DEEP LEARNING-BASED SEGMENTATION OF WHITE MATTER HYPERINTENSITIES IN MAGNETIC RESONANCE IMAGES: AN EARLY MARKER FOR DEVELOPMENT OF ALZHEIMER'S DISEASE

Martin Soria Røvang

Electronics, informatics and technology
Signal processing and imaging
60 ETC
The University of Oslo
Department of Physics / Department of Informatics

May 17, 2021

Summary

Background Alzheimer's disease is the most common cause of dementia, and the disease can be characterized by aggregation of the protein Amyloid-β [1]. Amyloid-β is poisonous for the nerve cells and forms pathologic lesions that can be visualized by medical imaging techniques. Currently, Positron Emission Tomography (PET) is the imaging-based gold standard for detecting amyloid-beta deposits. However, PET is an expensive and time-consuming method and requires the administration of a radioactive isotope.

Magnetic Resonance Imaging (MRI) is a non-invasive and more available alternative to PET. In MRI, amyloid depositions are associated with the formation of white matter hyperintensities (WMHs), and in Alzheimer's patients, a higher baseline of WMH is associated with a greater increase in Amyloid-β[2]. Accurate detection and quantification of WMH from MRI are therefore important in the diagnostic workup of Alzheimer's disease. Deep learning-based methods have proved powerful in a wide range of image segmentation tasks, and in this thesis, the aim is to test different deep learning approaches to automatically segment WMHs from MRI. A fully automated WMH segmentation tool is expected to aid the radiologist in the diagnostic workup in patients with early signs of Alzheimer's disease, speed up the diagnostic process, and may also reduce user bias.

Methods and data Different types of UNet[3] architectures were implemented and tested for the segmentation of WMH lesions in MRI data. These architectures were a reduced version of UNet, UNet with attention with & without pyramid input, and an input method where we added three vicinity 2D slices stacked together as channels to get a bit of 3D information. In the experiments we used a Tversky focal loss with two different setting, where the best parameters used were, $\alpha = 0.85$, $\beta = 0.15$ and $\gamma = 4/3$. This allows the gradient to be more focused on both the false negatives and hard examples.

Data from a large ongoing pre-dementia multi-center imaging study was used for model training and testing and included the two different MRI sequence types; T2-weighted Fluid-Attenuated Inversion Recovery (FLAIR) and T1-weighted MRI volumes. The experiments used data from two datasets. The first experiments used data from one scanner, *Philips Ingenia*. The other experiments used a larger dataset that contained MRI from different scanner models, which are shown in the table below.

Data splits	Skyra	Prisma	Ingenia	Optima MR450w	Achieva	Avanto	Avg. voxel size
Training	51	156	216	85	57	14	$0.863\,\mathrm{mm}^3$
Validation	7	35	50	13	11	5	$0.856\,\mathrm{mm}^3$
Test	7	38	48	20	11	4	$0.865\,\mathrm{mm}^3$

The best model in each experiment was found when the validation loss was at the lowest point during 50 epochs. We then tested how the model performed for whole 3D lesions and computed the average mini-batch $F_3 score$. To find the 3D lesions, we used 3D pixel-connectivity for the 3D predicted mask and ground truth mask. Here we used the average recall score for a given lesion, where we separated lesions into four main lesion size brackets,

from small to large. The lesions were detected as true positive if more than 60% of the lesions were detected.

Results and discussion Overall, good segmentation performance was obtained compared to the current state-of-the-art[4]. In the test data, an F_3 score of $= 0.74$ was obtained on the test data where F_3 is the weighted harmonic mean of precision and recall, reaching its optimal value at 1. The segmentation accuracy decreased with decreasing lesion size, as expected. However, for clinically relevant lesion sizes, an average recall score of 0.93 was achieved. Finally, the model also correctly detected some WMH lesions that were missed by the experts, which suggests some level of generalization of the models.

Keywords: White matter hyperintensities, Brain lesions segmentation, UNet

Abbreviations

MRI - Magnetic Resonance Imaging

WMH - White Matter Hyperintensities

PET - Positron Emission Tomography

SGD - Stochastic Gradient Descent

GT - Ground truth

FLAIR - Fluid-Attenuated Inversion Recovery

CNN - Convolutional Neural Network

MSD - Medical Segmentation Decathlon

AD - Alzheimer's Disease

Contents

Preface

This report documents the writer's master thesis (M.Sc.) that have been a part of the master's program *Electrical engineering, informatics, and technology: Signal processing and imaging* at *Department of informatics* (IFI) at *University of Oslo* (UiO). The thesis is written for Computational radiology and artificial intelligence (CRAI).

I want to thank my family, Ina, Antonio, Mira, Adrian, Knut and Kari, which have been very encouraging throughout the whole master's program, especially since the COVID19 pandemic has been a part of the whole master thesis.

I also want to thank my advisors Anne H. S. Solberg (IFI), Atle Bjørnerud (CRAI, OUS, and Dept of Physics, UIO) which have helped me a lot, and Per Selnes, who has both helped me learn much about the brain and collecting a lot of data for me to work with.

1 Introduction

Diagnostic imaging refers to the medical field of radiology and nuclear medicine, where digital images generated by advanced scanner technology are used to detect and characterize the disease and to follow up on the effect of treatment and intervention. Like most other fields of technology, medical imaging has been transformed by the digital revolution, which has led to a sharp increase in the amount of data generated for each patient examination. This rapid increase in data generation has not been matched by a similar increase in the number of experts (radiologists and other medical professionals) available to inspect and interpret the imaging data. This means that each medical expert must review an increasing amount of medical image data for each examination done, with the risk of missing important findings, which can impact patient treatment and outcome. There is, therefore, a well-established need for robust automated computer-based tools to aid the human expert in the image interpretation process. The most obvious (and so far most established) use of computer-aided diagnostic imaging support is in the field of detection and quantification of focal pathology in medical images. Here, focal pathology refers to any pathological process that manifests itself as visual (to humans or computers) changes in the medical images caused by pathological changes in the tissues. Typical examples of focal pathological changes include tumors and neurodegenerative disease.

In this thesis, we have specifically been analyzing magnetic resonance imaging (MRI) data of patients with early signs of Alzheimer's disease. MRIs from these patients often have pathological focal changes in brain white matter, referred to as white matter hyperintensities (WMH), and manual delineation and quantification of these changes is an extremely laborious and time-consuming task even for an expert.

Automated WMH detection is a typical semantic segmentation problem, and the use of deep learning-based methods for this purpose has already proved very useful [5]. More specifically, the branch of deep learning using convolutional neural networks (CNNs) has become the standard approach for image segmentation. One very popular type of CNN is the UNet[3].

UNet uses an encoding-decoding structure with skip connections which helps attain the finer details in the larger resolution layers while also having a large receptive field. After UNet was published, the architecture has been used in many different segmentation problems for both medical and non-medical tasks.

Magnetic resonance imaging is a medical imaging technique used in radiology. MRI uses a strong magnetic field to create an interaction between particles and their spin state and charge in the body. The patient is exposed to the electromagnetic energy at the correct frequency that is absorbed by the body, and then a short time later, the energy is reemitted and can be detected in the form of radiofrequency signals in a received coil, and which is subsequently digitized and computer-processed to generated volumetric images representing the biophysical properties of the signal emitting water molecules in tissue.

MRI is particularly useful to visualize soft tissues like the brain, and different types of image contrasts can be generated depending on the parameters used and the specifics of the data acquisition process.

In this thesis, two types of MRI sequences with different contrast properties were used; namely FLAIR and T1-weighted sequences. Fluid-attenuated inversion recovery, or FLAIR for short, uses a method for suppressing the cerebrospinal fluid (CSF) during imaging [6].

Since FLAIR sequences suppress the otherwise bright CSF signal, it is very sensitive to detecting WMHs and is, therefore, the sequence of choice for this purpose in clinical use. T1-weighted images, on the other hand, renders WMHs dark in the image and are thus less sensitive to WMH detection but may add values in WMH characterization (categorizing different sub-types of WMHs).

T1-weighted images are also the preferred modality for a general assessment of brain structure. In Figure 1.1 we can see one sample of an axial slice with WMH in FLAIR. The MRI's are volumes, which allows us to slice a 2D sample in three different orthogonal orientations as seen in Figure 1.2.

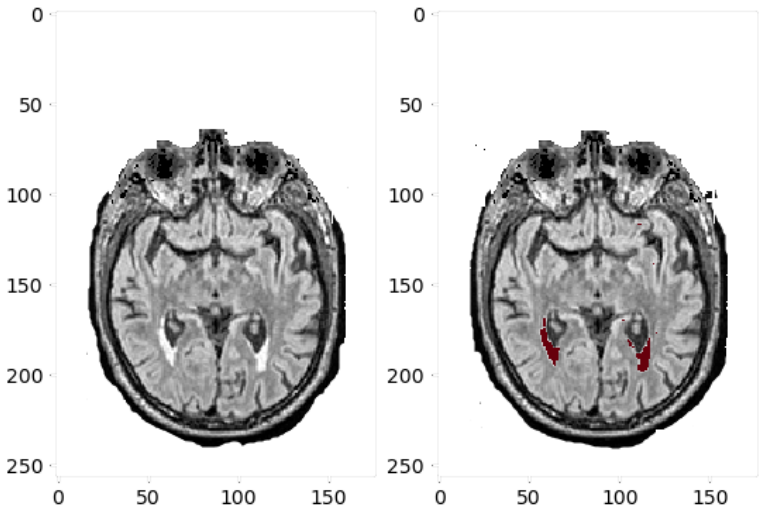


Figure 1.1: *(a)(b). Example of a 2D axial slice with WMH marked as red in (b).*

1.1 Goal of this thesis

The goal of this thesis is to experiment with different UNet architectures for automatic WMH lesion detection in MRI volumes. This work comprised of two main points:

- Use UNet architecture as the base model with limited GPU memory footprint.
- Find the best segmentation model.

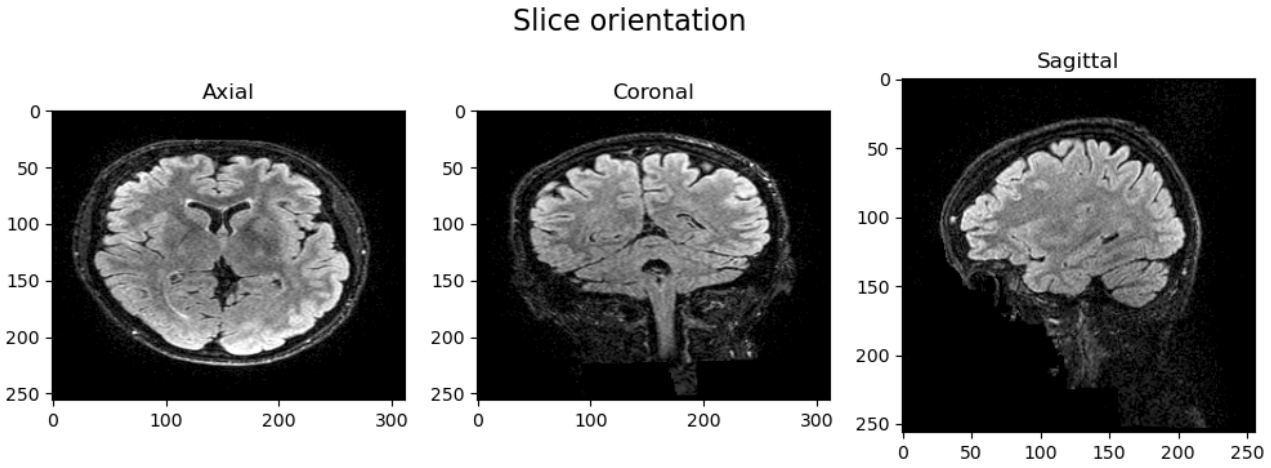


Figure 1.2: *(a)(b)(c) Sample slice of all three orientations.*

1.2 Outline

In this thesis, we follow an evolving process where an experimental outcome affects how we proceed with the next experiments. The main chapters are organized as follows:

Chapter 2: Intro to CNNs explains some of the theory behind convolutional neural networks. Here we introduce several of the popular activation functions and optimizers used in deep learning-based segmentation.

Chapter 3: Related literature gives an introduction to some of the related literature and their methods upon which various of the experiments in this thesis is based upon.

Chapter 4: Metrics and loss functions discuss and explains the metrics and loss function used in experiments.

Chapter 5: Dataset and preprocessing shows the statistics of the data and how we preprocessed the dataset before using it in both training and inference.

Chapter 6: Models explains the model architectures used in experiments.

Chapter 7: Methology explains the experimental methods used.

Chapter 8: Experiments and results shows the experiments and their results. These results affected what model we used going forward.

Chapter 9: Discussion is the chapter we discuss results and some of the possible future work to enhance the model.

Chapter 10: Conclusion gives a summary of the conclusion made after the experiments and discussion.

2 Intro to CNN

Image segmentation is a process of partitioning images into one or multiple regions. This makes it easier to change the representation or interpret the images. In deep learning, we can have a model learn the per-pixel probability of belonging to a given class by using supervised learning with pre-labeled target masks. In this section, we reiterate some of the core principles behind the model architectures used in the thesis and the metrics used for evaluation. The loss function is also discussed, and the derivation is shown.

2.1 Convolutional neural networks

Convolutional neural networks (CNN) have shown impressive results in image processing. CNN uses layers of convolutions to output feature maps which are learned by the gradient flow of back-propagation given by a loss function. Neural networks use the idea of a neuron depicted in Figure 2.1. The input goes through a series of weighting and adding operations. The output is then activated by a activation function, more details in Chapter 2.6.

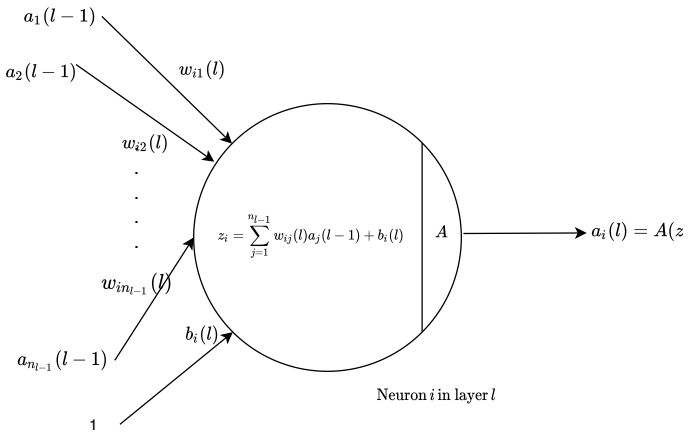
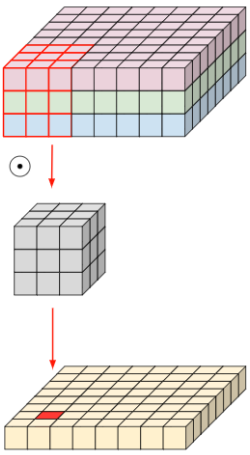
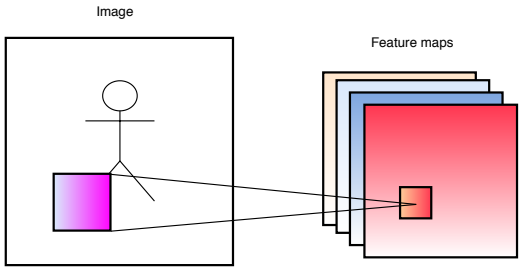


Figure 2.1: *Artificial neuron. [p.937][7].*

In Figure 2.2 (b), we see four convolution kernels extracting four different feature maps, and then each neuron can be activated by an activation function. We can apply different techniques like drop out, batch normalization, and much more to enhance the network. Figure 2.3 shows the convolution map output from three to one channel using a 1x1 kernel. This is used in the last layer to reduce the features to one dimension. After reducing the channels, we can, in a binary segmentation or classification model, apply a sigmoid activation to squeeze the values to the range [0, 1].



(a) *Convolution map calculated by a convolution kernel over three channels, this is therefore a 3 to 1 channel transformation. 2D convolution has kernels that iterate in 2D but weighs all pixels inside the kernel over all channels.(Image courtesy of Eli Bendersky [8])*



(b) *An example with four different filters outputting four different feature maps. You could also add a bias term which is added to the output of the convolution layer.*

Figure 2.2: *This figure shows how the convolution layers work. Adding a lot of these layers can extract incredible feature details.*

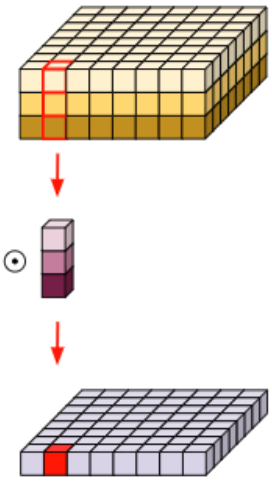


Figure 2.3: *In the models shown, the last layers use 1x1 convolutions to merge the last feature map layers into one map, which goes through softmax or sigmoid for prediction.*

In the activation layer is a sigmoid function, Equation 2.9, which forces the output between [0, 1] such that we get probabilities for a given pixel to belong to the positive class. This output is then used to compare with the ground truths, which are compared using the Tversky focal loss. The loss function will be the source of gradients propagating through the network.

2.2 Receptive Field

Receptive fields are important to understand when setting up a CNN network. The receptive field is how many neurons each neuron in the deeper layer is connected to. This can be viewed as the *field of vision* of the neurons.

2.3 Effective Receptive Field

The effective receptive field (ERF) is the area of the original image that influences a given neuron in a given layer. ERF shows the connection back to the original image from a given neuron in the layer l. The ERF using the *buttom-up approach* can be calculated as shown in Equation 2.1 [9]:

$$R_k = R_{k-1} + (f_k - 1) \prod_{i=1}^{k-1} s_i, \tag{2.1}$$

where, R_k is the ERF for a neuron in layer k and $R_0 = 1$ (input layer), f_k is the filter size in layer k and s_i is the stride of the layer i. The bottom-up approach projects the ERF from a layer k onto the input image. The advantage of this method is that it produces the ERF for all layers from one feed-forward pass.

The receptive field is important for knowing how much context each neuron has. If this field is very small, neurons will only use information in the vicinity, which could decrease the model's understanding of the input and prediction results. In medical imaging, we might want to know what information is taken into account for a decision. We might want to know how large the area of the image is taken into account for deciding to weigh the pixel more than others. For example, if a feature from a different part of the body is inside the receptive field, the network could use this information from a different part of the body to find something interesting in other parts of the body, depending on the problem. In Figure 2.4 we can see an illustration of the receptive field. With the UNet architectures used in all the experiments, the effective receptive field is at 40x40 in the deepest layer, which should be sufficient. Although the values inside the field do not weigh the same as described in [10], only a fraction of the theoretical field seems to be used.

2.4 Batch normalization layers

During training, the parameters of the layers change. Hence, changes to the distribution as well. The network, therefore, learns slower as it needs a more fine-tuned learning rate and initialization. Batch normalization standardizes the inputs to the activation function such that the mean is zero and the standard deviation is one. Batch normalization will smooth out the optimization landscape causing a more stable learning process[11] and will force the response to be in the linear region of the sigmoid function. Batch normalization is given as:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}] + \epsilon}}, \tag{2.2}$$

where $\hat{x}^{(k)}$ is the input to the activation layer, $x^{(k)}$ is the output from the convolution layer, and k is the batch index. Since this normalization constrains the value between [-1, 1] (good for gradient flow for sigmoid) the transform in Equation 2.3 is used for scaling and shifting. Using the scale- and shift transformation, the output could represent an identity transform if it is found to be optimal.

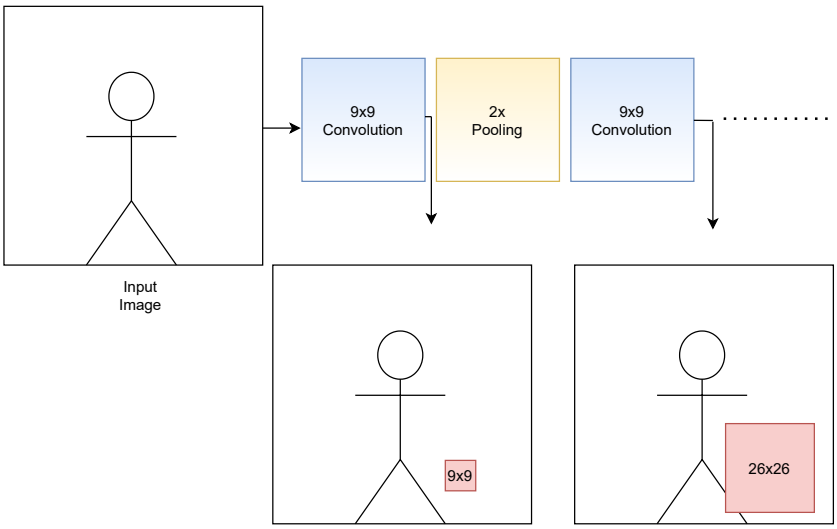


Figure 2.4: *Effective receptive field of a network. By adding cascading convolutions, the network convolves the previous weighted pixel into another pixel, increasing the receptive field, which is important concerning how large the lesions are.*

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{2.3}$$

These operations are differentiable and store the parameters for use in evaluation. In evaluation, the learned parameters are used instead of the input batch mean and variance. One big problem with this is that it uses batch statistics, which could be a bad representation if the mini-batch size is small. During inference the batch normalization uses the running mean and running variance estimates in Equation 2.4 and Equation 2.5:

$$\begin{aligned} \hat{\mu}_{(k)new} &= m\hat{\mu}_{(k)} + (1-m)\hat{\mu}_{(k)}, \\ \hat{\sigma}^2_{(k)new} &= m\hat{\sigma}^2_{(k)} + (1-m)\hat{\sigma}^2_{(k)}, \end{aligned} \tag{2.4}$$

where m is momentum and $\hat{\mu}_{(k)}$ and $\hat{\sigma}^2_{(k)}$ is the running mean and variance.

$$y^{(k)} = \frac{\gamma}{\sqrt{\hat{\sigma}^2_{(k)new} + \epsilon}} x^{(k)new} + \left(\beta - \frac{\gamma\hat{\mu}_{(k)new}}{\sqrt{\hat{\sigma}^2_{(k)new} + \epsilon}} \right). \tag{2.5}$$

2.5 Group Normalization

Group normalization does not depend on batch size and uses the same statistics for both training and inference[12]. Group normalization normalizes G groups of channels instead of normalizing across batch as shown in Figure 2.5. Variable G is the number of groups to normalize. If we use $G = C$, we get instance normalization[13]. Instance normalization is to normalize each channel independently. The output is then normalized using Equation 2.6:

$$\hat{x}^{(c)} = \frac{x^{(c)} - \mathrm{E}[x^{(c)}]}{\sqrt{\mathrm{Var}[x^{(c)}] + \epsilon}}, \tag{2.6}$$

where x_c is the input from channels in the respective group, and $\hat{x}^c$ is the normalized input to the activation function.

Group normalization does have the same scale and shift mechanic per channel as described in Chapter 2.4 in Equation 2.3.

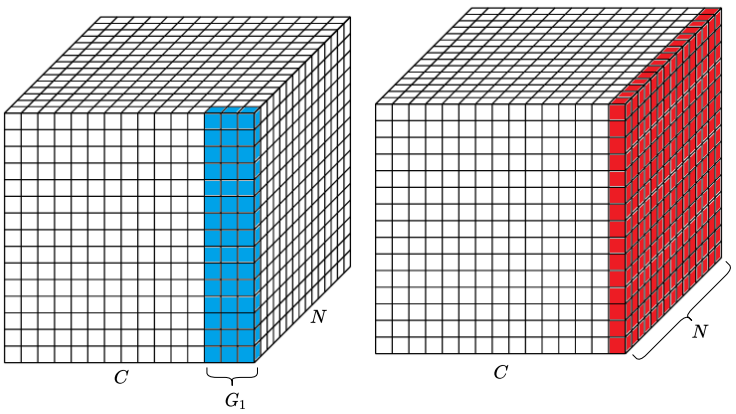


Figure 2.5: *Left: Group normalization layer, Right: Batch normalization layer. By using $G = 5$, we get three channels in each group if we have 15 channels.*

2.6 Activation functions

Activation function makes weighted features *click*. By looking at activation functions in the way of the biological brain, it let features weighted in such a way to pass through the network in a given way. For instance, if a ReLU function is used, the values below zero are set to zero, and values above are the same. All negative values will therefore be suppressed by a ReLU. Other activation functions are discussed below.

ReLU One of the most used activation functions as of the year 2020 is the Rectified Linear Unit (ReLU)[14]. The function is given in Equation 2.7:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & else. \end{cases} \tag{2.7}$$

Equation(2.7) applies a non-linear transformation but contains two linear piecewise functions. The linearity makes the network generalize more and easy to optimize. The great advantage of ReLU is by zeroing out negatives will create sparsity in the activations assuming randomly initialized weights. By outputting input ≥ 0 leads to unbounded scale Equation 2.8:

$$\max(0, ax) = a \max(0, x)\, a \geq 0. \tag{2.8}$$

One of the negative properties of the activation is that it is non-differentiable at zero. However, this is fixed by setting the derivative to zero or one. The dying ReLU problem, since the function forces all outputs below zero

to zero, the network could end up with dead neurons. The function is plotted in Figure 2.6. This activation function is used in almost every architecture, including UNet.

Figure 2.6: *ReLU activation function. All negative values are zero and the same output as input if elsewhere. ReLU is not differentiable at zero, therefore, thresholding to zero is performed if* $x = 0$. *Image grabbed from [15].*

Sigmoid The sigmoid function is a very known probability transforming function. The function squeezes the values $x \in [-\infty.\infty]$ to $y \in [0, 1]$. Hence, this function is very appropriate for the last layer of the network. A plot of sigmoid is shown in Figure 2.7) and shown in Equation 2.9. Sigmoid is nearly linear around -1 and 1, while exponentially flattens outside this interval. One problem with this is that the gradient vanishes as the neurons saturate, causing a problem with the gradient flow. Batch normalization discussed in Chapter 2.4 alleviates some of this problem as it normalizes the output to be inside the linear region of the sigmoid. This function was only used as the last layer in the models used in the experiments and in the attention gates. Attention gates are explained in Chapter 6.2.

$$S(x) = \frac{1}{1 + e^{-x}}. \tag{2.9}$$

In Equation 2.9 x is the output of previous layer.

