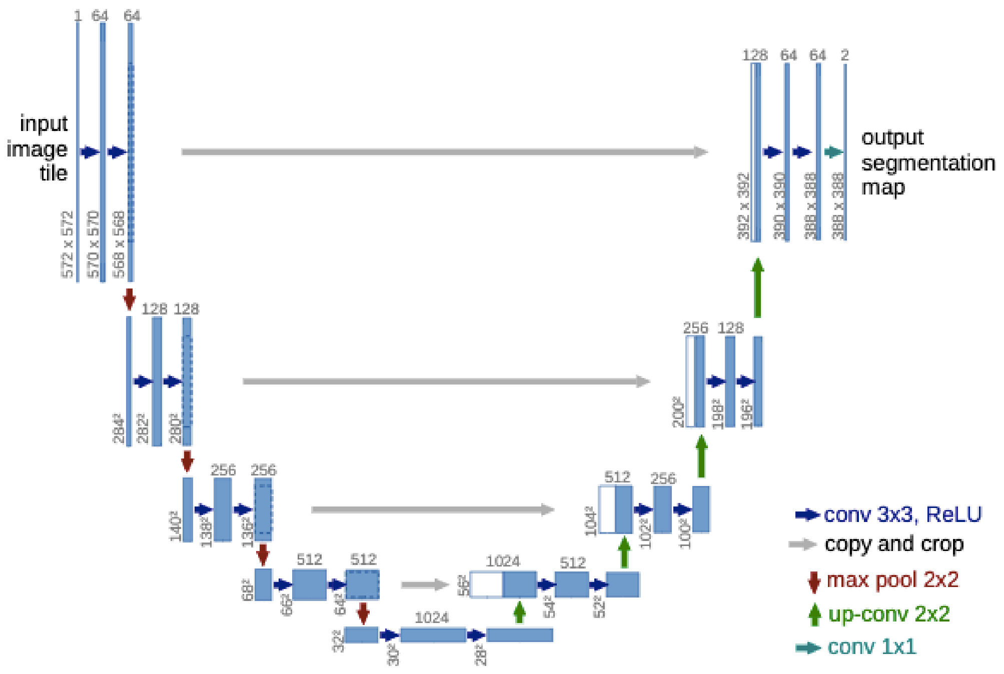


Figure 3.6: U-Net architecture in which convolutions, max pooling operations, up-convolutions and copy and crop operations are used. The blue bars represent the feature maps with on top the number of feature maps and at its left lower corner the x and y sizes of the feature maps [38]

corresponding layer in the expanding path (see figure 3.6). As a consequence, the expanded path is able to better localize pixels, which results in a more accurate segmentation. So, to be more precise, an expanding block starts with two $3x3$ convolutions having stride 1, which are each followed by a ReLU. In the second part of the expanding block the feature map is doubled in size by upsampling, after which a $2x2$ convolution with stride 1 is applied that in this case halves the number of feature maps. Finally, each block ends with a copy and crop operation by concatenating a copy of the feature maps in the contracting path to the feature maps in the expanding path.

At the end of the network another two $3x3$ convolutions with stride 1 are applied, which are each again followed by a ReLU function. Finally, a last $1x1$ convolution with stride 1 is performed to output the segmentation map. If an original image is used as input for the network, the output segmentation map will have slightly smaller dimensions than its input. This is caused by the convolution layers that reduce an images with a few pixels as can be seen in figure 3.4. To prevent the image from getting smaller, such that an output image of the same dimensions as the input can be obtained, one can attach extra pixels around the input of every convolution layer. This process is called padding.

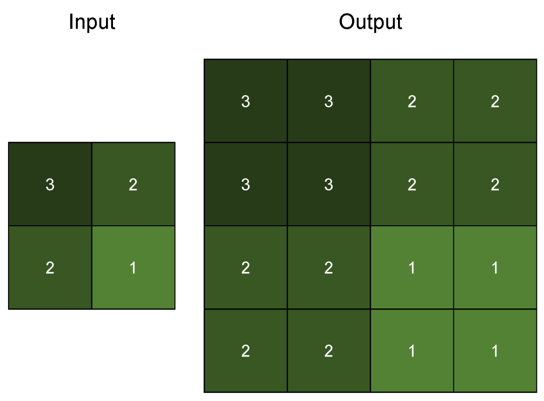


Figure 3.7: Example of upsampling. The value of a pixel is duplicated to three attached neighboring pixels.

4 Related work

This chapter gives an overview of the previous work on lacune detection and segmentation. With lacune detection, lacunes are localized by providing them with a bounding-box or a single point. With lacune segmentation every voxel is being classified, which as a result can, additionally to the location, also give the shape and size of the element. The lacune detection methods are described in the first section, which is followed by a summary of the segmentation methods. The chapter concludes with the contribution and proposal of the current research.

4.1. Detection methods

The first automated lacune detection method was developed by Yokoyama et al. [57]. This lacune detection method is a rule-based approach and consists of two steps: a lacune detection step and a false positive reduction step. In the first step all possible lacune candidates are detected using the brain images of the T2-weighted sequence. As lacunes occur only in some parts of the brain, the method starts with extracting a circular area of the brain in which lacunes are assumed to appear. This area was empirically determined based on the training dataset. After this, the images are binarized: every voxel is evaluated based on its intensity and mapped to either 0 or 1. In the paper it is mentioned that the intensity of lacunes can change according to its phase, which can be acute, sub-acute or chronic. To cope with these differences in intensity, the binarization technique is executed 15 times with 15 different thresholds. From the binarized images, candidates are selected based on their area (number of pixels), circularity (measure of how closely the candidate approaches a perfect circle) and gravity center. Thresholds for these features were optimized based on the training dataset. As this detection step is a very rough procedure, other lesions, brain structures and tissues will be incorrectly identified as candidates as well. Candidates that are misclassified as lacunes are called false positives. In the second step of the model these false positives are reduced using the brain images of the T1-weighted sequence. The brain tissue is extracted from the image as lacunes can never occur outside the brain tissue. Furthermore, on T1-weighted images most lacunes have high intensity differences compared to their surrounding area while this difference is not always present for false positives. Therefore, the intensity difference between the candidate and its peripheral area is measured and evaluated using the T1-weighted images. The threshold for the intensity difference was empirically determined based on the training dataset. For the development of the method 832 images are used of which 295 images contained together 695 lacunes. The method was able to detect 90.1% of the lacunes with an average of 1.4 false positives per image. However, even with the misclassification reduction step, there were still several misclassified candidates. Analyzing these false positives, the researchers found that especially the edge of the brain tissue, high-signal regions near the ventricles (figure 4.1 - b), a part of the cerebral ventricle and perivascular spaces were misclassified as lacunes.

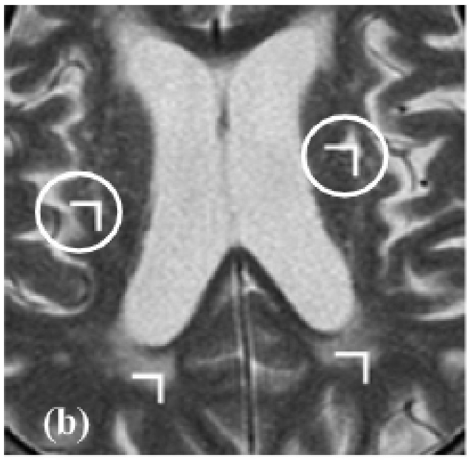


Figure 4.1: Examples of true positives and false positives. The half-boxes represent detections from the method of Yokoyama et al. [57]. A half-box with a circle represents a lacune, a true positive. The half-boxes without a surrounding circle indicate false positives. In this case the false positives are high-signal regions near the ventricle.

Uchiyama et al. [44] adopted the rule-based method of Yokoyama et al. [57], adjusted some parts and added a machine learning algorithm to the end. That is, the method from [44] also exists of two steps: a lacune detection step and a false positive reduction step. The only change that was made to the lacune detection step, is the extraction of the area in which lacunes can occur. The research group of Uchiyama extract in their method the entire brain region based on T1-weighted scans, while the group of Yokoyama extracts only a smaller circular shaped region. After the extraction, the procedure of detection is equal to the approach in [57]. This means that using T2-weighted scans binarization is applied and candidates are detected based on the size of their area, circularity and the gravity center. However, the false positive reduction step from the method of Yokoyama et al. [57] was replaced by a new approach. In the positive reduction step of Uchiyama et al [44], false positives are eliminated based on their location, signal intensity difference (in both T1- and T2-weighted images) and shape of the structure. The location is used, because lacunes occur within cerebral vessel regions and thus candidates on the periphery of the cerebral region are more likely to be false positives. Next, signal intensity difference is evaluated, as again lacunes show an intensity difference with their surrounding area and false positives may not. Finally, false positives are eliminated based on the shape of the structure, as lacunes are more likely to be of a nodular shape and some false positives such as the cerebral sulcus will have a more linear shape. The cut-off thresholds that are used to eliminate the misclassified candidates are determined empirically based on the detected lacunes during the candidate detection phase. After the elimination of the false positives, the remaining candidates are subdivided using a support-vector machine (SVM) into lacunes and false positives. An SVM is a machine learning method that produces a hyperplane to separate different classes for a classification task. To train and test the SVM, the data was split up into groups A and B. Both of these groups were alternately used for training and testing. The final result was then obtained by averaging over the results of both test datasets. To optimize the parameters of the SVM, the training and testing process was repeated. The method is developed based on 1143 scans of which 80 scans contained a lacune leading to a total lacune count of 93. In comparison with the method of Yokoyama et al. [57], the method shows improved results with detecting 96.8% of the lacunes and an average of 0.76 false positive per slice. Although their new approach appears to be better at detecting initial candidates and in reducing the false positives, the method did not succeed in eliminating all false positives. The main types of remaining false positives included a part of the cerebral sulcus (figure 4.2a), a part of the cerebral ventricle (figure 4.2b) and perivascular spaces (figure 4.2c).

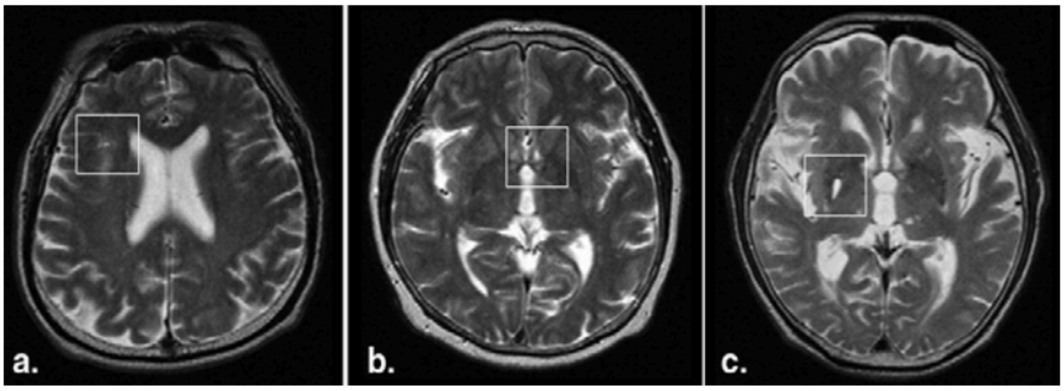


Figure 4.2: Examples of false positives, which are surrounded by boxes. In (a) the false positive is a part of the cerebral sulcus, in (b) the false positive is a part of the cerebral ventricle and in (c) the false positive is a perivascular space [44]

To tackle specifically these remaining false positives, the research group of Uchiyama changed their method in a later paper [43]. In this new method they adopted the rule-based part of their previous method from [44] and replaced the SVM from the false positive reduction step by four neural networks. The first neural network consists of 3 layers and outputs the likelihood of being a lacune. As input for the network the location, intensity difference and shape features are used. Thereafter, three parallel neural networks are applied to attack specific classes of false positives: classifier A distinguishes lacunes from parts of the cerebral sulcus, classifier B distinguishes lacunes from parts of the cerebral ventricle and classifier C distinguishes lacunes from perivascular spaces. For training and testing the neural networks, the dataset was split into two groups A and B. Group A was first used for training and group B for testing. After that, the groups were reversed: group B was used for training and group A for testing. By averaging the results of both test datasets, the final result was obtained. For the development of the method 1143 images are used. 80 of these images contained a lacune with a total of 93 lacunes. This method identified 96.8% of the lacunes with an average of 0.30 false positive per scan. Therefore it is concluded that the neural networks help in reducing the false positives.

As the research group of Uchiyama experienced that especially the differentiation between lacunes and perivascular spaces is challenging, they focused more on addressing this aspect in the method that followed. In [45] all lesions are first enhanced and segmented in T2-weighted images using a thresholding technique. Then, features of location, size, signal intensity difference in both T1- and T2-weighted images and degree of irregularity (a measure of how much the lesion deviates from a perfect circle) are used as input for a 3-layer neural network to classify the lesions. To train and test the network, one example was left out for testing the network and the rest of the examples were used for training. This procedure was repeated until every example was used for testing once. By averaging over all testing results, the final result was obtained. 109 images containing 89 lacunes are used for the development of the method. With this method they showed that size greatly contributes in distinguishing between lacunes and perivascular spaces and that location features are useful for differentiation between perivascular spaces and lacunes located in the periphery of the lateral ventricle. Furthermore, 93.3% of the lacunes were detected and 94.5% of the predictions were correct.

In a later paper [46] the research group of Uchiyama refocused to reducing all of the false positives again. They extended their original method, which was rule-based with a machine learning step [44], with an extra false positive reduction step, which aims at eliminating all false positives [46]. This means that the selection of candidates and the false positive elimination approach is similar to the method in [44]. However, after the two phased elimination step consisting of reducing the false positives using the location features, signal intensity difference features and shape features of the structure in the first phase and grouping the false positives and lacunes using a hyperplane (SVM) in the second, another reduction phase was added. The dataset for this new reduction phase was created by taking a smaller area of $51x51$ around the candidates that resulted from the candidate detection step. These smaller areas were divided into two groups A and B and both were used once for training and once for testing. Each of the training examples were then stretched and attached to each other to obtain an array with a size of the total number of training examples times 2061. After this, the principal component analysis (PCA) is applied to the training data. PCA tries to reduce the total number of variables, in this case

2061, to k number of variables. The aim of the reduction is to remove the information that can lead to misclassifications of the candidates. For the reduction, k vectors $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(k)} \in \mathbb{R}^n$ should be found onto which the data could be projected and such that the distance from the original points to these new vectors is minimized. With these vectors, the eigenspace can then be created onto which all training data is projected. Several values for k were considered for this. For testing, a testing example is taken and projected to the same eigenspace as the training set by using the k vectors from before. Finally, on the eigenspace the correlations between the test example and all training examples are calculated. The test example is labeled as lacune if the correlation with one of the training data lacunes is higher than the correlation with one of the training data false positives. It is labeled as false positive, if the opposite occurs. 1143 images are used to develop the method. From these 1143 images, 80 images contained a lacune with a total lacune count of 93. As a result of this addition, 96.8% of the lacunes were detected while showing an average of 0.47 false positive per image. The best result was achieved with using both T1- and T2-weighted images. It is concluded that if only the features directly around the candidate are used for the differentiation, it can improve the differentiation between lacunes and false positives such as parts of the cerebral ventricles and cerebral sulci.

The most recent method for the segmentation of lacunes is the two-staged deep learning method of Ghafoorian et al. [17], which uses 1075 T1-weighted images and FLAIR images. In the first stage candidates are detected using a 7-layer CNN architecture. For this, small sub-images, which are called patches, are used to capture a local neighborhood around each candidate. During training each of these patches is evaluated for whether it contains a lacune or not. In order to segment the candidates, the fully connected layers are rewritten into convolution layers to obtain an output image with a segmentation. If the network outputs a coarser segmentation of the candidate, it can be fused with the segmentation of other candidates leading to one large segmentation. To detach the distinct candidates, the local maximum is taken from a $10x10$ window that slides over the segmentation region. Then, based on an optimized threshold, only the local maxima with the highest values are considered to be a candidate. The second stage was created to eliminate the false positives with a 9-layer CNN. As location can be a discriminative factor for the differentiation between lacunes and perivascular spaces, explicit location features are added to this second CNN. In addition to this, the network is fed with three different sized images of a same candidate, such that the biggest sized image contains more neighborhood information and as a consequence also more information about the location than the smallest more detailed sized image. 97.4% of the lacunes were detected with this method with an average of 0.13 false positive per slice.

Al-Masni et al. [2] applied a 3D deep learning CNN method to detect lacunes in 3D patches and distinguish them from elements that look similar to lacunes. That is, the network does not work with entire images but expects patches that are suspected to contain a lacune. Based on these patches it should be able to identify whether the patch contains a lacune or not. The patches are of 3 different voxel sizes: $32x32x5$, $48x48x5$ and $64x64x5$. Smaller sized patches include all lacunes and should give more detailed information, while the bigger sized patches also contains information about the anatomical surroundings. To meet the standard size requirements of the network, all patches are resized to a size of $32x32x5$ voxels. T1-weighted and FLAIR images of 288 subjects are used to develop the method. These images included a total of 696 lacunes. The applied deep learning method consists of two parts: a feature extraction network and a classification network. The feature extraction network consists of 6 identical parallel paths. A path consists of a 17-layered CNN. For each suspected lacune, there are 3 different sized patches of both the T1-weighted image and FLAIR image. These 6 images run independently parallel trough the network, after which their results are concatenated and fed to the classification network. The second network consists of 2 layers and classifies the suspected lacune as lacune or non-lacune. Five-fold cross-validation is used to train and test the method. This means that the dataset is divided into 5 groups of which 4 groups are used for training the network and 1 group is used for testing. The procedure of training and testing is repeated 5 times such that every group is used for testing once. The patch-based method is able to detect 96.41% of the lacunes with an average of 1.32 false positives per subject.

In conclusion, the first lacune methods were developed by the group of Yokoyama and Uchiyama. These are mainly rule-based methods, which means that rules had to be manually defined. As a consequence, these methods are not fully automated. With a detection of 96.8% and with an average of 0.30 false positives per image, the best result was obtained in [43]. Compared to the other methods of the same research group, this method applied neural networks in the false positive reduction step. Ghafoorian et al. [17] proved that a more automated deep learning approach can improve the results to detecting 97.4% of the lacunes with an average of 0.13 false positives per image. Where the previous mentioned methods all used 2-dimensional data, Al-Masni et al. [2] applied their method on 3-dimensional data. However, their approach can only be applied to patches instead of entire images. All detection methods consist of at least two stages. This makes these methods more computationally expensive and less fast than a method that has only one stage.

4.2. Segmentation methods

Wang et al. [49] developed the first segmentation method, which is a rule-based method that is able to segment three different types of lesions at once. These types are lacunes, white matter hyperintensities and cortical infarcts. For this, they used scans of 272 subject of which 36 scans contained a lacune covering a total amount of 62 lacunes. The procedure starts with delineating brain tissues based on T1-weighted images, after which hyperintense regions are segmented using FLAIR and T1-weighted images. These hyperintense regions are then further classified into white matter hyperintensities and cortical infarcts using the FLAIR images. Both appear hyperintense on these images, but they can be differentiated based on their location. Then, the lacunes are segmented based on the T1-weighted images, T2-weighted images and FLAIR images. To detect the lacunes near white matter hyperintensities, the segmented white matter hyperintensities are first dilated. After this, voxels within this dilation can be identified as a lacune by comparing the voxel intensity with the average intensity within the white matter hyperintensity region. For segmentation of lacunes in subcortical structures, the intensity of the voxels is compared to the averaged intensity of the specific subcortical structure to determine whether a voxel belongs to a lacune or not. With this method 80.6% of the lacunes were detected while having an average of 0.06 false positive per scan. There is no accuracy score of the lacune segmentation reported.