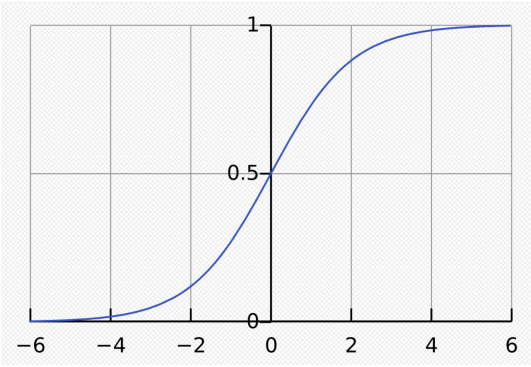


Figure 2.7: *Sigmoid function. This function squeezes the values between 0 and 1. A problem with this function is that the gradients are mostly linear in the middle and dies out exponentially at the tails, causing vanishing gradients. Since the sigmoid is almost linear between 0 and 1, the sigmoid either has a very small gradient or linear gradient. This function does not have a mean of zero, which could make the network converge slower as mentioned in* Speeding learning, *[16]. This function works well with batch normalization as the output is standardized to be within the linear region. The image is taken from [17]*

Mish Mish uses a self-gating property by multiplying non-modulated input with the output of non-linear function of the input. This activation function also eliminates the dying ReLU problem by preserving some of the negative valued information. The function is plotted in Figure 2.8 and the activation function and its derivative is shown in Equation 2.10, Equation 2.11:

$$f(x) = x\tanh(\text{softplus}(x)) = x\tanh\left(\frac{1}{\beta}\ln(1+e^{\beta x})\right), \tag{2.10}$$

$$f'(x) = \frac{e^x\omega}{\delta^2}, \tag{2.11}$$

where we used $\beta = 1$ in the experiments. Soft-plus also has a threshold parameter where values above will revert to a linear function for stability, input $\cdot\ \beta >$ threshold.
In Equation 2.11, $\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^x(4x+6)$ and $\delta = 2e^x + e^{2x} + 2$. The function is also bounded which acts as a strong regularizer. Mish function has a much smoother optimization landscape, hence gradient flow is more smooth[18]. In Figure 2.8 an example of input noise to a toy model shows the difference during inactivation. The Mish activation function lets most of the input have a response, while the ReLU activation has a lot of dead neurons. According to [18] using Mish might increase performance as Mish has smoother gradient flow, elimination of dead neurons, avoids saturation, has self-gating property, where the scalar input is provided to control the zero and sign of the activation. The function is also self-regularized by bounding the larger negative values to zero, like ReLU forcing negative values to zero can cause sparsity in the neuron activations and add generalization.

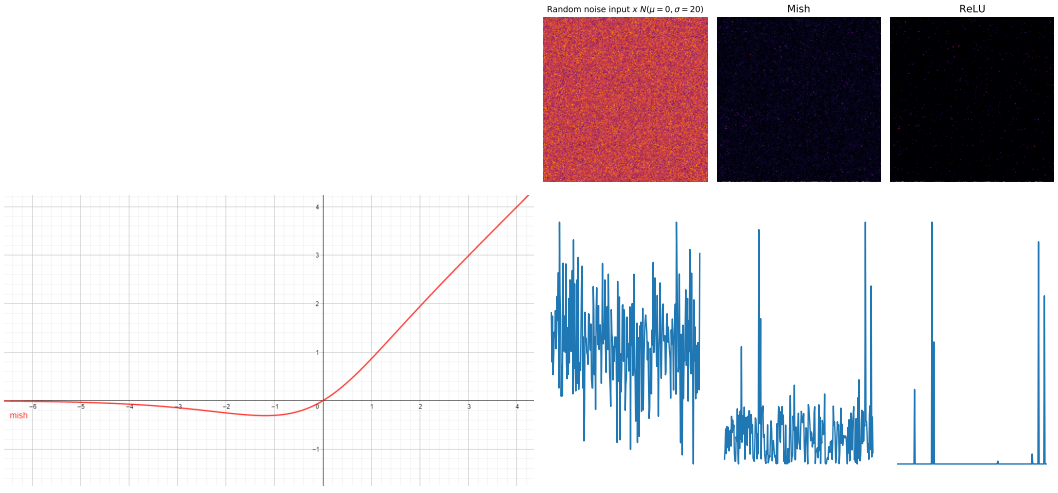


Figure 2.8: *Left: Mish activation function. Right: Gaussian noise image* $x \sim N(\mu = 0, \sigma = 50)$ *input to a toy model consisting of 3 convolution layers using mish activation and one with ReLU activation. The ReLU kills off almost everything, while Mish have similar activation on the spikes, it does not have the same amount of "dead neurons" as ReLU.*

2.7 Optimizers

Optimizers are functions that tie the loss function together with the model's parameters. Since the model function is very complex and has to be estimated from the dataset, we have to minimize it with respect to the loss function in an iterative manner. This can be done in many ways, but some of the more popular optimizers use techniques like momentum to *remember* previous changes to help estimate the path towards the minimum.

SGD with momentum One of the most popular optimizers is the Stochastic gradient descent (SGD) with momentum. SGD used a single example from the training data and updates the parameters by finding the gradient to a loss function and multiply it with a learning rate parameter that scales down the gradient. Which is to avoid forgetting what was learned. The SGD has the following update rule shown in Equation 2.12:

$$\Delta w_t = -\eta g_t, \tag{2.12}$$

where Δw_t is the change to the weights in the network during Back-propagation, η is the learning rate and $g_t = \nabla f(\theta)$ is the gradient for a loss function $f(\theta)$. Momentum is an extra term added to the update function. Momentum adds weights from previous weight updates that add "stopping power" to the update rule. This can make the update bypass noisy regions of the loss landscape and help bypass local minima. Update rule for SGD with momentum shown in Equation 2.13:

$$\Delta w_t = -\eta g_t + \alpha \Delta w_{t-1}, \tag{2.13}$$

where α is the momentum parameter which is scaled depending on how much momentum is needed but is usually between 0.5 and 1.0 [19]. SGD with momentum therefore has the update equation shown in Equation 2.14.

$$w_t = w_t - \eta g_t + \alpha \Delta w_{t-1}, \tag{2.14}$$

Adam The Adam optimizer[20] is the most popular as of today[21]. It uses an adaptive learning rate, which means that it computes individual learning rates for different parameters. It uses the first and second moments of the gradients to adapt the learning rate for all the weights. The moments of Adam are calculated as shown in Equation 2.16 on the mini-batches, not just single samples.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.15}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.16}$$

Here m, v is moving averages, and g is the gradient on the current mini-batch.

By initializing the running average with zeros the expectation leaves bias terms, dividing the moving averages by the bias gives the bias corrected equations (2.17, 2.18)

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{2.17}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{2.18}$$

Finally, the weights are updated as shown in Equation 2.19.

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{(\sqrt{\hat{v}_t} + \epsilon)} \tag{2.19}$$

The initialized learning rate will act as the *bounded* learning rate as the Adam optimizer changes the learning rate for the different parameters. The term ϵ is a small value to stabilize the update. In most cases, this value is something in the order of $1e^{-6}$.

2.8 Regularization

Regularization is a method in machine learning that helps the model from overfitting. Regularization is added in different ways to *punish* the optimization, for example, adding extra loss in the loss function based on the number of parameters the network has or add extra loss based on the size of the weights. For instance, if a model has extremely large weights on a few weights, the model will most likely be overfitted to various types of features. Punishing the model to spread out the weight values can make the model more generalized.

Dropout is a technique that removes layers with a given probability and, therefore, a **stochastic regularization technique**. One motivation behind this is the natural process of gene mutation. Dropping out layers with a given probability for each training sequence will produce thinned versions of the main network, and each back-propagation is for these thinned networks. The removed layers get a gradient of zero. Hence, the trained network is built from 2^n thinned networks where each thinned network rarely gets trained. In test time, the layers are not dropped but rather multiplied with the probability p, which was assigned to that given layer. This retains the expected value from training to testing [22].

3 Related literature

This section discusses and reiterates the highlights of related literature.

3.1 U-Net: Convolutional Networks for Biomedical Image Segmentation

Architecture The UNet architecture is shown in Figure 3.1. It consists of contracting and expansive paths. The contracting path consists of repeated application of two convolutions (unpadded), each followed by a ReLU activation function and a 2x2 max pooling with stride = 2 for downsampling. At each downsampling step, the number of feature channels is doubled. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 upsampling that halves the number of feature channels. A skip-connection is concatenated from the contracting to the expansive path, and two 3x3 convolutions, each followed by a ReLU. In the paper, they had to crop the maps from the skip-connection since they did not pad the convolutions, hence, losing some of the borders for each convolution. As the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes.

Method Medical images can be very large, so a *overlap-tile* strategy can be implemented to larger images. A crop of the big image is taken (to have even size, so the contracting/expanding part of the network works without dimension problems) and then segmented. The borders of the images are mirrored for the convolution kernels to fit. U-Net does not use any padding which, causes the output to be a bit smaller in size than the input. The borders between cells are hard to segment correctly, hence, the author added a weight map for each mask image that has loss weights at the border used in the cross-entropy loss function. The weights map was computed using morphological operations and computed as shown in Equation 3.1.

$$w(\boldsymbol{x}) = w_c(\boldsymbol{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\boldsymbol{x}) + d_2(\boldsymbol{x}))^2}{2\sigma^2}\right) \tag{3.1}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the different classes, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of nearest cell, $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell and the positions are $\boldsymbol{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$. In the paper they used $w_0 = 10$ and $\sigma \approx 5$ pixels. These weight maps are used in the cross entropy loss function given in Equation 3.2.

$$E = \sum_{\boldsymbol{x} \in \Omega} w(\boldsymbol{x}) \log(p_{l(\boldsymbol{x})}(\boldsymbol{x})) \tag{3.2}$$

where $p_{l(\boldsymbol{x})}(\boldsymbol{x}))$ is the probability of correct class in that position and $l : \Omega \to \{1, ...K\}$ are the true labels.

The neuron weights are initialized such that the output variance is unit variance. This is done by drawing

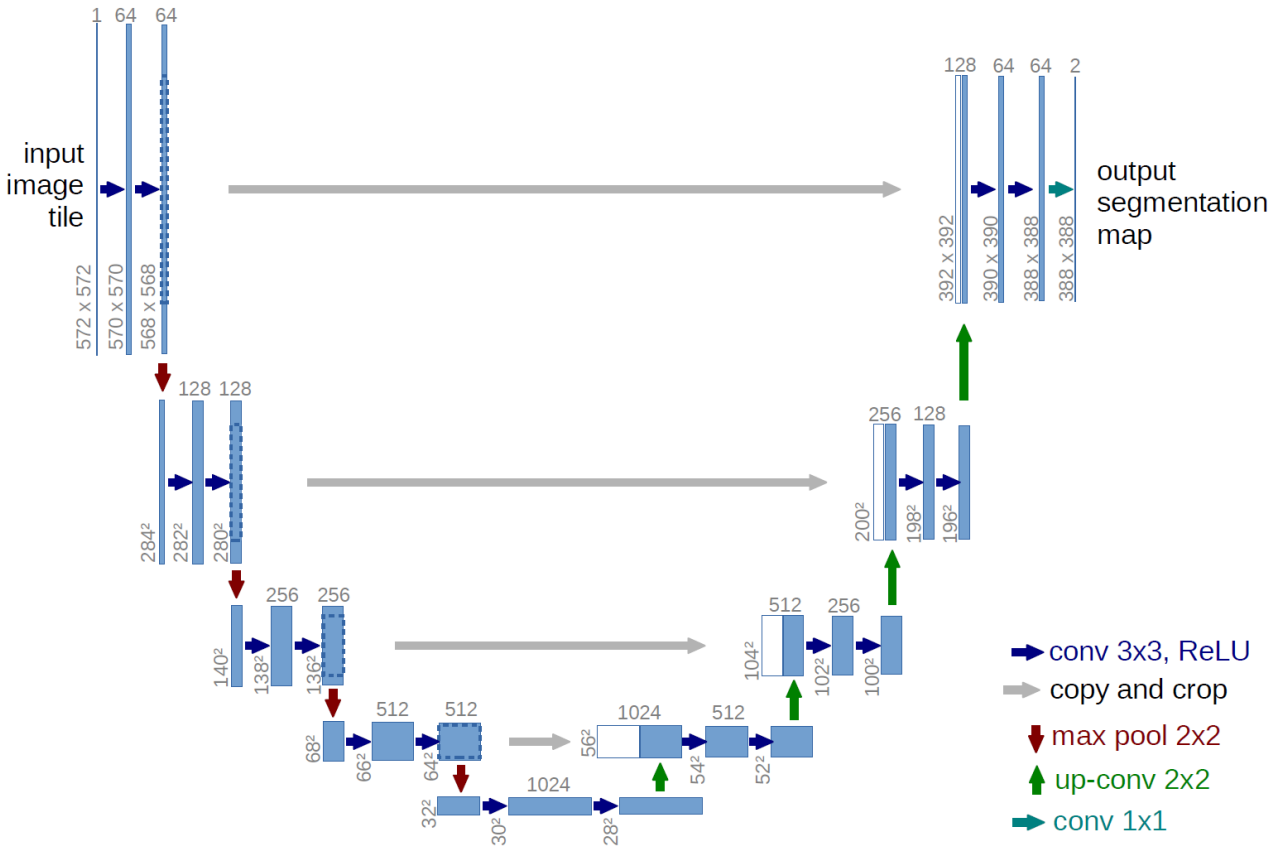


Figure 3.1: *UNet architecture from the paper [3].*

values from a gaussian distribution and using a standard deviation of $\sqrt{\frac{2}{N}}$ where N denotes the number of incoming nodes to one neuron. In the paper, they favored using large input tiles instead of large batch sizes, hence, they used a batch size of one single image. This was to make maximum use of the GPU memory.

The augmentation used were primarily shift, rotation, deformation, and gray value variations. Random elastic deformation was used to increase the size of the dataset, as it was very small. They also used drop out for the contracting path.

Optimizer and loss The optimizer was stochastic gradient descent with momentum = 0.99, and the loss function was softmax with cross-entropy.

Dataset The dataset used in the original UNet paper are cells imaged with light microscopy. The data is from the ISBI cell tracking challenge 2014 and 2015. The first dataset "PhC-U373" contains Glioblastaoma-astrocytoma U373 cells on a polyacrylamide substrate recorded by phase-contrast microscopy. This contains 35 partially annotated training data. So here we can see there is not much training data. The second data set was "DIC-HeLa", which are HeLa cells on a flat glass recorded by differential interference contrast (DIC) microscopy. It contains 20 partially annotated training images. When images were very large, the overlap-tile strategy was used, and at the edges, the pixels were symmetrically padded.

3.2 Alzheimer's Disease Classification through Transfer Learning

Method In the paper[23], they adapted two popular CNN architectures for an AD (Alzheimer's Disease) diagnosis problem through transfer learning. Transfer learning is the concept of using the trained weights and architecture of a pre-existing network that has been used on another domain and re-train it for a different task.

The idea is that by using the pre-trained weights, we *skip* a lot of the local minimums in the domain, such that we do not need to train very much for the new problem and need less training data. The architectures are *VGG16* and *Inception* which were winners of the previous ImageNet Large Scale Visual Recognition Challenge.

These architectures were trained on natural images, which is another domain than the problem in this paper(MR images). The pre-trained weights are open source and can be downloaded. In the paper, they did not choose the dataset at random but rather chose images with most information using entropy for each image slice using Equation 3.3.

$$H = -\sum_{i=1}^{M} p_i \log p_i \tag{3.3}$$

where M is the set of intensity values in the image if gray-scale (256) and p_i is the probability for a given intensity.

The pre-trained models *VGG16* and *Inception* with the last layer changed to fit the AD detection task, so this layer has been trained from scratch. The network architectures are as follows:

VGG16:
VGG16[24] uses 16 layers with 3x3 convolution layers. This network has a lot of parameters making it very heavy to train, and the pre-trained weights are very large, although very intuitive to understand.
Inception:
The inceptions[25] breakthrough is in the realization that non-linear functions can be learned by changing how the convolutional layers are connected. By replacing the fully connected layer with a global average pooling and then connect it with a softmax layer for classification. Thus, there are fewer parameters, and as a result, less overfitting.

Dataset The dataset contains MR images of patients with AD or HC. These contain both cross-sectional and longitudinal slices. These networks have then been trained on a small dataset(most informative 32 images) taken from the OASIS dataset[1]. To test the generalization power of the transfer learning a limited amount of images were used to train the network. The entropy method discussed in Methods has been used to get the most informative 32 images from the axial plane of each 3D scan. This resulted in 6400 training images, 3200 of which were AD nad the other 3200 were HC. The MR images are resized to fit the models, for VGG16 the images have been resized to (150x150) and (299x299) for *Inception*.

Contrary to the MRI volumes in our dataset, these volumes have been skull-stripped, leaving only the brain visible. One of the goals of our models is to have volumes *straight out of the scanner* such that the model can predict as fast as possible and without eventual error-prone pre-processing.

[1]https://www.oasis-brains.org/ - latest 05.04.2021

3.3 *A Novel Focal Tversky loss function with improved Attention UNet for lesion segmentation*

Method In [26] they used a generalized dice loss function, Focal Tversky loss. This function uses the parameters α, β, and γ to tune the loss function for a given problem. Increasing α weights the false-negative more than false positives, and increasing β weights false positives more. The γ changes the slope of the function to increase the gradient for easy/hard examples. The function is shown in Equation 4.12,

$$TI_{floss} = \left(1 - \frac{TP + \Omega}{TP + \alpha FN + \beta FP + \Omega}\right)^{\gamma}. \tag{3.4}$$

Architecture The architecture is a UNet structure with attention gates[**?**]. The architecture is shown in Figure 3.2. The paper did experiments with both multi-scale input and deep supervision layers. CNN's with attention gates (AGs) focus on the target region, with respect to the classification goal, and can be trained end-to-end. At test time, these gates generate soft region proposals to highlight salient ROI features and suppress feature activations by irrelevant regions [26][Intro].

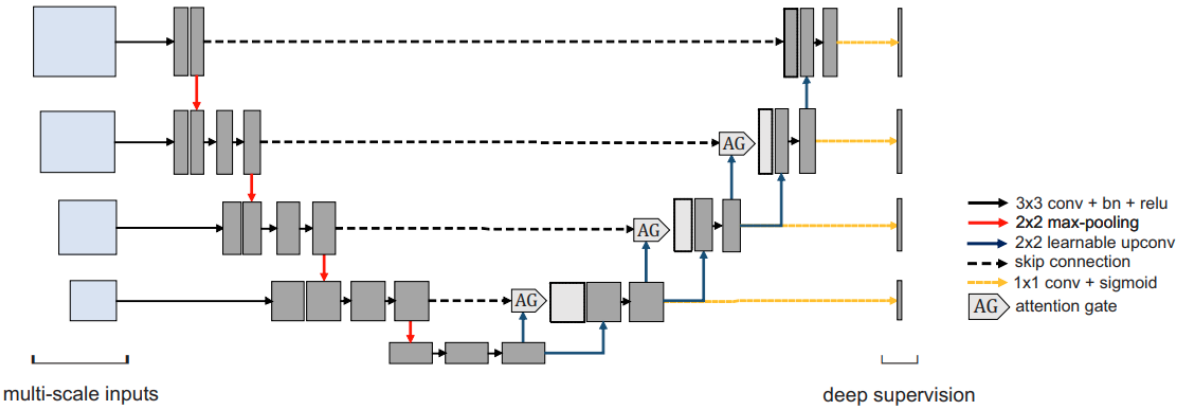


Figure 3.2: *Attention UNet architecture from the paper [26].*

Dataset In the paper, they used the Breast Ultrasound Lesions 2017 dataset (BUS)[27] dataset where lesions occupy 4.84% of the images area.

Optimizer and loss The optimizer used was stochastic gradient descent with momentum, using an initial learning rate at 0.01, which decays by 10^{-6} on every epoch. The loss function is Focal Tversky loss as discussed in **Method**.

3.4 *Fully Convolutional Network Ensembles for White Matter Hyperintensities Segmentation in MR Images*

Method and Architecture In [4], each 3D scan they employed a z-standardization for *pre-processing* to normalize the intensities distribution for each individual 3D scan. This paper used ensembles of UNet architecture where many models were trained. These models were then used to vote for each individual pixel by computing

the mean for all model outputs for final prediction during evaluation. This reduces overfitting and variance by taking the mean of the final results. The augmentation consisted of sheering, rotation, and scaling. The scaling was set to have a random interval between the smallest and largest pixel resolutions in the dataset.

Dataset This data comes from the WMH segmentation challenge at https://wmh.isi.uu.nl/, consisting of FLAIR and T1 MRI volumes.

Parameters The U-Net hyperparameters were set as follows: batch size for computing the training loss was set to 30; learning rate was set to 0 0002; the number of epochs was set to 50. The number of models in the ensemble was set to 3.

3.5 Thickened 2D Networks for Efficient 3D Medical Image Segmentation

Method In [28], they experimented with an alternative to 3D convolutional networks by using more than one 2D slice from the MRI volume as channel information. This is a way to include 3D information in the model, but also decrease the memory usage and increase the speed of both training and inference. In experiments, they tested different sizes of thickness, from 3-slice to at most 24-slice. In the experiments, results keep increasing until slice thickness reaches 15. They also tested both the 2D and 3D thickened approach, where 2D is for one orientation only, while 3D uses all three orientations, axial, coronal, and sagittal.

Architecture The architecture proposed in this paper was split into two parts, where the 2D backbone model were DeepLabV3+ [29] based on ResNet50 [30]. The input is first split into mini groups and then goes through the first 2D backbone. The outputs are fused with concatenation from the different groups on channel dimensions. A two-layer convolution compresses the channel number to its half and the other to 256. The features then go through a slice-sensitive attention module. The architecture is shown in Figure 3.3.

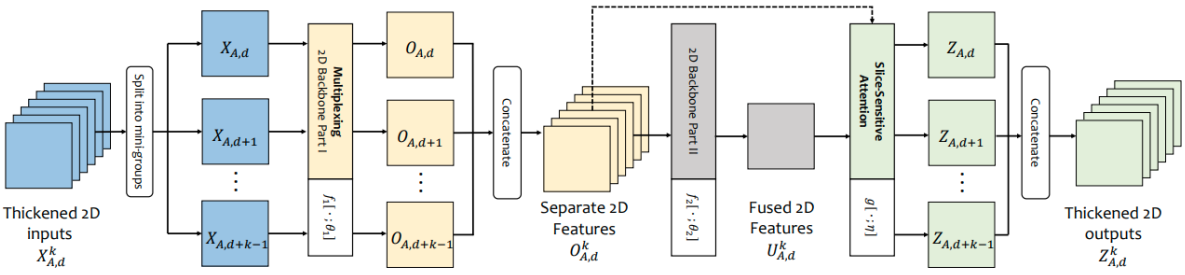


Figure 3.3: *Thickned architecture from the paper [28].*

Dataset Experiments are conducted on several abdominal organs individually, including two regular organs and three blood vessels in their dataset, and the hepatic vessels in the Medical Segmentation Decathlon (MSD) [31] dataset.

Optimizer and loss and parameters The loss function used was the Dice loss. . For the 2D setting, they initialized the model with ImageNet [32] pre-trained weight and trained for 100k iterations with SGD optimizer, where the momentum is 0.9 and weight decay 0.0005. The batch size is 8, and the learning rate is 0.005, which decays by a factor of 10 at iteration 70k and 90k. The batch size is 8, and the learning rate is 0.005, which decays by a factor of 10 at iteration 70k and 90k. For the 3D setting, they adopt an SGD optimizer with a polynomial learning rate scheduler starting from 0.01 with a power of 0.9.

4 Metrics, optimizers and loss functions

This section explains the metrics used for evaluation, some explanation of the most popular optimizers, and the one used in the experiment. The loss function used in experiments is explained.

4.1 Evaluation Metrics

Metrics measure the results which are important for a given task. These measurements are done on the validation and test data. Probabilities from the last sigmoid layer are thresholded to zero if they are below 0.5 and 1 if equal to or above 0.5 and then use the validation metrics. The best model found during training is done by using the weights from where the validation data has the lowest Tversky focal loss for 50 epochs. Tversky focal loss is described in section(4.2. The metrics are found by flattening the whole batch and then computing the metric. In Figure 4.1 and Figure 4.2 the two methods are shown.

Recall (smooth) Shows how much of the lesion labels are found with a value between zero and one, where a value of zero is no true positives and a value of one if there are no false negatives. Recall is shown in Equation 4.1:

$$R = \frac{TP + \Omega}{TP + FN + \Omega}, \tag{4.1}$$

where TP is true positive, FP is false positive, FN is false-negative and Ω is a smoothing factor which provides numerical stability and can be used to smooth out the function[33], this parameter is further discussed in section(4.2). The outputs from the network are from a sigmoid function. Hence the following equations(4.2):

$$\begin{aligned} TP &= \sum_i \hat{y}_i y_i, \\ FP &= \sum_i (1 - y_i)\hat{y}_i, \\ FN &= \sum_i y_i (1 - \hat{y}_i), \end{aligned} \tag{4.2}$$

where $\hat{y_i}$ is the probability of a pixel belonging to class 0 or 1 (negative/positive) this value is given by the sigmoid function shown in Equation 2.9 which squeezes the output value into the range $[0, 1]$. y_i is the ground truth pixel, which has the binary value of either zero or one. Since as few false negatives as possible are very important in medical problems, the recall will be one of the most important metrics during experimentation.