Another segmentation method was proposed by Sudre et al. [41], which is a 3-dimensional three-staged deep learning method to detect and segment perivascular spaces and lacunes. The data includes 16 T1-weighted, T2-weighted and FLAIR images of 16 elderly subjects having an elevated vascular burden. These images together contain 2442 elements. 14 images are used for training and 2 images are used for testing. The deep learning model exists of 3 independently learned CNNs: a network for extracting features, a network for proposing regions of where the elements might be located and a network for the final classification. For the extraction of the features, a CNN containing 20 layers is used for which patches of size $64x64x64$ are used. The extracted features are used as input for the region proposal network, which is a 2-layered CNN that proposes possible locations of the elements. Boxes containing proposed regions are extracted from the patches and fed to the final classification network. The final network is a 3-layer CNN to classify the boxes and refine the location of the detections. The method is able to detect 72.7% of the elements. Additionally, a median overlap between the manual segmentations and predicted segmentations of 59% is achieved on objects that all raters agreed on, while a median overlap of 30% is obtained for objects that were less certain.

Duan et al. [12] develop a deep learning system in which four cSVD lesion types are segmented. These cSVD types include lacunes, white matter hyperintensities, subcortical infarcts and cerebral microbleeds. The deep learning system consists of four different 8-layers U-net based architectures. That is, for every lesion a separate architecture is applied. All of these architectures are trained independently and output a segmentation of the specific lesion. To obtain a final output, the results of the four architectures are eventually merged together. The deep learning system takes 2-dimensional images as input and output, which are obtained from 3-dimensional images. 3-dimensional prediction images are recreated by concatenating the 2-dimensional output images predicted by the method. For the segmentation of the lacunes 854 T1-weighted images and 854 FLAIR-images from Chinese ischemic stroke or transient ischemic attack patients are used. 30 of those 854 T1-weighted and FLAIR images are used for testing, the remaining images are used for training the method. From the 824 images for training, 98.3% contained a lacune. The 30 images for testing all contained a lacune. DSC and region-wise detection F1 are used to evaluate the method. The region-wise detection F1 metric gives an indication of the detection accuracy, where 0 means no accurate detection and 1 means a perfect detection. The DSC gives an indication of the overlap, where 0 means no overlap and 1 means a perfect overlap. The method achieved a pixel-wise DSC accuracy of 0.496 and a region-wise detection F1 accuracy of 0.783 for specifically the lacune segmentations.

To conclude, three methods have been developed for the segmentation of lacunes. One rule-based approach from Wang et al. [49], which is not fully automated, and two more automated deep learning approaches from Sudre et al. [41] and Duan et al. [12]. From the three segmentation methods only one can be applied to full 3-dimensional images. However, this method from Sudre et al. uses patches instead of entire images and additionally contains three separate networks, which makes the method computationally expensive and less fast. Duan et al. apply their method to entire images with a single network. However, to obtain a prediction of a 3-dimensional image, one needs to slice the image and feed every slice independently to the network, which means that this method is also limited in computational cost and speed when 3-dimensional evaluation is desired.

4.3. Contribution of the current research

We have seen that both rule-based methods and deep learning methods are applied to detect and segment lacunes. From these methods, the deep learning based approaches seem to achieve better results with respect to the detection and average false positive rate per image. Regarding the segmentation methods, only one method [41] is designed for 3-dimensional images. However, this approach uses patches and consists of three separate networks making the process more computationally expensive and slower compared to using entire images and a single network. One method [12] consists of a single network which needs entire images as input. But to apply this method to 3-dimensional data, every slice needs to go through the network, which makes it more time consuming and computationally expensive than when entire 3-dimensional images are used.

This thesis proposes a fully automated lacune segmentation method, which consists of a single convolutional neural network that can be applied to full 3-dimensional images. This will be developed on a dataset of 222 manually segmented images.

5 Data

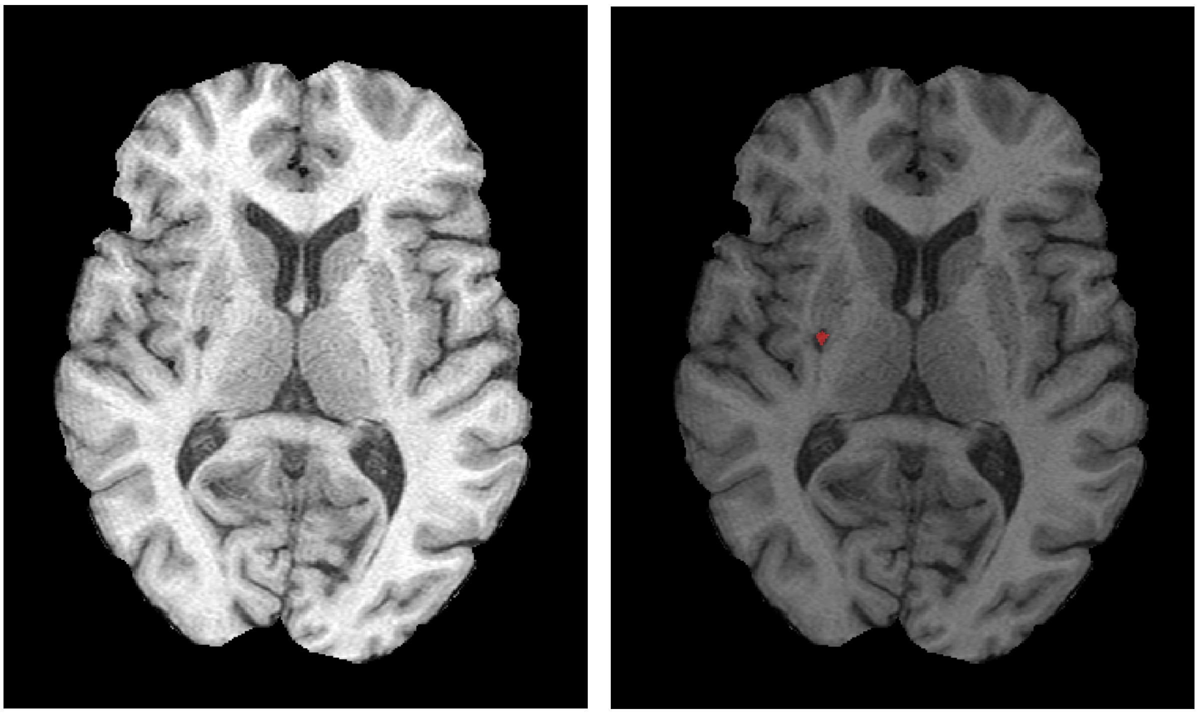
5.1. Rotterdam Scan Study

For the segmentation of the lacunes, brain MRI scans from the Rotterdam scan study are used. The Rotterdam scan study (RSS) is a study with the aim to examine causes and consequences of neurological diseases among the elderly by imaging the underlying pathological changes in the brain [21]. Its participants originate from the related Rotterdam study (RS), a population-based study with the goal of unraveling causes and consequences of chronic disease in mid-life and late-life. The group of participants in this Rotterdam study consists of around 18,000 inhabitants of the district Ommoord in the city of Rotterdam who are all aged 40 years and over. After every 3 to 6 years they undergo extensive physical examinations and are interviewed [22]. A selection of the same group of participants was approached to participate in the RSS to additionally undergo MRI-exams. More specifically, within the group of participants of the Rotterdam study the people that gave consent, and additionally did not have dementia, MRI contra-indications or claustrophobia were considered for the Rotterdam scan study. As a result, by 2015 the amount of 12,147 brain MRI scans have been collected of over 5,800 different participants of 40 years and over [21].

The MRI scans were all performed by a 1.5 Tesla scanner with an 8-channel head coil (General Electric Healthcare). The Tesla value and the number of channels are indicators of the image quality and the examination times. In general, the higher the Tesla and channel number, the quicker the images can be made and the higher their quality. Scanners can have a channel number of up to 32 and can have a Tesla value of up to 7.0, but can also go beyond. The examinations were all executed by trained radiology technicians according to a standardized protocol. Several high-resolution MRI sequences were performed of which, for the purpose of this research, the images of the T1-weighted sequence will be used. The slice thickness of the T1-weighted sequence is 0.8 mm and the slices were contiguous [21]. The researchers of the RSS executed some steps to preprocess the data. To obtain a scan with only brain tissue, the brain is extracted from the image: the cerebellum, eyes and skull are removed. Subsequently, scans are corrected for non-uniformity in intensity. The scans have a dimension of $512x512x192$ voxels, where a voxel is the 3D equivalent of a pixel and has a dimension of $0.49x0.49x0.8$ mm. More information about the data can be found in [21].

5.2. Manual segmentations

In the Rotterdam scan study, lacunes are defined as focal lesions of ≥ 3 and < 15 mm in size with a hypointense appearance on both the T1-weighted and FLAIR images (see figure 5.1a), and with in some cases an additional hyperintense rim on the FLAIR sequence [21]. Around 5,000 of the 12,147 MRI scans have been examined for the presence of these lesions. If lacunes were identified, they were manually segmented by a rater on the T1-weighted image, that is, they were provided with an overlaying mask as is seen in figure 5.1b. As a result, we have 529 lacune segmented scans and around 4,500 scans without annotation.


(a) T1-weighted brain scan.

(b) Annotated T1-weighted brain scan. Here, the lacune is given a red color.

Figure 5.1: Brain MRI T1-weighted scan containing a lacune.

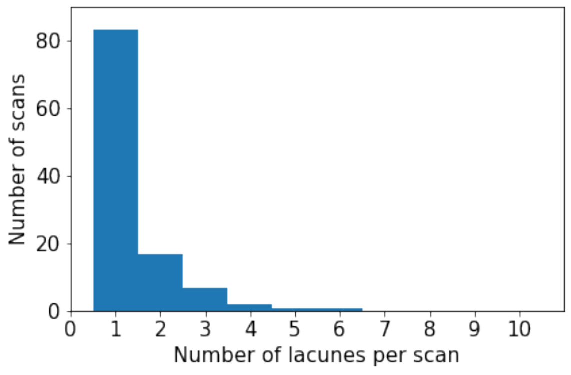
5.3. Data selection

From the 529 manually segmented scans that were available, 307 scans had to be removed: for 12 scans the linked participant numbers could not be found. As a consequence, it was unable to check whether these scans had a follow-up which was linked to the same participant. Therefore it was decided to remove those scans. Another 54 scans had problems with their segmentations, of which most of them seemed shifted. When the scan and segmentation were placed on top of each other, the segmentation was located a few voxels away from the actual lacune on the brain MRI scan. If the manual segmentations are not at their right positions, ambiguous information is given to the network which might prevent the network from learning the right features of a lacune. Lastly, as it was unclear whether the manual segmentations of the scans belonging to patients that underwent more than one MRI scan were correct, these scans are excluded as well. As a consequence, an additional amount of 241 MRI scans were removed. So, after these 307 scans were withdrawn from the dataset, 222 lacune segmented scans remained. Only those manually segmented scans are used for developing an automated lacune segmentation method. These scans are all T1-weighted images.

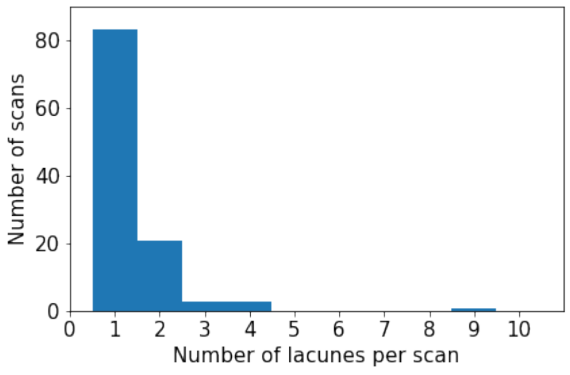
5.4. Data split

The dataset should be subdivided into a training set, a validation set and a test set. The training set is the set of examples that will be used for training the network. During this training phase, the weights of the network will be optimized based on only these training set examples. The examples of the validation set are used to check whether the method is able to generalize to examples that it has not seen during training. When the network shows good results for the training examples, but poor results on other examples, it means that it fails to generalize and changes should be made to the network. During the development stage, the effect of each adjustment to the network should be evaluated using the validation set. This way, parameters of the network can be optimized without depending on the actual test set. The test set is used to evaluate the performance of the final network, when everything is optimized. The training set, validation set and test set contained respectively 40%, 10% and 50% of the total 222 examples.

While the split between the training set and validation set can be made repeatedly, the test set should be left untouched after the split is made. Because we would like to develop a network that is also performing well on unseen data, we would like to test this with the test set. For that reason, the examples of the test set should not be used during the development and kept aside until the end. Therefore, the dataset is first subdivided into two sets, where the training and validation examples are first being taken together to form one set and the other set is the test set. From the 222 examples, 111 examples were randomly allocated to the training- and validation set and the other 111 examples were randomly allocated to the test set. From figure 5.2 it can be seen that the lacunes seem equally distributed over both sets. After the first data split between the training- and validation set and test set is made, a second split is needed to divide the training- and validation set into two separate sets. For this, 89 examples are randomly allocated to the training set and 22 examples are randomly allocated to the validation set.



(a) Distribution of lacunes within the training- and validation set.



(b) Distribution of lacunes within the test set.

Figure 5.2: Histogram per set showing the frequency of scans containing a certain amount of lacunes per scan.

6 Methods

6.1. Architecture

The network architecture of Ronneberger et al. [38] as described in subsection 3.4 and shown in figure 3.6 was adopted to use as initial architecture for the lacune segmentation method. However, as lacunes are different from the cell structures that were segmented in [38], it also needs a different approach. Therefore, some adjustments were made to the original architecture of Ronneberger et al. [38]. Compared to the original network, the initial architecture for the lacune method contained no convolution layers after the upsampling steps and less feature maps. Additionally, it is reformulated from a 2-dimensional to a 3-dimensional network architecture to handle the 3-dimensional images. Lastly, with every pooling operation the image is downsized by mapping parts of the output feature maps to one summary statistic. If too many pooling layers are applied to an image containing a lacune, which is very small, the lacune might end up being described in the deeper layers by only a few or even one voxel. This might lead to a loss of information. Therefore, it was decided to use only two pooling layers.

To be more precise, the network architecture of the lacune segmentation method (see figure 6.1) consists of two paths: a contracting path (the left side of the architecture) and an expanding path (the right side of the architecture). In the contracting path, the images are downsampled while in the expanding path the images are upsampled to obtain its original size and resolution again. The contracting path consists of two blocks, each containing two $3x3x3$ convolutions and a max pooling operation. The expanding path contains two blocks of two $3x3x3$ convolutions, followed by an upsampling operation and a skip connection, which is comparable to a copy and crop operation from the U-net described in section 3.4 and concatenates a copy of the feature maps in the contracting path to the feature maps in the expanding path. After the expanding path two last $3x3x3$ convolutions and a final $1x1x1$ convolution are applied. Each $3x3x3$ convolution of the network is directly followed by a ReLU activation function and the $1x1x1$ convolution is followed by a sigmoid end activation function. Convolutions can increase or decrease the number of feature maps of the image, while each max pooling operation halves the size of the image and each upsampling operation doubles the size of the image. The initial weights of the network are randomly drawn from the standard normal distribution.

With each convolution in the contracting path of the network 16 feature maps are created in the upper part of the network, 32 in the middle part and 64 in the lower part. However, to reduce memory, in the expanding path of the network the images in the upper and middle part of the network consist of only 6 and 3 feature maps, respectively. By taking a smaller number of feature maps in the expanding path, less memory is needed for training the network. With reducing the number of feature maps, we might lose feature information that might be useful to correctly segment the lacunes. However, it is hypothesized that important features can already be recognized by the contracting path of the network and, as this information is transferred with skip connections to the expanding path of the network, it is expected that it is still possible to develop a well performing lacune segmentation method.

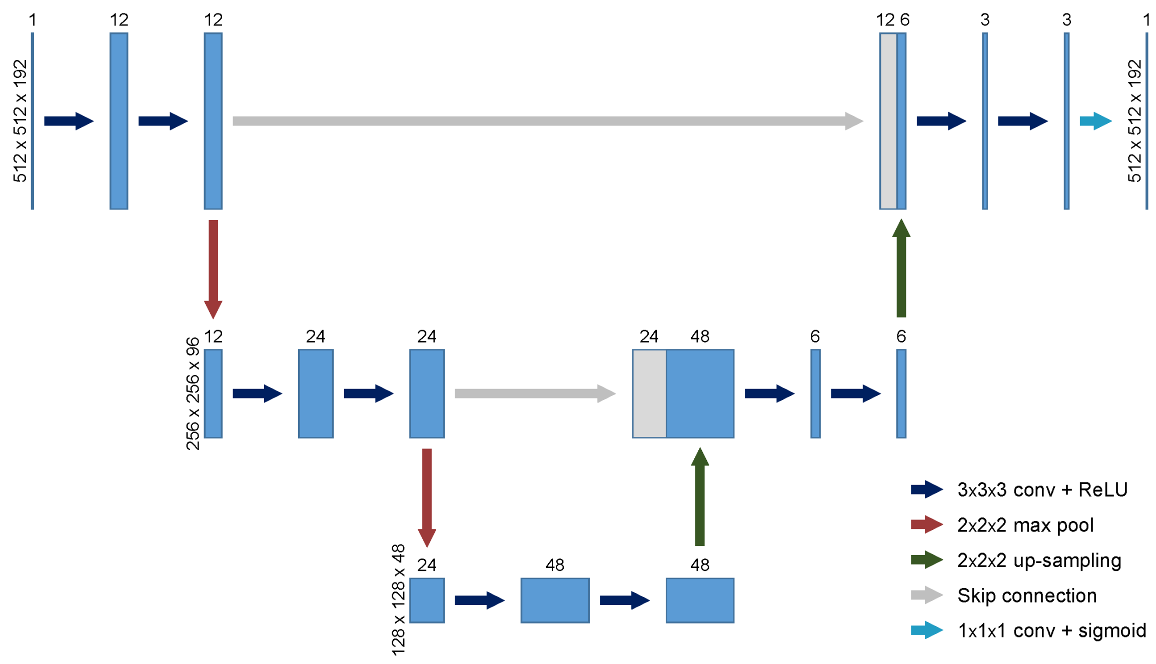


Figure 6.1: Visual representation of the applied network architecture. The architecture consists of convolutions, max pooling operations, upsampling operations and skip connections. The blue bars represent the feature maps. The number of feature maps is reported above these bars and on the left side their size.

6.2. Loss functions

Compared to the entire brain, lacunes are very small. This means that on a brain MRI image, only a few voxels are occupied by the lacune. As a result, with the largest size of the lesion, which has a diameter of 15 mm, a voxel size of 0.5 mm in all directions and an image size of $512x512x192$, the lacune:non-lacune voxel ratio is 1:3,560. This is referred to as class imbalance and can cause a problem when a network is trained. If there are many more samples of one class (referred to as the majority class) than of the other class (the minority class) the learning process often gets stuck in a local minimum. When a network needs to learn from class imbalanced data, it will over-classify the majority class, the non-lacune voxels, because of its increased prior probability. As a consequence, the minority class, the lacune voxels, will often be misclassified [24].

To tackle this challenge of data imbalance, two loss functions that have proven to be able to cope with data imbalances before [33, 38], will first be applied to develop a method for automating lacune segmentations. These loss functions are the weighted binary cross-entropy loss and the Dice loss, where the weighted binary cross-entropy loss is an adaption of the binary cross-entropy loss. In order to see if the model can be optimized even more, two other loss functions, which are adaptations of the Dice loss, will be tested as well. More details about these loss functions follow below.

6.2.1. Binary cross-entropy loss

As described in 3.2.1, an often used loss function for a binary classification task, is the binary cross-entropy (BCE) loss. Therefore, since the lacune segmentation problem is a binary classification task, the first experiments are executed using the binary cross-entropy loss. Let us assume that we have a set of m 3-dimensional images $\{X^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U \times V \times W}$ with manual segmentation output images $\{Y^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U \times V \times W}$ and the predicted output images from the network $\{\hat{P}^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U \times V \times W}$. Additionally, suppose that the manually segmented images are binary, where a voxel value of 1 corresponds to a lacune and a voxel value of 0 corresponds to the background, and the predicted images have voxel values between 0 and 1, where a higher value indicates a higher probability of the voxel being a lacune. Then, with $y_k^{(i)}$ and $p_k^{(i)}$ being the k^{th} voxel values of the segmented image $Y^{(i)}$ and predicted image $P^{(i)}$, the binary cross-entropy loss is computed by

$$\textit{BCE loss} = \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{n}\sum_{k=1}^{n}\left(y_k^{(i)}\log\left(\hat{p}_k^{(i)}\right) + \left(1-y_k^{(i)}\right)\log\left(1-\hat{p}_k^{(i)}\right)\right)\right), \tag{6.1}$$

where the first sum runs over the total number of images m in the set and the second sum runs over the total number of voxels n in an image. Due to the sigmoid (equation 3.4) applied at the end of the network architecture, the prediction voxel values range between $0 < p_k^{(i)} < 1$.

From the notation of this loss we can see that within the second sum, the loss for a lacune voxel ($y_k^{(i)} = 1$) is determined by the first term of the sum, while the loss for a background voxel ($y_k^{(i)} = 0$) is determined by the second term of the sum. The loss for a lacune voxel ($y_k^{(i)} = 1$) will be high when the corresponding voxel from the prediction approaches 0, which is the case when the model predicts background for that voxel. The loss for a lacune voxel ($y_k^{(i)} = 1$) will approach 0 if the prediction approaches 1, i.e. a lacune is predicted. The opposite is true for the background voxels ($y_k^{(i)} = 0$). By the definition of the binary cross-entropy loss, the majority class can dominate the outcome of the loss when the majority class is much bigger than the minority class. This is the case for the current lacune segmentation task, as there are many more background voxels than lacune voxels in the images of the dataset. Therefore, when training with this loss, the model is likely to get stuck in a local minimum and overly-classify the background.

6.2.2. Weighted binary cross-entropy loss

The first loss function that is used to tackle the class imbalance problem, is the weighted binary cross-entropy (WBCE) loss. The weighted binary cross-entropy loss is in fact the binary cross-entropy loss as given by equation 6.1 where a weight α can be applied, as is done by Ronneberger et al. [38], to every voxel of the minority class. If classes are extremely imbalanced, the contribution of the majority class to the binary cross-entropy loss is enormous. As a consequence of this, to reduce the loss, the method primarily focuses on correctly predicting the samples of the majority class, while neglecting the minority class samples. By giving the minority class a higher weight, more importance is given to correctly predict the samples of the minority class as well. With $y_k^{(i)}$ and $\hat{p}_k^{(i)}$ being the k^{th} voxel values of the manually segmented image $Y^{(i)}$ and the predicted image $\hat{P}^{(i)}$ and with α being the weight applied to the minority class, the weighted binary cross-entropy loss is given by,

$$\textit{WBCE loss} = \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{n}\sum_{k=1}^{n}\left(ay_k^{(i)}\log\left(\hat{p}_k^{(i)}\right) + \left(1-y_k^{(i)}\right)\log\left(1-\hat{p}_k^{(i)}\right)\right)\right), \tag{6.2}$$

where once more the first sum runs over the total number of images m in the set and the second sum runs over the total number of voxels n in an image.

Several definitions have been given to the weight α before [33, 38]. However, to cope with the lacune data imbalance, in this report a different definition is used for the weight α. Preliminary experiments showed poor performance when only the minority class is given a weight. To overcome the data imbalance, a very high value of the weight α is needed, resulting in an enormous loss value. As optimizers might be tuned for a certain range in loss values and the loss values obtained by the preliminary experiments achieved much higher values, weighing only the minority class might have complicated the optimization process. Therefore, to prevent the loss value from becoming too high, an additional weight β is applied to the majority class to reduce the importance of the background class from the other side. As the goal was to balance the contribution of the lacune pixels and the background pixels to the loss, the weights α and β were chosen to be $\frac{n}{2q}$ and $\frac{n}{2r}$. Here, $n = q + r$ is the total voxel number of an image, $q = \sum_{k=1}^{n} y_k$ is the number of lacune voxels in the manually segmented image and $r = \sum_{k=1}^{n}(1 - y_k)$ is the number of background voxels in the segmented image. With these definitions for the weights, the final weighted binary cross-entropy loss is defined as follows

$$\begin{aligned}
\textit{WBCE loss} &= \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{n}\sum_{k=1}^{n}\left(\alpha y_k^{(i)}\log\left(\hat{p}_k^{(i)}\right)+\beta\left(1-y_k^{(i)}\right)\log\left(1-\hat{p}_k^{(i)}\right)\right)\right)\\
&= \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{n}\sum_{k=1}^{n}\left(\frac{n}{2q}y_k^{(i)}\log\left(\hat{p}_k^{(i)}\right)+\frac{n}{2r}\left(1-y_k^{(i)}\right)\log\left(1-\hat{p}_k^{(i)}\right)\right)\right)\\
&= \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{2}\sum_{k=1}^{n}\left(\frac{1}{q}y_k^{(i)}\log\left(\hat{p}_k^{(i)}\right)+\frac{1}{r}\left(1-y_k^{(i)}\right)\log\left(1-\hat{p}_k^{(i)}\right)\right)\right)\\
&= \frac{1}{m}\sum_{i=1}^{m}\left(-\frac{1}{2}\left(\frac{1}{q}\sum_{s=1}^{q}\log\left(\hat{p}_s^{(i)}\right)+\frac{1}{r}\sum_{t=1}^{r}\log\left(1-\hat{p}_t^{(i)}\right)\right)\right),
\end{aligned} \tag{6.3}$$

From this notation we can see that the loss per image is calculated by averaging over the mean of the lacune voxel losses and the mean of the background voxel losses. As a consequence, the lacune voxels and background voxels are valued as equally important to the loss. Therefore, when training with this loss, the model is expected to perform better at predicting lacune voxels.

6.2.3. Dice loss

The second loss that is expected to handle the data imbalance is the Dice loss, as this loss has proven to cope well with data imbalances in other applications [33]. The Dice loss is derived from the Dice similarity coefficient (DSC), which measures the similarity between two sets A and B and can be written as

$$\textit{DSC} = \frac{2|A \cap B|}{|A| + |B|},$$

where $|A|$ is the cardinality of set A and $|B|$ is the cardinality of set B. Its value ranges between 0 and 1, where 0 means that there is no similarity between the sets while 1 means that the sets are completely equal to each other. Milletari et al. [33] was the first to use this Dice similarity coefficient as a loss function for binary images. Suppose we have a set of m 3-dimensional images $\{X^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$ with manually segmented images $\{Y^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$ and their predictions $\{\hat{P}^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$. Furthermore, suppose that the manually segmented images are binary, where a lacune is given by 1 and background is given by 0, and the predicted images have continuous values between 0 and 1, where a high value indicates a high probability of the voxel being a lacune. Then, with $y_k^{(i)}$ and $p_k^{(i)}$ being the k^{th} voxel values of the manually segmented image $Y^{(i)}$ and predicted image $P^{(i)}$, the Dice loss reads as

$$\textit{Dice loss} = \frac{1}{m}\sum_{i=1}^{m}\left(1-\frac{2|Y^{(i)}\cap P^{(i)}|}{|Y^{(i)}|+|P^{(i)}|+\epsilon}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(1-\frac{2\sum_{k=1}^{n}y_k^{(i)}\hat{p}_k^{(i)}}{\sum_{k=1}^{n}y_k^{(i)}+\sum_{k=1}^{n}\hat{p}_k^{(i)}+\epsilon}\right), \tag{6.4}$$

where ϵ is a small number, the first sum runs over all images m in the set and the sums in the fraction run over all n voxels in each image. Furthermore, as losses are expected to go down with increased performance, the Dice value is subtracted from 1.

The Dice loss mainly focuses on the similarity of the foreground voxels between the manually segmented image and the predicted image. Unless the background voxels are predicted incorrectly, the Dice loss completely ignores the background voxels. This makes the background voxels less important to optimize. Although it is still important that the background voxels are predicted correctly as well, the loss benefits more from predicting a foreground voxel right than from predicting a background voxel right. Thus, with using the Dice loss, more emphasis is placed on the foreground and therefore it is hypothesized that it is able to cope with the imbalanced images and therewith can help to find the lacunes.

6.2.4. Dice-ReLU loss

Although the formulation of the Dice loss limits the influence of the background voxels, if the background voxels are not predicted exactly as zero they can still dominate the outcome of the loss if the data imbalance is extreme. Even if all background voxels have a small value, adding up millions of small background values can still result in a large number. As a consequence, the denominator of the Dice loss (equation 6.4) becomes small and the loss becomes large. Optimizing the model to predict lower values for the background voxels, getting closer and closer to 0, might thus be more rewarding than predicting the lacune voxels correctly. To prevent the method from optimizing these background voxel values, values below 0.1 are clipped by using a shifted ReLU activation function where a shift of 0.1 is applied to every prediction voxel value $p_k^{(i)}$, such that

$$f\left(p_k^{(i)}\right) = \max\left(0.1, p_k^{(i)}\right).$$

As a consequence, all prediction voxel values with a value that is smaller than 0.1 will be set to 0.1 when calculating the loss. With $y_k^{(i)}$ and $\hat{p}_k^{(i)}$ being the k^{th} voxel values of the i^{th} manually segmented image $Y^{(i)}$ and the i^{th} prediction image $\hat{P}^{(i)}$, we get the following definition of the Dice-ReLU loss,

$$\textit{Dice-ReLU loss} = \frac{1}{m}\sum_{i=1}^{m}\left(1 - \frac{2\sum_{k=1}^{n} y_k^{(i)} \max\left(0.1, \hat{p}_k^{(i)}\right)}{\sum_{k=1}^{n} y_k^{(i)} + \sum_{k=1}^{n} \max\left(0.1, \hat{p}_k^{(i)}\right) + \epsilon}\right), \tag{6.5}$$

where ϵ is a small number, the first sum again runs over the m images in the set and the sums from the fraction run over the voxel values from an image.

With this formulation, putting the background values to 0.1 will give the same value for the loss as when the background values are actually 0.1. As the optimizer tries to minimize the loss, it will most likely not choose to go into the direction in which the background voxels are pushed below 0.1 as this step will not minimize the loss. Instead, to minimize the loss it is expected that the method will focus more on reducing the false positives and finding and maintaining the true positives.

6.2.5. Constrained Dice-ReLU loss

The final loss is the constrained Dice-ReLU (CDR) loss. The CDR loss aims at reducing the false positives, while obtaining and maintaining the true positives by using constraints, as using constraints in loss functions has shown great promise in various applications [6, 26]. Two constraints are added to the Dice-ReLU loss from equation 6.5: a constraint $C_B(V_B, V_T)$ to the volume of the prediction voxels that according to the manually segmented image should be predicted as background and a constraint $C_L(V_L, V_T)$ to the volume of the prediction voxels that according to the manually segmented image should be predicted as lacune.

Let us assume that a set contains m 3-dimensional images $\{X^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$ with manually segmented images $\{Y^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$ and predicted images $\{\hat{P}^{(i)}\}_{i=1}^{m} \in \mathbb{R}^{U\times V\times W}$. Additionally, assume that the segmented images are binary, where a value of 1 represents a lacune and a value of 0 represents the background, and that the predicted images contain values between 0 and 1, where these values indicate the probability of the voxel being a lacune. Then, with $y_k^{(i)}$ and $\hat{p}_k^{(i)}$ being the k^{th} voxel values of the manually segmented image $Y^{(i)}$ and the predicted image $\hat{P}^{(i)}$ and with μ being a parameter defining the contribution of the constraint to the loss, this results in the following definition for the constrained Dice-ReLU loss

$$\textit{CDR loss} = \frac{1}{m}\sum_{i=1}^{m}\left(\left(1 - \frac{2\sum_{k=1}^{n} y_k^{(i)} \max\left(0.1, \hat{p}_k^{(i)}\right)}{\sum_{k=1}^{n} y_k^{(i)} + \sum_{k=1}^{n} \max\left(0.1, \hat{p}_k^{(i)}\right) + \epsilon}\right) + \mu\left(C_B^{(i)}\left(V_B^{(i)}, V_T^{(i)}\right) + C_L^{(i)}\left(V_L^{(i)}, V_T^{(i)}\right)\right)\right), \tag{6.6}$$

where ϵ is a small number, $V_B^{(i)} = \sum_{k=1}^{n}\left(1 - y_k^{(i)}\right)p_k^{(i)}$ is the volume of the background prediction voxels, $V_L^{(i)} = \sum_{k=1}^{n} y_k^{(i)} p_k^{(i)}$ is the volume of the lacune prediction voxels and $V_T^{(i)} = \sum_{k=1}^{n} y_k^{(i)}$ is the volume of the manually segmented lacune voxels. Furthermore, the first sum runs over the total m images of a set, while the sums in the fraction run over the total number of voxels in an image.

The first constraint, $C_B^{(i)}\left(V_B^{(i)}, V_T^{(i)}\right)$, is applied to the voxels of the predicted image that correspond to the background voxels of the manually segmented image to reduce the amount of false positives. Every voxel that has a value bigger than zero, which according to the corresponding voxel in the manually segmented image should be a background voxel and thus should have a value of 0, is considered to be a false positive. In the ideal situation, we would like to have zero false positives, which means that the voxels that according to the manually segmented image should be predicted as background all have a value of 0, leading to a background volume of 0 as well. In order to approach this situation, we constrain the background prediction volume, $V_B^{(i)}$, to be smaller than a portion of the volume of the manually segmented lacune voxels, $V_T^{(i)}$. With FP_{max} being the maximum possible false positive volume, the constraint reads as follows:

$$C_B^{(i)}\left(V_B^{(i)}, V_T^{(i)}\right) = \begin{cases} \frac{\left(V_B^{(i)} - 0.25V_T^{(i)}\right)^2}{\left(FP_{max} - 0.25V_T^{(i)}\right)^2} & \text{if } V_B^{(i)} > 0.25V_T^{(i)}, \\ 0 & \text{otherwise} \end{cases}.$$

By normalizing with FP_{max} the constraint value lies approximately between 0 and 1.

The second constraint, $C_L^{(i)}\left(V_L^{(i)}, V_T^{(i)}\right)$, is applied to the voxels of the predicted image that correspond to the lacune voxels in the manually segmented image to obtain and maintain the true positives. For a well-performing method, we would like to have that all lacunes are segmented correctly. This means that the values of all prediction voxels that correspond to lacune voxels of the manually segmented image should be equal to the values of all manually segmented lacune voxels and thus that their volumes should also be equal to each other. Therefore, the prediction volume of the lacune voxels, $V_L^{(i)}$, is constrained by the volume of the manually segmented lacune voxels, $V_T^{(i)}$, such that

$$C_L^{(i)}\left(V_L^{(i)}, V_T^{(i)}\right) = \begin{cases} \frac{\left(V_L^{(i)} - 0.75V_T^{(i)}\right)^2}{\left(0.75V_T^{(i)}\right)^2} & \text{if } V_L^{(i)} < 0.75V_T^{(i)}, \\ 0 & \text{otherwise} \end{cases}.$$

As a result of this notation, the values of the constraint range approximately from 0 to 1.

Both constraints give some freedom to the method to make some errors by not expecting the background prediction value to be exactly 1 and not expecting the lacune prediction value to be exactly the same as the manual segmentation volume. As a result, more freedom is given to learn new features.

With the notation of the constrained Dice-ReLU loss, the method will be punished by the background volume constraint with an increasing loss when the background prediction volume becomes too large. Additionally, the constraint on the prediction lacune volume punishes the method by increasing the loss when the lacune prediction volume becomes too small. Therefore, as the optimizer tries to find the minimum loss, it is expected that the background constraint will reduce the number of false positives and the lacune constraint helps with obtaining and maintaining the true positives.

6.3. Preprocessing

All images are cropped around the brain to limit GPU memory usage. To obtain one crop size that contains the entire brain of all images, the minimum length, width and height that is needed to contain the entire brain is determined for every 3-dimensional training image. Then, for each dimension the maximum value of all the determined minima is chosen to be the size for the corresponding dimension of the crop. As a result of this, the cropped scans have a size of $304x384x188$ voxels.

To make the optimization process more efficient, all images of the dataset are standardized. With x_i being the i^{th} voxel value of the image and μ and σ being the mean and standard deviation of all voxel values of the image, the following change of scale is performed for the standardization

$$z = \frac{x_i - \mu}{\sigma}.$$

6.4. Network Training

Adam [27] and AdaDelta [58] are used as optimizer for the optimization of the learning process. Both optimizers are a variant of the stochastic gradient descent (SGD) optimizer, which is explained in more detail in subsection 3.2.2. Compared to SGD, which uses a fixed learning rate, Adam and AdaDelta use an adaptive learning rate. This means that SGD updates by applying the same fixed learning rate to every mini-batch and consequently also to every parameter in the network, while Adam and AdaDelta calculate a specific learning rate for each network parameter. As a result, the learning process becomes more efficient. Adam and AdaDelta use a slightly different technique to obtain an adaptive learning rate. More details on how this is achieved can be found in [27] for Adam and in [58] for AdaDelta. To limit memory usage, the mini-batch size consists of only one image. This means that the network parameters will be updated after every image that is passed through the network.

An experiment was terminated when the validation loss of the experiment obtained its minimum and the validation loss did not show further improvement in at least 100 epochs after this minimum was obtained. During the training process of an experiment, weights are updated every epoch. This means that every epoch a new model is created. From all those models, the model at the epoch where the minimum validation loss is obtained is chosen to be the best model.

To provide the network with more variation in images during training, the training images are shifted, rotated and randomly vertically flipped in each epoch. Every image is randomly shifted in each dimension with a maximum shift of 20% of the corresponding dimension length. Random rotation is applied to every dimension of the image with a maximum of 30 degrees.

The method is developed in Python, using Keras, a neural network library, in combination with TensorFlow, an open-source library for machine learning tasks. The experiments are run on a GPU cluster with NVIDIA Quadro P6000 GPU's (24 GB GPU memory) and NVIDIA GeForce RTX 2080 Ti GPU's (12 GB GPU memory). The weighted binary cross-entropy loss experiments are run on the NVIDIA Quadro P6000 GPU's as these experiments need more than 12 GB of GPU memory. All other experiments are able to run on the NVIDIA GeForce RTX 2080 TI GPU's.

6.5. Evaluation methods

This section describes how the results are evaluated. For the evaluation the terms true positive, false negative and false positive are used. A true positive (TP) is a positive predicted outcome that is indeed positive, a false negative (FN) is a negative predicted outcome that should have been positive and a false positive (FP) is a positive predicted outcome that should have been negative.

6.5.1. Thresholding

In order to properly compare the binary manually segmented images with the continuous prediction images, the prediction images need be binarized as well. Therefore, the values of the predicted images are thresholded. With γ being a threshold and y_k a prediction value of voxel k, the predictions are thresholded as follows

$$y_k = \begin{cases} 0, & \text{for } y_k \leq \gamma \\ 1, & \text{for } y_k > \gamma \end{cases}.$$

Every threshold will give another result. To investigate which threshold leads to desired results, multiple thresholds are applied. Each method is thresholded from 0 to 1 with steps of 5×10^{-3}. The methods with the WBCE loss, Dice-ReLU loss and the CDR loss are additionally thresholded from 0.995 to 1 with steps of 2×10^{-5} and from 0.99998 to 1 with steps of 1×10^{-7}, while the method with the Dice loss is additionally thresholded from 0 to 0.005 with steps of 2×10^{-5} and from 0 to 0.00002 with steps of 1×10^{-7}.

6.5.2. Definition of a true positive element

A prediction element is defined as a true positive if it has at least one voxel overlap with the corresponding manually segmented lacune. To prevent trivial solutions from being a true positive (e.g. the entire image is predicted as lacune), the volume of the prediction element should be smaller than the volume obtained with three times the diameter of the corresponding manually segmented lacune. If two or more prediction elements have overlap with the corresponding manually segmented lacune, the prediction element with the smallest distance from its center of mass to the center of mass of the manually segmented lacune is considered to be the true positive. The other prediction element is seen as a false positive. If one prediction element is considered to be a true positive for two or more manually segmented lacunes, the prediction element is a true positive for only the lacune that has the smallest distance from its center of mass to the center of mass of the prediction element. The other manually segmented lacunes are considered to be false negatives. Here, by the previous explanation of differentiating between two predicted elements that have overlap with the same manually segmented lacune, it is assumed that the other manually segmented lacunes can not have an overlap with other prediction elements and therefore are seen as false negatives.

6.5.3. Metrics

This subsection describes the metrics (measures to quantify performance) that are used for the evaluation of the results. The succeeding subsections give a more detailed explanation of how the metrics are used.