Precision (smooth) Shows how much of the lesion pixels are found with a value between zero and one, where a value of zero is that there are no true positives and a value of one if there are no false positives. The

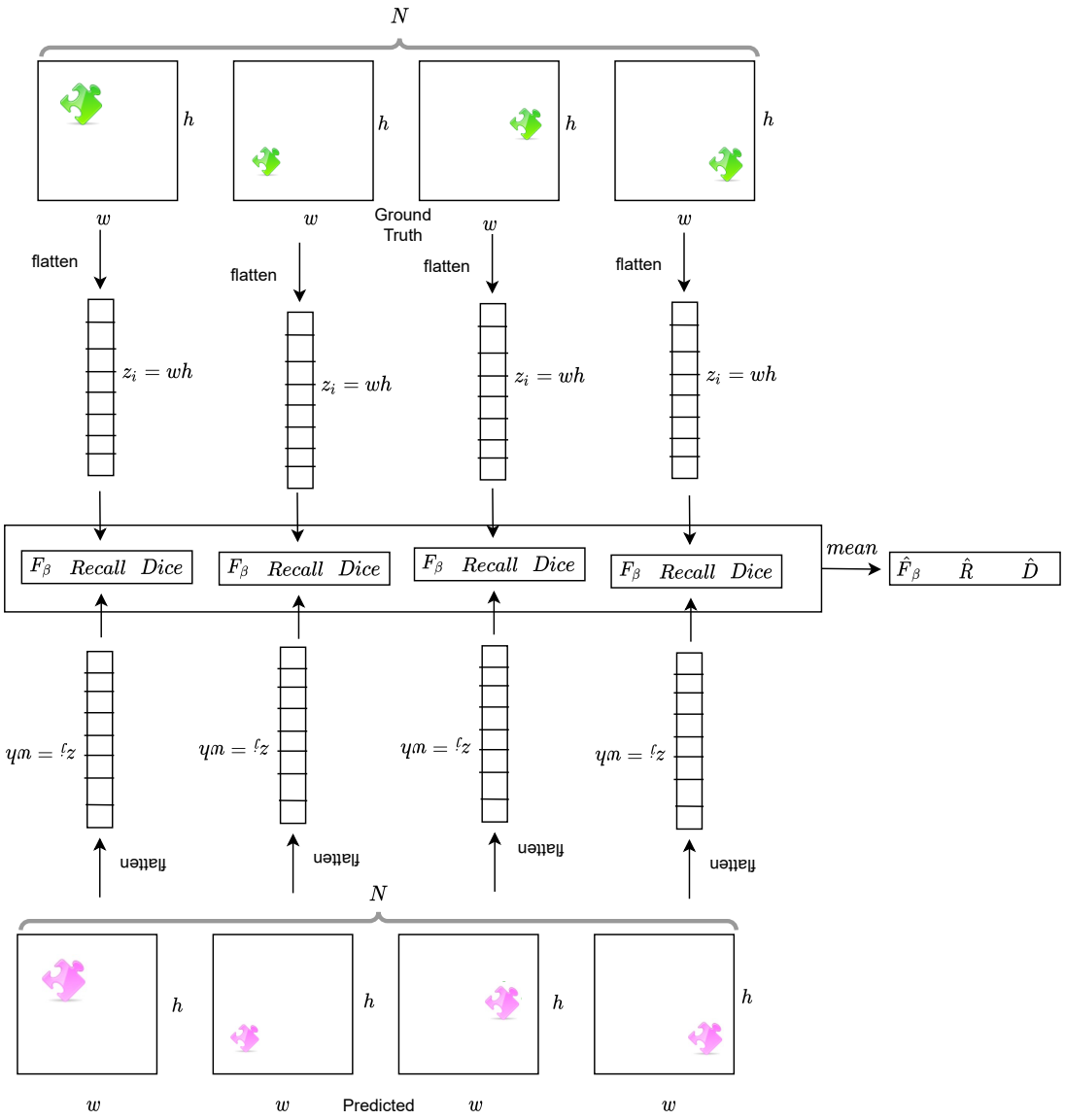


Figure 4.1: *In this case, the very small lesions will dominate the metric score as they are much more frequent. N =mini-batch size, z_i one-dimensional array of one ground truth sample, z_j one-dimensional array of one predicted sample, w is the width of the mask, and h is the height of the mask.*

smoothed precision score can be found in Equation 4.3:

$$P = \frac{TP + \Omega}{TP + FP + \Omega}. \tag{4.3}$$

Dice (smooth) One of the most commonly used metrics in segmentation is the dice score. The metric is the harmonic mean between precision and recall and can be defined as shown in Equation 4.4).

$$D = \frac{2TP + \Omega}{2TP + FP + FN + \Omega}. \tag{4.4}$$

Generalized F-score In many cases, recall is more important than precision. The general F_β score can

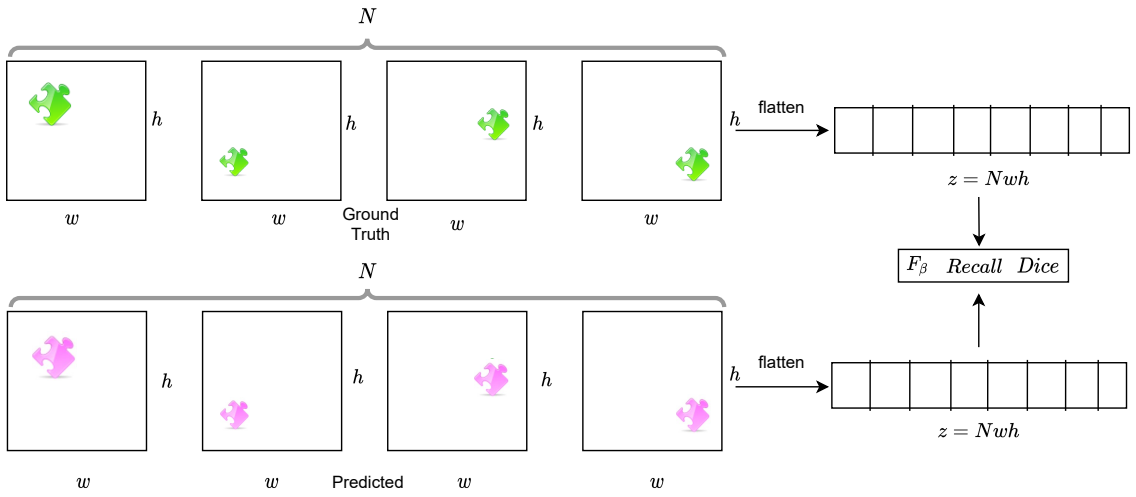


Figure 4.2: *This was the method used in this thesis. In this case, the larger lesions will have more impact on the metric score if it is inside the mini-batch. N = mini-batch size, z one-dimensional array of the whole flattened mini-batch. w is the width of the mask, and h is the height of the mask.*

add importance to either recall or precision. This metric is shown in Equation 4.5:

$$F_\beta = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + FP + \beta^2 FN}, \tag{4.5}$$

if $\beta > 1$ recall is weighted more than precision, and if $\beta < 1$ precision is weighted more than recall. If we want the recall to be two times as important as precision, we have F_2 score, and if we want it to be three times as important, we have F_3 and so on. If $\beta = 1$ the generalized F-score turns into the non-smooth dice score.

4.2 Loss functions

Medical image segmentation, in most cases, suffers from an imbalance in the dataset, for example, much more negative examples than positives. Another imbalance is that the backgrounds and foregrounds are much larger than the lesions, as shown in Figure 4.3, seen as white dots. As shown in the figure, both the background and the brain tissue not having lesions is about 98% of the 2D slice. Classical loss functions such as entropy would create large gradients towards predicting the massive background and brain tissue correctly.

Dice Loss A very popular loss function for image segmentation is the Dice loss[34]. Dice loss is a loss that uses the harmonic mean between precision and recall, which prevents large backgrounds from overwhelming the gradient. The smooth dice loss is found in Equation 4.6:

$$DSCL_c = \left(1 - \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \Omega}{\sum_{i=1}^{N} p_{ic} + g_{ic} + \Omega}\right), \tag{4.6}$$

where for the binary problem $g_{ic} \in \{0, 1\}$ and $p_{ic} \in \{0, 1\}$ represents the ground truth and the predicted label, respectively. The N is the total number of pixels. If we have an image segmentation problem that has a very small ROI(region of interest) we might need to weigh recall(false negatives) more.

Tversky Loss and focal loss A loss function that deals with large data imbalance is the Tversky focal loss[26][35]. This loss function is derived from the generalized dice loss called the Tversky loss. The focal loss

adds an exponential parameter that causes the gradient in either the lower or higher end of the loss to saturate. By choosing the parameter in such a way that the higher end of the Tversky score is flattened, will saturate the gradient for the easy/already understood examples in the training data. Hence, focus more on the harder examples. In [26] they introduced a novel focal Tversky loss, where a parameter γ has been introduced to the Tversky loss [36]. This parameter prevents the easy negative examples to dominate the gradient by focusing on harder examples by setting $\gamma > 1$ as shown in Figure 4.4. The Tversky similarity index is defined as shown in Equation 4.7:

$$TI(\alpha, \beta, \Omega) = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \Omega}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} p_{i\bar{c}} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} g_{i\bar{c}} + \Omega}, \tag{4.7}$$

where p_{ic} is the probability that pixel i is of positive class c and $p_{i\bar{c}}$ is the probability pixel i is of the non positive class $\bar{c}$, the same for g_{ic} and $g_{i\bar{c}}$, respectively.
This can be simplified for binary classification as shown in Equation 4.8:

$$TI(\alpha, \beta, \Omega) = \frac{TP + \Omega}{TP + \alpha FN + \beta FP + \Omega}. \tag{4.8}$$

Some 2D slice examples have small lesion sizes (down to one pixel). These examples should not necessarily give high loss because they are not important clinically and may cause problems with the generalization. As previously mentioned, an example of small lesions are shown in Figure 4.3. Therefore Ω can also act as a

Figure 4.3: *In these 2D slices small lesions (bright white spots) are pointed our by the red arrows. The small ROI can be hard to segment.*

regularizer. For example, if the model predicts zero positives, while the ground truth has one positive lesion pixel, the output of the Tversky index is then as shown in Equation 4.9:

$$TI_{\Omega=0} = \frac{0 + 0}{0 + \alpha + 0 + 0} = 0. \tag{4.9}$$

This yields a high gradient and maximum loss, but this is not necessarily good since predicting one single pixel is not very important. Changing Ω to two gives the Equation 4.10.

$$TI_{\Omega=2,\alpha=0.7} = \frac{0 + 2}{0 + 0.7 + 0 + 2} = \frac{2}{0.7 + 2} \approx 0.74. \tag{4.10}$$

We can see that this is very different. Here we have a very high Tversky index score instead, which is more appropriate. Using the smoothing factor will also fix the dividing by zero limits if all scores are zero. Hence, when the numerator and denominator metrics are all zero, yield a Tversky score of one and a loss of zero. In

most medical problems, we are more interested in the least FP(false positives) as possible, weighting this is done with the α parameter. To obtain the loss function, we subtract the Tversky index from 1, which is shown in Equation 4.11:

$$TI_{loss} = 1 - \frac{TP + \Omega}{TP + \alpha FN + \beta FP + \Omega}. \tag{4.11}$$

To obtain the focal loss we add γ as an exponent as shown in Equation 4.12:

$$TI_{floss} = \left(1 - \frac{TP + \Omega}{TP + \alpha FN + \beta FP + \Omega}\right)^{\gamma}. \tag{4.12}$$

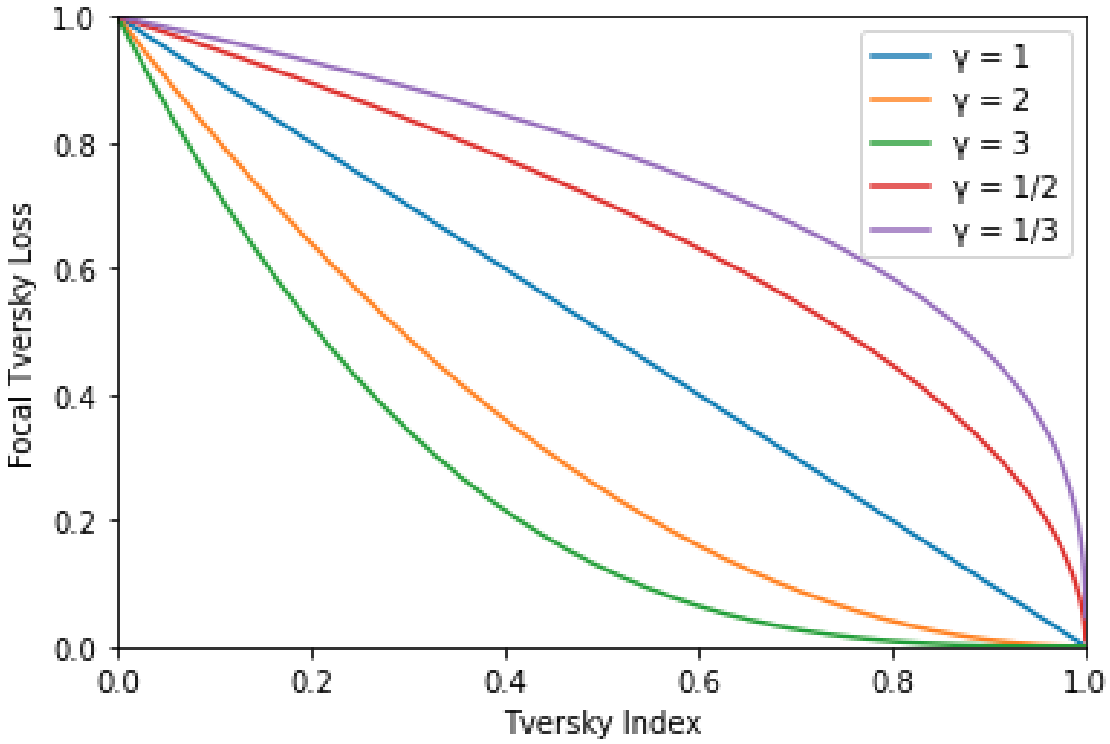


Figure 4.4: *Loss values for different values of γ. The focal loss with $\gamma < 1$ focuses more on the easier examples, as the gradient is higher when the Tversky index > 0.5. If $\gamma > 1$, the gradient is higher for the lower Tversky index and, therefore, focuses more on the harder examples. One thing to notice is that by having a $\gamma > 1$, the gradient towards the Tversky index > 0.8 is starting to saturate. This can help the model from overfitting and help in generalization.*

This leads to one of the derivatives for backpropagation shown in Equation 4.13:

$$\frac{dTI_{floss}}{d\hat{y}_i} = \frac{dTI_{floss}}{dTI}\frac{dTI}{d\hat{y}_i} = -\gamma(1 - TI)^{\gamma - 1}\frac{dTI}{d\hat{y}_i}. \tag{4.13}$$

The derivative of the loss function with respect to the sigmoid activation is smaller if TI increased by Ω.

5 Dataset and preprocessing

In this section, we explain the dataset and the preprocessing steps used.

5.1 Datasets

The dataset contains 100 MRI volumes, where domain experts have annotated ground truth masks for the MRI volumes. Most of these masks have very few positive pixels creating a severe class imbalance. The size of the 2D slices is 256x256, and they have channels FLAIR and T1. The data is separated into training, validation, and test[37][p. 222], using NumPy functions *shuffle* function[1] with random seed 123.

In this thesis, we have two datasets. One is the small dataset, which we obtained early during experimentation. The first experiments used data from one scanner, *Philips Ingenia* where MRI volumes have a voxel size of $0.954\,\mathrm{mm}^3$. The other experiments used a larger dataset that has MRI volumes from many different scanners and hospitals, which added different resolutions to the dataset. These scanners are shown in Table 5.1.

Table 5.1: *Scanner models for the large dataset.*

Data splits	Skyra	Prisma	Ingenia	Optima MR450w	Achieva	Avanto	Avg. voxel size
Training	51	156	216	85	57	14	$0.863\,\mathrm{mm}^3$
Validation	7	35	50	13	11	5	$0.856\,\mathrm{mm}^3$
Test	7	38	48	20	11	4	$0.865\,\mathrm{mm}^3$

Since there are different resolutions in the large dataset resolution, the receptive field of the model will be able to look at the different contexts of different parts of the brain in exchange for less context inside a given area. The UNet model has skip-connections making the architecture have a fixed range of receptive fields. If the resolution is low, then more parts of the brain will be inside the receptive field. If the resolution is high, then fewer parts of the brain will be inside the receptive field in exchange for more information inside the small region of the brain.

5.2 Statistics of data

Small dataset The mean and standard deviation of a sample of the training data is shown in Figure 5.2 and the validation data in Figure 5.3. One problem with the small dataset is that validation and test data are very small after performing a 70,15,15 split.

In Table 5.3 some important lesion metrics are shown. These metrics will be used in comparison with predicted lesions. These metrics are important because it is not generally the pixel predictions themselves who

[1] https://numpy.org/doc/stable/reference/random/generated/numpy.random.shuffle.html

are important, but the size of the 3D lesion. The lesions are grouped as discussed in Chapter 7.5, where we use pixel connectivity to group the pixels into groups of 3D lesions.

All the 100 MRI volumes in the small initial dataset have been scanned by the same machine of type *Philips Ingenia* which can cause generalization issues discussed in Chapter 9. In Table 5.2 information about the dataset is shown. There are no MRI volumes that have zero lesions, but three MRI volumes in the training data have below 100 pixels, which is very little. Figure 5.1 describes the distribution of the number of WMH-identified pixels (by the experts) per image slice across the entire dataset for each volume.

Table 5.2: *Data information for the small dataset splits. In this table, we can see how many 2D slices have WMH pixels.*

Dataset	Total Number of 2D slices	Masks $\geq$ 100 WMH	Empty masks	Masks $<$ 100 WMH	Volumes with zero lesions
Training	11712	1297	8700	3	0
Validation	2562	324	1851	0	0
Test	2562	335	1863	0	0

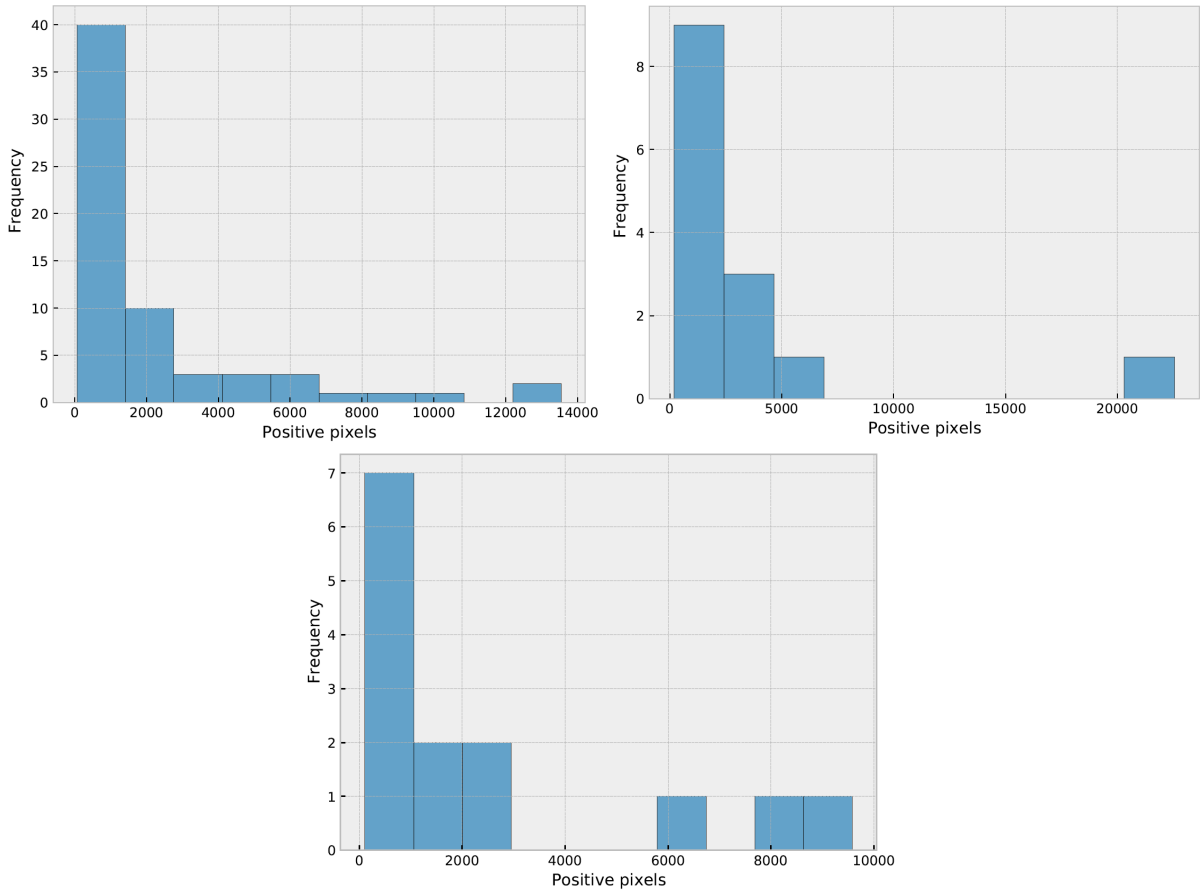
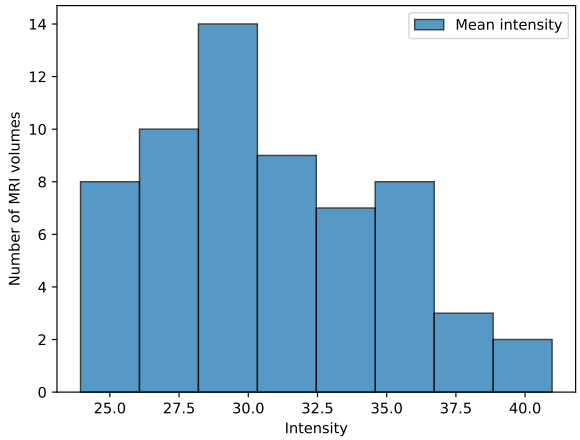
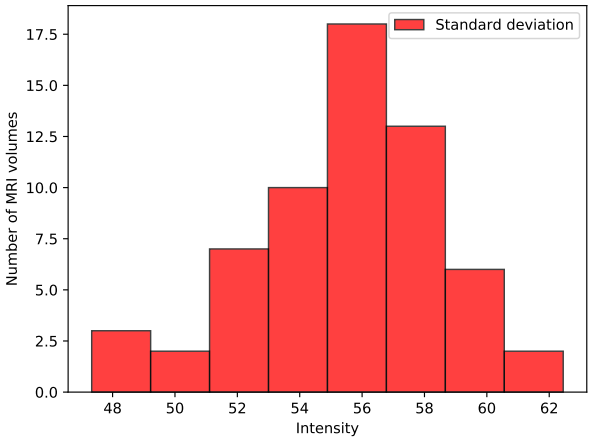


Figure 5.1: *(a), (b) / (c). Distribution of lesion pixels for each MRI volume. (a) The training dataset. (b) Validation dataset (c) Test dataset. 64 MRI volumes in training, 15 MRI volumes in the validation dataset, and 13 in the test dataset. The frequency is in MRI volumes, not 2D slices.*

Large dataset This dataset contained MRI volumes from many different scanners and hospitals. Using

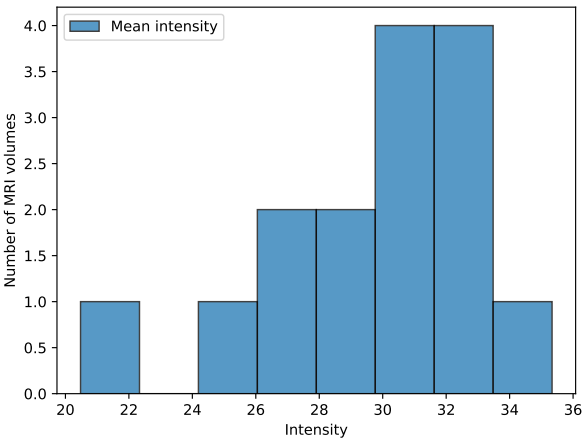


(a) *Mean values of samples from the small training data.*

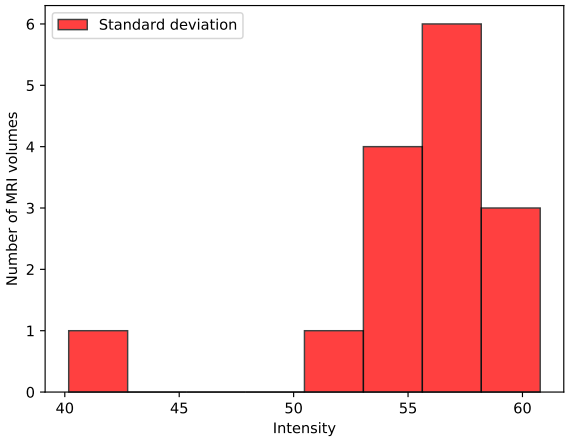


(b) *Standard deviations from the small training data.*

Figure 5.2: *The mean and standard-deviation of the small training data with 60 samples is shown. The standard deviation seems to almost follow a gaussian distribution. The same statistics are plotted in Figure 5.3, but with the validation data instead.*



(a) *Mean values of small validation data.*



(b) *Standard deviations of the small validation data.*

Figure 5.3: *The mean and standard deviation of the small validation data. The validation data is very small at $N = 9$ after removing MRI volumes not containing FLAIR and T1, so comparing this to the small training data is not that significant. There seems to be at least one outlier in the standard deviation of this sample.*

Table 5.3: *Ground truth lesion metrics for the small validation dataset. Depths and voxels are averages over total lesions.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	52	566	3960
Avg. depth	1	6	28	42
Total	555	189	9	7

a more diverse dataset helps with the generalization of the model. Different scanners have their own noise distributions and different imaging settings like magnetic strength etc. Lesion Information for the large dataset is shown in Table 5.4. Figure 5.4 show the sum of all positive pixels in the MRI volumes masks in the new dataset. In Figure 5.5 a sample of 60 MRI volumes in the training data is shown, and in Figure 5.6 the sample of 60 MRI volumes from the validation data is shown. Comparing these distributions, we can see that the standard deviation is much larger than in the smaller dataset. The reason for this is that the larger dataset has many MRI volumes containing much more noise and more/fewer background values. In Figure 5.8 and Figure 5.7 some 2D slices are shown with and without color sliced intensities. We can see that it is hard to see the low-intensity noise in the original image, but after color slicing, we see that values in the [1,19) intensity range contain a lot of noise in some MRI volumes. The noise can add generalization to the model as the model need to adapt to the disruption of the intensity values in the region of interest.

Table 5.4: *Data information for the small dataset splits. In this table, we can see how many 2D slices have WMH pixels.*

Dataset	Total Number of 2D slices	Masks $\geq$ 100 WMH	Empty masks	Masks $<$ 100 WMH	Volumes with zero lesions
Training	132070	27946	80303	0	0
Validation	27274	5996	16331	0	0
Test	28425	5525	17468	0	0

Table 5.5: *Ground truth lesion metrics for the large validation dataset. Depths and voxels are averages over total lesions.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	44	631	3817
Avg. depth	2	6	23	42
Total	10436	4210	116	133

Image presentation Domain experts like to view the 2D slices in the axial orientation as shown in Figure 4.3. For the model training, the orientation of the input data is not important if the data has isotropic resolution; that is, the same resolution along all three axes as was the case for the data available here. After prediction, the output could then be viewed along any chosen plane through the 3D volume making up each MRI dataset.

In experiments, we explore the axial and sagittal orientations to see if it has any effects on prediction. If the dimensions become uneven or less than 256 pixels, zero padding is needed, which may affect the model. The most important part of the prediction is that the input is in the same way as it was during training and validation.

How the volume is stacked together from 2D slice predictions are shown in Figure 7.4.