Sensitivity

The sensitivity can be used to assess how well the method performs in detecting the lacunes by measuring the proportion of the true positives. It is given by

$$\textit{sensitivity} = \frac{TP}{TP + FN}.$$

Dice similarity coefficient (DSC)

From section 6.2.3, we saw that the Dice similarity coefficient (DSC) measures the similarity between two sets (equation 6.4). We can use this score to assess the performance on overlap between a manually segmented image and a prediction image. With y_k being the k^{th} voxel value of the manually segmented image Y and $\hat{p}_k$ the k^{th} voxel value of the predicted image P, the DSC reads as

$$\textit{DSC} = \frac{2\sum_{k=1}^{n} y_k \hat{p}_k}{\sum_{k=1}^{n} y_k + \sum_{k=1}^{n} \hat{p}_k + \epsilon},$$

where ϵ is a small number and the sum runs over all voxels in an image. Values range between 0 and 1, where 0 indicates no overlap and 1 indicates a perfect overlap.

Relative volume difference

The relative volume difference measures the factor by which the segmentation volume of the predicted image differs from the segmentation volume of the manually segmented image. It is calculated as follows

$$\textit{Relative volume difference} = \frac{\left|\sum_{k=1}^{n} y_k - \sum_{k=1}^{n} \hat{p}_k\right|}{\sum_{k=1}^{n} y_k + \epsilon},$$

where ϵ is a small number and the sum runs over all voxels in an image. A value between 0 and 1 can both indicate undersegmentation and oversegmentation, while a value larger than 1 indicates oversegmentation.

Absolute volume difference

To quantify the exact total number of voxels of which the predicted segmentation differs from the manual segmentation, we also calculate the absolute volume difference:

$$\textit{Absolute volume difference} = \left|\sum_{k=1}^{n} y_k - \sum_{k=1}^{n} \hat{p}_k\right|,$$

where ϵ is a small number and the sum runs over all voxels in an image.

6.5.4. Detection performance

To assess the detection performance of a method, free-response receiver operating characteristic (FROC) curves are plotted. For every threshold, the average number of false positives per image is plotted against the average sensitivity. Both the number of false positives and the sensitivity are calculated element-wise, where the definition of a true positive element is given in subsection 6.5.2. The false positives and the sensitivity are calculated over the total number of images in the test set. With this curve we can see how a certain value for the average sensitivity can be obtained with a certain average number of false positives per image.

6.5.5. Segmentation performance

The segmentation performance is measured voxel-wise in two ways. To obtain the overall segmentation performance, metrics are applied to the entire image. The segmentation performance of a TP-element is assessed by applying the metrics to only the voxels of that TP-element.

Overall segmentation performance

To measure the overall segmentation performance, metrics are applied voxel-wise to the entire image. These metrics include the DSC, the relative volume difference and the absolute volume difference. They are averaged over the total images in the test set.

TP-element-wise segmentation performance

To assess how well the method performs on segmenting specifically the TPs, only the prediction voxels of the TP element will be evaluated. For this, the DSC, the relative volume difference and the absolute volume difference are calculated and averaged over the total number of TP elements in the test set.

6.5.6. Consistency performance

By means of a random number generator, the initial weights of the network are randomly drawn from the standard normal distribution. The random number generator needs a seed for this: a number to start with. To assess if the method can obtain comparable results with different seeds, all experiments are executed three times with for each time a different seed. This way the consistency performance of a method is evaluated.

7 Experiments

This chapter describes the executed experiments. First, a section follows with the description of the preliminary experiments. These are used to get an indication of which methods might perform well and, therefore, are executed based on only a small part of the dataset. The section will be concluded with the findings from these exploratory experiments.

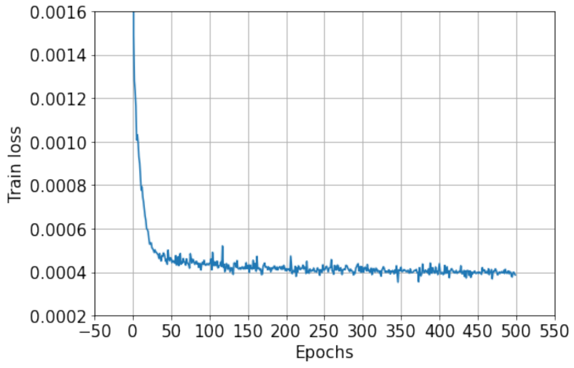
The main experiments are explained in the next section. These experiments are executed based on the entire dataset. The results of the main experiments follow in chapter 8.

7.1. Preliminary experiments

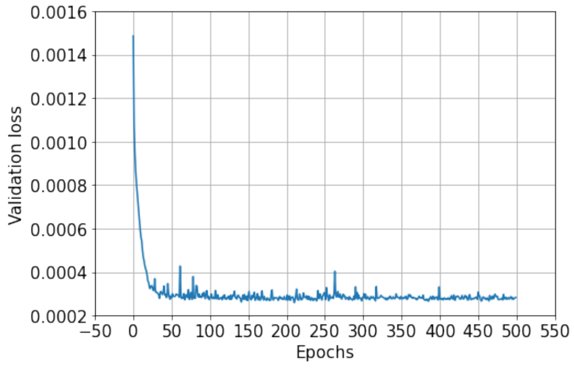
The goal of the first experiments is to explore which loss function might have potential to lead to a well performing lacune segmentation method. For this, the binary cross-entropy loss, weighted binary cross-entropy loss and the Dice loss are tested on only a selection of the dataset: 10 scans of the training dataset and 1 scan of the validation set. Compared to the network shown in figure 6.1 the network architecture used for the preliminary experiments is symmetric and contains more feature maps. That is, the architecture globally resembles the architecture explained in section 6.1, but differs in the number of feature maps having 16 features in the upper layer, 24 features in the middle layer and 48 features in the lower layer. To compensate for the extra GPU memory that a more elaborated network requires, a tighter crop is needed. Therefore, the images are cropped to a size of $212x232x136$ voxels. Although these tighter crops do not capture the entire brain, they still contain all lacunes of the examples in the small dataset. Both Adam and AdaDelta are applied for the optimization of the experiments. Several learning rates are tested to see which learning rate should be applied to obtain the best performance. As the losses have different ranges (the Dice loss ranges between 0 and 1, while the binary cross-entropy loss and the weighted binary cross-entropy loss can obtain values beyond 1), they might benefit from a different learning rate value. Therefore, the learning rates are varied for every loss. The findings of the preliminary experiments are described below.

Binary cross-entropy loss

Figure 7.1 shows the evolution of the losses during training of an experiment trained with the binary cross-entropy loss and AdaDelta optimizer with learning rate 1. The figure of this example displays how a loss evolves during optimization. In the ideal case both losses approach zero with an increasing number of epochs indicating that the model, having learned the right features, is able to resemble the manually segmented image perfectly. Both the train loss and the validation loss of this example show a sharp decrease during the first few epochs. In the remaining epochs, the losses keep decreasing, but at a much lower pace. The lowest value of the validation loss is obtained after 210 epochs. Figure 7.2 shows image slices of the predictions at this epoch. Figures 7.2a and 7.2c are the manually segmented images of a train and a validation example respectively. Figures 7.2b and 7.2d are their thresholded predictions. For the plotted images a threshold of 0.1 was used. However, despite of this low threshold, the prediction images do not display any foreground and thus also no lacunes. This means that the prediction values are very close to zero in these particular slices. Inspecting the values of the entire images, it appears that all prediction values have a value near zero. This applies to all examples in the dataset.

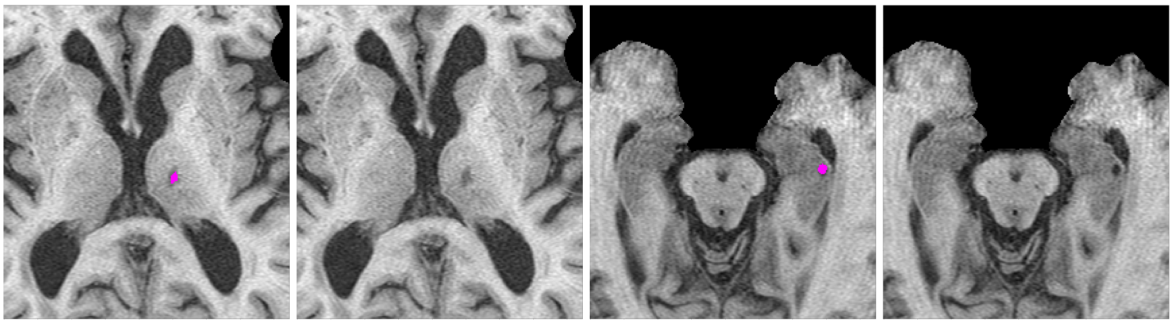


(a) Train loss.



(b) Validation loss.

Figure 7.1: Evolution of the train and validation loss over the number of epochs. The results are obtained with the binary cross-entropy loss and AdaDelta optimizer with learning rate 1.



(a) Manually segmented train image. (b) Predicted train image. (c) Manually segmented validation image. (d) Predicted validation image.

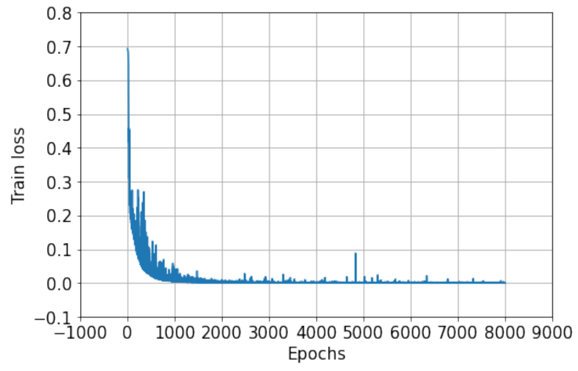
Figure 7.2: Plots of a train example slice and a validation example slice, where (a) and (c) are the manually segmented image slices and (b) and (d) are the predicted image slices. A threshold of 0.1 is used for the prediction plots. The results are obtained with the binary cross-entropy loss and AdaDelta optimizer with learning rate 1.

Similar results are obtained for the AdaDelta optimizer with learning rates 1×10^{-1}, 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6} and the Adam optimizer with learning rates 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}. Based on these results it seems that experiments with the binary cross-entropy loss get stuck in a local minimum. With all prediction values being close to zero, the background is over-classified which might be the result of its increased prior probability. Therefore it looks like the binary cross-entropy loss suffers from the data imbalance and is not able to learn correct features for recognizing lacunes.

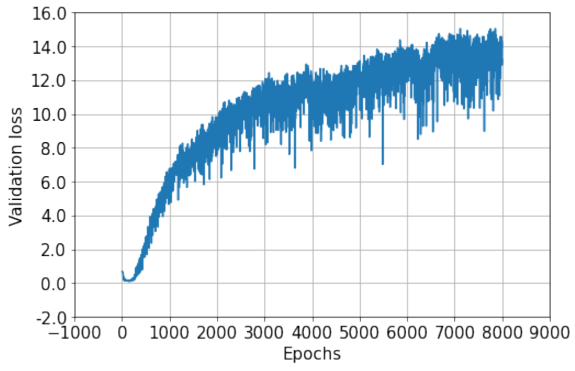
Weighted binary cross-entropy loss

Figure 7.3 shows the losses of the experiment with the weighted binary cross-entropy loss and AdaDelta optimizer with learning rate 1×10^{-2}. We can see that both the train and validation losses are decreasing during the first epochs. However, while the train loss keeps decreasing, the validation loss goes up very fast after these first few epochs. The validation loss obtains its minimum at epoch 147. At this point the method seems to be able to roughly segment the lacunes of both train (figure 7.4b) and validation images (figure 7.4e). However, we can see that other structures are segmented by the method as well. An explanation for this behaviour might be that the model has not learned enough features yet that specifically apply to lacunes, but rather that it has learned more general features that apply to both lacunes and other similar looking structures. More specific lacune features might be obtained by giving the method more time to learn from the data.

When we look at the results at epoch 8000 (figures 7.4c and 7.4f), we find that the method is able to recognize the lacune of the train example without segmenting other structures, but that it fails to recognize the lacune of the validation example. At epoch 8000 the method seems to perform better in predicting the lacunes and removing the false positives on the train images than at epoch 147. This is

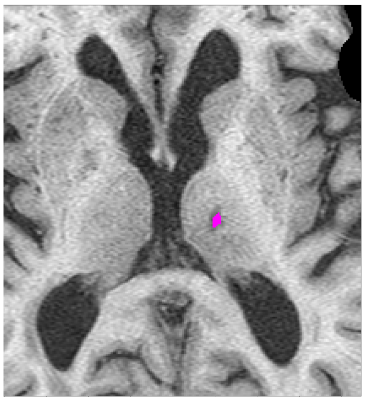


(a) Train loss.

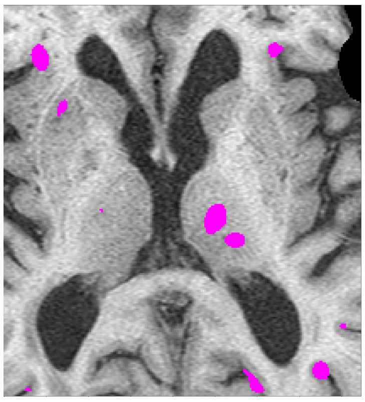


(b) Validation loss.

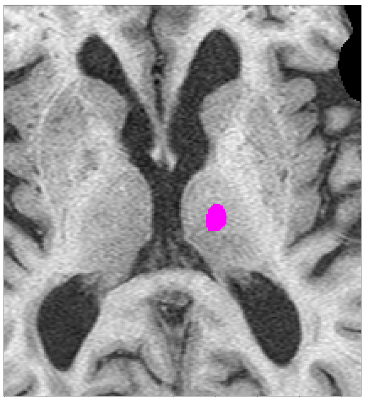
Figure 7.3: Evolution of the train and validation loss over the number of epochs. The results are obtained with the weighted binary cross-entropy loss and AdaDelta optimizer with learning rate 1×10^{-2}.



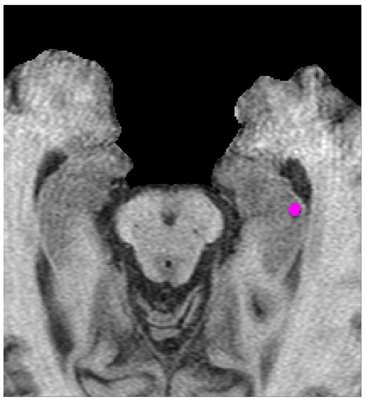
(a) Manually segmented train image.



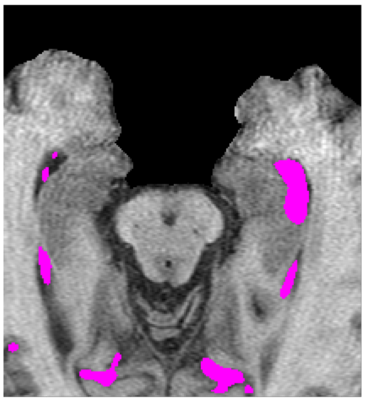
(b) Predicted train image. Epoch=147. Threshold=0.9.



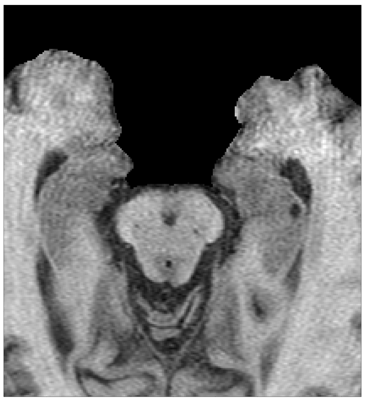
(c) Predicted train image. Epoch=8000. Threshold=0.9.



(d) Manually segmented validation image.



(e) Predicted validation image. Epoch=147. Threshold=0.9.



(f) Predicted validation image. Epoch=8000. Threshold=0.9.

Figure 7.4: Plots of a train example slice and a validation example slice. The first column contains the manually segmented image slices, the second column contains the predicted image slices at epoch 147 with threshold 0.9 and the last column contains the predicted image slices at epoch 8000 with threshold 0.9. The results are obtained with the weighted binary cross-entropy loss and AdaDelta optimizer with learning rate 1×10^{-2}.

in line with the pattern of the train loss in figure 7.3a, as the loss keeps decreasing after epoch 147. However, for the validation example the method seems to perform worse with increasing epochs as the lacune that was segmented in epoch 147 is not present anymore in epoch 8000. This corresponds to the behaviour of the validation loss in figure 7.3b, which is increasing after epoch 147. An explanation for the difference in performance on the train and validation data, might be that the features have become too specific for the lacunes in the train examples. That is, it might have learned features that apply specifically to the lacunes in the train set and because lacunes in the validation set might look slightly different, the method fails to recognize the lacunes of the validation example. This is also known as overfitting and can occur when a small dataset is used. It can be prevented by providing the method with more variability in lacunes and their surroundings, such that the method can learn features that are more generalizable to the validation examples as well. This can be achieved by taking more examples and thus by taking a larger dataset.

As these are explorative experiments, we cannot draw hard conclusions from them. However, the weighted binary cross-entropy loss looks promising in tackling the data imbalance and in segmenting the lacunes. A larger dataset is needed to prevent the method from overfitting.

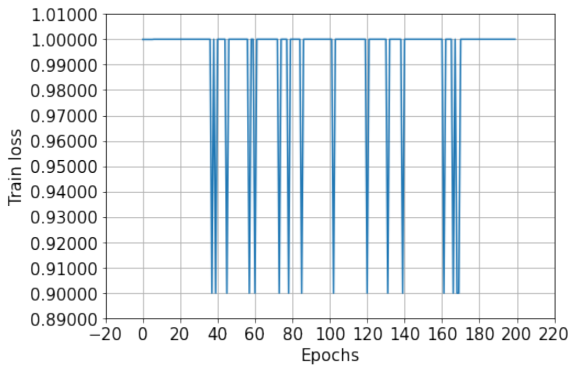
Dice loss

In figure 7.5 we find the evolution of the losses for the experiment that is executed with the Dice loss and Adam optimizer with learning rate 1×10^{-3}. Neither of these losses show a gradual decreasing pattern, which might indicate that the method has trouble with learning. The validation loss obtains its minimum at epoch 4. After 4 epochs the train loss in figure 7.5 seems not to obtain lower values than 1. However, when inspecting these values more closely, they appear to decrease with very tiny steps, which makes it barely visible on the plot. Figures 7.6b and 7.6e show that at this epoch the method is able to roughly segment the lacunes, but the segmentation does not have the correct size and shape of the manually segmented annotation yet. Furthermore, additional regions and structures are falsely segmented as well. After the fourth epoch the validation loss increases rapidly (figure 7.5b) and remains at a high value for the rest of the experiment. The train loss displays some drops, but always returns to a high loss value too (figure 7.5a). Figures 7.6c and 7.6f show the results of the method at epoch 200 where both losses have these high values. We can see that at this epoch the prediction image slices do not display any foreground at a threshold of 0.1. This means that the prediction values of these image slices are close to zero. In fact, the prediction values of all images of the dataset appear to be near zero.

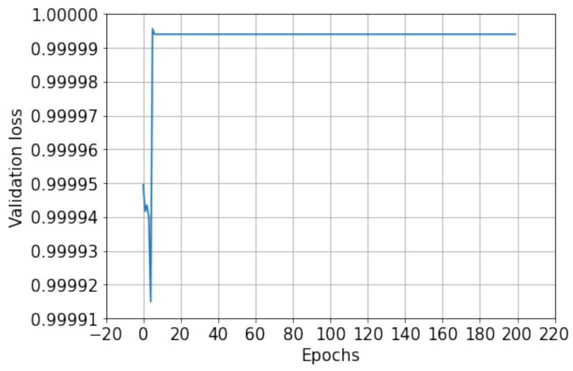
Comparable results are obtained for the Adam optimizer with learning rates 1×10^{-2}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}, 1×10^{-7} and for the AdaDelta optimizer with learning rates 1, 1×10^{-1}, 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}, 1×10^{-7}. Based on figures 7.6b and 7.6e it seems that only a few lacune features have been learned at epoch 4 as the segmentation is not very accurate yet. One might expect the method to perform better on segmenting lacunes with increasing epochs, as with a greater amount of epochs the method has more time to learn more specific and advanced features. However, with increasing epochs the method seems to fail in further improving the segmentation of the lacunes as the loss does not decrease further and the figures 7.6c and 7.6f at a later epoch show no lacune at all.