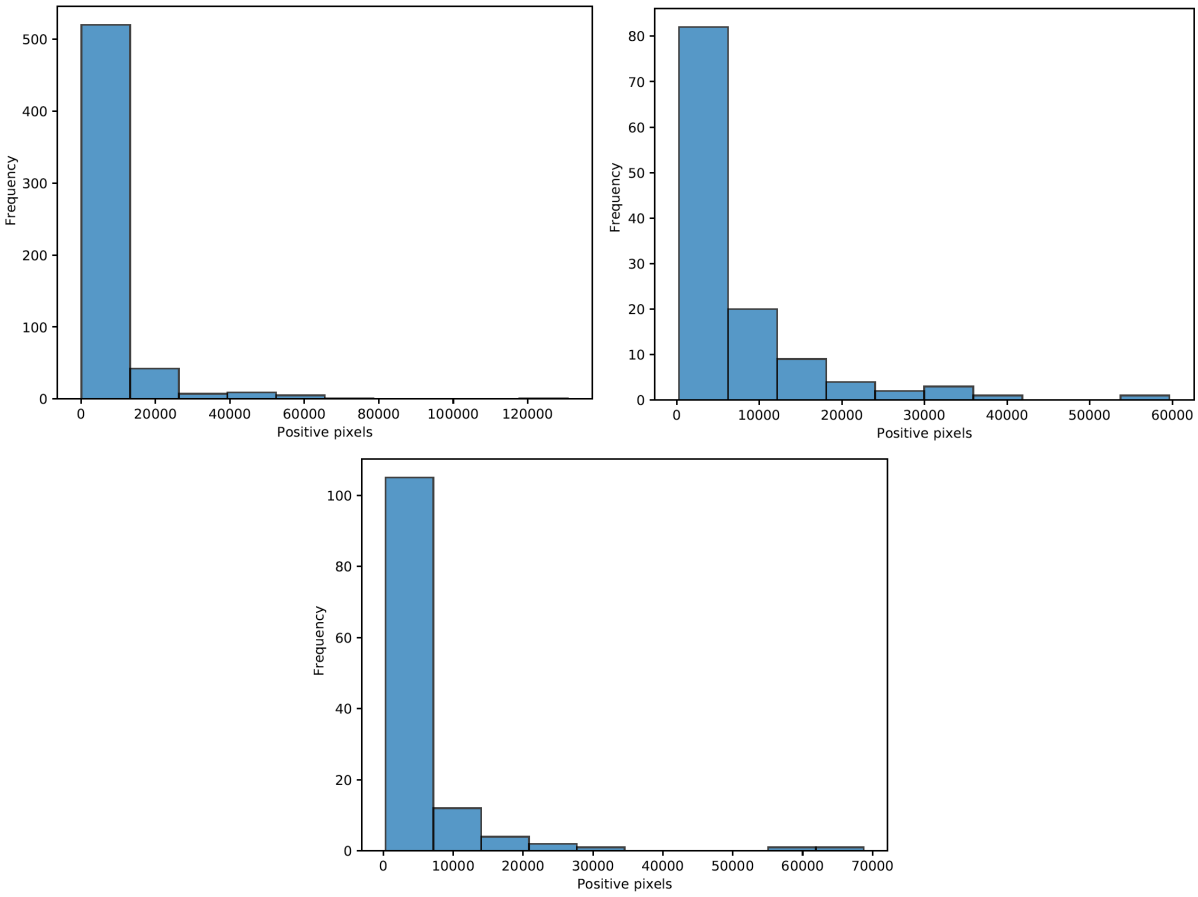


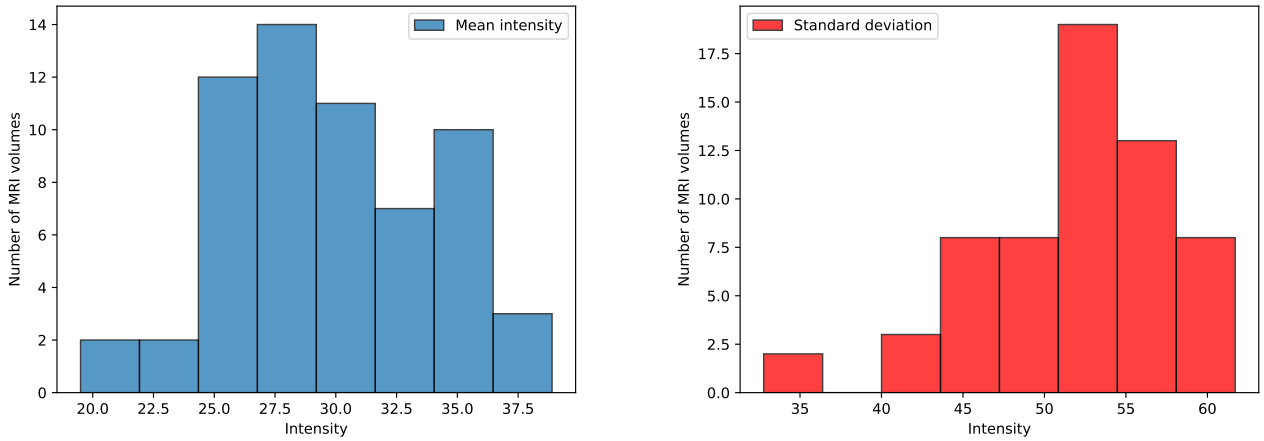
Figure 5.4: *(a), (b) / (c). Distribution of positive pixels(lesion pixel) for each MRI volume in the large dataset. (a) The training dataset. (b) Validation dataset (c) Test dataset. The frequency is in MRI volumes, not 2D slices.*

5.3 Pre-processing

The general intensity values can be quite different from volume to volume because of noise, magnetic field strength, and others. Because of this, we need to standardize the dataset in a way such that these factors do impact the model as little as possible. **Z-standardization** scale the values such that the values are between [-1, 1]. Since the lesions usually are the brightest parts of the image, they will end up around one, mean values will end up at zero, and the background values will end up at around a negative one. As pre-processing the dataset is **z-standardized** over the whole MRI volumes independently from other volumes using Equation 5.1:

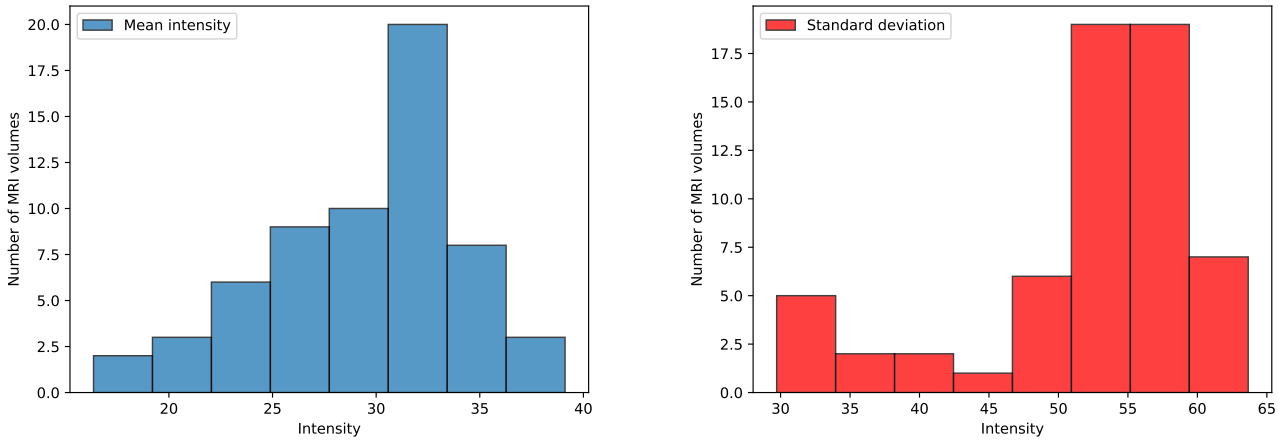
$$Z_k = \frac{\boldsymbol{x}_k - \mu_{\mathcal{V}}}{\sigma_{\mathcal{V}}}, \tag{5.1}$$

where $\mu_{\mathcal{V}}$ is the mean of MRI volume, $\sigma_{\mathcal{V}}$ is the standard deviation of the MRI volume, $\boldsymbol{x}_k$ is the MRI volume slice $x_k \in \mathcal{R}^{M \times N}$, M being the height of the slice, N is the width, and k is the depth index for volume depth K. This operation will convert the data into *intensites from the mean* and yield a zero mean and a standard deviation of one over the whole MRI volume. Having data with zero mean and normalized standard deviation could help with convergence[16]. This is the pre-processing step used in all experiments. For the training data,



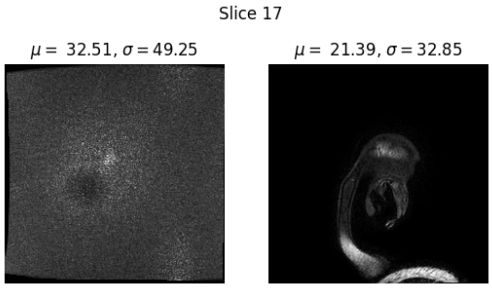
(a) *Mean values of training data.* (b) *Standard deviations of the training data.*

Figure 5.5: *The first and second-order moment of the large training data. This dataset has much more data allowing for 60 samples in the validation data as well.*

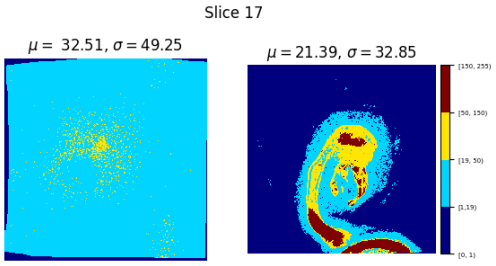


(a) *Mean values of validation data.* (b) *Standard deviations of the validation data.*

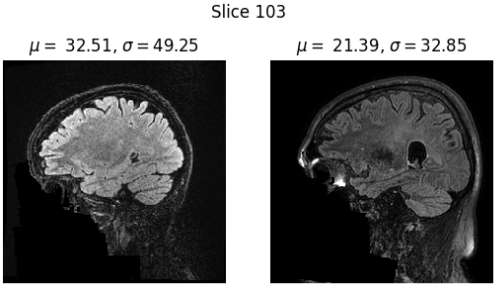
Figure 5.6: *The first and second-order moment of the large validation data.*



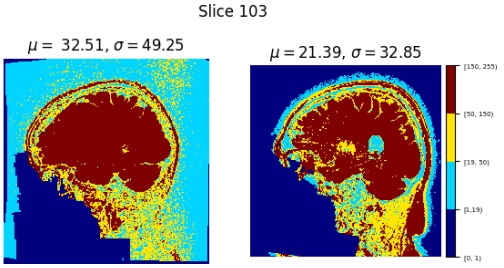
(a) *Gray scale 2D slice from validation data. Left: 2D slice example with a lot of noise. Right: 2D slice from another volume.*



(b) *Color sliced 2D slice from validation data. Left: 2D slice from a volume. Right: 2D slice from another volume.*

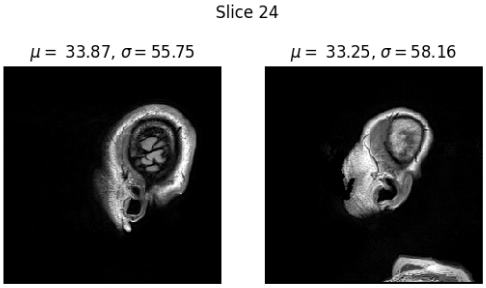


(c) *Gray scale 2D slice from validation data. Left: 2D slice from a volume. Right: 2D slice from another volume.*

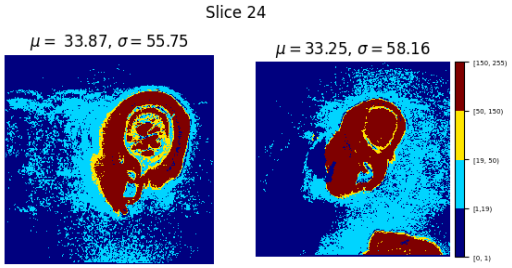


(d) *Color sliced 2D slice from validation data. Left: 2D slice from a volume. Right: 2D slice from another volume.*

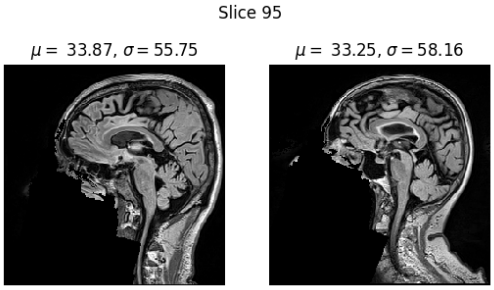
Figure 5.7: *Some volume seems to have a lot of noise bringing the mean and variance up. This skews the intensity values of the brain and can have an effect on the model. Note: The statistics shown as the subtitle is for the whole volume, not for that given slice.*



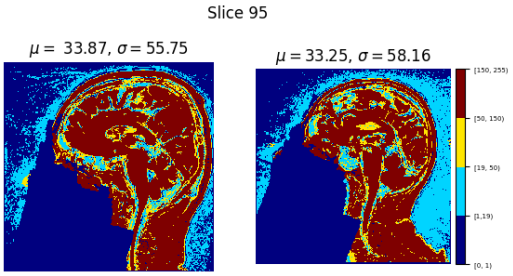
(a) *Gray scale 2D slice from validation data. Here are more examples of some 2D slices in the more commonly occurring percentile.*



(b) *Color sliced 2D slice from validation data. Here are more examples of some 2D slices in the more commonly occurring percentile.*



(c) *Gray scale 2D slice from validation data. Here are more examples of some 2D slices in the more commonly occurring percentile.*



(d) *Color sliced 2D slice from validation data. Here are more examples of some 2D slices in the more commonly occurring percentile.*

Figure 5.8: *There seem to be some noisy examples overall in the dataset. Noise is not necessarily bad as it could enhance the generalization of the model. Note: The statistics shown as the subtitle is for the whole volume, not for that given slice.*

we also used entropy measures to remove 2D slices containing small amounts of information.

5.4 Feature analysis

In this section, we show some basic feature analysis using t-SNE[38]. t-SNE is a statistical method for visualizing high-dimensional data. The algorithm converts the similarities between data points to join probabilities by minimizing the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. Since t-SNE is an optimization algorithm that used a non-convex loss function, a different initialization will yield different results. For these tests, we kept the NumPy random seed at 123 as we have done for all experiments.

In Figure 5.9 one example of the reduced features based on the slice-orientation is shown. The dimensionality have been reduced from $\boldsymbol{x}_i \in \mathcal{R}^{256x256} \rightarrow \boldsymbol{y}_i$, where $\boldsymbol{y}_i$ is in the **map space** $\mathcal{R}^2$. There is an bijection in the reduced map space, hence, every point represents one slice in the volume.

It is interesting to note that the sagittal and axial orientations have very different feature clusters. In the first column, the dimensionality-reduced results of 8-bit volume are shown. In the second column is the z-standardized slices, and in the last column, one example of the image presentation is shown.

As observed, the features reduce in different ways depending on the image orientation. The bubbles depict the lesion sizes. The larger bubbles are the larger lesions. The most amount of lesion pixels in a slice was 124 pixels. In axial orientation, it seems like the lesions group up in a leaf-like structure, while in the sagittal slice, the lesions group up in one given area.

In Figure 5.10 one example with almost no lesions are shown together with two random volumes. The z-standardization seems to help the algorithm to separate the clusters. Since we are able to separate the lesions into their own clusters, we hypothesize that we are able to predict the lesions with fairly good accuracy and that the axial slice orientation is the best.

The lesions seem to stay within the *leaves* which might suggest that the lesions occur in distinct parts of the brain. The leaf patterns also look very alike for the respective image orientations. These patterns suggest that there is a pattern in how the lesions are formed, and that the deep learning model might pick this up.

For all feature reduction we used all the default values from the sci-kit learn T-SNE function at [2], where the main parameter perplexity = 30. Even though we see patterns for the lesions, it might not mean anything special as the interpretation of T-SNE is quite complex. [3]

[2]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html - 09.04.2021
[3]https://distill.pub/2016/misread-tsne/ - 09.04.2021

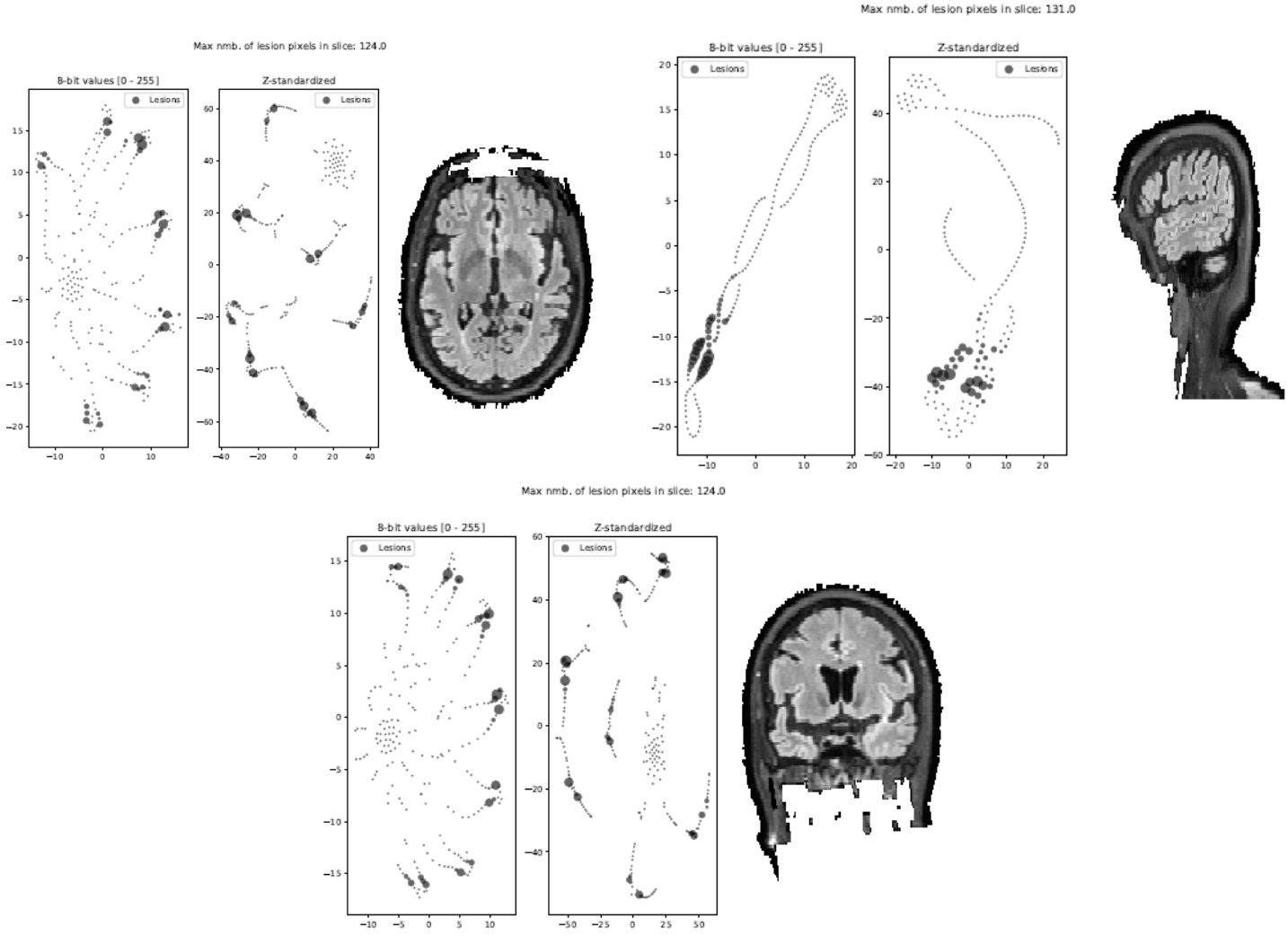


Figure 5.9: *(a)(b)/(c). Features per slice from the different orientations. In each plot: (a) (b) and (c) different orientation of the same volume is shown. T-SNE can be used to reduce the dimensions of data to make it easier to interpret the data. In this example, each slice was reduced from 256x256 to 1x1 point in 2D space. This was done overall slices and added to the final 2D plot as shown. The bubble sizes follow the sum of lesion pixels in that given 2D slice. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

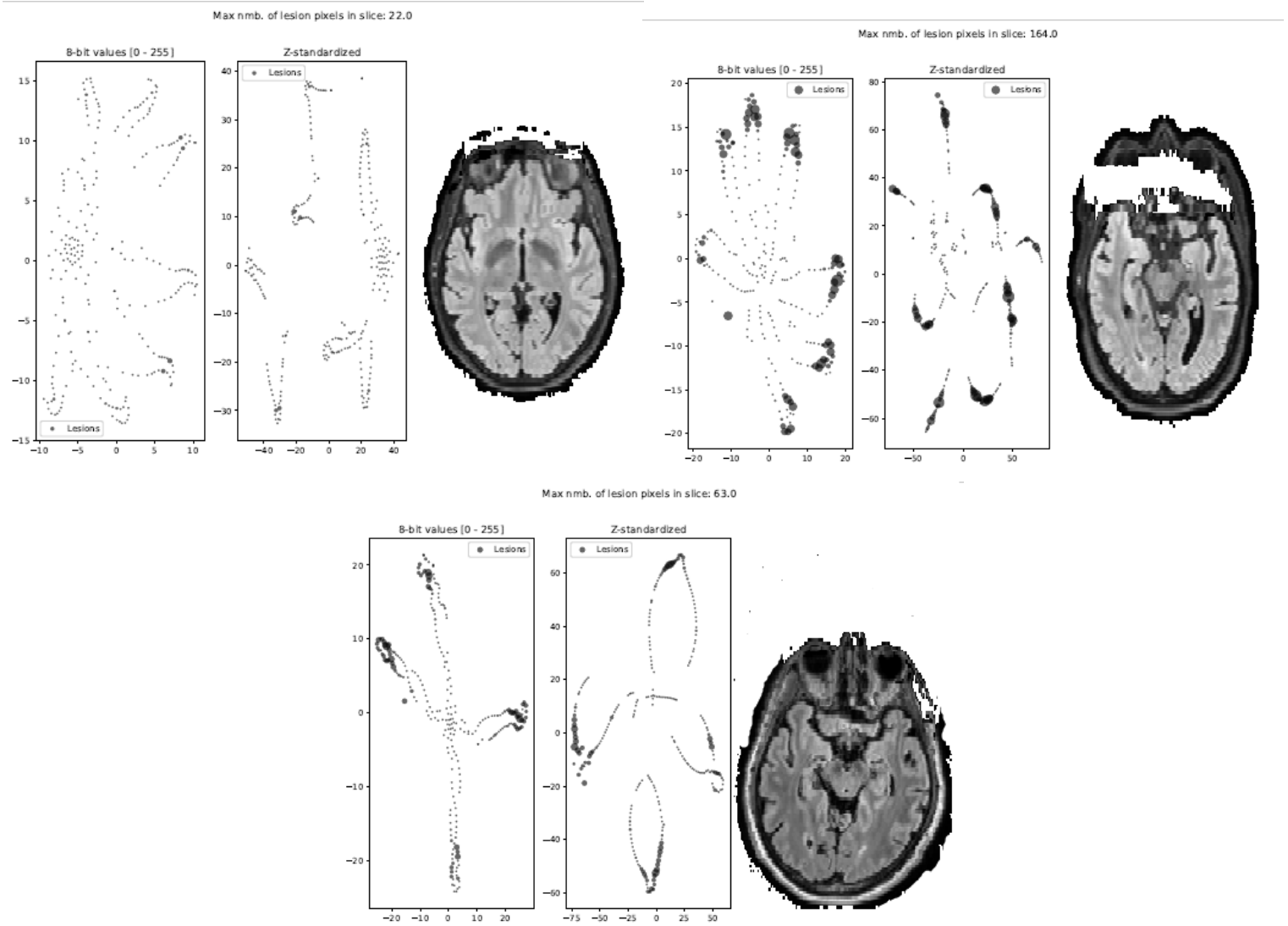


Figure 5.10: *(a)(b)/(c). An example with very few lesion pixels is shown in (a). In (b) and (c) two random volume examples are shown. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

6 Models

In this section, we explain the architecture of the models used during experimentation.

The models are built with modules of layers usually comprised of convolution layers, activation layer, normalization layer, and pooling layers. There can also be some type of regularization layer, such as dropout. One of the most popular segmenation model is the UNet[3].

6.1 UNet

In 2015 Olaf Ronneberger, Philip Fischer, and Thomas Brox introduced the U-Net architecture for medical image segmentation[3]. This architecture is good at localizing by using the elegant concatenated **skip-connections** for each down-sampling. **Skip-connections** help with gradient flow between the layers, as the gradient may saturate in deep and large networks. The network architecture is shown in Figure 6.2. The original architecture does not pad the layers, decreasing the size of the output map. In the deepest layer, the feature maps are 32×32 when the input images are 572×572. In our architecture, we use zero paddings to get the same output as the input size after feeding it to the model, even though padding affects the network[39]. We also used 256×256 input images to get feature maps of 16×16 in the deepest layer.

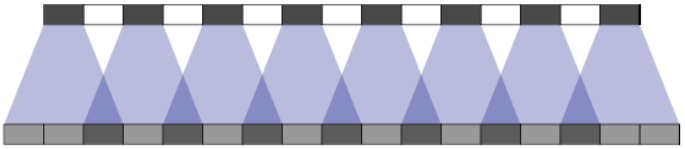
The architecture used in our experiments has half the feature map size compared to the original. In our experiments, we use 256x256 images, while in the original paper, it was used 512x512. Feature map size in our experiments are [32, 64, 256, 512], and then up-sampled using transposed convolutions through the decoder part of the architecture. The last layer was changed from softmax to sigmoid for probability since we have a binary classification problem. The kernel size for all the convolution layers are 3x3 with stride = 1, padding = 1 and dilation = 1. The last convolution before sigmoid has 1x1 with stride = 1 padding = 1, dilation = 1 and bias = False. The downsampling was done using max-pooling with stride = 2 and kernel size = 2, cutting the resolution in half.

The contracting part of the network finds the features, and the expanding part localizes the area to segment in the final output. For the up-sampling, we could either employ bilinear interpolation or learnable up-sampling/ transposed convolutions. In our experiments, we used transposed convolutions for upsampling. One problem with transposed convolutions is that the convolved output can have a checkerboard pattern. Figure 6.1 show a checkerboard pattern when using unequal stride to kernel size.

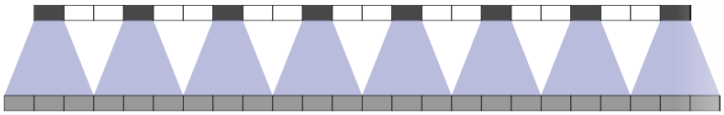
6.2 Attention UNet

The attention UNet[41] attempts to identify the salient regions and to prune the features to preserve only the activations relevant to the specific task, which in this case is low ROI segmentation. The architecture is seen in

Figure 6.1: *Transposed convolutions with different size and stride. Figures are from [40].*



(a) *Transposed convolution with stride = 2 and size = 3. Double overlap causes a checkboard pattern in 2D.*



(b) *Transposed convolution with stride = 3 and size = 3. No overlap between nearby upsampled pixels.*

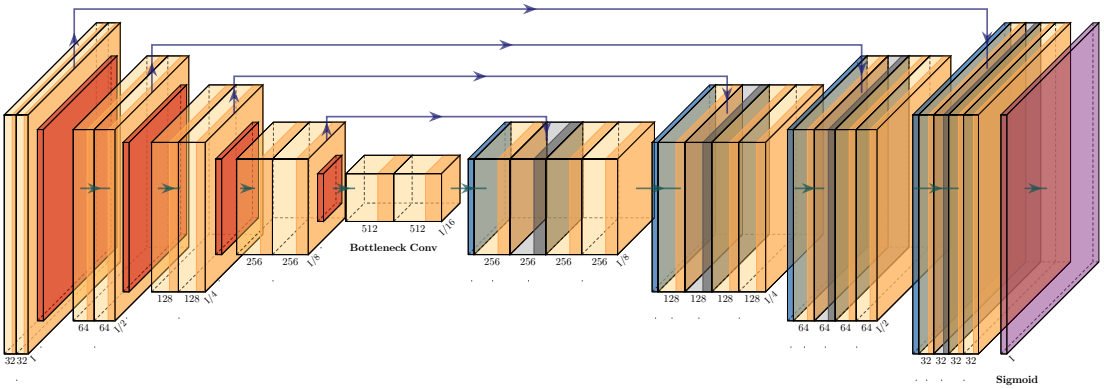


Figure 6.2: *The UNet model used in our experiments. The original layer had 1024 feature maps in the bottleneck, but since we work with 256x256 2D slices, we decreased the feature maps in the bottleneck to 512 feature maps.*

Figure 6.3. In this figure, the Attention UNet with pyramid input is shown. The same parameters were used as in the regular UNet, except for the attention gates and in some experiments the pyramid input.