Looking at the results, it seems like the method has trouble finding the right lacune features that are needed to accurately segment the lacunes. It is unsure why the method is not able to learn properly, but an explanation might be that it is suffering from the data imbalance. As the Dice loss mainly focuses on the similarity of the foreground voxels between the manually segmented image and the predicted image, it was expected that it is able to handle the data imbalance. However, since there are many more background voxels than foreground voxels, the method might have a problem with finding those foreground voxels. As a consequence of this, as long as it can not find those foreground voxels, the Dice loss stays at a maximum value of 1 and the method updates its parameters randomly and thus is not able to learn any features yet. However, this does not explain the drops in loss after which it returns back to 1. The cause of this behaviour should be further investigated.

From these Dice loss preliminary experiments, we saw that the method has trouble with learning on a small dataset. Although it appears to learn some lacune features, it seems to fail to improve these features. However, when a larger dataset is used, which means more information about the lacunes is provided, the method might perform better on learning the lacune features.

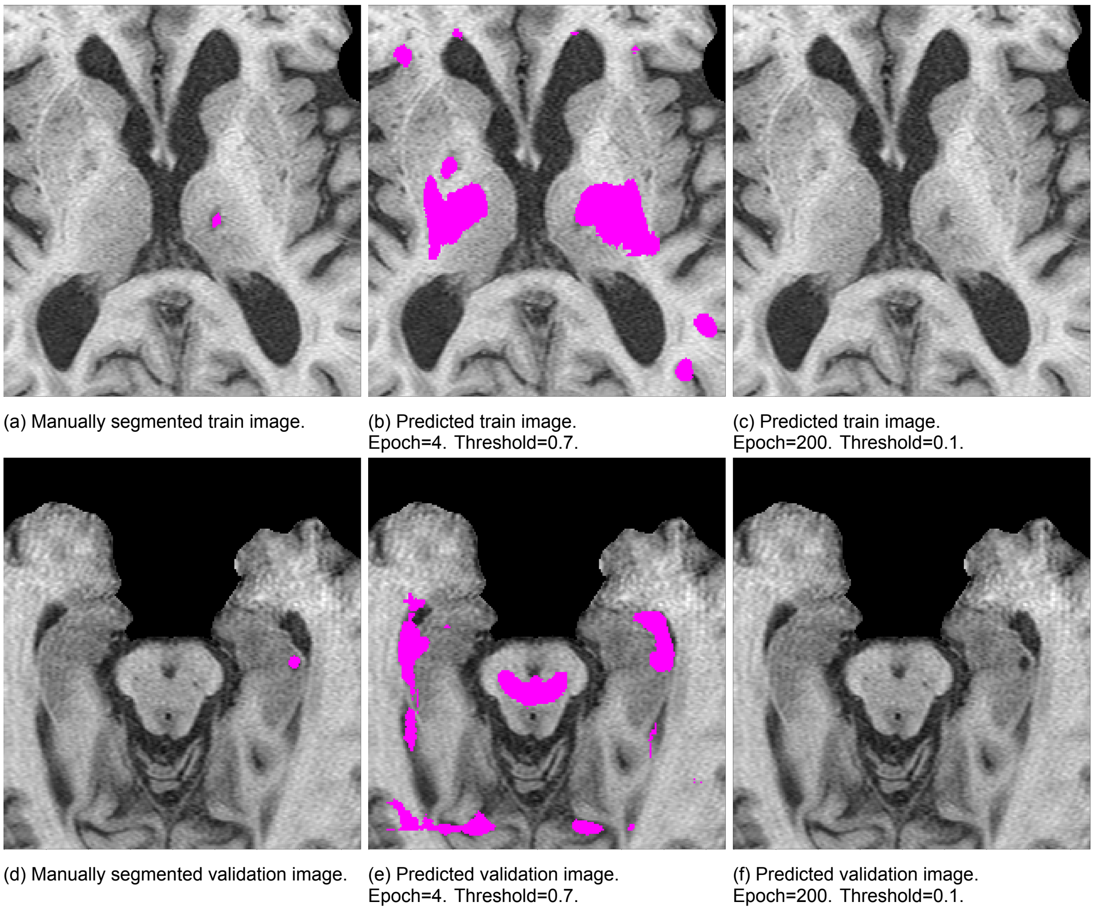


(a) Train loss.



(b) Validation loss.

Figure 7.5: Evolution of the train and validation loss over the number of epochs. The results are obtained with the Dice loss and Adam optimizer with learning rate 1×10^{-3}.



(a) Manually segmented train image. (b) Predicted train image. Epoch=4. Threshold=0.7. (c) Predicted train image. Epoch=200. Threshold=0.1.

(d) Manually segmented validation image. (e) Predicted validation image. Epoch=4. Threshold=0.7. (f) Predicted validation image. Epoch=200. Threshold=0.1.

Figure 7.6: Plots of a train example slice and a validation example slice. The first column contains the manually segmented image slices, the second column contains the predicted image slices at epoch 4 with threshold 0.7 and the last column contains the predicted image slices at epoch 200 with threshold 0.5. The results are obtained with the Dice loss and AdaDelta optimizer with learning rate 1×10^{-3}.

Preliminary thoughts

Since for these preliminary experiments a small dataset was used and only a few experiments were executed, no conclusions can be drawn from these exploratory experiments. However, based on this small dataset it seems like both the binary cross-entropy loss and the Dice loss might suffer from the data imbalance. The method might perform better with the Dice loss if a larger dataset is used. Lastly, the weighted binary cross-entropy loss looks promising in tackling the data imbalance.

7.2. Main experiments

The goal of the main experiments is to compare the performance of the methods with the loss functions proposed in section 6.2. These loss functions include the binary cross-entropy loss, the weighted binary cross-entropy loss, the Dice loss, the Dice-ReLU loss and the constrained Dice-ReLU loss. For this, the network architecture described in section 6.1 and shown in figure 6.1 is used. As described in 5.4, 89 scans are used for training, 22 for validation and 111 for testing. All experiments are executed based on the entire dataset, where the images are cropped as described in section 6.3 to a size of $304x384x188$ voxels.

As the preliminary experiments of the binary cross-entropy loss and the weighted binary cross-entropy loss showed better performance with the Adam optimizer, Adam is used for experiments with these losses. The preliminary experiments of the Dice loss showed better performance with the AdaDelta optimizer. Therefore AdaDelta is used for all Dice loss related experiments. The experiments with the binary cross-entropy loss and the weighted binary cross-entropy loss are optimized by AdaDelta with learning rate 1×10^{-2}. The Dice loss, Dice-ReLU loss and the constrained Dice-ReLU loss are all optimized by the Adam optimizer, where for the Dice loss and the Dice-ReLU loss a learning rate of 1×10^{-4} is used and for the constrained Dice-ReLU loss a learning rate of 1×10^{-7} is adopted. These learning rates are chosen based on the preliminary experiments.

The constrained Dice-ReLU (CDR) loss will be applied to the best model trained using the Dice-ReLU loss. That is, the parameter settings that lead to the minimum validation loss for the Dice-ReLU loss, were adopted and used as initial parameters for the constrained Dice-ReLU loss. As a consequence, the final loss will continue the optimization process from where the Dice-ReLU loss has left off.

For the experiments of the CDR loss (equation 6.6), the parameter μ is set equal to 0.5 times the validation loss value of the best Dice-ReLU loss model from where the constrained Dice-ReLU loss will continue. As both constraints can obtain a maximum value of 1, together they can have a maximum value of 2. Therefore, when both constraints obtain those maximum values, putting μ to half of the validation loss, leads to an equal contribution of the first part of the loss function (Dice-ReLU loss) and the constraints. To determine the value for the FP_{max}, the total predicted background volume is calculated for every validation example, which are predicted by the best model from the Dice-ReLU loss method. The FP_{max} is then set equal to the maximum of those calculated background volumes.

8 Results

This chapter describes the results of the main experiments. Methods will be compared based on their detection performance, overall segmentation performance and their segmentation performance of specifically the true positives. Lastly, the consistency performance of the methods will be assessed.

Figure 8.1 shows the free-response receiver operating characteristic (FROC) curves of the weighted binary cross-entropy (WBCE) loss, the Dice loss, the Dice-ReLU loss and the constrained Dice-ReLU (CDR) loss. In the figure the average number of false positive elements per image is plotted against the average element-wise sensitivity, where sensitivity is the detection rate. Every star represents a measure point, a different threshold that is used to binarize the prediction image. The binary cross-entropy (BCE) loss failed to detect and segment lacunes and as a consequence predicted the entire image as background. Therefore, the BCE loss is not included in the FROC plot.

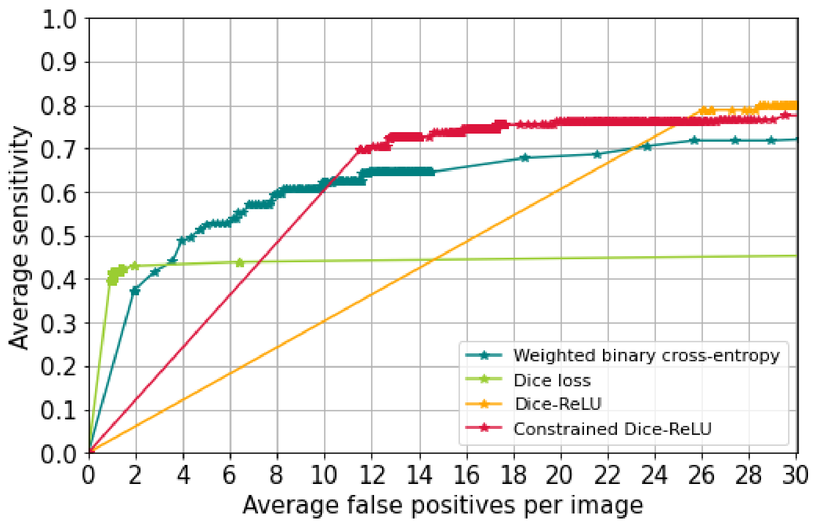


Figure 8.1: Plot with FROC curves of the average false positives per image against the average sensitivity over a range of thresholds. FROC curves of the WBCE loss, the Dice loss, the Dice-ReLU loss and the CDR loss are displayed.

The FROC of the Dice loss shows that with the Dice loss, only a few false positives per image are obtained. But this occurs with a low sensitivity. So, less than half of the lacunes are detected with around 1.0 false positives per image. Compared to the Dice-ReLU loss, the FROCs of the WBCE loss, the Dice-ReLU loss, the CDR loss all perform better on the average sensitivity. However, the methods also produce many false positives. The Dice-ReLU loss produces twice as many false positives than the CDR loss with a comparable sensitivity. Compared to the Dice-ReLU loss and the CDR loss, the WBCE loss produces less false positives per image, but performs poor on the detection of the lacunes. All methods show few measure points near the origin. This occurs because the prediction outputs of all losses contain many values of exactly 1. The first measure point that follows after the origin is the point, as a result of thresholding, at which only the values of 1 remain. As voxels can not obtain a value higher than 1, using more thresholds will not help in obtaining more measure points near the origin.

The FROC of the Dice loss has many measure points aggregated together around one point. This occurs because all values of the prediction output appear to be very close to either 0 or 1. As a consequence, when the threshold is varied between 0 and 1, we obtain barely any differences in sensitivity and false positives leading to aggregated measure points.

Table 8.1 shows the overall segmentation performance of the loss functions, which is the segmentation performance in the full image. It gives an indication of how well methods are able to correctly classify voxels as lacune or background in the full image. This performance is evaluated by computing the overall Dice similarity coefficient (DSC), the overall relative volume difference and the overall absolute volume difference, which were all computed on the entire image. With the DSC an indication is given of the overlap between the manually segmented image and the prediction image. The relative volume difference is the rate at which the prediction volume differs from the manually segmented volume. With the absolute volume difference the exact difference in number of voxels between the manually segmented image and the prediction image is given. From the table we find that the Dice loss performs best on all metrics, while the Dice-ReLU loss and the CDR loss have the lowest performance. The results show that the volume differences of the Dice-ReLU loss can be halved by applying the CDR loss, which supports the findings of the FROC curves of these loss functions.

When choosing the threshold per method based on the best overall DSC value (see table 8.1), the sensitivity is 0.49 with 3.96 false positives per image for the WBCE loss, the sensitivity is 0.43 with 1.93 false positives per image for the Dice loss, the sensitivity is 0.79 with 26.03 false positives for the Dice-ReLU loss and the sensitivity is 0.70 with 11.52 false positives per image for the CDR loss.

Overall segmentation performance			
Loss function	DSC (mean ± STD)	Relative volume difference (mean ± STD)	Absolute volume difference (mean ± STD)
BCE	-	-	-
WBCE	0.14 ± 0.19	1.07 ± 1.37	243.10 ± 395.77
Dice	**0.19 ± 0.25**	**0.89 ± 1.69**	**204.01 ± 370.38**
Dice-ReLU	0.05 ± 0.05	42.28 ± 43.78	5409.05 ± 2445.62
CDR	0.08 ± 0.08	18.28 ± 20.59	2424.08 ± 1575.61

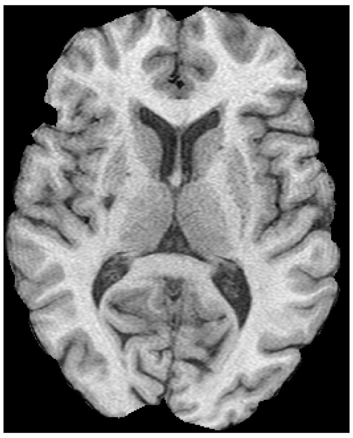
Table 8.1: These metrics quantify the overall segmentation performance and are computed voxel-wise over the full image. The overall segmentation performance gives an indication of how well methods are able to correctly classify voxels as lacunes or background in the full image. The table shows the values corresponding to the threshold value that gave the best metric score for each loss function. For each metric the mean and standard deviation are given, where the best obtained result is displayed in bold.

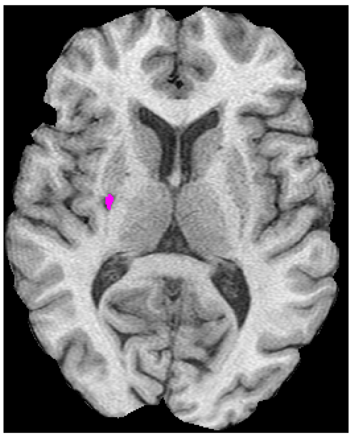
To quantify the segmentation of specifically the TP elements, the same metrics from above are computed exclusively on the correctly detected elements (true positives). Every true positive (TP) element is evaluated using the TP-element-wise DSC value, the TP-element-wise relative volume difference and the TP-element-wise absolute volume difference by comparing the predicted element voxel-wise with the corresponding manually segmented element. DSC indicates the degree of overlap between the manually segmented lacune and the predicted lacune. The relative volume difference is the rate at which the predicted lacune volume differs from the manually segmented lacune volume. The absolute volume difference measures the exact difference in number of voxels between a manual lacune segmentation and a predicted lacune segmentation. The results are given in table 8.2. These results show that the Dice loss performs best on the DSC and the relative volume difference. The WBCE loss achieves a comparable DSC value and performs better on the absolute volume difference. Once more, the Dice-ReLU loss and the CDR loss show the lowest performance.

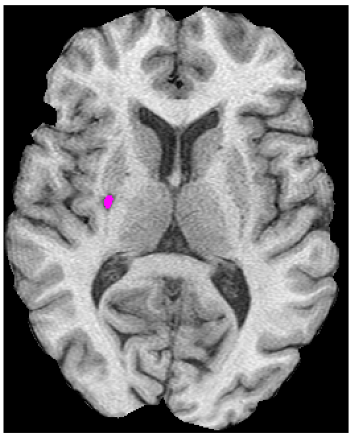
The relative volume difference indicates how much methods over- or undersegment. Undersegmentation is limited to segmenting no voxels at all, which would give a relative volume difference of 1. Any values above 1 therefore indicate oversegmentation. The Dice-ReLU loss and the CDR loss both clearly oversegment the lacunes, with the Dice-ReLU loss predicting almost 6 times the manually segmented volume and the CDR loss almost 5 times. As a relative volume difference smaller than 1 can indicate both oversegmentation and undersegmentation, it is unclear from these results whether the WBCE loss and the Dice loss are mainly oversegmenting or undersegmenting the lacunes. After inspecting several predicted images, we suspect that both methods have a tendency to undersegment the lacunes. Figure 8.2 shows an example of a lacune that is correctly detected by all loss functions. Here, we see that the WBCE loss and the Dice loss resemble the shape of the manual segmentation, while the Dice-ReLU loss and the CDR loss are oversegmenting the lacune.

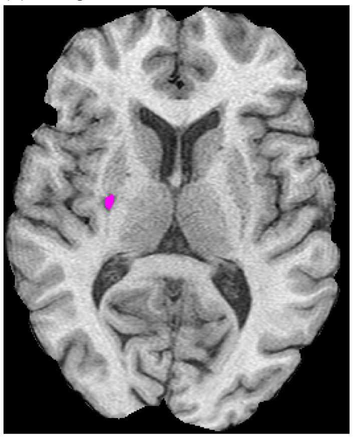
TP-element-wise segmentation performance			
Loss function	DSC (mean ± STD)	Relative volume difference (mean ± STD)	Absolute volume difference (mean ± STD)
BCE	-	-	-
WBCE	0.45 ± 0.21	0.75 ± 0.61	**106.74 ± 87.05**
Dice	**0.47 ± 0.23**	**0.49 ± 0.30**	132.88 ± 314.79
Dice-ReLU	0.28 ± 0.15	5.93 ± 5.65	738.66 ± 509.36
CDR	0.29 ± 0.14	4.77 ± 4.41	601.65 ± 428.37

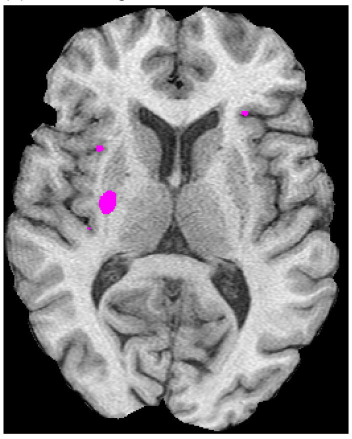
Table 8.2: These metrics quantify the segmentation performance of specifically the TP elements and are applied to only the voxels of the TP element. The TP-element-wise segmentation performance gives an indication of how well methods are able to segment individual lacunes. The table shows the values corresponding to the threshold value that gave the best metric score for each loss function. For each metric the mean and standard deviation are given, where the best obtained result is displayed in bold.


(a) Unsegmented lacune.


(b) Manual segmentation of the lacune.


(c) TP predicted with the WBCE loss.


(d) TP predicted with the Dice loss.


(e) TP predicted with the Dice-ReLU loss.

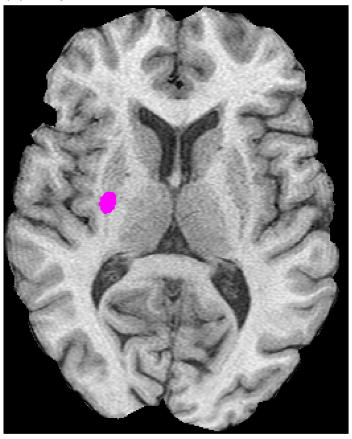
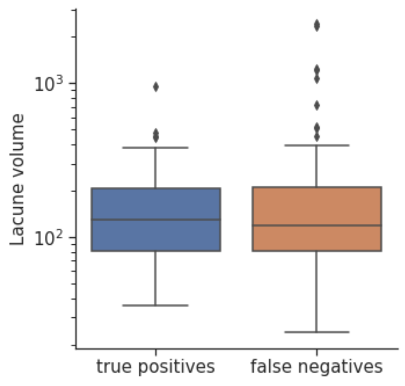
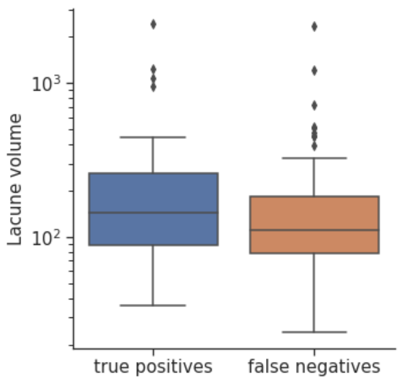

(f) TP predicted with the CDR loss.