The pyramid downsampling was done using max-pooling with stride = 2 and kernel size = 2 on the input image. The image pyramid is an attempt to improve segmentation accuracy since the small ROI features can get lost in cascading convolutions. The attention gate diagram is shown in Figure 6.4.The attention coefficients can be formulated as shown in Equation 6.2:

$$q^i_{att,i} = \psi^T(\sigma_1(\boldsymbol{W}_x^T \boldsymbol{x}_i^l + \boldsymbol{W}_g^T \boldsymbol{g} + \boldsymbol{b}_{xg})) + b_\psi, \tag{6.1}$$

$$\alpha^l = \sigma_2(q^l_{att}(\boldsymbol{x}^l, \boldsymbol{g} : \Theta_{att})), \tag{6.2}$$

where Θ_{att} is the set of parameters containing the linear transformations $\boldsymbol{W}_x \in \mathbb{R}^{F_l \times F_{int}}$, $\boldsymbol{W}_g \in \mathbb{R}^{F_g \times F_{int}}$, $\boldsymbol{\psi} \in \mathbb{R}^{F_{int} \times 1}$ and the bias terms, $\boldsymbol{b}_{xg} \in \mathbb{R}^{F_{int}}$ and $b_\psi \in \mathbb{R}^{F_{int}}$

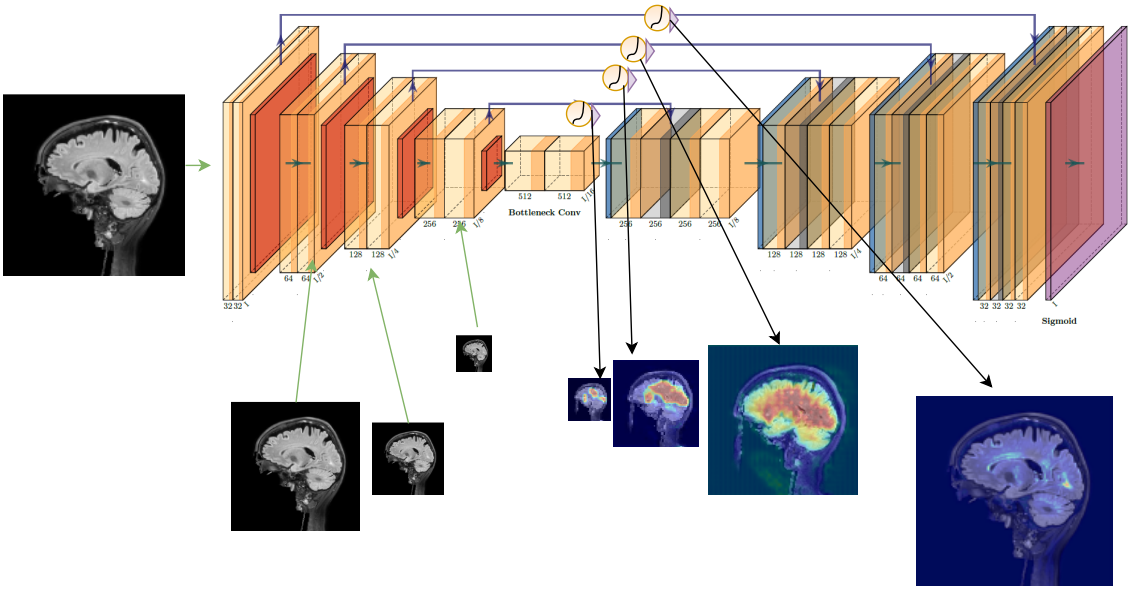


Figure 6.3: *Model of attention UNet. The circles in the skip connection are the attention gates. These attention modules output an attention map which is multiplied with the feature maps to learn self-attention. The architecture is the same as UNet, the only difference is the attention gates.*

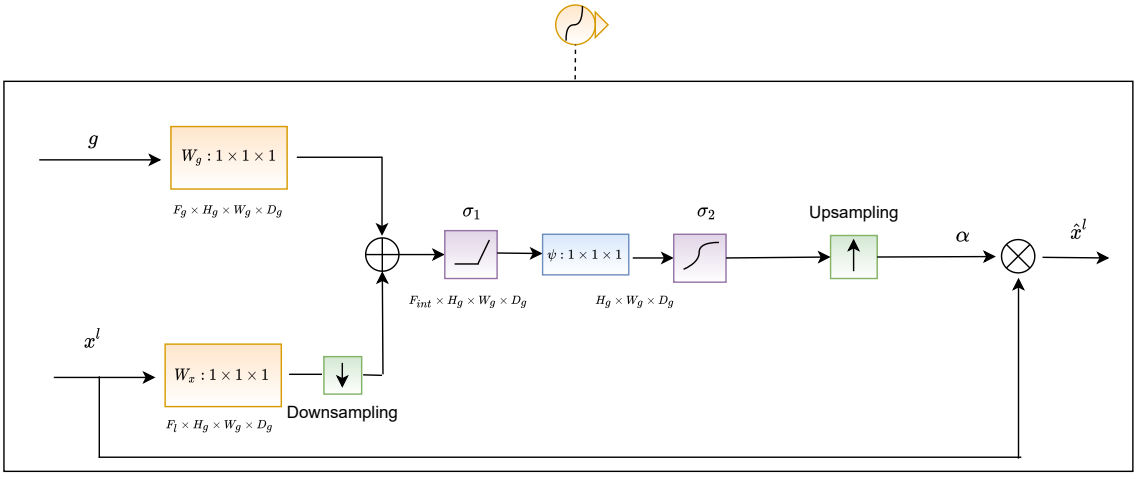


Figure 6.4: *The architecture of the attention gates used in the attention UNet. The gating signal g coming from the coarser scale feature maps is added together with the input signal x^l, which is the skip connection coming from the higher spatial resolution. This input is down-sampled to be able to sum it with the gating signal. After going through the ReLU, σ_1, 1x1 mapping ψ, and sigmoid σ_2, the output is up-sampled using a bilinear. The attention weight map α is multiplied with the skip connection to create the attention weighted output map.*

6.3 3D UNet256 alternative - 3 slices as channels

The dataset is made of MRI volumes, therefore, it is logical to think of 3D neural networks as a way to optimize the objective. One of the big problems with 3D neural nets is the immense memory usage. Typical volumes in the dataset consist of 256x256x183 float value (when z-standardized) plus x amount of batch size, including

now 3D weight kernels in the architecture, which leads to an explosive amount of memory usage.

One way to lessen the usage of memory is to use patches of the 3D volume during training. After prediction, the output can be stitched back together, creating the original volume. This patching method could lead to the creating of edges across lesion volumes, by, for example, if the patching is done in between a lesion, etc. This might lead to a sub-optimal understanding of the lesion features.

Using this method still uses a lot of memory. In most of the experiments, the channels were FLAIR and T1, but we could instead use either one slice from all three orthogonal orientations or stack three or more slices in one orientation to get depth information. Here we can stack, for example, three vicinity slices and predict a segmentation map for the center slice. The model can use one down and one up information for a given orientation for prediction as shown in Figure 6.5. This method does not need any alterations to the architecture either.

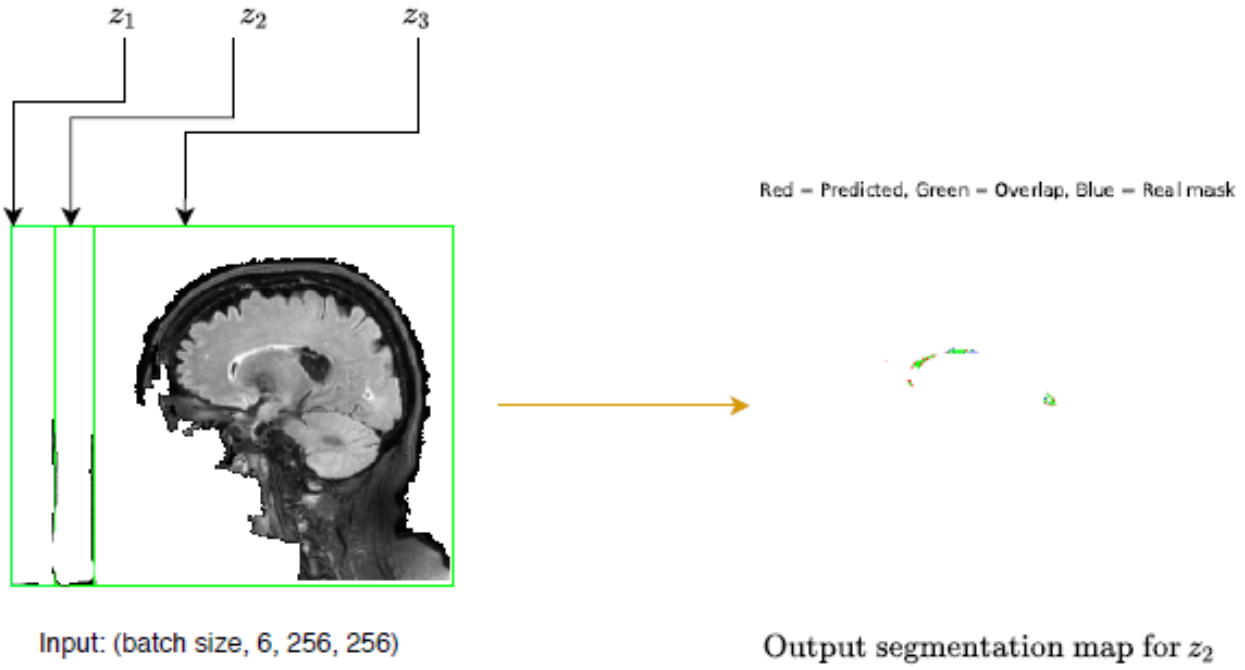


Figure 6.5: *Using three slices as channels, where we want to predict a single prediction map for the center slice. If we use three slices of FLAIR and T1, the channels would be six. The channels can be the concatenation of FLAIR(3x) and T1(3x). This method is a thickened 2D approach, not 3D since we only use information from one orientation.*

7 Methology

In this section, we explain the methods used during experimentation.

In this thesis, we have one small subset of data coming from one scanner, which is used in the first couple of experiments. Later, we received a large dataset coming from many different hospitals and scanners, which are used for the last experiments and used for the final results. This is also mentioned in the experiments. The dataset contains Fluid-attenuated inversion recovery (FLAIR) and T1 MRI volumes, which are both used as channels in some of the experiments. Since it is more practical to only use one of these, in the last experiments, the only FLAIR was used, which showed promise and were used for the final results.

White matter hyperintensities (WMH) can be hard to segment because most of the lesions can be very small. One example of a small to medium-sized lesions are shown in Figure 1.1. A loss function used in our experiment, Tversky focal loss[26], can help the model not overfit and to focus on both hard examples and false negatives, depending on how you tune the parameters.

7.1 Transfer Learning

Transfer learning is the method of using the pre-trained weights of a model trained on another dataset and using it on a new dataset with a different objective. Most low-level features extracted are the same across different categories. By using pre-trained weights, we leverage the power of an already trained model for prediction by making small tweaks to the weights in order to minimize the loss for the new objective.

Instead of training from scratch and the need for a massive training dataset, we might only need a small dataset and a couple of epochs in order to attain good results. One problem with using transfer learning is that the gradient might start off very small since it is already very close to the minimum. This will make our momentum low and might cause the optimization to end up in one of the local minima. Some experiments in this thesis are done using pre-trained weights from a Kaggle competition for lower grade glioma tumor segmentation on 2D MR slices.

7.2 Image Entropy

As described in [7][p. 545] "How few bits are needed to represent the information? That is, is there a minimum amount of data that is sufficient to describe an image without losing information in an image? Information theory provides the mathematical framework to answer this and related questions. Its fundamental premise is that the generation of information can be modeled as a probabilistic process that can be measured in a manner that agrees with intuition. Following this supposition, a random event E with probability $P(E)$ is said to

contain:

$$I(E) \log \frac{1}{P(E)} = -\log P(E), \tag{7.1}$$

units of information. If $P(E) = 1$ (that is, the event always occurs), $I(E) = 0$ and no information is attributed to it. Because no uncertainty is associated with the event, no information would be transferred by communicating that the event has occurred [it *always* occurs if $P(E) = 1$]" By using $\log_2$ we have the unit of information in a bit. If there is only one value across the whole image, the information is zero. Therefore for all experiments, the slices with the information above a threshold of two were used during training. This was found by visually inspecting the 2D slices around this value. Information content was found by using the Shannon entropy as defined in Equation 7.2. By removing images with low entropy, the slices-with-lesion to the slices-without-lesion ratio was decreased, this is good because of the large imbalance.

$$\hat{H} = -\sum_{k=0}^{L-1} p_r(r_k) \log_2(p_r(r_k)). \tag{7.2}$$

In Equation 7.2[7][p. 546] L is the intensity levels and $p_r(r_k)$ is the probability of that given intensity. The result is in bits/pixel. The entropy of all training MRI volumes is shown in Figure 7.1.

Entropy is high if there are many different intensities to describe an image. Hence, noisy 2D slices will have high entropy, where uniform distribution will have the highest. All data has the same depth $N = 183$. The red line indicates the threshold for removing 2D slices. In Figure 7.2 we can see some examples of 2D slices at the threshold border. Since this entropy line is almost the same for all MRI volumes, we can suspect that there is not much difference in the data. This makes sense as it later was discovered to be all from the same MRI scanner. Looking at Figure 8.18 the entropy is a lot more diverse.

In all experiments, a threshold of two was used. This was found by looking at the 2D slices directly by taking the value where brain tissue started to form. From Equation 7.2, the entropy is maximized if the intensities of the image follow a uniform distribution and are minimized if all intensities are the same.

Removing 2D slices under the threshold left us with 9421 training images and 2562 validation images. No images were removed in the validation set.

7.3 Training/Validation evaluation procedure

The loss and evaluation metrics have been calculated by first finding the loss and scores in a mini-batch (12 images). This loss is found by flattening the mini-batch and using Tversky focal loss. When evaluating the metrics like recall, dice, and F_3, the mini-batches are all flattened before the score is found. After all mini-batches for the epoch is complete, the final result is then the average over all the mini-batch step results. In Figure 7.3 the training procedure is shown. The metrics use a hard threshold of 0.5 to binarize the outputs from the sigmoid function before computing the score.

The dice and recall are only calculated for the validation set. During training, the weights are saved when the validation loss is at the lowest value.

7.4 Lesion metric evaluation volume prediction

When the best model is found, the weights are loaded, and the MRI volumes in the validation and/or test data are predicted by chunking each volume into batches. These batches are segmented and stacked into a volume

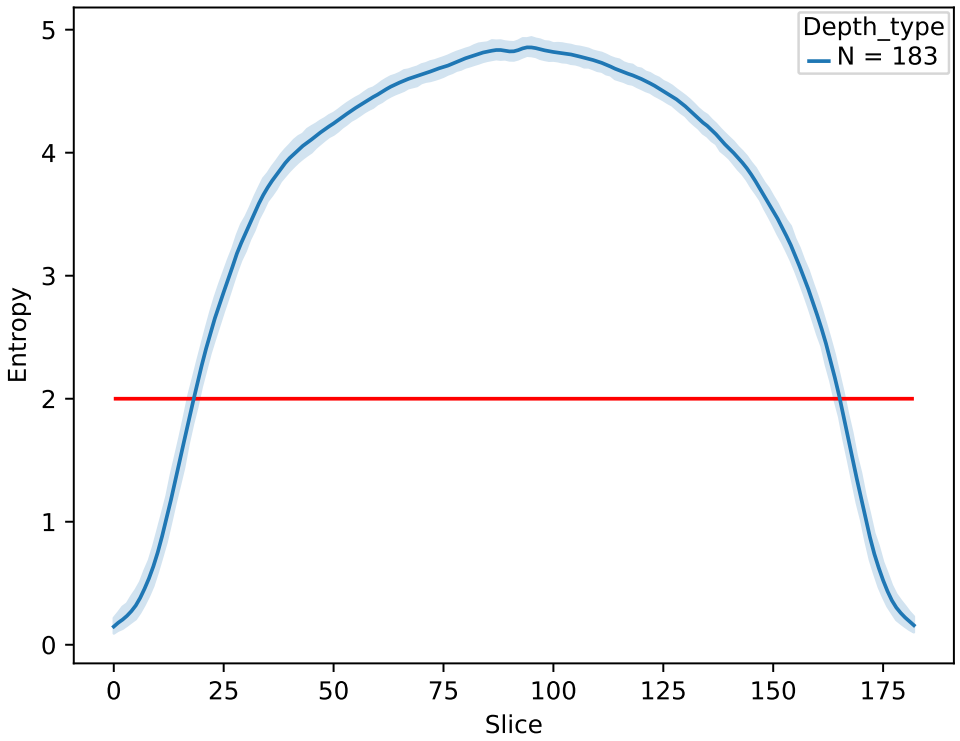


Figure 7.1: *Entropy content over MRI volumes over the slices in training data. The dataset has a very small standard deviation in information over slices. All slices/images above 2 bits/pixel were used. Here the maximum entropy is at around five bits/pixel.*

array of the size of the original FLAIR volume. This is shown in Figure 7.4.

7.5 Pixel connectivity (lesions)

For **lesion analysis** the functions *label*[1] and *regionprop*[2] from scikit-image library[42] has been used to group the pixels to lesion clusters. The label function uses pixel connectivity. This method groups pixels together if they connect in either 4 or 8 connection, some examples are shown in Figure 7.5. In this thesis, we used 8-connectivity. The grouping is done after the whole volume is predicted.

For different pixel values V={1,2,3,..M} or binary V = {1} we assign into one group if the pixels is 8-connected, $((x+1, y+1, z+1))$, $((x-1, y-1, z+1))$, $((x-1, y-1, z-1))$, $((x+1, y-1, z-1))$, $((x+1, y+1, z-1))$, $((x+1, y-1, z+1))$, $((x+1, y+1, z-1))$ and $((x-1, y+1, z-1))$

Regions It would be interesting to know how many lesion regions the model can detect, not just how many pixels of regions. We will use the function "label" from scikit-image[3] to group regions and count them. In Figure 7.6 there is an example of regions. Regions/Lesion counting is further discussed in Chapter 9.6.

[1]https://github.com/scikit-image/scikit-image/blob/master/skimage/measure/_label.py#L32-L120 - fetchdate: 11/01/2021
[2]https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops - fetchdate: 11/01/2021
[3]https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.label - fetchdate: 11/01/2021

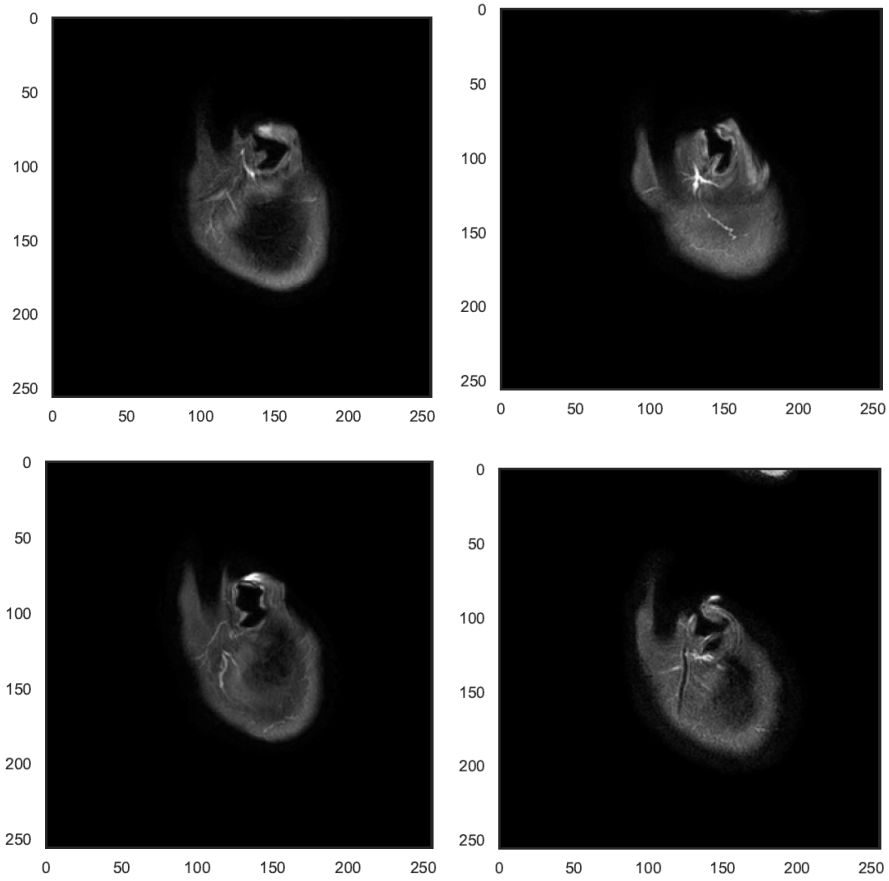


Figure 7.2: *Examples of 2D slices at the entropy threshold border. There is not that much information, but we do not want to remove too much either. The threshold at two seems reasonable. Each 2D slice is from a different MRI volume.*

7.6 Setup

The initial learning rate was $\eta = 0.0001$, and a learning rate scheduler reduced it by a factor of 0.2 every five plateau epoch[4]. A mini-batch size of 12 2D random slices was used with an Adam optimizer with parameters: betas $= (0.9, 0.999)$ eps $= 1e-08$ and weight decay $= 0$. These parameters were used for all experiments unless specified. Adam optimizer was used since it is the most popular as of this moment [21].

To maintain the same size after convolution the images are expanded using zero paddings. The weights were initialized with uniform distribution, $\mathcal{U}(-\sigma_N, \sigma_N)$, where $\sigma_N = 1/\sqrt{N}$, N is the size of the layer. The random seed was 123 for both numpy and torch. The torch `functions torch.backends.cudnn.benchmark = False` and `torch.backends.cudnn.deterministic = True`[5] was used for reproduceability.

For all experiments was a smoothing factor of $\Omega = 2$ was used. The loss function was *Tversky focal loss* from Equation 4.11, with parameters $\gamma = 4/3$, $\alpha = 0.7$ and $\beta = 0.3$, unless specified otherwise. These parameteres were used because they gave the best result in [26]. Note that in [26] $1/\gamma$ was used in Equation 4.12 with a value of $\gamma = 4/3$, which will focus on the easy examples as $\gamma^{-1} = (4/3)^{-1} = 3/4$. This can be seen in the Figure

[4]If the validation loss does not decrease for five epochs, the learning rate is reduced.
[5]https://pytorch.org/docs/stable/notes/randomness.html

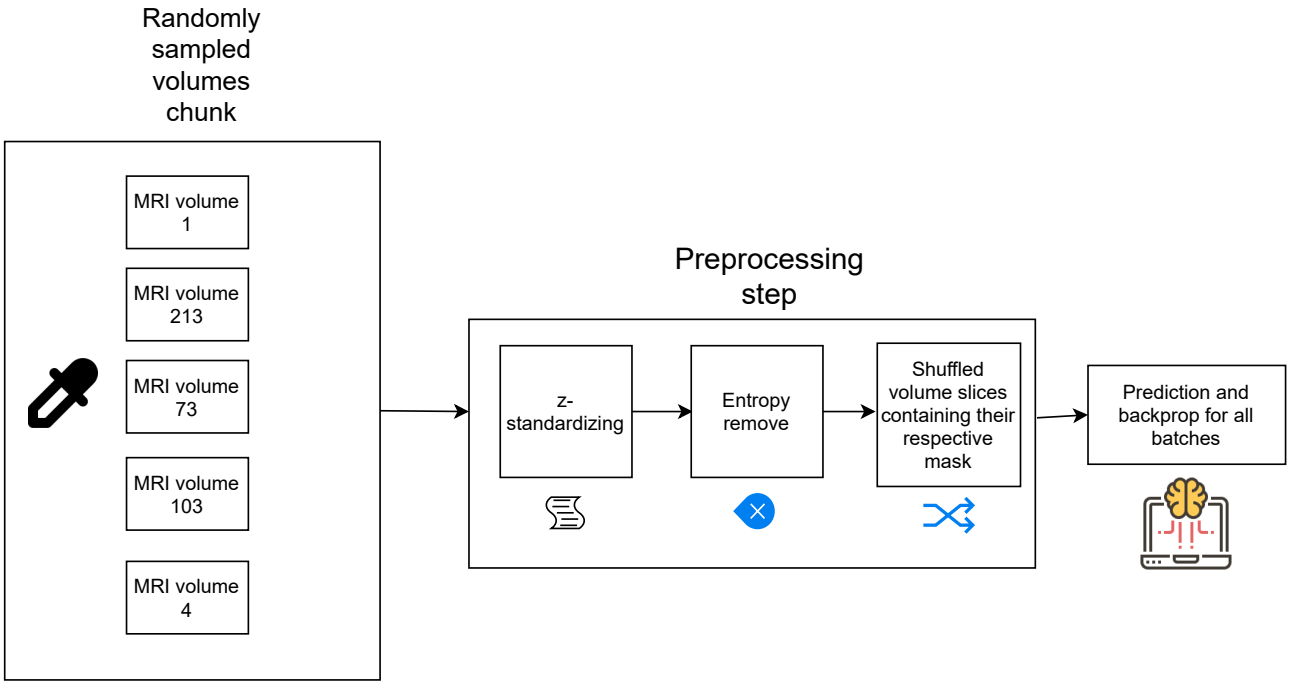


Figure 7.3: *Data pipeline for the training procedure. This cycle was done until all the training data have gone through the training procedure. The chunking of MRI volumes was done because of memory constraints. The MRI data was grabbed randomly until all volumes have had been seen during training. The validation procedure goes through the same steps except for entropy removal and back-propagation. Then it all starts over again for the next epoch.*

4.4. Assuming this was a typo we use $\gamma = 4/3$ for how the focal loss is defined in Equation 4.12.

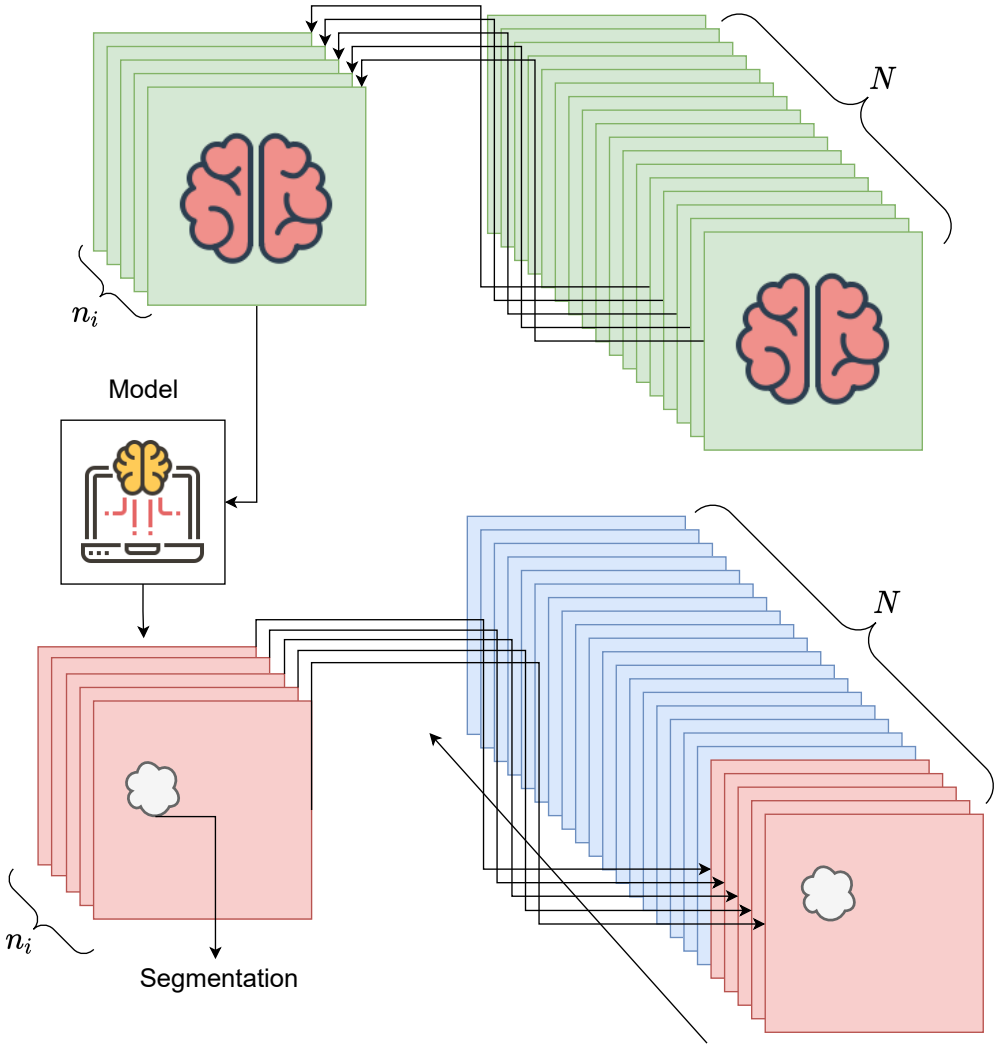


Figure 7.4: *Data pipeline for the volume prediction procedure. Since the models use 2D slices and not volumes, a batch of slices are taken in sequence and stack up a volume until the whole volume is segmented. N is the depth of the FLAIR volume and n_i is batch number i with batch size n.*