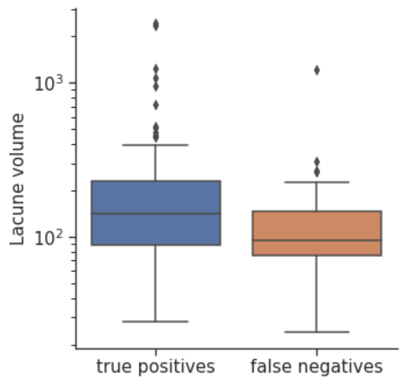
Figure 8.2: An example of a lacune that is correctly detected by every method. The TPs for each loss function are obtained with the threshold that gave the best performance on the overall segmentation DSC metric.



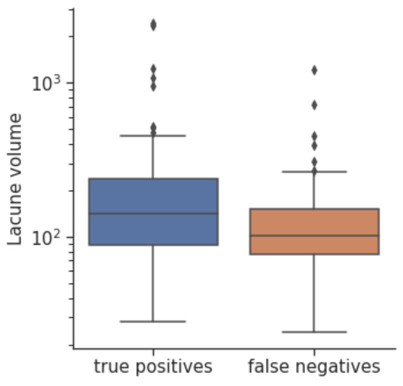
(a) WBCE loss.



(b) Dice loss.



(c) Dice-ReLU loss.



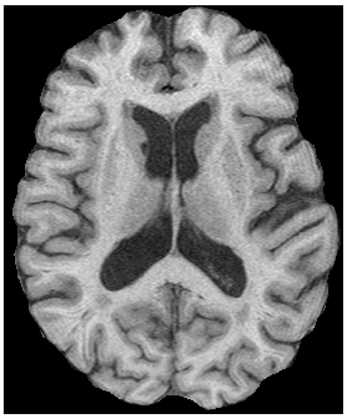
(d) CDR loss.

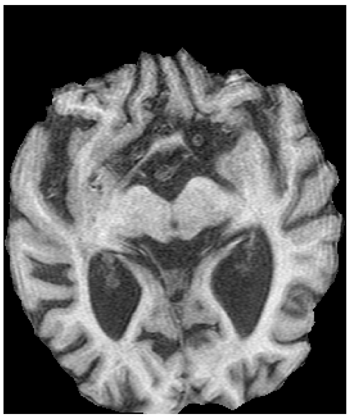
Figure 8.3: Boxplots of lacune voxel volumes for both the TPs and the FNs. The TPs and FNs for each loss function are obtained with the threshold that gave the best performance on the overall segmentation DSC metric.

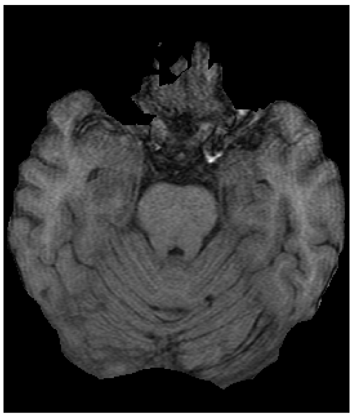
The relation between the lacunes that are detected (true positives) and lacunes that are not detected (false negatives) and their volume is shown in figure 8.3. From these boxplots we see that for the Dice loss, the Dice-ReLU loss and the CDR loss there seem to be slightly more small lacunes among the false negatives than among the true positives. The WBCE loss does not show a difference of the lacune volume between the true positives and false negatives.

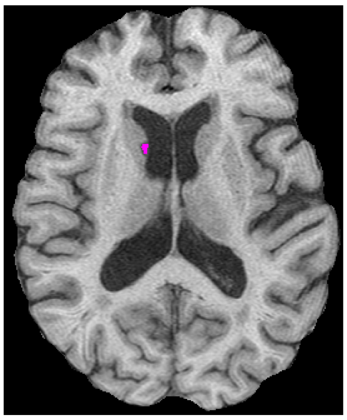
Figure 8.4 shows examples of often occurring false negatives, where figures 8.4a-c and 8.4g-i are the unsegmented images and figures 8.4d-f and 8.4j-l are the manually segmented images. All loss functions failed to detect the first four examples (figures 8.4d-f and 8.4j). Among these four often occurring false negatives are lacunes that are located near the ventricles, lacunes that show little or no difference in intensity with their background, lacunes located in the cerebellum and lacunes located in the outer parts of the brain. The Dice loss generally also fails to detect the lacunes in the brain stem (figure 8.4k) and the lacunes in the upper parts of the brain (figure 8.4l).

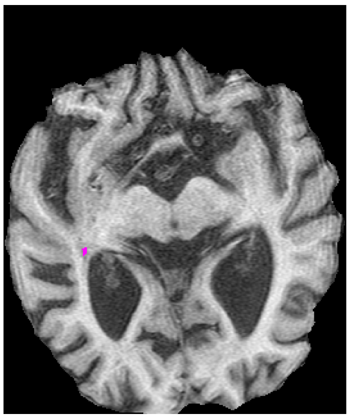
Examples of false positives are shown in figure 8.5, where figures 8.5a-c and 8.5g-i are the unsegmented images and figures 8.5d-f and 8.5j-l are the predicted segmented images. These false positives are all created by the method with the Dice loss. However, methods with the WBCE loss, the Dice-ReLU loss and the CDR loss, create comparable types of false positives. Structures that look similar in shape and intensity to a lacune (figures 8.5d-f and 8.5j-k) and parts of the gyrus (figure 8.5l) are generally often misclassified as a lacune.

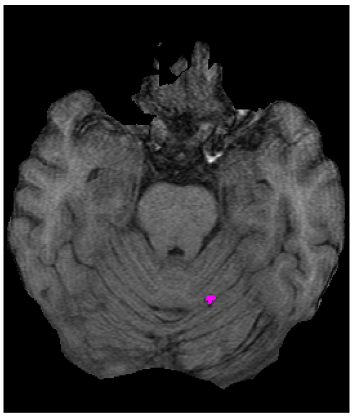

(a) FN near the ventricle.

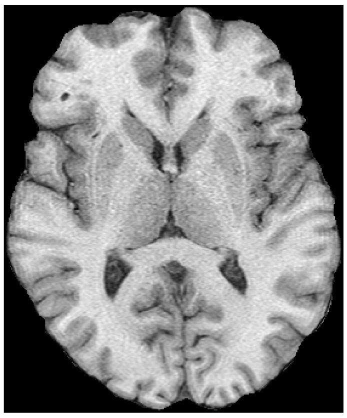

(b) FN with little intensity difference.

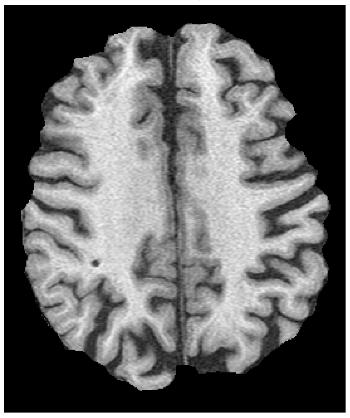

(c) FN in cerebellum.

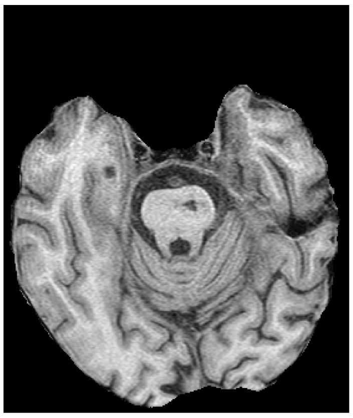

(d) FN near the ventricle.

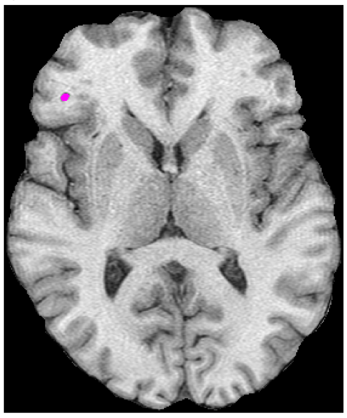

(e) FN with little intensity difference.

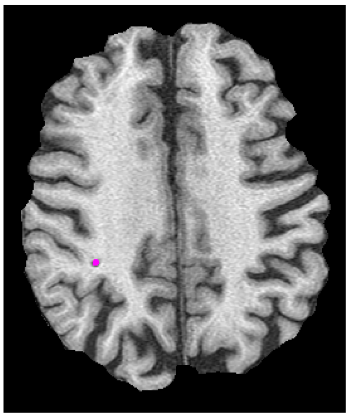

(f) FN in cerebellum.


(g) FN in outer part of the brain.


(h) FN in upper part of the brain


(i) FN in brainstem.


(j) FN in outer part of the brain


(k) FN in upper part of the brain

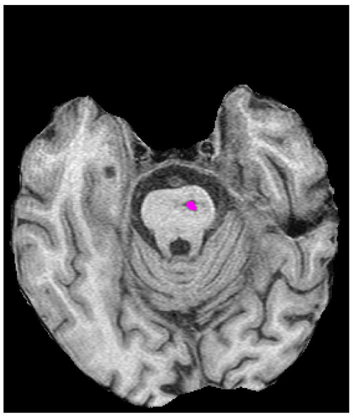
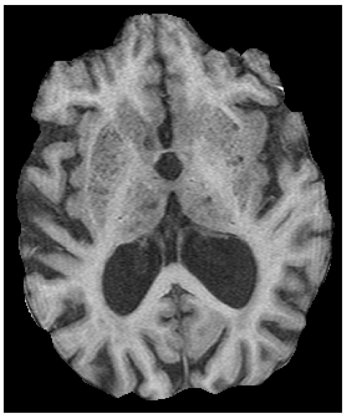
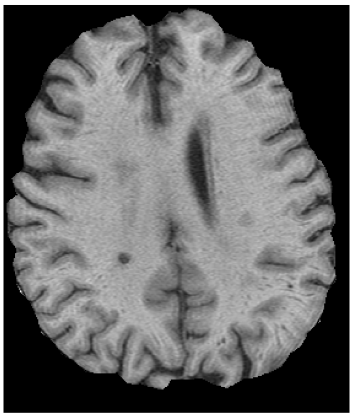
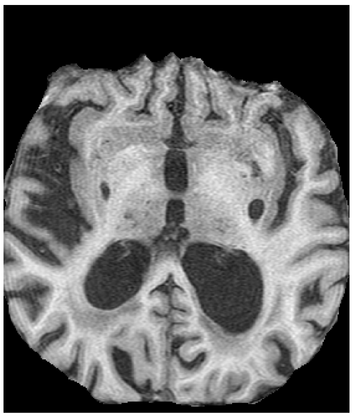
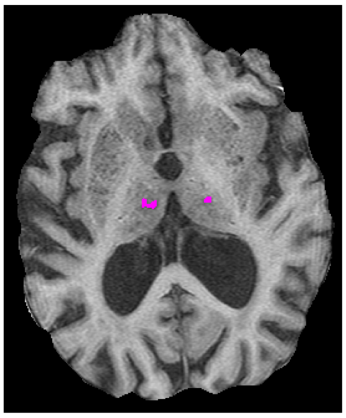

(l) FN in brainstem.

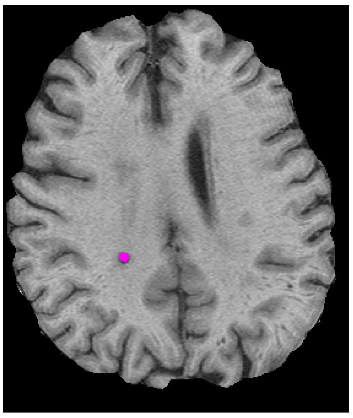
Figure 8.4: Examples of often occurring false negatives (FNs) obtained with the threshold that gave the best performance on the overall segmentation DSC metric. The first and third row display the unsegmented images, the second and fourth row display the manually segmented images. d-f and j are examples that occur for the WBCE loss, the Dice loss, the Dice-ReLU loss and the CDR loss. k and l are examples of FNs that generally only occur with the Dice loss.

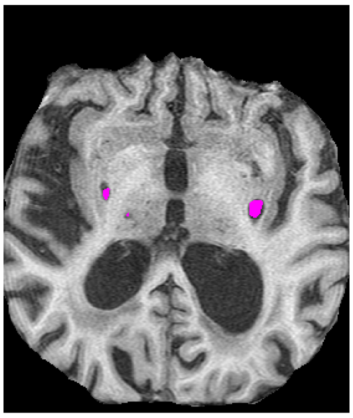

(a) FP with characteristics of a lacune.

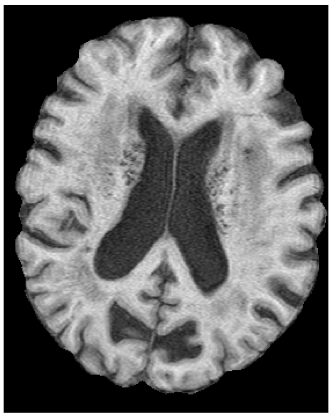

(b) FP with characteristics of a lacune.

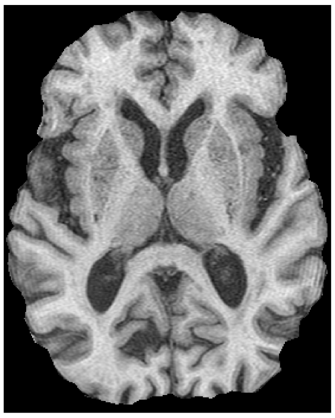

(c) FP with characteristics of a lacune.

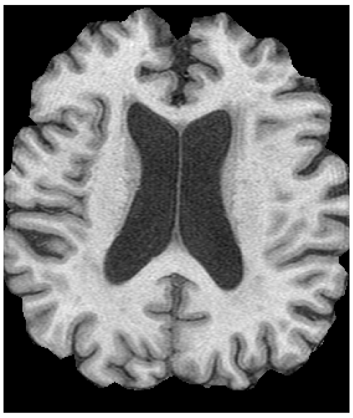

(d) FP with characteristics of a lacune.

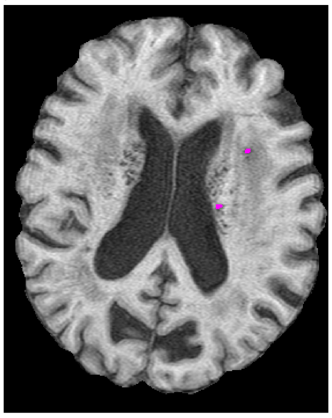

(e) FP with characteristics of a lacune.

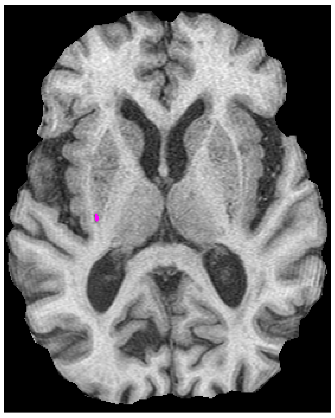

(f) FP with characteristics of a lacune.


(g) FP with characteristics of a lacune.


(h) FP with characteristics of a lacune.


(i) FP as part of gyrus.


(j) FP with characteristics of a lacune.


(k) FP with characteristics of a lacune.

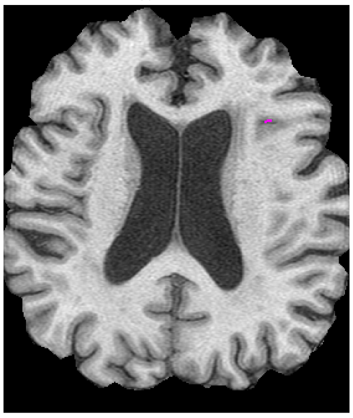
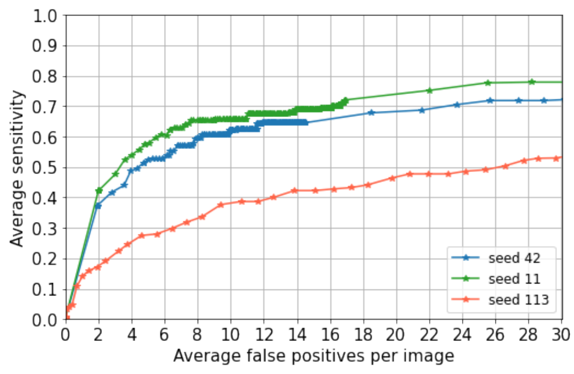

(l) FP as part of gyrus.

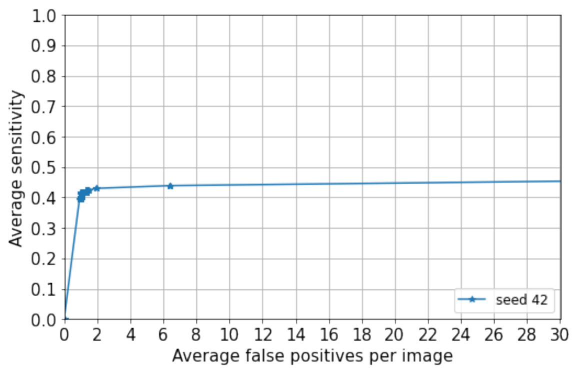
Figure 8.5: Examples of false positives (FPs) obtained with the threshold that gave the best performance on the overall segmentation DSC metric. The first and third row display the unsegmented images, the second and fourth row display the predicted segmented images. These FPs are created by the method with the Dice loss. Predictions of the methods with the WBCE loss, the Dice-ReLU loss and the CDR loss show similarly looking FPs.

In order to measure the consistency of the methods, each method was executed three times, using a different seed for each run. The FROC curves of all loss functions with these varying seeds are shown in figure 8.6 and the TP-element-wise performance, evaluated by the DSC value per seed for all loss functions, is shown in 8.7. In this plot the error bars represent the standard deviation. For the segmentation performance of TP elements, the difference in perfomance on DSC is little between the various seeds. However, the FROCs show that larger differences in both the sensitivity and false positives per image can be obtained when another seed is used. Note that the Dice-ReLU loss and the CDR loss have results for only two seeds and that the Dice loss displays the result of only one seed. The results of the other seeds were excluded because these methods failed to train to detect and segment lacunes.

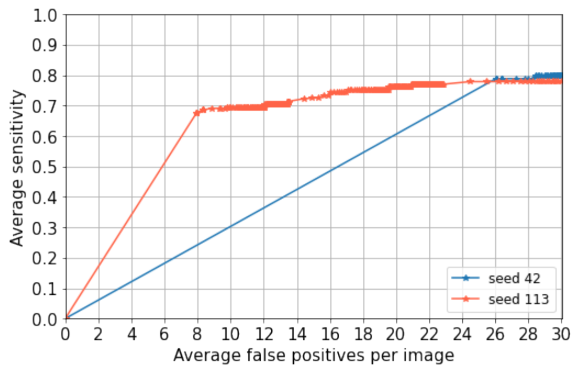
As there was only one seed that worked for the Dice loss, the Dice loss could not be evaluated on consistency based on the first three seeds. Therefore, seven other seeds were tried as well. It turned out that for only two out of ten seeds the Dice loss was able to detect and segment lacunes, while the WBCE loss performed well on all of these seeds. Both the Dice-ReLU loss and the CDR loss worked for six out of ten seeds. The results of the two seeds for which the Dice loss could learn are shown in figure 8.8. The figure shows that also for the Dice loss another seed can lead to larger differences in the sensitivity than in the TP-element-wise DSC.



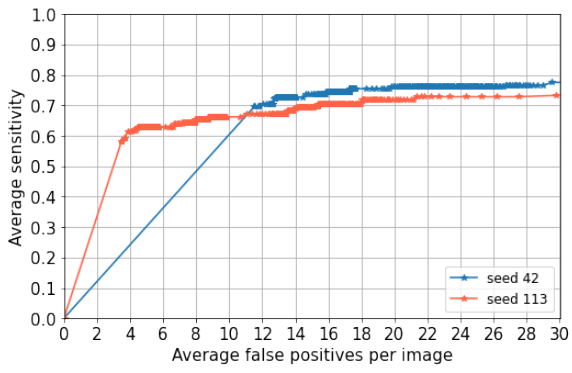
(a) WBCE loss.