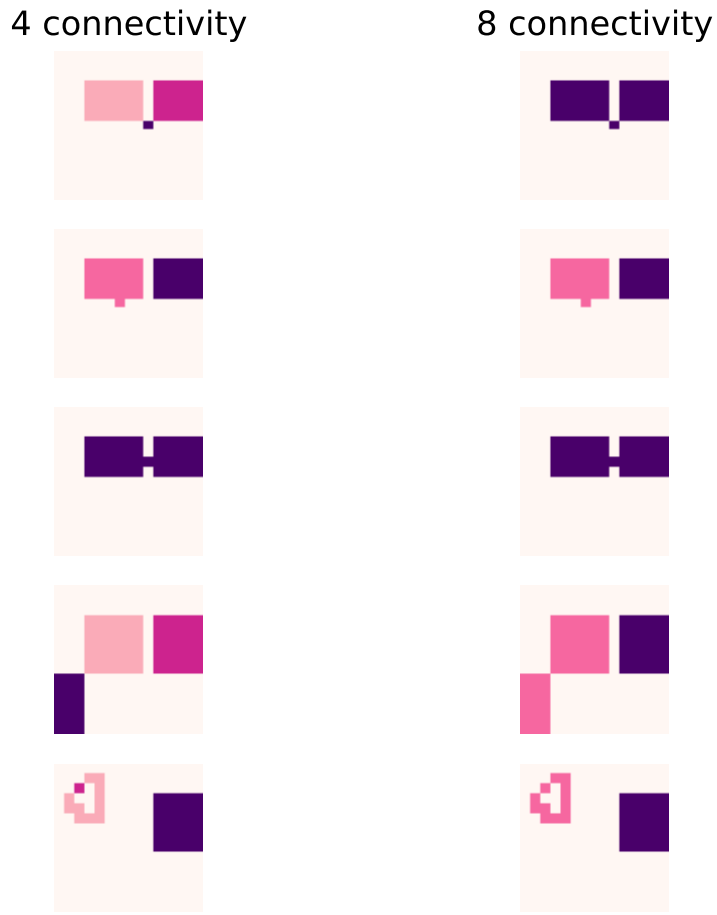


Figure 7.5: *Examples of connected regions in 2D using 4- and 8-connected. Colors represent the different regions. 8-connected seems the most appropriate.*

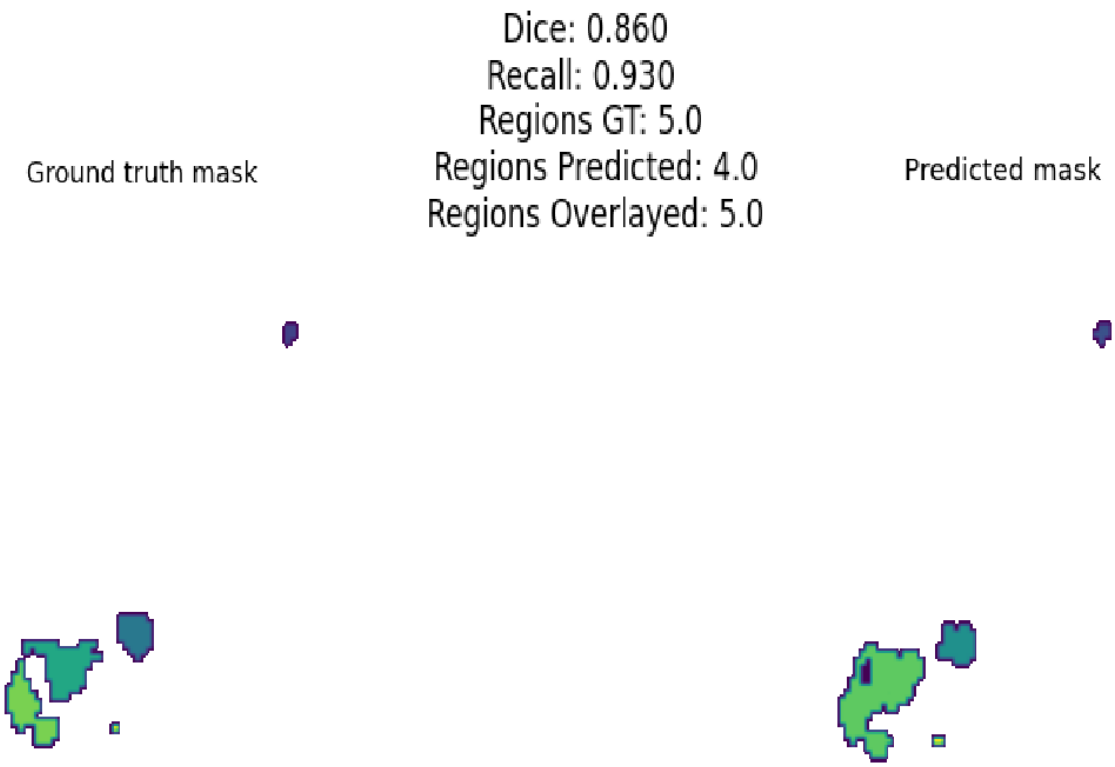


Figure 7.6: *Left: GT(Ground truth). Right: Predicted. Example of ground truth masks and predicted mask regions. It can be seen here that the prediction of the large regions is one larger region, and in the ground truth, there are two regions very close together, but all regions are overlapping. In the lesion metric tables shown in experiments, coordinates of the GT lesions were used to find the recall score by finding the prediction on the same coordinates.*

8 Experiments and results

In this section, we show results for different experiments and give a short explanation.

8.1 Experiments with pre-trained UNet256

In Chapter 3, some of the model's used pre-trained weights such as ImageNet. In our pre-trained experiments we use weights from [1] ([43]). The pre-trained weights were trained on a dataset that has MRI volumes with lower-grade gliomas. That dataset contained 110 pre-contrast, FLAIR (fluid-attenuated inversion recovery), and post-contrast MRI volumes as channels. In our data, we do not have all these channels, hence, transfer learning might not work well.

All weights were retrained in this experiment. The losses are shown in Figure 8.1. The weights were trained on three channels, thus, in this experiment, the input was 256x256 sagittal slice orientation slices concatenated with FLAIR, T1, and FLAIR(again), such that the input size is (mini-batch size, 3, 256, 256). We added FLAIR twice to not having to remove any of the pre-trained weights, which uses three input channels. In this experiment, no augmentation was used. In the experiment shown in Figure 8.2, the parameters were changed to $\alpha = 0.85$, $\beta = 0.15$ to increase the recall.

Results for the metrics are shown in Table 8.1.

Table 8.1: *Results for the pre-trained model for the given focal Tversky loss parameters. The higher recall is better because finding lesions are considered to be more important than false positives.*

γ	α	β	Dice	Recall
4/3	0.70	0.30	0.66	0.74
4/3	0.85	0.15	0.63	0.81

8.2 Experiment with non pre-trained UNet256

We performed experiments with random uniform initialized weights to get a baseline for all the experiments. The model seems to perform relatively well in both pre-trained and not pre-trained experiments. **Results** are shown in Table 8.2 and the respective loss is plotted in Figure 8.3 and Figure 8.4. The loss seems to oscilates in the early epochs. The validation loss seems to start diverging close to epoch 50, but might still have some minimization left as the divergence is not that clear.

[1] https://github.com/mateuszbuda/brain-segmentation-pytorch

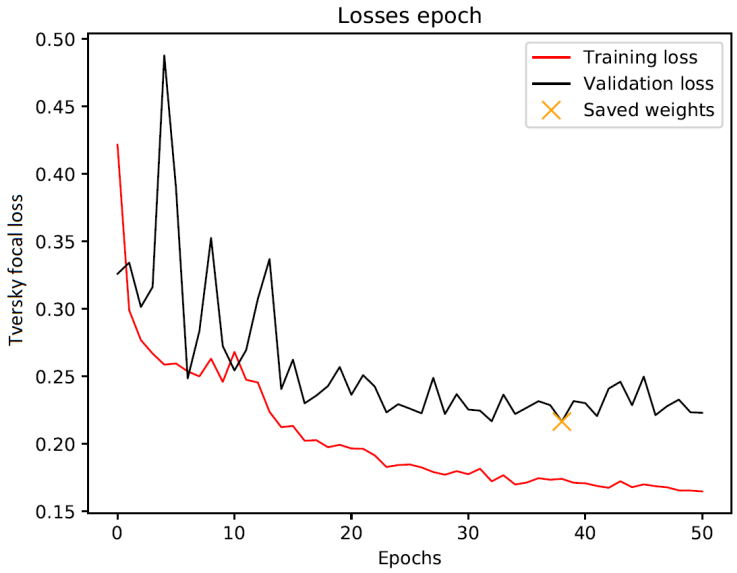


Figure 8.1: *The average mini-batch Tversky focal losses for the pre-trained model. Since this is a medical problem, as we get a high enough dice score, we would like a large recall score to make sure we find the positives. According to this plot, we might want to weigh the α (FN) parameter a bit more to further increase the recall.*

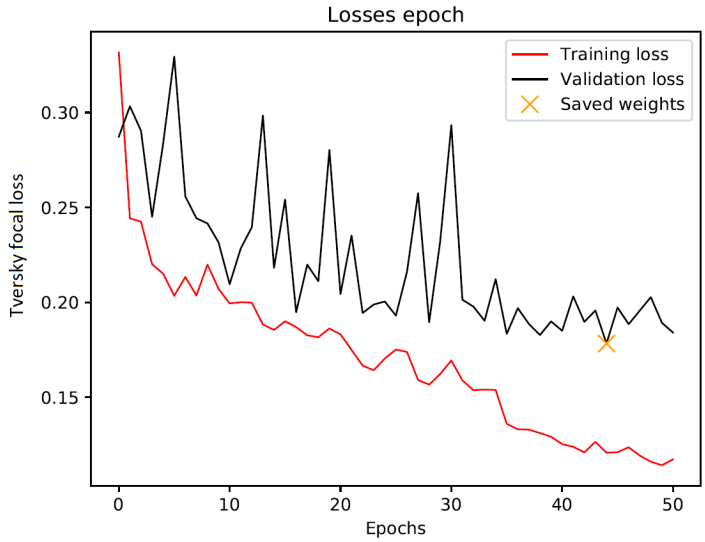


Figure 8.2: *The average mini-batch Tversky focal losses for the pre-trained model, with focal Tversky loss parameters: $\alpha = 0.85$, $\beta = 0.15$, $\gamma = 4/3$. The validation loss oscillates more rapidly, and the validation loss seems to plateau/diverge at the end. This might suggest that the minimum found is likely the best for the parameters with respect to validation.*

Table 8.2: *Results for the model trained from scratch for the given Tversky focal loss parameters. The pre-trained model seems to be slightly better.*

γ	α	β	Dice	Recall
4/3	0.70	0.30	0.65	0.74
4/3	0.85	0.15	0.63	0.80

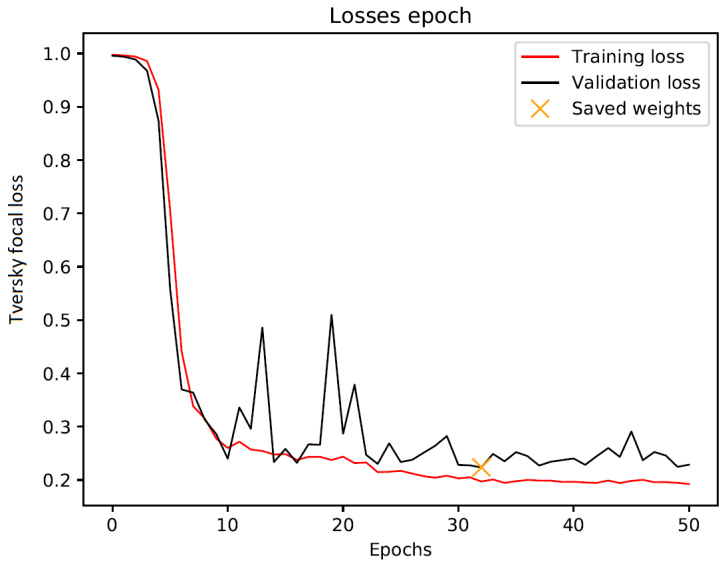


Figure 8.3: *The average mini-batch Tversky focal losses for the model without pre-trained weights. Here the parameters were* $\alpha = 0.7$, $\beta = 0.3$ *and* $\gamma = 4/3$. *The loss starts at about 1, but the loss is rapidly decreasing.*

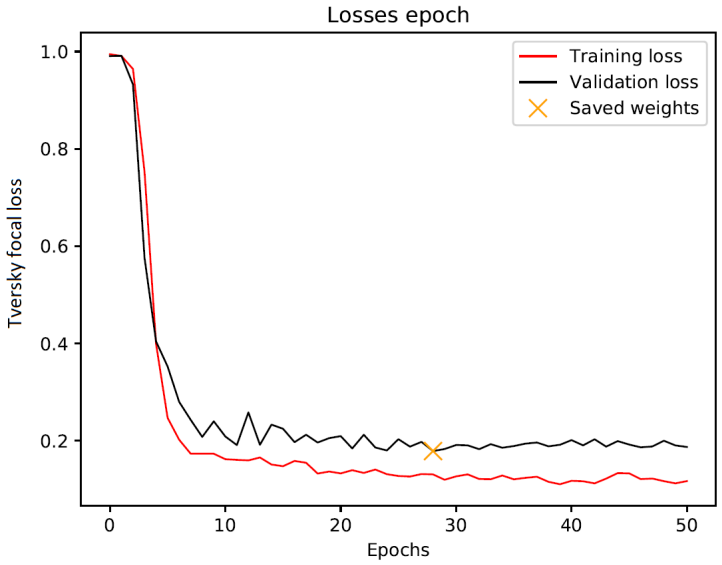


Figure 8.4: *The average mini-batch Tversky focal losses for the model without pre-trained weights.* $\alpha = 0.85$, $\beta = 0.15$ *and* $\gamma = 4/3$. *Changing the parameters seems alleviate the large loss spikes encountered in Figure 8.3.*

8.3 Experiment UNet256 with attention gates, pyramid input, no pre-trained.

Many of the lesions are very small, so we perform experiments with the attention modules as described in Chapter 6.2 and Chapter 3.3. In this experiment, the attention architecture from Section 6.2 was used. The same parameters as for the regular UNet256 were used to compare the results. Pyramid input was used as seen in Figure 6.3. The down-sampling was done with max-pooling with kernel = 2x2 and stride = 2. Small spatial details get lost during cascading convolutions and nonlinearities. Hence, pyramid input is a way to introduce the

smaller details to the more feature-rich layers by concatenating them with the non-convolved but downsampled input. The soft attention gates are also a way to avoid the interesting features in the lower resolution getting lost during the decoding phase.

In Figure 8.5 an example result for the validation data is shown. In the last layer, we can see that the true positive region is highlighted, but also another region that is not a true positive. We can also observe that the ReLU sets all values below zero to zero, which then goes through a sigmoid layer, forcing the zero values to 0.5. This can be seen in the background of the attention map.

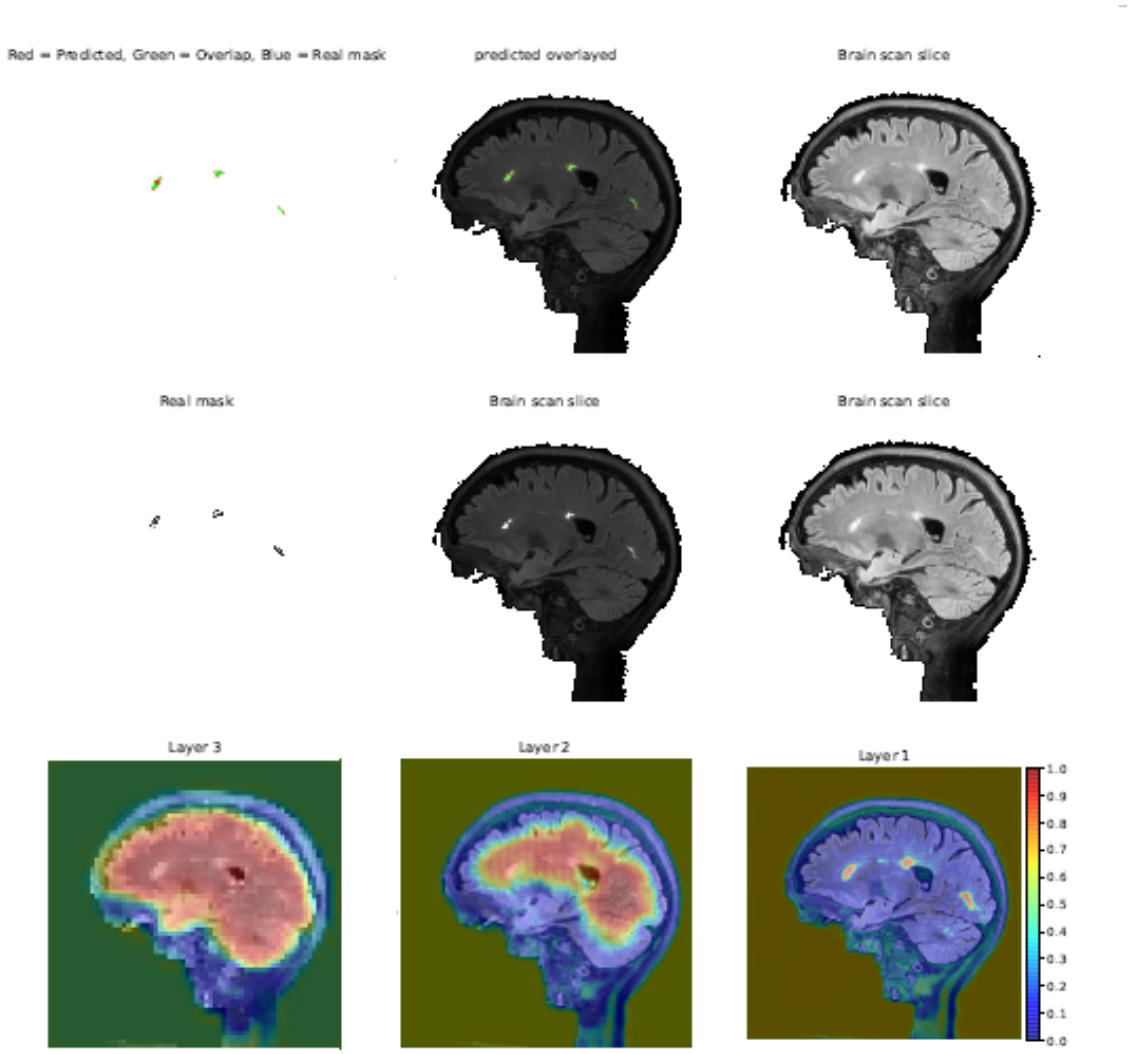


Figure 8.5: *The positive regions can be very small, as seen in these predictions. The last three columns show the attention weighting map for the three deepest layers of the network. As the resolution increases, the weighting gets more concentrated on the lesions.*

Results were dice: 0.66 and recall: 0.75. This model also has an interesting attention map that could be used for prediction analysis and used to explain what regions are being "looked at". The loss can be seen in Figure 8.6. The loss oscillates wildly to 50 epochs. The minimum loss is also very close to 50 epochs. Therefore, more optimization can most likely be done here.

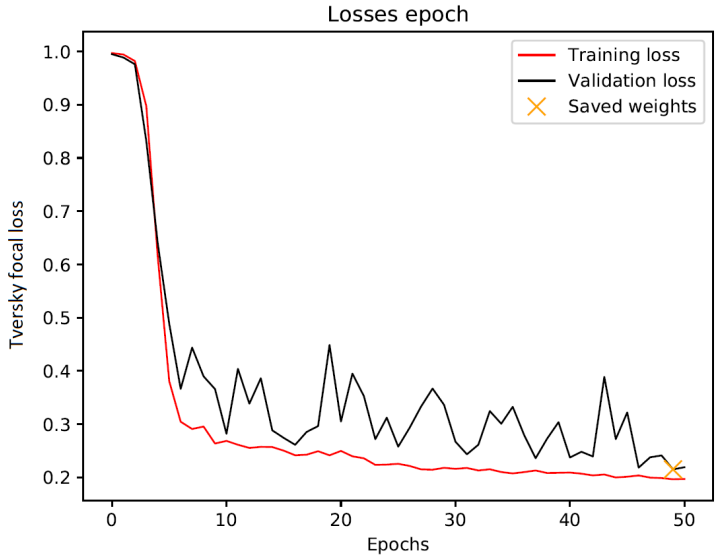


Figure 8.6: *The average mini-batch Tversky focal losses for the attention model without pre-trained weights and with pyramid input. Here the parameters were* $\alpha = 0.7$, $\beta = 0.3$ *and* $\gamma = 4/3$. *Here it seems like the training could go on a bit further since the best result were at the very end of the epoch length. The metrics for the minima were dice: 0.66 and recall: 0.75.*

8.4 Experiment pre-trained UNet256 with attention gates without pyramid input.

In this experiment, we used pre-trained weights with attention. One problem with using the pre-trained weights is that attention modules are not in the weights list because they were not in the original model. Therefore, we had to use initial weights in the attention modules and having the rest of the model using the pre-trained weights.

Results of the scores are shown in Table 8.3, and the losses are shown in Figure 8.8 and Figure 8.9. In the loss figures, we can see that we have probably found the optimum for this model as the validation is diverging. In Figure 8.7 an example prediction with attention map is shown. Comparing an example output of the attention map in figures(8.7) and (8.5, we can see that the pyramid input seems to make the attention more precise.

Table 8.3: *Results for the pre-trained attention model for the given focal Tversky loss parameters. In this experiment, we got a better recall, but at the cost of dice, this means the precision score went down. This suggests that the pyramid input helped to decrease false positives.*

γ	α	β	Dice	Recall
4/3	0.70	0.30	0.67	0.72
4/3	0.85	0.15	0.63	0.79

8.5 Experiments with 3-slices as a channel.

In this experiment, we added depth information as channels. Even though we do not add any slice attention modules to the model, as discussed in relevant literature Chapter 3.5, we hypethize that adding three slices in

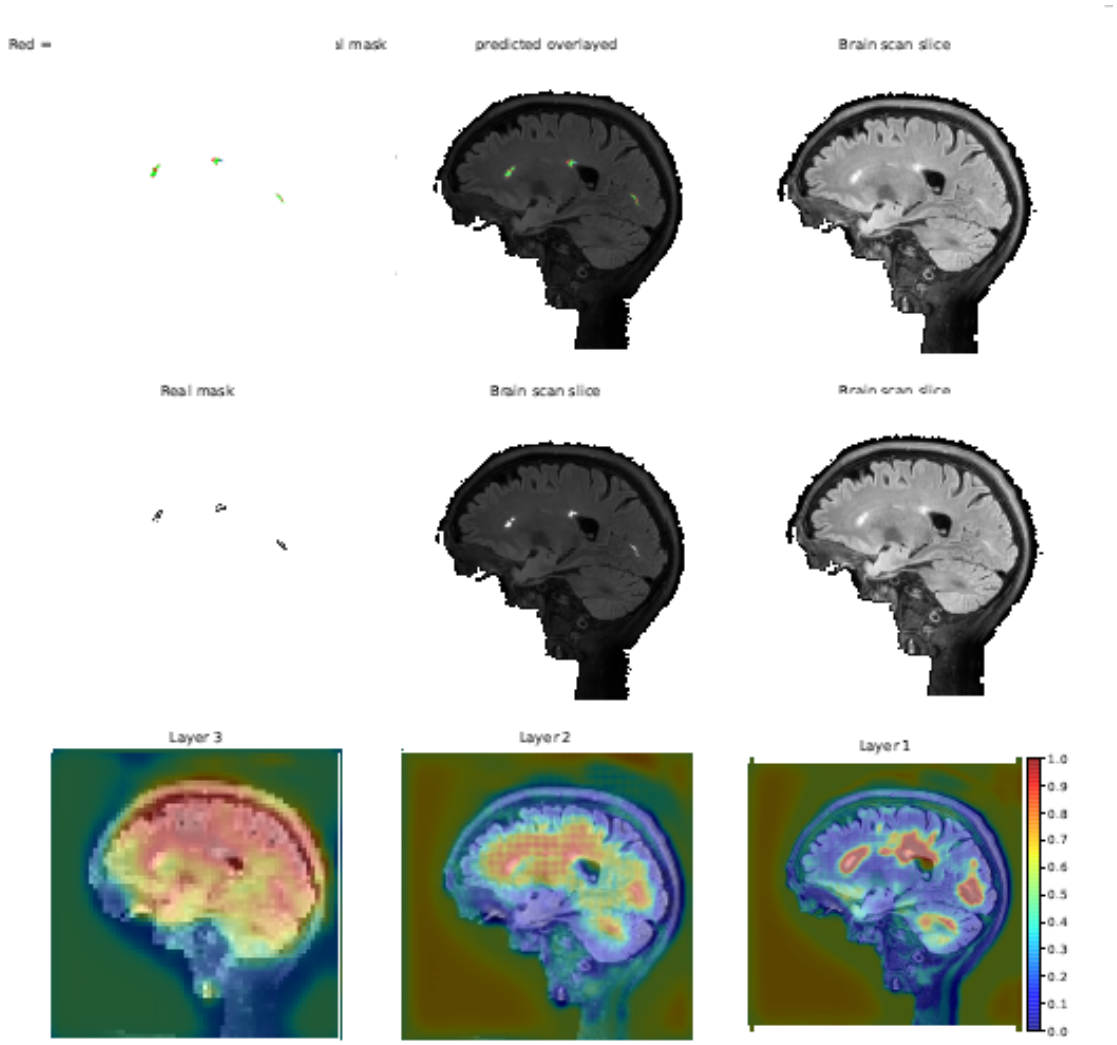


Figure 8.7: *There is a large difference in how the attention map is weighted compared to Figure 8.5. The lowest layer weighs parts of the skull, and the largest resolution layer has much larger attentions around the lesions. The pyramid input seems to help the attention gates to perform better.*

one orientation will help in detection. Instead of one 2D slice from FLAIR and T1, we add three slices in depth from both. The depth slice stacking is done by taking three vicinity slices as shown in Figure 6.5 and then the model predicts the middle slice mask. Using the depth as channels in input is described more in detail in Section 6.3.

In this experiment we obtained the loss shown in figures(8.10) and (8.11). The best metric results are shown in Table 8.4. Using nearby depth information seems to increase both dice and recall score. Adding more slices might help because if we look at the lesion metric Table 5.3, we see that the lesions span up to 40+ voxels on average in the larger lesions. But we hypothesize that it won't help very much as one up and one down slice is enough to predict one slice of lesions.

Results for the losses are shown in Figure 8.10 and Figure 8.11. The score results are shown in Table 8.4.

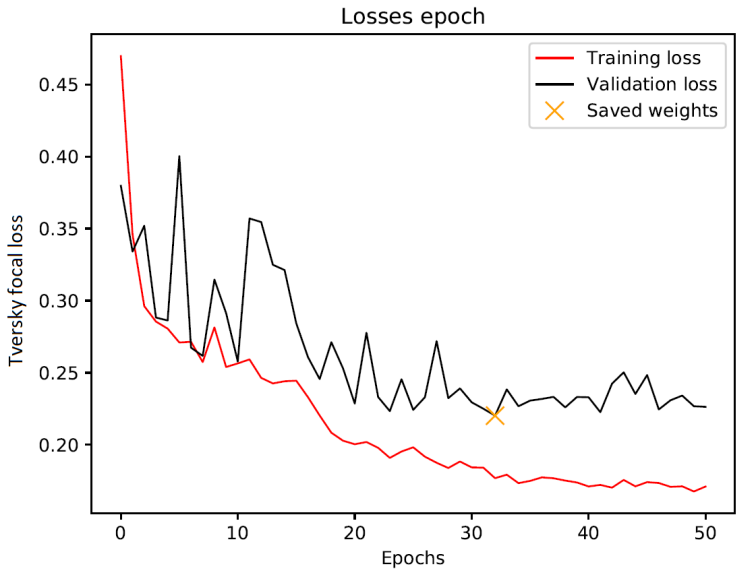


Figure 8.8: *Average mini-batch losses for the model without pre-trained weights. Parameters used were* $\gamma = 4/3$, $\alpha = 0.70$ *and* $\beta = 0.30$.

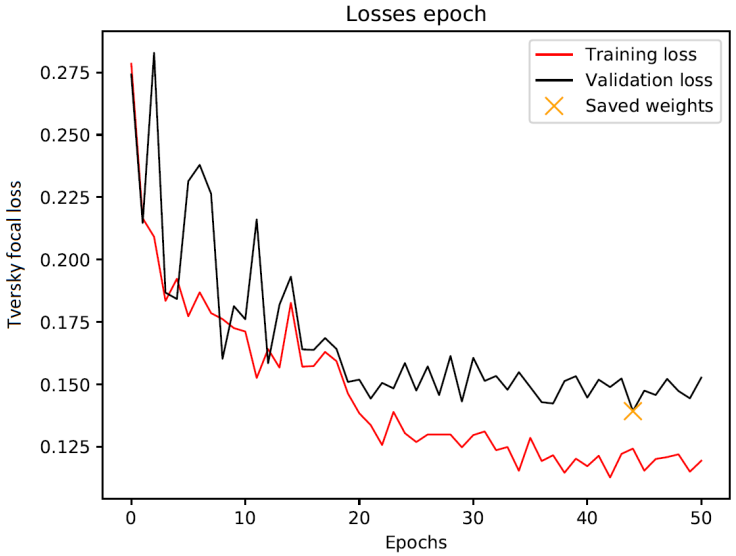


Figure 8.9: *The average mini-batch Tversky focal losses for the model without pre-trained weights. Parameters used were* $\gamma = 4/3$, $\alpha = 0.85$ *and* $\beta = 0.15$.

Table 8.4: *Results for UNet256 using 3-slices as input for the given Tversky loss parameters. The experiment with* $\gamma = 4/3$, $\alpha = 0.85$ *and* $\beta = 0.15$ *yield best results.*

γ	α	β	Dice	Recall
4/3	0.70	0.30	0.70	0.75
4/3	0.85	0.15	0.68	0.80

8.6 Robustification of best result

Adding augmented examples of the data can help the model generalize more. Augmentations like having vertical and horizontally flipped examples of the data in the dataset can help the model understand these flipped features.

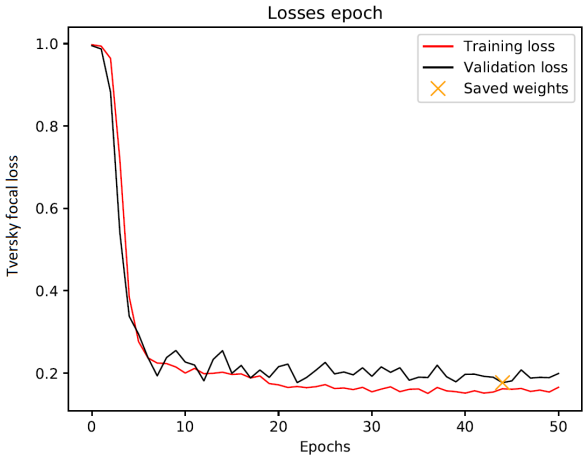


Figure 8.10: *The average mini-batch Tversky focal losses for the model using 3-slices as input in channel. Parameters used were* $\gamma = 4/3$, $\alpha = 0.70$ *and* $\beta = 0.30$. *The validation seems more stable than the other experiments.*

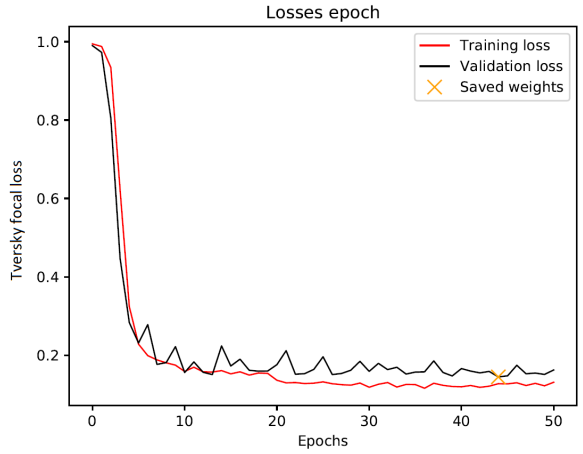


Figure 8.11: *The average mini-batch Tversky focal losses for the model using 3-slices as input in channel. Parameters used were* $\gamma = 4/3$, $\alpha = 0.85$ *and* $\beta = 0.15$. *The validation seems more stable than the other experiments*

Convolutional neural networks are not rotation invariant and therefore need these augmentations, or use some type of rotation equivariant filters to understand rotated examples[44]. In the augmented experiments, 300 random examples were flipped randomly in both vertical and/or horizontal, with both operations having the probability of $p = 0.5$.

In [45] they argued that flipping along the horizontal axis is valid because of the left and right symmetry of the brain regions, which is also mentioned in [46]. Although they did not have a segmentation problem, a classification problem, we hypothesize that adding vertical and horizontal flips can help the model generalize more on how the features of the lesion look like as it is mostly in the proximity of the lesion where the correlated features are. **Continuing training the best 3-slice model** in Section 8.5 with augmentation increases the score and generalization. For augmentation, the 300 images of each chunk (12 patients' worth of images were used for each chunk) were flipped both horizontally and/or vertically with a probability of 0.5. Adding vertical flips might not be completely correct, as mentioned in [46], where vertical flips are not always "interchangeable"

as in horizontal flips. Other typical augmentations in MR include, but are not limited to: Translation, which is changing the position of the object/slice. Scaling and cropping, which can help the model learn important features independently of original size[46]. Caution must be made when doing augmentation for segmentation as we must not create any errors in the mask, which usually needs the same transformation as the 2D slice. Other types of augmentations are noise, rotations, and sheering, which we can see some natural examples of in our dataset. In [4] they also used rotation, sheering, and scaling as augmentation.

Adding augmentation increased our training data from 9421 to 11221 images. The result shown in Figure 8.12. The training data still used data not removed by the entropy threshold described in Section 5.3, while the validation uses all slices. Here the dice score becomes worse, but the recall increases. The lesion metrics is shown in Table 8.5 below.

Results for the loss is shown in Figure 8.12 and the lesion metric is shown in Table 8.5.

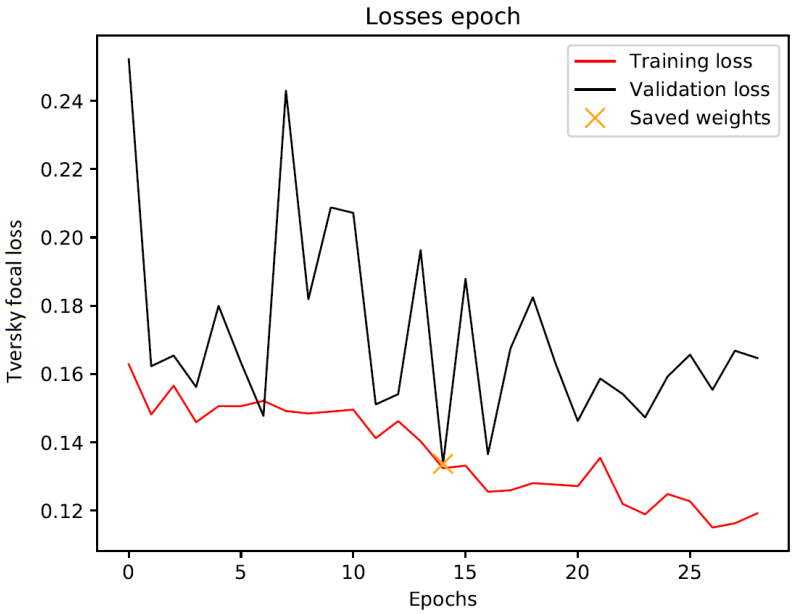


Figure 8.12: *The average mini-batch Tversky focal losses. Parameters used were* $\gamma = 4/3$, $\alpha = 0.85$ *and* $\beta = 0.15$. *In this figure, the mini-batch mean loss is shown. This is a continuation of the best 3-slice experiment. The weights from the experiment were loaded and augmentation was added, and then trained again. Result here is* $F_3 = 0.75$, *dice* $= 0.66$ *and recall* $= 0.84$ *on the validation data.*

8.7 Experiments with 3-slices as a channel using FLAIR only.

All previous experiments used both FLAIR and T1 as channels. In deployment, it would be much easier and less time-consuming if the model only needed FLAIR as input. As experiments show, the use of T1 does not give much higher scoring. We first trained the model as before, and then retrained these weights with the same augmentation as in Section 8.6. **Results**: The loss is shown in Figure 8.13 and the lesion metric is shown in Table 8.6.

Table 8.5: *Lesion prediction metrics. The model predicts more lesions with a small depth decreasing the average depth and voxels. The recall is overall pretty good, but it increases with lesion size. Depths and voxels are averages over lesions. *TP, FN, and FP are not pixel-wise, but for predicted lesions, The TP is set if the predicted lesions are at least 60% of GT lesion. *Avg. recall(smooth) is calculated using the pixels inside of the lesions and then average over them in all their respective lesion bracket. These scores are from the validation data, and the lesions are in 3D.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	51	636	4255
Avg. depth	2	5	24	43
Total	441	172	9	7
Total TP*	266	147	9	7
Total FN*	289	42	0	0
Total FP*	175	25	0	0
Avg. recall(smooth)	-	0.79	0.92	0.94
Lesion precision	0.60	0.85	1.0	1.0
Lesion recall	0.49	0.78	1.0	1.0

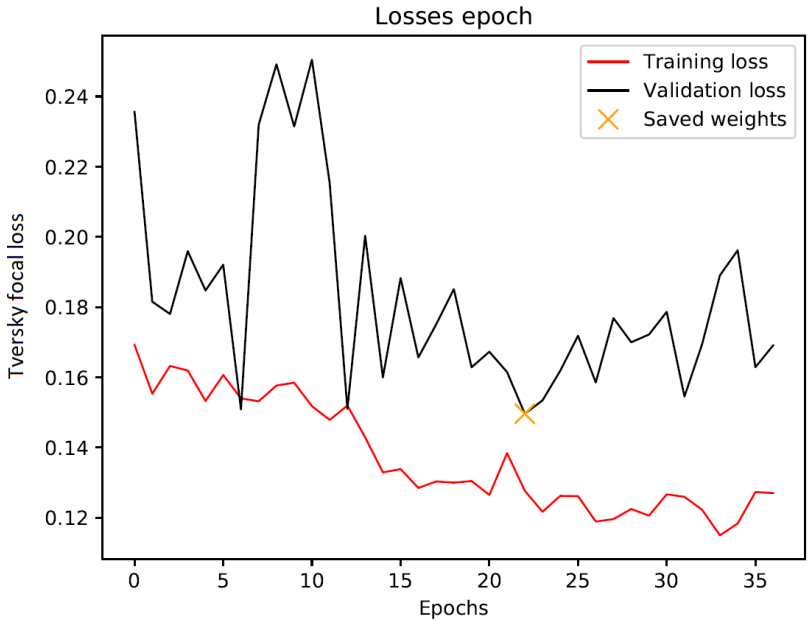


Figure 8.13: *The average mini-batch Tversky focal losses. Parameters used were $\gamma = 4/3$, $\alpha = 0.85$ and $\beta = 0.15$. In these figure the mini-batch mean loss is shown. Score: $F_3 = 0.74$, dice = 0.66 and recall = 0.82.*

8.8 Experiment Mish and group normalization

Adding augmentation seems to boost recall. In this section, we further improved by changing the ReLU activation function to Mish. In this training process, augmentation was used from the beginning of the training. Since batch normalization works better with large batch size, we also performed two experiments by changing to group normalization as described in Section 2.5. One problem with this change is that we now have a new hyperparameter G. We did two experiments with Mish as activation, one with $G = 4$ and another with $G = 32$. **Results** are shown in Figure 8.15. As seen in Table 8.8, the model predicted one large false positive, which is not even part of the brain. The image is shown in Figure 8.16. These results were worse than using batch

Table 8.6: *Lesion prediction metrics. The detection of the lesion is not as great as the model with both FLAIR and T1.*TP, FN, and FP are not pixel-wise, but for predicted lesions, It is counted if the predicted lesion is at least 60% of GT lesion. Avg recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	58	620	3849
Avg. depth	2	6	24	40
Total	456	155	9	7
Total TP*	241	138	9	7
Total FN*	314	51	0	0
Total FP*	215	17	0	0
Avg. recall(smooth)	-	0.77	0.93	0.93
Lesion precision	0.53	0.89	1.0	1.0
Lesion recall	0.43	0.73	1.0	1.0

normalization with Mish. Henceforth, we use batch normalization for the rest of the experiments.

Results for Mish with augmentation using FLAIR only losses is shown in Figure 8.14 and the lesion metric is shown in Table 8.7.

Results for Mish and group normalization losses is shown in Figure 8.15 and lesion metrics are shown in Table 8.8 and Table 8.9.

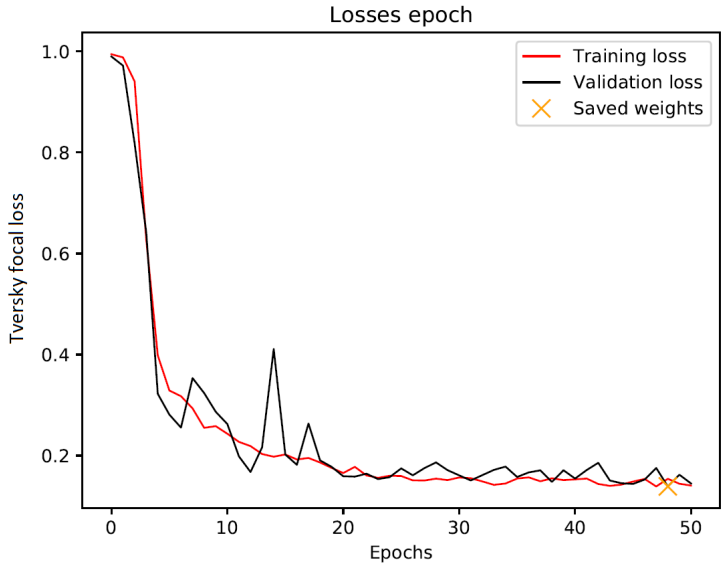


Figure 8.14: *The average mini-batch Tversky focal losses. Parameters used were $\gamma = 4/3$, $\alpha = 0.85$ and $\beta = 0.15$. In these figure the mini-batch mean loss is shown. Score: $F_3 = 0.76$, dice = 0.64 and recall = 0.85. The F_3 score is two percentage point better.*

8.9 Test data experiment

The results on the test data are the most interesting and important part. This part of the data had no interaction with the training process and, therefore, yields the best representation of the real-world result. In this experiment, we used the Mish, batch normalization, and augmentation with verical and/or horizontal flips

Table 8.7: *Prediction lesion metrics for flair only model using Mish and group normalization. This model seems to detect more of the lesions with sizes 10-400 but at the cost of more false negatives. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	51	589	4219
Avg. depth	2	5	23	37
Total	546	168	9	7
Total TP*	284	143	9	7
Total FN*	271	46	0	0
Total FP*	262	25	0	0
Avg. recall	-	0.79	0.93	0.96
Lesion precision	0.52	0.85	1.0	1.0
Lesion recall	0.51	0.76	1.0	1.0

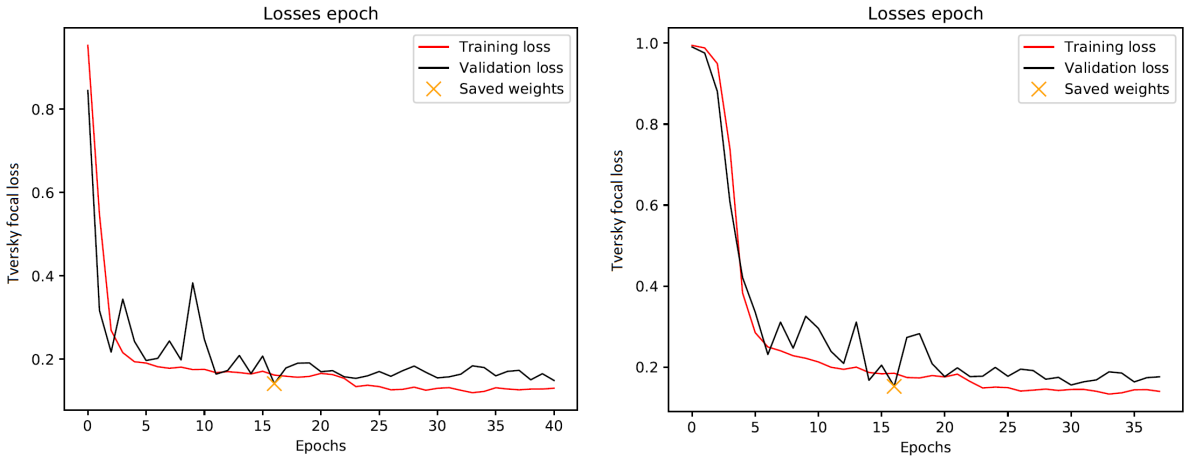


Figure 8.15: *The average mini-batch Tversky focal losses. Parameters used were* $\gamma = 4/3$, $\alpha = 0.85$ *and* $\beta = 0.15$. *In these figure the mini-batch mean loss is shown. Left:* $G = 4$ *Score:* $F_3 = 0.75$, *dice* $= 0.59$ *and recall* $= 0.89$. *Right:* $G = 32$ *Score:* $F_3 = 0.73$, *dice* $= 0.59$ *and recall* $= 0.86$.

Table 8.8: *Prediction lesion metrics for only flair model with mish and* $G = 32$. *Using* $G = 32$, *led to one large false positive, which is very bad. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion bracket. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	57	608	2886
Avg. depth	2	6	22	37
Total	508	193	10	7
Total TP*	243	146	9	7
Total FN*	312	43	0	0
Total FP*	265	47	1	0
Avg. recall	-	0.78	0.97	0.95
Lesion precision	0.48	0.76	1.0	1.0
Lesion recall	0.44	0.77	1.0	1.0

model, which gave the best F_3 scoring in previous validation experiments. **Results** F_3 score: 0.80, Dice: 0.66, Recall: 0.90. From now on the dice and recall score will be skipped as the F_3 is the most important score. The

Figure 8.16: *Large false positive in the validation data, which is not even part of the brain. This model is too aggressive in segmentation during prediction.*

Table 8.9: *Prediction lesion metrics for flair only model with mish and $G = 4$.Using fewer groups made the network not predict a large false positive in the validation data. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	51	640	3355
Avg. depth	2	5	24	37
Total	660	199	9	7
Total TP*	314	152	9	7
Total FN*	241	37	0	0
Total FP*	346	47	0	0
Avg. recall	-	0.82	0.97	0.96
Lesion precision	0.48	0.76	1.0	1.0
Lesion recall	0.57	0.80	1.0	1.0

lesion metrics are shown in Table 8.10. Some examples are shown in Figure 8.17. In these examples, we can observe that the prediction is very precise.

Table 8.10: *Prediction lesion metrics for test data. The results are better than the results from validation data. This is because the model does much better on the larger lesions, which the test data had more examples. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion bracket. These scores are from the test data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	53	631	1978
Avg. depth	2	5	20	31
Total	551	195	11	9
Total TP*	287	176	11	9
Total FN*	232	20	0	0
Total FP*	264	19	0	0
Avg recall	-	0.87	0.94	0.97
Lesion precision	0.52	0.90	1.0	1.0
Lesion recall	0.55	0.90	1.0	1.0

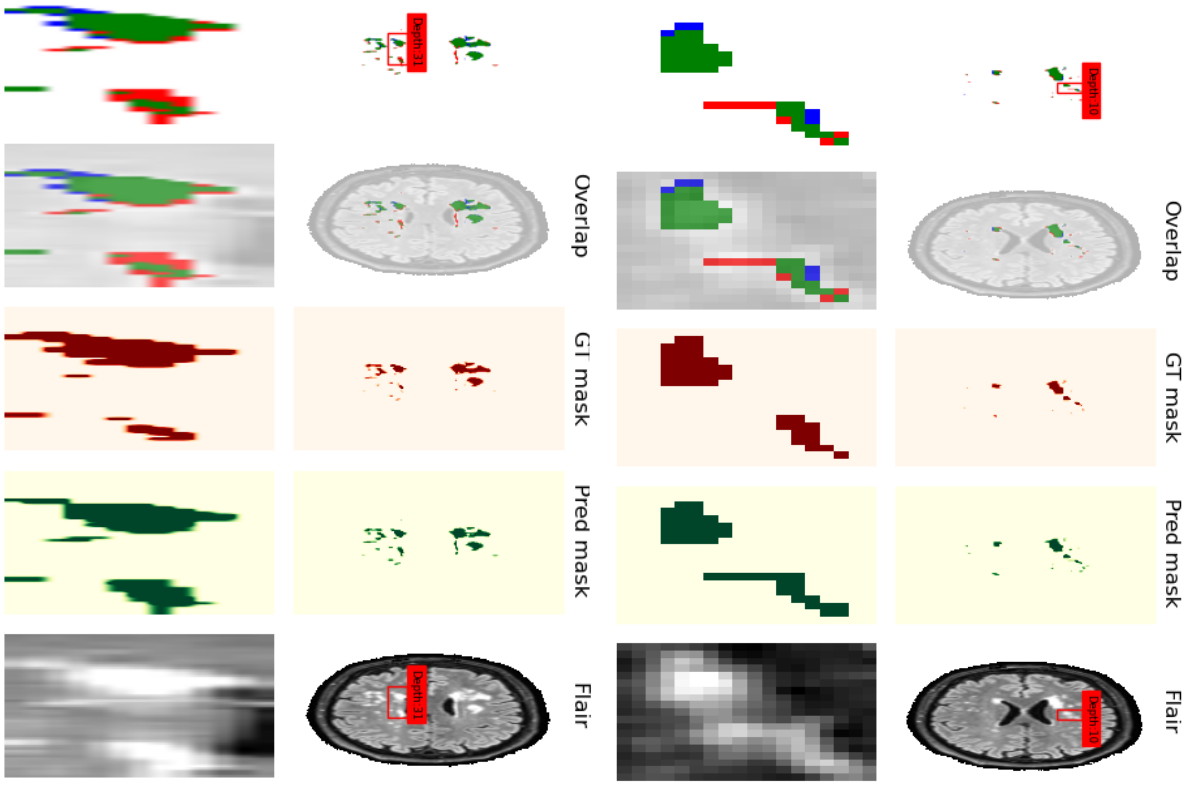


Figure 8.17: *Example predictions on a 2D slice for two different MRI volumes from the test data. The first row shows a zoomed-in version of the lesion inside the bounding box. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

8.10 Large dataset experiments

In these experiments, we introduce the large dataset, which will provide the main experiments as this dataset contain a much broader range of different MRI volumes. The dataset was preprocessed the same way with entropy measure thresholding for the training data as well. In Figure 8.18 we can see some difference in entropy compared to the previous dataset. Since this data is from different machines, hospitals, etc, it will have some differences in imaging settings and different MRI volume sizes. In Figure 8.18 some MRI volumes are much larger in-depth and are much noisier, creating large entropy over the whole volume. The largest standard deviation is in depths $N = 176$ and $N = 300$, but it is expected to have some deviation. All slices/images above 2 bits/pixel were used. Here the maximum entropy is at around 6 bits/pixel. The red line indicates the threshold for removing 2D slices. As observed in this figure, the information seems to follow mostly the same patterns when they are in the same depth. N is the depth of the MRI volume. The number of MRI volumes in each depth bracket is shown in Table 8.11. The one MRI volume at depth $N = 120$ starts where we have brain tissue, therefore every 2D slice above the threshold.

The entropy threshold is kept at two since we do not want to remove too much data, this was first introduced in Section 7.2. This dataset has a lot of different types of MRI volumes indicated by the vast difference in information over slices. Here we might hypothesize that the model trained on the smaller dataset, which only had MRI volume depths $N = 183$ slices will not do good on this large dataset. Some examples of entropy border

cases is shown in Figure 8.19. In all these experiments we used Tversky focal loss with $\gamma = 4/3$, $\alpha = 0.85$ and $\beta = 0.15$, and chunk sizes are set to 4 volumes per chunk. Since the chunk size is smaller, it might impact the results a bit as we used 12 in the smaller dataset. The chunk size was decreased as many of the MRI volumes were larger and therefore increased the memory footprint.

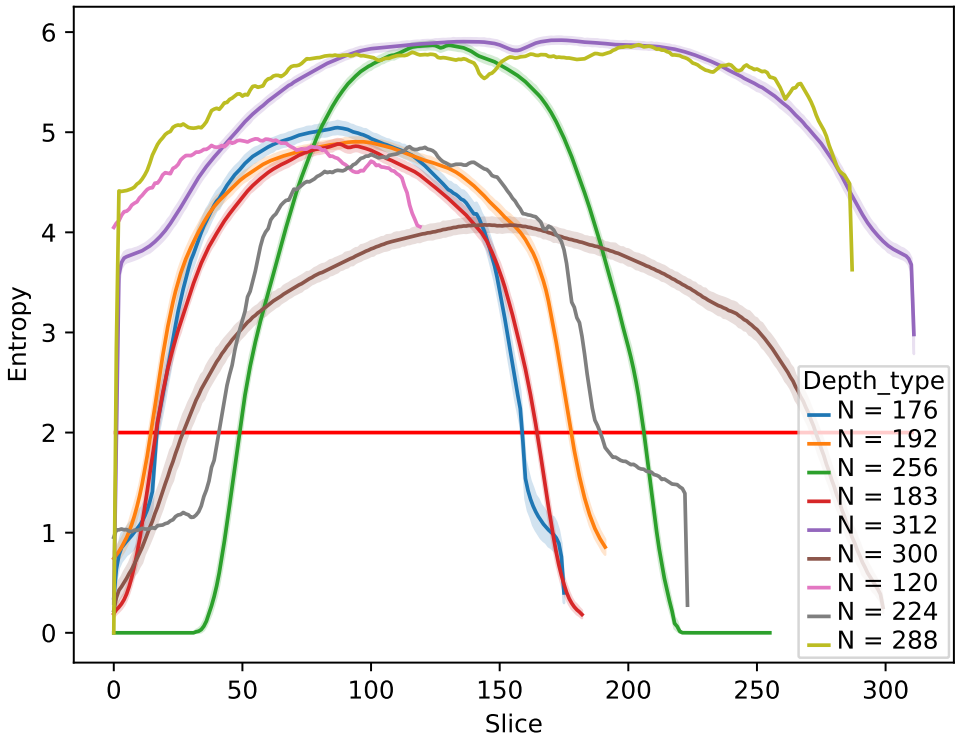


Figure 8.18: *Entropy content over MRI volumes over the slices in training data. The dataset has a very small standard deviation in information over slices which is indicated by the confidence-band.*

Depths	120	176	183	192	224	256	288	300	312
Num. MRI Volumes	1	64	134	155	1	87	1	57	85

Table 8.11: *Some MRI volumes with a given depth only occurs one time. This table shows the amount of MRI volumes in the different depth brackets shown in Figure 8.18.*

8.11 3-slice as channels FLAIR only, [Large data] sagittal slices with ReLU

Using FLAIR as the only channel shows great results in experimentation on the small dataset. Hence, we continue using this as the only channel.