(b) Dice loss.



(c) Dice-ReLU loss.



(d) CDR loss.

Figure 8.6: Plots with FROC curves of the average false positives per image against the average sensitivity over a range of thresholds. Every plot shows an FROC curve for each seed that is used to initialize the weights of the method. Three different seeds are used for this. Some FROC curves of seeds for the methods of the Dice loss, the Dice-ReLU loss and the CDR loss are missing, as these methods failed to learn to detect and segment lacunes with these seeds.

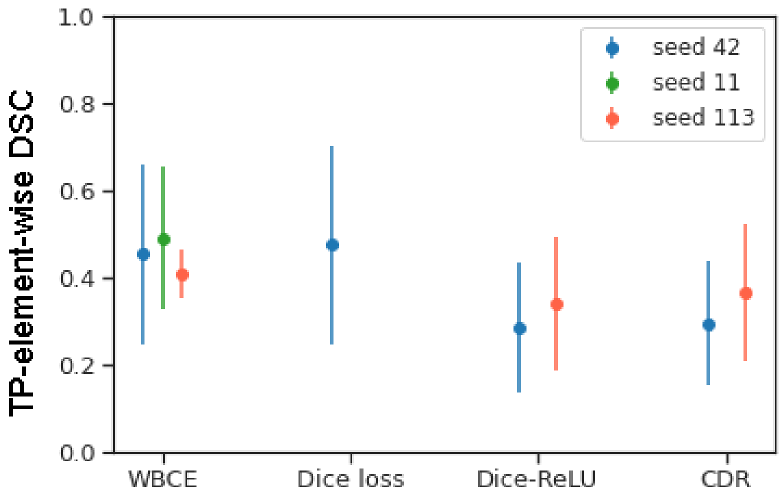
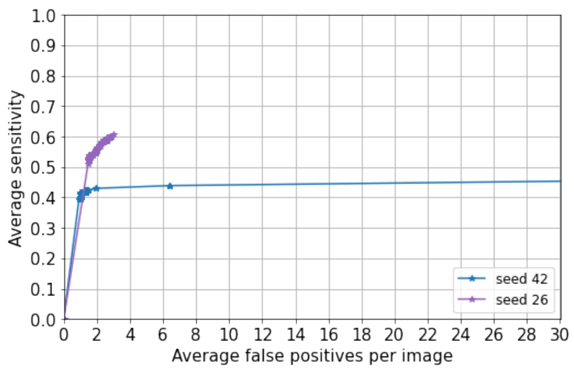


Figure 8.7: Scatterplot of the TP-element-wise DSC values per loss function for each seed that is used to initialize the weights of the method, where the error-bars represent the standard deviations. Three different seeds are used for this. TP-element-wise DSC values of seeds for the methods of the Dice loss, the Dice-ReLU loss and the CDR loss are missing, as these methods failed to learn to detect and segment lacunes with these seeds. DSC values corresponding to the threshold value that gave the best performance on this metric are used for each loss function.



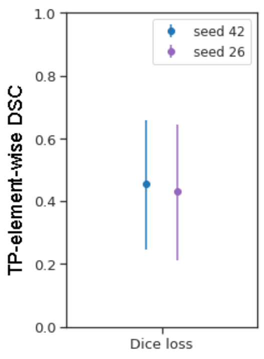


Figure 8.8: FROC plot (left) and scatterplot (right) of experiments executed with the Dice loss. The FROC plot displays curves of the average false positives per image against the average sensitivity over a range of thresholds. The plot shows an FROC curve for each seed that is used to initialize the weights of the method with the Dice loss. The scatterplot shows the TP-element-wise DSC values per loss function for each seed that is used to initialize the weights of the method with the Dice loss, where the error-bars represent the standard deviations. DSC values corresponding to the threshold value that gave the best performance on this metric are used.

9 Discussion

9.1. Discussion of results

The experiments showed that the networks trained with all loss functions except the binary cross-entropy (BCE) loss were able to detect lacunes and thus proved that these loss functions are able to cope with the data imbalance. Only the BCE loss failed to learn detecting lacunes, indicating that this loss function suffers from the large imbalance between the number of lacune voxels and background voxels.

The Dice loss performed best on the overall segmentation having the highest values for all metrics: overall Dice similarity coefficient (DSC), overall relative volume difference and overall absolute volume difference. Additionally, this loss performed best on the TP-element-wise DSC and the TP-element-wise relative volume difference. This indicates that compared to the other losses, it performs well on resembling the segmentation of a lacune. However, on lacune detection it performed less well. The detection sensitivity was quite low for this loss, although it did have a very low number of false positive elements per image. The prediction values of the Dice loss appeared to be very close to either 0 or 1, meaning that the method is very certain of the prediction of both the lacunes and background. However, this limits the freedom in choosing the right balance between sensitivity and false positives per image. By inspecting the prediction images during training, we found that at the beginning of the learning process the method actually achieves a higher sensitivity on the validation set than when the loss has converged. However, later in the optimization process it loses these lacunes again. This might be caused by the data imbalance. The Dice loss depends on the sum over the predicted image, which consists of millions of voxels with values that can range between 0 and 1. As a consequence, if these values are not exactly 0, they can have a large influence on the loss. Therefore, decreasing millions of background voxels with only a tiny step, can greatly improve the value of the loss. Consequently, the loss might benefit more from further reducing the background values than maintaining the lacune detections consisting of only a few voxels.

To shift the focus of the Dice loss from optimizing the background to correctly detecting the lacunes, the ReLU was applied to the Dice loss leading to the Dice-ReLU loss. By adding the ReLU in the Dice loss, the values below 0.1 are clipped, which means that they are set to 0.1. Consequently, the loss does not benefit from optimizing the background voxel values further than 0.1. As a result, the emphasis is shifted from bringing the background values near zero to predicting the lacune voxels correctly. The Dice-ReLU loss proved to successfully increase the sensitivity. However, this result is accompanied by a much larger number of false positives per image. As the Dice-ReLU loss obtains a significant higher sensitivity than the Dice loss it seems like the ReLU helps in preventing the Dice loss from optimizing the background.

The constrained Dice-ReLU (CDR) loss aimed to reduce the resulting false positives from the method trained with the Dice-ReLU loss, while maintaining the TPs. To achieve this, two constraints were applied in the Dice-ReLU loss. The first to the volume of the prediction voxels that according to the manually segmented image should be predicted as background and the second to the volume of the prediction voxels that according to the manually segmented image should be predicted as lacune. The results showed that the CDR loss is able to halve the false positive elements with only a limited decrease in sensitivity and thus, placing a constraint seems to help reducing the false positives.

Among the false negatives of the methods with the Dice loss, the Dice-ReLU loss and the CDR loss there were slightly more smaller lacunes. This might indicate that the Dice loss related methods have more trouble to detect these smaller elements.

The BCE loss failed to detect and segment lacunes. Inspecting the prediction images showed that every value of every image was close to 0, meaning that every voxel was predicted as background. Therefore, it seems like the BCE loss suffers from the data imbalance.

The weighted binary cross-entropy (WBCE) loss was an adaptation to the BCE loss in which the lacune voxels and background voxels are both given a weight to balance their contribution. The WBCE loss was able to detect and segment lacunes. This already proves that giving more weight to the lacune voxels helps to overcome the data imbalance. Compared to the Dice loss, the WBCE loss performed slightly less on the TP-element-wise relative volume difference, while it performed slightly better on the TP-element-wise absolute volume difference. An explanation for this disagreement on volume difference, might be that the WBCE loss performs poorly on segmentation for especially the smaller lacunes, while on the other hand the Dice loss performs most poorly on the larger lacunes, leading to a larger contribution to the absolute volume difference.

With respect to consistency, the results showed that the choice of seed for initialization has a clear effect on the performance of especially the sensitivity and false positives per image for all methods. Furthermore, the Dice related methods seem very unstable as for many seeds the methods failed to learn, while the WBCE loss method is able to learn with every seed.

To conclude, if the purpose of using a method is to get insight in the relation between the characteristics of a lacune, such as its shape and volume, and its physical and psychological consequences, the Dice loss and the WBCE loss are preferred as those methods are better in resembling the shape and volume of the manual segmentations. Although with the WBCE loss slightly less good results are obtained for the segmentation performance than with the Dice loss, the WBCE loss reaches a higher detection sensitivity. This is however at the cost of more false positives, but this can give more freedom in choosing the right balance between sensitivity and false positives per image as the predictions are less bimodal (i.e. values are either close to 0 or 1). If the goal is to detect cerebral small vessel disease (cSVD), the detection of lacunes is more important and for this, the Dice-ReLU loss or the CDR loss are most suitable. However, as these methods still contain many false positives an expert is needed to differentiate between the detected lacunes and the false positives. Since this expert would only have to inspect a couple false positives per image instead of inspecting entire images, this could already save quite some time.

9.2. Limitations

The weighted binary cross-entropy (WBCE) loss, the Dice loss, the Dice-ReLU loss and the constrained Dice-ReLU (CDR) loss all performed limited on the Dice similarity coefficient (DSC) and volume differences evaluation metrics. During the data preparation phase some manual segmentations displayed a shift in their segmentation. The segmentations with large shifts were excluded, but it might be that the dataset still contains some images with slightly shifted manual segmentations. When these shifts are present, it affects the outcome of the evaluation metrics. Furthermore, potential shifts could also have complicated the optimization process, which might have lead to worse performance on sensitivity and false positives per image.

The methods trained with the Dice loss, the Dice-ReLU loss and the CDR loss might have an advantage when evaluating with the DSC value as they are optimized on almost exactly this metric (the only difference being whether a threshold is used on the predicted image or not). Therefore it is easier to perform well on this metric for these losses.

To prevent the Dice loss from overfitting, a threshold was put on the prediction values of the Dice-ReLU loss method. However, by thresholding these predictions, the loss is not differentiable. Although the threshold was applied by using a work-around provided by Tensorflow that should be able to cope with this better, it might still have complicated the optimization process.

Only scans that contained a lacune are used for the development of the methods. As a consequence, it is unsure how the methods would perform when they are trained with images that do not contain a lacune as well. As the images have a large imbalance between lacune voxels and background voxels and as the methods have proved to cope with this imbalance, the methods might obtain similar results when trained on a set of scans that include scans with no lacunes (containing only background voxels). Furthermore, no scans without lacunes were included in the test set. Therefore it is unsure how the trained methods would perform on scans with no lacunes.

Manual segmentation can be quite subjective and observer dependent. The scans are manually segmented by only one rater. As a consequence, it is unclear how accurate these manual segmentations are. Furthermore, the method might have learned the segmentation style of the rater. To assess if the method learned this segmentation style and to assess how reliable the segmentations are, a second rater would be needed. It would also be interesting to compare the variability between raters and the variability with our methods.

9.3. Comparison to related methods

To evaluate the performance of the methods used in this thesis, we compare the methods with other detection and segmentation methods proposed in the literature.

As all detection methods mentioned in chapter 4 *Related work* report average detection sensitivity and average number of false positives per image, we compare these methods with our methods based on these metrics. All previously developed detection methods [2, 17, 43–46, 57] obtain an average sensitivity value of more than 90% with an average of less than 1.5 false positives per image. The methods of the current work show an average sensitivity of 0.49 with an average of 3.96 false positives per image for the WBCE loss, an average sensitivity of 0.43 with an average of 1.93 false positives per image for the Dice loss, an average sensitivity of 0.79 with an average of 26.03 false positives per image for the Dice-ReLU loss and an average sensitivity of 0.70 with an average of 11.52 false positives per image for the CDR loss. The methods of the current work appear to perform worse than previously proposed detection methods on both detection sensitivity and number of false positives. However, it should be noted that apart from one method, all detection methods in previous work are based on 2D images, while the current work is based on 3D images. As the class imbalance is less present in 2D images, it might be harder to detect lacunes in 3D scans than on 2D scans. Furthermore, these detection methods report false positive scores per slice, while the current methods are evaluated based on entire images. As a consequence, since a 3D image is bigger, it has more chance to produce false positives. The only detection method that used 3D images, used preselected patches for this. That is, a radiologist inspected scans for lacunes and similar looking structures and patches of these suspected lesions were given to the method. So for this method human intervention is required which makes it considerably more inconvenient and time-consuming to apply. Additionally, the radiologist already filtered out many potential false positives, which makes it easier for the method to produce less false positives.

For the segmentation methods it is more difficult to make a comparison as there are only a few methods developed for specifically lacune segmentation. Wang et al. [49] segmented several lesions at once and obtained an average sensitivity of 80.6% with an average of 0.06 false positives. These results are better than the results of this work. However, as for the previously mentioned detection methods, this method also uses 2D images. They did not report on the accuracy of the segmentations. Sudre et al. [41] obtained with their 3D method a sensitivity of 72.2% with a median overlap of 59% on objects that all raters agreed on and a median overlap of 30% on objects of which raters were less certain. These results are computed on a test set of only two scans. The results of the uncertain objects are comparable to the results of the Dice-ReLU loss method which obtained a TP-element-wise DSC value of 0.28 with a sensitivity of 79% and the CDR loss method which obtained a TP-element-wise DSC value of 0.29 with a sensitivity of 70%. However, these results are not specifically for lacunes as the researchers also included perivascular spaces into their method. This makes it difficult to compare. Duan et al. [12] also uses a U-net architecture to segment lacunes and other cSVD related lesions. Their method obtains a region-wise detection F1 score of 0.783, which is an evaluation metric that is comparable to the sensitivity. It achieves an overall DSC value of 0.50. The methods of all loss functions in the current work all perform worse compared to this method. However, once more this method is developed and evaluated based on 2D images.

9.4. Recommendations

Although the CDR loss, in which constraints are placed independently on both the lacune volume and the background volume, was able to halve the number of false positives, it might be possible to achieve even better results. The best value for the validation loss of the CDR loss was obtained after only 25 epochs, after which the method started to largely overfit on the training data. The fast overfitting might have occurred due to using constraints that were too strict. When the constraints are too strict, the method might have too limited freedom to learn new features that can satisfy the constraints and might be held back by the features that it learned when optimized with the Dice-ReLU loss. As a consequence, this might lead to overfitting on the training data. Putting constraints on the volumes that are less strict, might give the method more freedom and prevent it from overfitting. Another approach would be, instead of restarting the Dice-ReLU loss method, to run the experiment with the CDR loss from the beginning. Additionally, further fine-tuning the contribution of the two constraints to the Dice-ReLU loss might also lead to better performance.

As the WBCE loss produced relatively many false positives, this loss function might also benefit from constraining the background voxels. It would be interesting to see if the constraints can help to improve the WBCE loss as well. Additionally, since the Dice loss showed difficulty in maintaining the lacunes, a constraint on the lacune voxels might improve the sensitivity of the Dice loss. Such that, if the Dice loss tends to lose lacune voxels, it will be punished for it. Furthermore, one could consider approaches that prevent the predictions of the Dice loss from becoming so close to binary.

The current methods only use T1-weighted images. As images of other modalities might show other features of a lacune, the method might be improved in differentiating between lacunes and other similar looking structures by adding other modalities as well. Especially FLAIR images could be interesting to add, as on this modality some lacunes display a characteristic hyperintense rim. Lacunes that are often among the false negatives (figure 8.4) show little differences in intensity with their background on the T1-weighted images. These might display more variation in intensity on the FLAIR images. Furthermore, FLAIR images might also help in detecting lacunes that are located near the in intensity similar looking ventricles. If the lacunes contain a hyperintense rim, they could be more easily differentiated from the hypointense ventricles. The same might apply to lacunes occurring in the cerebellum.

Lastly, adding more scans with lacunes could result in more information about the variance in appearance of lacunes, which might help in differentiating between lacunes and other similar looking structures. False positives as well as false negatives might be reduced by this. The false negative lacunes were often located in uncommon parts of the brain or had an uncommon shape. With more training examples the methods might be better at detecting these as well.

10 Conclusion

The aim of this research was to develop a deep learning method that is able to detect and segment lacunes in 3D brain MRI scans. To achieve this, we needed to address the data imbalance problem. Furthermore, an additional approach was needed to differentiate the lacunes from similarly looking structures.

Several loss functions are compared on how well they deal with the data imbalance: the binary cross-entropy (BCE) loss, the weighted binary cross-entropy (WBCE) loss and the Dice loss. This thesis proposed to apply background clipping in the Dice loss to handle the data imbalance further, leading to the Dice-ReLU loss. To help the method reduce false positives and as a result differentiate better between lacunes and similarly looking structures, this thesis proposed to apply a constraint to the Dice-ReLU loss, leading to the constrained Dice-ReLU (CDR) loss.

In this thesis, it is shown that, apart from the BCE loss, all other loss functions are able to detect and segment lacunes and thus are able to cope with the data imbalance. Clipping the background in the Dice loss (Dice-ReLU loss) led to a higher detection sensitivity, but at the cost of many more false positives. Adding a constraint on the volume size successfully reduced the false positives with more than 50%. This indicates that adding a constraint can improve the optimization of the network and can lead to a method that can differentiate better between lacunes and similarly looking structures.

This research showed that the developed deep learning methods are able to detect and segment lacunes in 3D brain MRI scans. The WBCE loss and the Dice loss proved to be especially suitable for gaining more information into the relationship between the characteristics of a lacune and the manifestations of cerebral small vessel disease. The Dice-ReLU loss and the CDR loss are especially useful for detecting the disease.

Acronyms

BBB blood-brain barrier. 3

BCE binary cross-entropy. 30

CAA cerebral amyloid angiopathy. 3

CDR constrained Dice-ReLU. 33

CNN convolutional neural network. 8

cSVD cerebral small vessel disease. 1

DSC Dice similarity coefficient. 32

FLAIR fluid-attenuated inversion recovery. 4

FN false negative. 35

FP false positive. 35

FROC free-response receiver operating characteristic. 36

MRI magnetic resonance imaging. 3, 4

PCA principal component analysis. 21

ReLU rectified linear unit. 12

RS Rotterdam study. 25

RSS Rotterdam scan study. 25

SGD stochastic gradient descent. 14

SVM support-vector machine. 20

TP true positive. 35

WBCE weighted binary cross-entropy. 31

Glossary

Amyloid
: Aggregate of proteins. 3

Arteriole
: Small blood vessel that branches out from an artery into the capillaries. 3

Background
: All pixels, or voxels, that do not belong to the element under investigation and make up the rest of the image. 1

Backpropagation
: An algorithm that is used for training a network. 14

Biomarker
: Measurable indicator of the presence or severity of a disease state. 3

Brain atrophy
: Lesion that results from cerebral small vessel disease. 4

Capillary
: The smallest blood vessel in the body. They supply blood to the surrounding tissues. 3

Cardiovascular system
: System that consists of the heart and blood vessels. 3

Cerebellum
: Area at the back and bottom of the brain, which is associated with movement and coordination. 25

Cerebral small vessel disease
: A disease that can result in different types of brain lesions. 1

Cerebral sulcus
: Grooves on the surface of the brain. 20

Classification
: The allocation of a visual input into classes. 16

Convolutional neural network
: A deep learning algorithm that is mostly used for analyzing images and image classification problems. 8

Detection
: The localization and classification of objects in a visual input. 16

Epoch
: When every example of the dataset has passed forward and backword through the neural network once. 14

False negative

A negative predicted outcome that should have been positive. 35

False positive

A positive predicted outcome that should have been negative. 19

Feature map

Captures the result of applying a kernel to an input map. 8

Fluid-attenuated inversion recovery sequence

A particular setting of radiofrequency pulses and field gradients that influences the appearance of an image. 4

Foreground

All pixels, or voxels, that belong to the element under investigation. 16

Haemorrhage

Blood escaping from the cardiovascular system caused by damaged blood vessels. 4

Hyperintense

Brighter appearance of an abnormality or structure than the structures it is compared to. 4

Hypointense

Darker appearance of an abnormality or structure than the structures it is compared to. 4

Infarct

Resulting lesion from tissue death due to inadequate blood supply. 4

Ischaemic stroke

Blood supply to a part of the brain is decreased, leading to dysfunction of the brain tissue. 3

Kernel

Filter that is able to extract features. 8

Lacune

A round or ovoid, subcortical, fluid-filled cavity of between 3 mm and about 15 mm in diameter. Short for lacune of presumed vascular origin. 1

Learning rate

A stepsize for an optimizer. 15

Microbleed

Lesion that results from cerebral small vessel disease. 1

MRI sequence

A particular setting of radiofrequency pulses and field gradients that influences the appearance of an image. 4

Neuroimaging

The process of producing images of the structure or activity of the nervous system. 3

Overfitting

When too specific features have been learned, such that a network fails to generalize learned features to new unseen examples.. 42

Pathogen
Any disease-producing agent such as bacteria, a virus or other microorganisms. 3

Perivascular space
Lesion that results from cerebral small vessel disease. 1

Segmentation
The division of a visual input into segments. 1

Semipermeable
Some molecules or ions can pass through, but others can not. 3

Stride
Stepsize of a sliding window. 12

Subcortical
Part of the brain that is located below the cerebral cortex, the outer part of the brain. 4

T1-weighted sequence
A particular setting of radiofrequency pulses and field gradients that influences the appearance of an image. 19

T2-weighted sequence
A particular setting of radiofrequency pulses and field gradients that influences the appearance of an image. 19

Test set
Set of examples that are used to evaluate the performance of the final network. 26

Training set
Set of examples that are used for training the network. 26

True positive
A positive predicted outcome that is indeed positive. 35

Validation set
Set of examples that are used to check whether the network is able to generalize to new examples. 26

Ventricle
Fluid-filled space in the brain. 19

Venule
Small blood vessel that transports the regained blood from the capillaries to the veins. 3

Voxel
The equivalent of a pixel. 1

White matter hyperintensity
Lesion that results from cerebral small vessel disease. 1

Bibliography

[1] N.J. Abott, A.A.K. Patabendige, D.E.M. Dolman, S.R. Yusof, and D.J. Begley. Structure and function of the blood-brain barrier. *Neurobiology of Disease*, 37(1):13–25, 2010. doi: 10.1016/j.nbd.2009.07.030.

[2] M.A. Al-Masni, W.R. Kim, E.Y. Kim, Y. Noh, and D.H. Kim. 3d multi-scale residual network toward lacunar infarcts identification from mr images with minimal user intervention. *IEEE Access*, 9: 11787–11797, 2021. doi: 10.1109/ACCESS.2021.3051274.

[3] J. Bamford, P. Sandercock, L. Jones, and C. Warlow. The natural history of lacunar infarction: The oxfordshire community stroke project. *Stroke*, 18(3):545–551, 1987. doi: 10.1161/01.str.18.3.545.

[4] J.M. Biesbroek, H.J. Kuijf, Y. van der Graaf, K.L. Vincken, A. Postma, W.P.T.M. Mali, G.J. Biessels, and M.I. Geerlings. Association between subcortical vascular lesion location and cognition: A voxel-based and tract-based lesion-symptom mapping study. the smart-mr study. *PLOS ONE*, 8 (4), 2013. doi: 10.1371/journal.pone.0060541.

[5] A. Biffi and S.M. Greenberg. Cerebral amyloid angiopathy: A systematic review. *Journal of Clinical Neurology*, 7(1):1–9, 2011. doi: 10.3988/jcn.2011.7.1.1.

[6] G. Bortsova, F. Dubost, S. Ørting, I. Katramados, L. Hogeweg, L. Thomsen, M. Wille, and M. de Bruijne. Deep learning from label proportions for emphysema quantification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 768–776, 2018. doi: 10.1007/978-3-030-00934-2_85.

[7] A. Charidimou, Q. Gang, and D.J. Werring. Sporadic cerebral amyloid angiopathy revisited: recent insights into pathophysiology and clinical spectrum. *Journal of Neurosurg Psychiatry*, 83(2):124–137, 2012. doi: 10.1136/jnnp-2011-301308.

[8] F. Chollet. *Deep Learning with Python.* Manning Publications Co., Shelter Island, New York, 2018.

[9] E. Cuadrado-Godia, P. Dwivedi, S. Sharma, A. Ois Santiago, J. Roquer Gonzales, M. Balcells, J. Laird, M. Turk, H.S. Suri, A. Nicolaides, L. Saba, N.K. Khanna, and J.S. Suri. Cerebral small vessel disease: A review focusing on pathophysiology, biomarkers, and machine learning strategies. *Journal of Stroke*, 20(3):302–320, 2018. doi: 10.5853/jos.2017.02922.

[10] K.F. de Laat, A.M. Tuladhar, A.G. van Norden, D.G. Norris, M.P. Zwiers, and F.E. de Leeuw. Loss of white matter integrity is associated with gait disorders in cerebral small vessel disease. *Brain*, 134(1):73–83, 2011. doi: 10.1093/brain/awq343.

[11] J. Dolz, I.B. Ayed, and C. Desrosiers. Dense multi-path u-net for ischemic stroke lesion segmentation in multiple image modalities. In *Proceedings of the 4th International MICCAI Brainlesion Workshop*, pages 271–282, Granada, Spain, 2018. doi: 10.1007/978-3-030-11723-8_27.

[12] Y. Duan, W. Shan, L. Liu, Q. Wang, Z. Wu, P. Liu, J. Ji, Y. Liu, K. He, and Y. Wang. Primary categorizing and masking cerebral small vessel disease based on "deep learning system". *Frontiers in Neuroinformatics*, 14(17), 2020. doi: 10.3389/fninf.2020.00017.

[13] F. Dubost, G. Bortsova, H. Adams, M.A. Ikram, W.J. Niessen, M.W. Vernooij, and M. de Bruijne. Gp-unet: Lesion detection from weak labels with a 3d regression network. In *Proceedings of the 20th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 214–212, Québec City, Canada, 2017. doi: 10.1007/978-3-319-66179-7_25.

[14] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning, 2016. URL arXiv:1603.07285.

[15] M.T. Duong, J.D. Rudie, J. Wang, L. Xie, S. Mohan, J.C. Gee, and A.M. Rauschecker. Convolutional neural network for automated flair lesion segmentation on clinical brain mr imaging. *American Journal of Neuroradiology*, 40(8):1282–1290, 2019. doi: 10.3174/ajnr.A6138.

[16] C.M. Fisher. Lacunes: Small, deep cerebral infarcts. *Neurology*, 15(8):774–784, 1965. doi: 10.1212/wnl.15.8.774.

[17] M. Ghafoorian, N. Karssemeijer, T. Heskes, M. Bergkamp, J. Wissink, J. Obels, K. Keizer, F.e. de Leeuw, B. van Ginneken, E. Marchiori, and B. Platel. Deep multi-scale location-aware 3d convolutional neural networks for automated detection of lacunes of presumed vascular origin. *NeuroImage: Clinical*, 14:391–399, 2017. doi: 10.1016/j.nicl.2017.01.033.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, Massachusetts, 2016. http://www.deeplearningbook.org.

[19] L.L. Herrmann, M. Le Masurier, and K.P. Ebmeier. White matter hyperintensities in late life depression: a systematic review. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(6):619–624, 2008. doi: 10.1136/jnnp.2007.124651.

[20] F.C. Higham and Higham D.J. Deep learning: An introduction for applied mathematicians. *SIAM Review*, 61(4):860–891, 2019. doi: 10.1137/18M1165748.

[21] M.A. Ikram, A. van der Lugt, W.J. Niessen, P.J. Koudstaal, G.P. Krestin, A. Hofman, D. Bos, and M.W. Vernooij. The rotterdam scan study: design update 2016 and main findings. *European Journal of Epidemiology*, 30(12):1299–1315, 2015. doi: 10.1007/s10654-015-0105-7.

[22] M.A. Ikram, G. Bruselle, M. Ghanbari, A. Goedegebure, M.K. Ikram, M. Kavousi, B.C.T. Kieboom, C.C.W. Klaver, R.J. de Knegt, A.L. Luik, T.E.C. Nijsten, R.P. Peeters, F.J.A. van Rooij, B.H. Stricker, A.G. Uitterlinden, M.W. Vernooij, and T. Voortman. Objectives, design and main findings until 2020 from the rotterdam study. *European Journal of Epidemiology*, 2020. doi: 1oi.org/10.1007/s10654-020-00640-5.

[23] K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision*, pages 2146–2153, 2009. doi: 10.1109/ICCV.2009.5459469.

[24] J.M. Johnson and T.M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(27), 2019. doi: doi.org/10.1186/s40537-019-0192-5.

[25] R. Karthik, U. Gupta, A. Jha, R. Rajalakshmi, and R. Menaka. A deep supervised approach for ischemic lesion segmentation from multimodal MRI using fully convolutional network. *Applied Soft Computing Journal*, 84, 2019. doi: 0.1016/j.asoc.2019.105685.

[26] H. Kervadec, J. Dolz, M. Tang, E. Granger, Y. Boykov, and I.B. Ayed. Constrained-cnn losses for weakly supervised segmentation. *Medical image analysis*, 54:88–99, 2019. doi: 10.1016/j.media.2019.02.009.

[27] D.P. Kingma and L.J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, United States of America, 2015.

[28] P. Klarenbeek, R.J. van Oostenbrugge, R.P.W. Rouhl, I.L.H. Knottnerus, and J. Staals. Ambulatory blood pressure in patients with lacunar stroke: Association with total mri burden of cerebral small vessel disease. *Stroke*, 44(11):2995–2999, 2013. doi: 10.1161/STROKEAHA.113.002545.

[29] R.P. Kloppenborg, P.J. Nederkoorn, A.M. Grool, L.J.L. De Cocker, W.P.T.M. Mali, Y. van der Graaf, and M.I. Geerlings. Do lacunar infarcts have different aetiologies? risk factor profiles of lacunar infarcts in deep white matter and basal ganglia: The second manifestations of arterial disease-magnetic resonance study. *Cerebrovascular Diseases*, 43(3-4):161–168, 2017. doi: 10.1159/000454782.

[30] Y. Ling and H. Chabriat. Progression of mri markers in cerebral small vessel disease: Sample size considerations for clinical trials. *Journal of Cerebral Blood Flow & Metabolism*, 36(1):228–240, 2016. doi: 10.1038/jcbfm.2015.113.

[31] Y. Ling and H. Chabriat. Incident cerebral lacunes: A review. *Journal of Cerebral Blood Flow & Metabolism*, 40(5):909–921, 2020. doi: 10.1177/0271678X20908361.

[32] G. Litjens, T. Kooi, B.E. Bejnordi, A.A.A. Setio, F. Ciompi, M. Ghafoorian, J.A.W.M. van der Laak, B. van Ginneken, and C.I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. doi: 10.1016/j.media.2017.07.005.

[33] F. Milletari, N. Navab, and S.A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Proceedings of the 4th International Conference on 3D Vision*, pages 565–571, Stanford, United States of America, 2016. doi: 10.1109/3DV.2016.79.

[34] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, Haifa, Israel, 2010.

[35] L. Pantoni. Cerebral small vessel disease: from pathogenesis and clinical characteristics to therapeutic challenges. *The Lancet Neurology*, 9(7):689–701, 2010. doi: 10.1016/S1474-4422(10)70104-6.

[36] M. Pasi, G. Boulouis, P. Fotiadis, E. Auriel, A. Charidimou, K. Haley, A. Ayres, K.M. Schwab, J.N. Goldstein, J. Rosand, A. Viswanathan, L. Pantoni, S.M. Greenberg, and M.E. Gurol. Distribution of lacunes in cerebral amyloid angiopathy and hypertensive small vessel disease. *Neurology*, 88(23):2162–2168, 2017. doi: 10.1212/WNL.0000000000004007.

[37] A. Pogessi, G. Pracucci, H. Chabriat, T. Erkinjuntti, F. Fazekas, A. Verdelho, M. Hennerici, P. Langhorne, J. O'Brien, P. Scheltens, M.C. Visser, M. Crisby, G. Waldemar, A. Wallin, D. Inzitari, and L. Pantoni. Urinary complaints in nondisabled elderly people with age-related white matter changes: the leukoaraiosis and disability (ladis) study. *Journal of the American Geriatrics Society*, 56(9):1638–1643, 2008. doi: 10.1111/j.1532-5415.2008.01832.x.

[38] O. Ronneberger, P. Fisher, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th Medical Image Computing and Computer Assisted Intervention*, pages 234–241, Munich, Germany, 2015. doi: 10.1007/978-3-319-24574-4_28.

[39] E.E. Smith and S.M. Greenberg. Beta-amyloid, blood vessels and brain function. *Stroke*, 40(7):2601–2606, 2009. doi: 10.1161/STROKEAHA.108.536839.

[40] E.E. Smith and J.M. Lee. Lacunes: Black holes in our understanding of cerebral amyloid angiopathy. *Neurology*, 88(23):2158–2159, 2017. doi: 10.1212/WNL.0000000000004017.

[41] C.H. Sudre, B. Gomez Anson, C.D. Lane, D. Jimenez, L. Haider, T. Varsavsky, L. Smith, R.H. Jäger, and U.M. Jorge Cardoso. 3d multirater rcnn for multimodal multiclass detection and characterisation of extremely small objects. *Proceedings of Machine Learning Research*, pages 447–456, 2019.

[42] R. Topakian, T.R. Barrick, F.A. Howe, and H.S. Markus. Blood-brain barrier permeability is increased in normal-appearing white matter in patients with lacunar stroke and leucoaraiosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 81(2):192–197, 2010. doi: 10.1136/jnnp.2009.172072.

[43] Y. Uchiyama, R. Yokoyama, H. Ando, T. Asano, H. Kato, H. Yamakawa, H. Yamakawa, T. Hara, T. Iwama, H. Hoshi, and Fujita. Improvement of automated detection method of lacunar infarcts in brain MR images. In *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1599–1602, Lyon, France, 2007. doi: 10.1109/IEMBS.2007.4352611.

[44] Y. Uchiyama, R. Yokoyama, H. Ando, T. Asano, H. Kato, H. Yamakawa, H. Yamakawa, T. Hara, T. Iwama, H. Hoshi, and H. Fujita. Computer-aided diagnosis scheme for detection of lacunar infarcts on MR images. *Academic Radiology*, 14(12):1554–1561, 2007. doi: 10.1016/j.acra.2007.09.012.

[45] Y. Uchiyama, T. Kunieda, T. Asano, H. Kato, T. Hara, K. Masayuki, T. Iwama, H. Hoshi, Y. Kinosada, and H. Fujita. Computer-aided diagnosis scheme for classification of lacunar infarcts and enlarged virchow-robin spaces in brain MR images. In *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3908–3911, Vancouver, Canada, 2008. doi: 10.1109/IEMBS.2008.4650064.

[46] Y. Uchiyama, A. Abe, C. Muramatsu, T. Hara, J. Shiraishi, and H. Fujita. Eigenspace template matching for detection of lacunar infarcts on MR images. *Journal of Digital Imaging*, 28(1):116–122, 2015. doi: 10.1007/s10278-014-9711-2.

[47] W. M. van der Flier, E.C.W. van Straaten, F. Barkhof, A. Verdelho, S. Madureira, L. Pantoni, D. Inzitari, T. Erkinjuntti, M. Crisby, G. Waldemar, R. Schmidt, F. Fazekas, and P. Scheltens. Small vessel disease and general cognitive function in nondisabled elderly. *Stroke*, 36(10):2116–2120, 2005. doi: 10.1161/01.STR.0000179092.59909.42.

[48] K.M.H. van Wijnen, F. Dubost, P. Yilmaz, M.A. Ikram, W.J. Niessen, H. Adams, M.W. Vernooij, and M. de Bruijne. Automated lesion detection by regressing intensity-based distance with a neural network. In *Proceedings of the 22nd International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–242, Shenzhen, China, 2019. doi: 10.1007/978-3-030-32251-9_26.

[49] U. Wang, J.A. Catindig, S. Hilal, H.W. Soon, E. Ting, T.Y. Wong, N. Venketasubramanian, C. Chen, and A. Qiu. Multi-stage segmentation of white matter hyperintensity, cortical and lacunar infarcts. *NeuroImage*, 60(4):2379–2388, 2012. doi: 10.1016/j.neuroimage.2012.02.034.

[50] J.M. Wardlaw, P.A.G. Sandercock, M.S. Dennis, and J. Starr. Is breakdown of the blood-brain barrier responsible for lacunar stroke, leukoaraiosis, and dementia? *Stroke*, 34(3):806–812, 2003. doi: 10.1161/01.STR.0000058480.77236.B3.

[51] J.M. Wardlaw, F. Doubal, P. Armitage, F. Chappell, T. Carpenter, S. Muñoz-Maniega, A. Farall, C. Sudlow, M.S. Dennis, and B. Dhillon. Lacunar stroke is associated with diffuse blood-brain barrier dysfunction. *Annals of Neurology*, 65(2):194–202, 2009. doi: 10.1002/ana.21549.

[52] J.M. Wardlaw, C. Smith, and M. Dichgans. Mechanisms underlying sporadic cerebral small vessel disease: insights from neuroimaging. *Lancet Neurology*, 12(5):483–497, 2013. doi: 10.1016/S1474-4422(13)70060-7.

[53] J.M. Wardlaw, E.E. Smith, G.J. Biessels, C. Cordonnier, F. Fazekas, R. Frayne, R.I. Lindley, J.T. O'Brien, F. Barkhof, O.R. Benavente, S.E. Black, C. Brayne, M. Breteler, H. Chabriat, C. DeCarli, F.E. de Leeuw, F. Doubal, M. Duering, N.C. Fox, S. Greenberg, V. Hachinski, I. Kilimann, V. Mok, R. van Oostenbrugge, L. Pantoni, O. Speck, B.C.M. Stephan, S. Teipel, A. Viswanathan, D. Werring, C. Chen, C. Smith, M. van Buchem, B. Norrving, P.B. Gorelick, and M. Dichgans. Neuroimaging standards for research into small vessel disease and its contribution to ageing and neurodegeneration. *The Lancet Neurology*, 12(8):822–838, 2013. doi: 10.1016/S1474-4422(13)70124-8.

[54] J.M. Wardlaw, C. Smith, and M. Dichgans. Stroke subtype, vascular risk factors, and total mri brain small-vessel disease burden. *Neurology*, 83(14):1228–1234, 2014. doi: 110.1212/WNL.0000000000000837.

[55] J.M. Wardlaw, S.J. Makin, M.C. Valdés Hernández, P.A. Armitage, A.K. Heye, F.M. Chappell, S. Muñoz-Maniega, E. Sakka, K. Shuler, M.S. Dennis, and M.J. Thrippleton. Blood-brain barrier failure as a core mechanism in cerebral small vessel disease and dementia: evidence form a cohort study. *Alzheimer's & Dementia*, 13(6):634–643, 2017. doi: 10.1016/j.jalz.2016.09.006.

[56] B. Xu, Y. Chai, C. Galarza, C. Vu, B. Tamrazi, B. Gaonkar, L. Macyszyn, T. Coates, N. Lepore, and J. Wood. Orchestral fully convolutional networks for small lesion segmentation in brain MRI. In *Proceedings of the 15th IEEE International Symposium on Biomedical Imaging*, pages 889–892, Washington, United States of America, 2018. doi: 10.1109/ISBI.2018.8363714.

[57] R. Yokoyama, X. Zhang, Y. Uchiyama, H. Fujita, T. Hara, X. Zhou, M. Kanematsu, T. Asano, H. Kondo, S. Goshima, H. Hoshi, and T. Iwama. Development of an automated method for the detection of chronic lacunar infarct region in brain MR images. *IEICE Transactions on Information and Systems*, E90-D(6):943–954, 2007. doi: 10.1093/ietisy/e90-d.6.943.

[58] Zeiler. Adadelta: an adaptive learning rate method, 2012. URL arXiv:1212.5701.

[59] C.E. Zhang, S.M. Wong, H.J. van de Haar, J. Staals, J.F. Jansen, C.R. Jeukens, P.A. Hofman, R.J. van Oostenbrugge, and W.H. Backes. Blood-brain barrier leakage is more widespread in patients with cerebral small vessel disease. *Neurology*, 88(5):426–432, 2017. doi: 10.1212/WNL.0000000000003556.