Results $F_3 = 0.75$ and lesion metrics are shown in Table 8.12. Comparing axial and sagittal slice orientation experiments, it seems to suggest that it does not increase the prediction as long as inferred slice orientation is consistent with the trained slice orientation. The losses are shown in Figure 8.20. We can observe the loss is

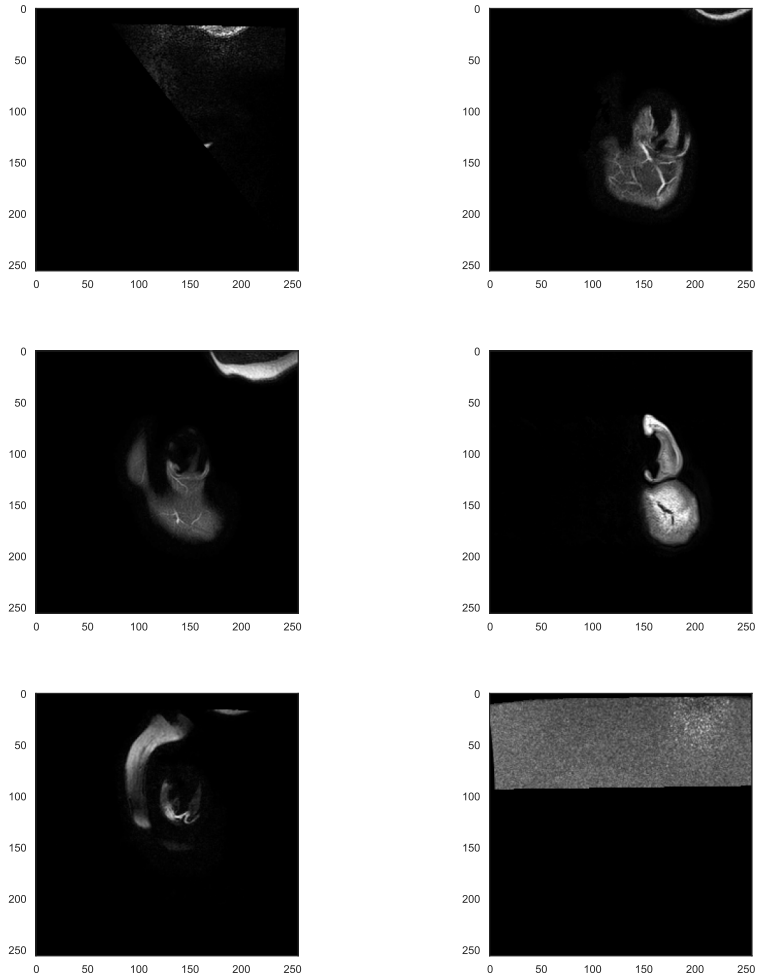


Figure 8.19: *Entropy threshold border cases from different depth volumes. As seen in the examples, most border examples do still not contain that much information, and in the last row of column two, we can see how noise has a lot of entropy. The threshold was set very low to not remove too much data. Each 2D slice is from a different MRI volume.*

fairly stable throughout the training process. This is interesting because most of the loss curves have strong spikes and oscillates a bit in the beginning.

8.12 Pre-trained FLAIR only, [Large data] axial slices, ReLU

In this experiment, we flipped the orientation to axial and tried the pre-trained model again. Since the original pre-trained model used three channels and we want to only use FLAIR, the channel is concatenated with the same FLAIR, C = [FLAIR, FLAIR, FLAIR], here we could add the 3-slices as in previous experiments, but we wanted to stay as close to the pre-trained data as possible, although in retrospect using three slices with pre-trained might be better.

Results F_3 score: 0.73. The lesion metric is shown in Table 8.13 and is visualized in Figure 8.22. The larger lesions seems to have examples of lower recall than in Figure 8.24 from Section 8.13 which used Mish activation

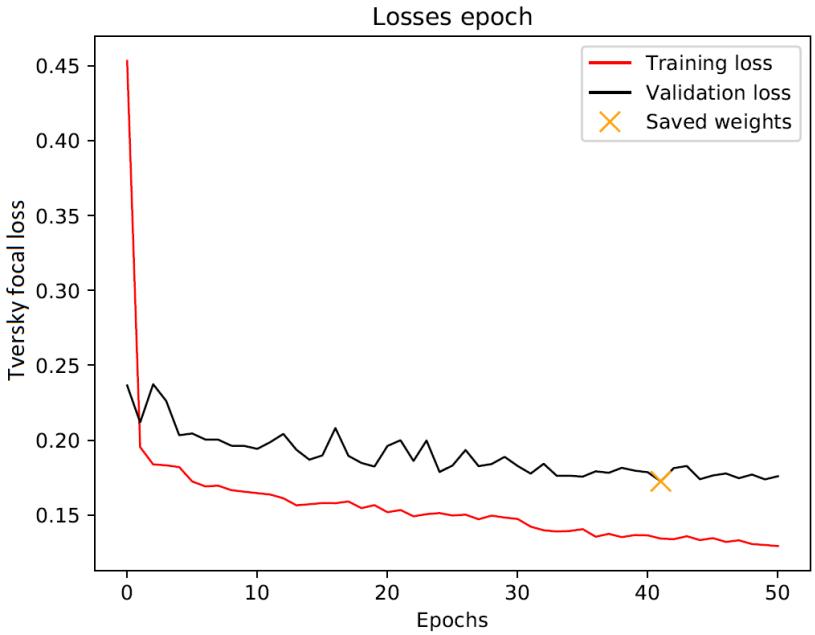


Figure 8.20: *The average mini-batch Tversky focal losses for the validation data 3-slice as channels FLAIR only, [Large data] sagittal slices with ReLU experiment.*

Table 8.12: *Prediction lesion metrics for FLAIR model. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	51	625	4348
Avg. depth	2	6	21	41
Total	9009	4036	119	136
Total TP*	3307	3282	119	136
Total FN*	7554	1059	2	2
Total FP*	5702	754	0	0
Avg. recall	-	0.76	0.92	0.94
Lesion precision	0.37	0.81	1.0	1.0
Lesion recall	0.30	0.76	0.98	0.99

and augmentation. In Figure 8.21 the losses are shown. In this loss plot we can see that the pre-trained weights puts the model closer to the minimum in the first epoch compared to the experiments trained from scratch.

8.13 3-slice as channels FLAIR only, [Large data] axial slices with Mish and augmentation

The experiment in Section 8.11 show four large false negative. To try removing these false negatives, we applied mish activation and augmentation as it seems to increase the recall score, as shown in Section 8.8.

Results F_3 score: 0.75. The lesions metrics is shown in Table 8.14 and losses is shown in Figure 8.23. The table is visualized in Figure 8.24, in this figure, some of the largest lesions have a very bad recall. After consultation with a professional, the large lesions with very bad scores were found to have errors in the ground

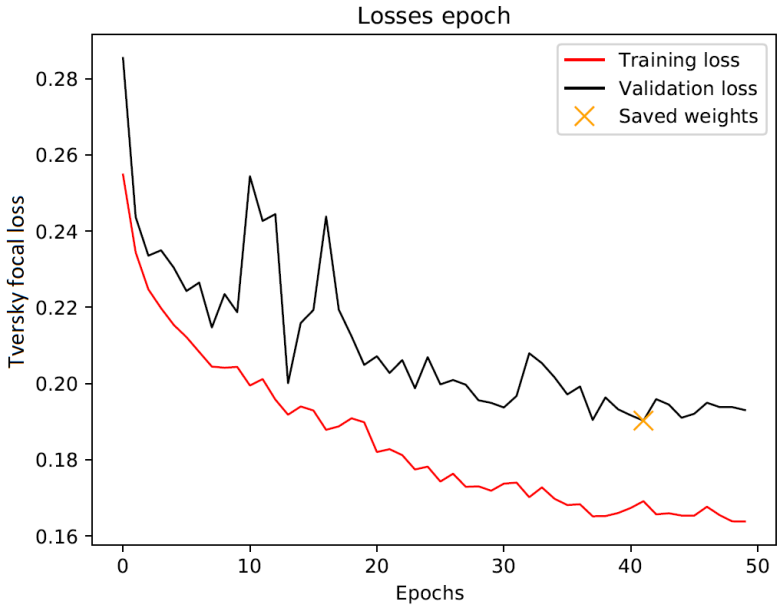


Figure 8.21: *The average mini-batch Tversky focal losses for the validation data for the Pre-trained FLAIR only, [Large data] axial slices, ReLU.*

Table 8.13: *Lesion prediction metrics. There is not much change in the larger lesions, but the pre-trained have higher false negatives in the second smallest lesions bracket. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	50	615	4226
Avg. depth	1	5	20	40
Total	10407	3868	121	135
Total TP*	3082	3100	120	135
Total FN*	7779	1241	2	3
Total FP*	7325	768	1	0
Avg. recall	-	0.72	0.91	0.92
Lesion precision	0.30	0.80	0.99	1.0
Lesion recall	0.28	0.71	0.98	0.98

truth masks. The false-positive, which was later found to have faulty ground truth, might indicate that the model has some level of generalization.

8.14 3-slice as channels FLAIR only, [Large data] axial slices with Mish

In this experiment, we trained from scratch using the mish activation function. As shown in Table 8.15 below, we do not have a large false positive and one less false negative than the pre-trained model with ReLU.

Results F_3 score: 0.74. The lesion metric is shown in Table 8.15. In this experiment, we have no large false positives, since we know there is at least one false positive which is actually true positive(as discussed previously and in Section 9.7 it may suggest that this generalize less than the other models that had one large

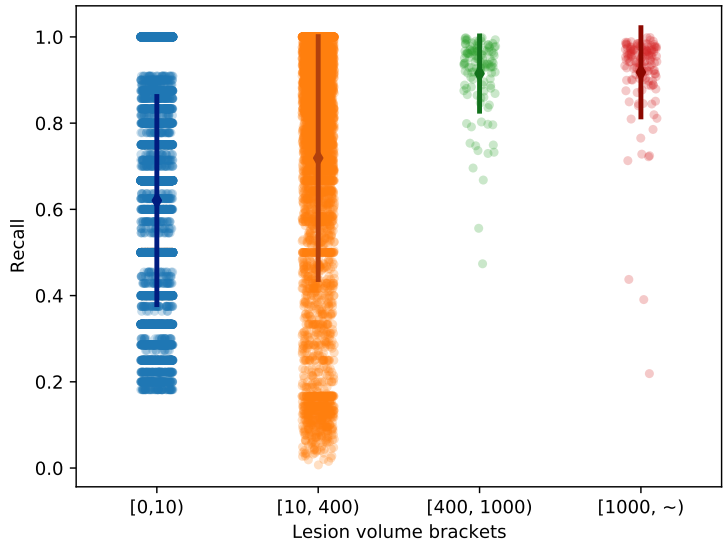


Figure 8.22: *This is a visualization of Table 8.13. There is not much difference between this and Figure 8.24, but the largest lesions is slighty worse.*

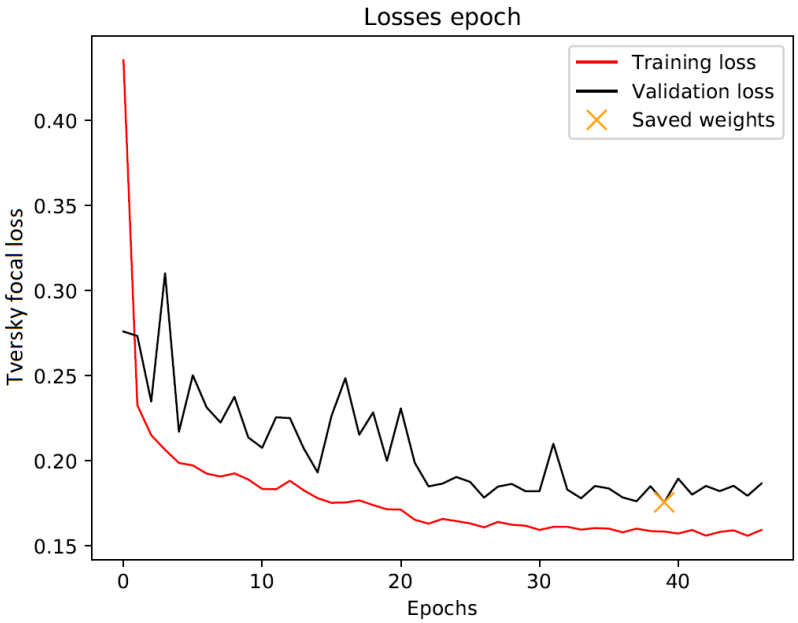


Figure 8.23: *The average mini-batch Tversky focal losses for the validation data for the 3-slice as channels FLAIR only, [Large data] axial slices with Mish and augmentation experiment.*

false positive. If the training data contain the same types of errors, it could be that this model is more prone to overfitting to these errors. The losses is shown in Figure 8.25. Even though the saved model is very close to the last epoch, which might suggest that we could find a better model by adding more epochs, it most likely won't as the loss seems to plateau in the last 15 epochs.

Table 8.14: *Lesion prediction metrics. There are one large false positives and a total of 4 large false negatives. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	52	621	4371
Avg. depth	2	6	21	41
Total	9331	4200	121	136
Total TP*	3399	3360	120	136
Total FN*	7462	981	2	2
Total FP*	5932	840	1	0
Avg. recall	-	0.77	0.93	0.94
Lesion precision	0.36	0.80	0.99	1.0
Lesion recall	0.31	0.77	0.98	0.99

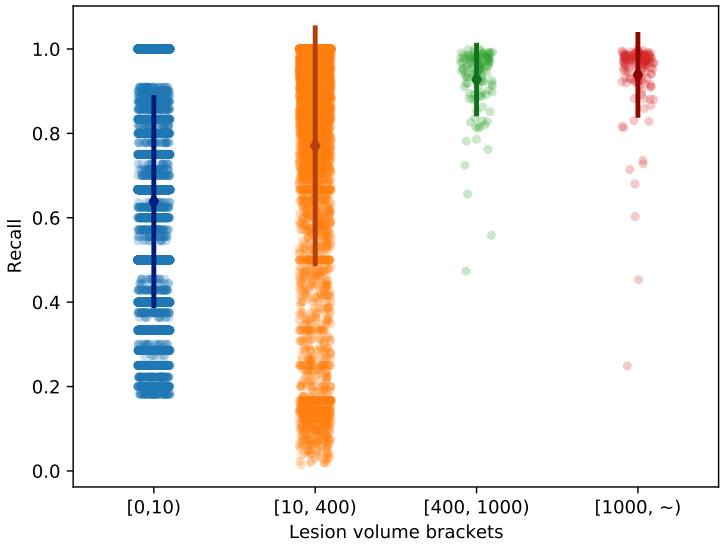


Figure 8.24: *This is a visualization of Table 8.14. The second smallest lesion bracket has the highest standard deviation and also has a lot of lesions at very low recall. Most of the highest lesions have a high score, but there are some outliers with a bad score. We suspect that systematic error is inside the data as the highest lesions have a higher standard deviation than the second-largest lesions bracket.*

8.15 3-slice as channels FLAIR axial slices with Mish and augmentation, with some of the largest GT errors removed

The ground truth in some volumes was not correct, as we found out in the previous section. The large false negatives and the large false positives did not have the correct masks. These were removed from the dataset and then continued training using the weights from Section 8.14).

The lesion metric is shown in Table 8.16 and the visualization of the table is shown in Figure 8.27. The loss plot is shown in Figure 8.26. As expected, the large false negatives and the false positive is gone as we removed them from the dataset. We can see that the scores, overall, are better for all lesion brackets.

Results F_3 score: 0.76. The lesion metric is shown in Table 8.16.

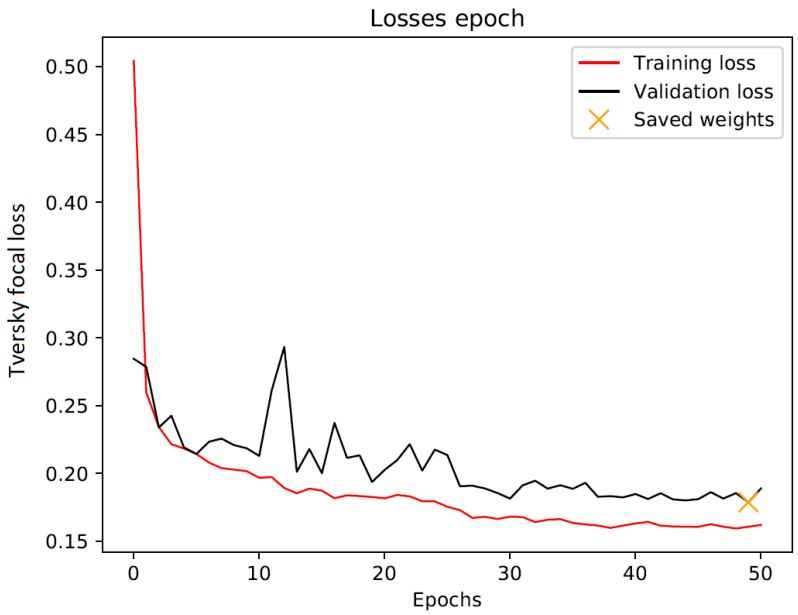


Figure 8.25: *The average mini-batch Tversky focal losses for the validation data for the 3-slice as channels FLAIR only, [Large data] axial slices with Mish experiment.*

Table 8.15: *Lesion prediction metrics. In this experiment, we have some large false-negative lesions. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	51	618	4062
Avg. depth	2	5	21	40
Total	9877	4154	121	135
Total TP*	3324	3304	121	135
Total FN*	7537	1037	1	3
Total FP*	6553	850	0	0
Avg. recall	-	0.76	0.93	0.93
Lesion precision	0.34	0.80	1.0	1.0
Lesion recall	0.31	0.76	0.99	0.98

Table 8.16: *Prediction lesion metrics. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the validation data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ∼)
Avg. voxels	3	52	626	4439
Avg. depth	2	5	20	41
Total	10514	4178	116	133
Total TP*	3646	3329	116	133
Total FN*	6790	881	0	0
Total FP*	6868	849	0	0
Avg. recall(smooth)	-	0.79	0.95	0.96
Lesion precision	0.35	0.80	1.0	1.0
Lesion recall	0.35	0.79	1.0	1.0

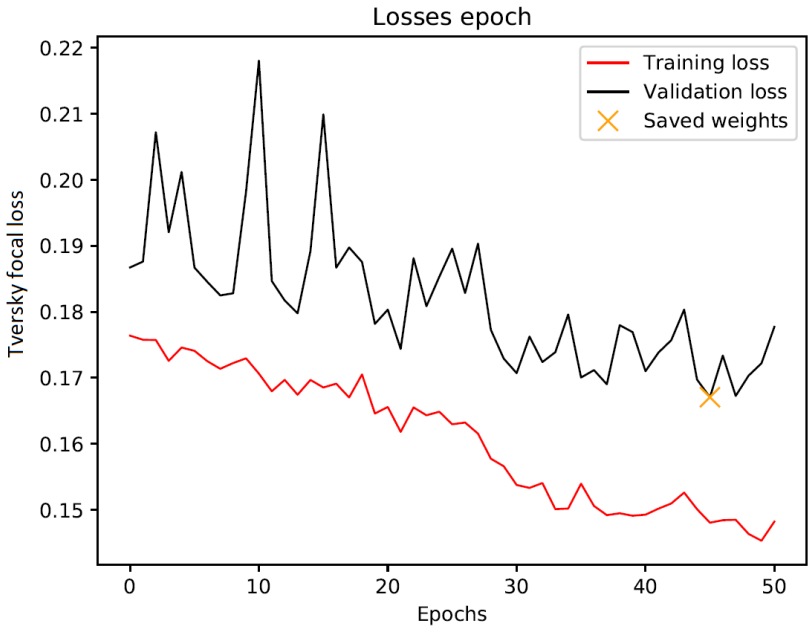


Figure 8.26: *The average mini-batch Tversky focal losses for the validation data for 3-slice as channels FLAIR axial slices with Mish and augmentation, with some of the largest GT errors removed experiment. The validation loss oscillates a lot. The best model was found close to the max epoch of 50, indicating that a better model could be found by increasing epochs.*

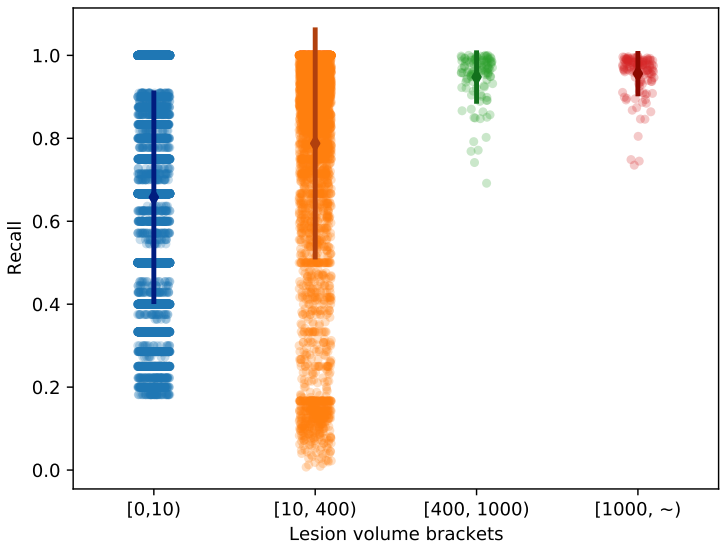


Figure 8.27: *This is a visualization of Table 8.16. The bad recall scores in the two largest lesion brackets are now obviously gone.*

8.16 Test results [Large data]

Using the model from Section 8.15) we get **results** F_3 score: 0.74. Lesion metrics are shown in Table 8.17 and visualized in Figure 8.28. Some of the bad/corrupted annotated examples leaked into the test dataset when the

dataset was split into training, validation and test. This is further discussed in Section 9.7).

Table 8.17: *Prediction lesion metrics for the test data. As expected, we have some large false negatives and false positives as we have not* repaired *the test data. The underlying issue was separated into training, validation, and test data. Avg. recall is calculated using the pixels inside the lesions bounding box and then average over them in all the corresponding lesion brackets. These scores are from the test data.*

Lesion Size	[0, 10)	[10, 400)	[400, 1000)	[1000, ~)
Avg. voxels	3	52	629	3826
Avg. depth	2	5	22	37
Total	9420	3895	117	97
Total TP*	3144	3103	116	97
Total FN*	6159	734	1	3
Total FP*	6276	792	1	0
Avg. recall(smooth)	-	0.80	0.94	0.93
Lesion precision	0.33	0.80	0.99	1.0
Lesion recall	0.34	0.81	0.99	0.97

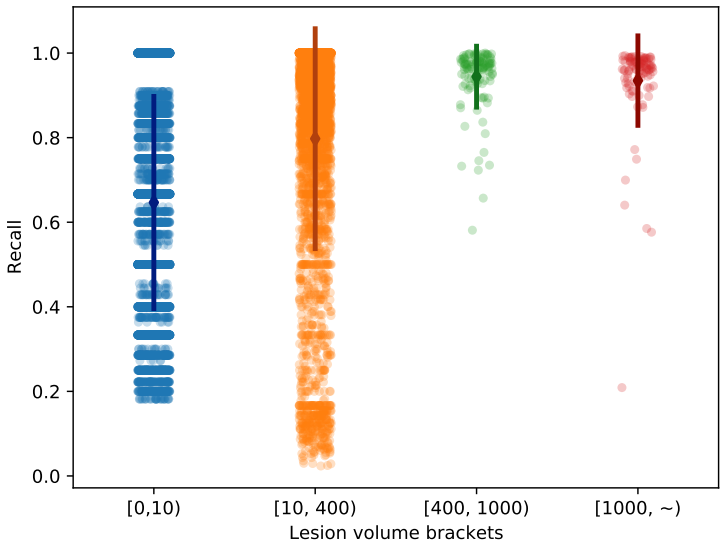


Figure 8.28: *Some bad results in the two largest lesions bracket in the test data. This was expected as the initial split into training, validation, and test added a few faulty annotated examples to the splits. The test was the only part that did not get any corrections, hence, the scoring may therefore be better than these results imply. It is interesting to note that the model can classify well on the few examples with faulty ground truth.*

9 Discussion

In this section, we discuss the important parts of the experimentations. The results, in general, seem to be very good, at least for the largest lesions. The smallest lesion bracket has such small lesions (1-4 voxels) that many of them are most likely errors in the annotations. Annotation errors were discovered during a review of results by an expert.

9.1 Data imbalance

Data imbalance can cause a bias in the model towards the data which is encountered the most during training. Table(5.2 and Table 5.4 show that the data does not contain any volumes with zero lesions. This can make the model biased towards these types of patients. The model might be overconfident in a real-world setting, and therefore, segment lesions on volumes with zero lesions or predict very unexpectedly, as the data might be outside the trained domain. This assumes that volumes with lesions have some underlying bias that spans over the whole volume or interacts with many parts throughout the brain. We never experimented with full 3D models, which might reduce this bias as the models trained with random 2D slices. Hence, we assume that this bias is negligible in our models. Since the test data did not have any MRI volumes with zero lesions, the experiments could not test for this issue.

The lesion-to-background ratio is very high in all the slices and volumes. This was balanced more using the focal Tversky loss discussed in Chapter 4.2.

The examples predicted with high confidence and are correct (high Tversky score) yield a small gradient causing small changes to the weights. Examples with bad predictions yield higher gradients, causing harder examples to be more in focus during training. This could also lead to better generalization as lesions predicted well enough will not be overfitted. The way the backpropagation is done in the experiments do also punishes the smaller lesions since a mini-batch size of 12 is used and is flattened into one large image, and then the loss and metrics are found. This will make the smaller lesions less important if the very large lesions are in the mini-batch.

Taking the average for all single slices could punish too much, as the smaller lesions are much harder to segment correctly. Adding ground truth errors in the mix and the optimization ends up in a tug of war between the large and small lesions, although here, the smoothing parameter Ω could help. The mini-batch loss could most likely be handled better than this way since the second lesions bracket might be punished too much, as seen in the lesion metric visualizations in Figure 8.22.

9.2 Change of orientation

To enhance the prediction metrics, we tried to change the input slice orientation. We had to pad the uneven dimensions to 256 for it to fit the model. The padding method was done using the minimum value in the slice instead of zeros after z-standardization, as zero's have a new meaning after standardization. Some volumes had larger width than 256, which were center cropped to 256. Cropping could lead to the outer parts of the brain being at the border. In [39] they showed that different border values impact the convolutions. During experimentation, the model seems to perform the same for all orientations, but during inference, the input has to be on the same orientation as it was during training to perform well. This is because the feature structure is different for the three axes. The experiments show that using different orientations did not seem to increase the results.

9.3 Batch normalization and standardization

In these experiments, we have z-standardized over the patient volumes. Thus, when predicting, we must have the whole volume, not single-sliced images, even if single/three slices are used in prediction. This is not a problem if the use case is always predicting a whole volume. If single image prediction is needed, a retrained model with a new normalization or standardization method is needed. In these experiments, the data was z-standardized over each patient volume.

Batch normalization is a method that usually needs a large batch size to work well, and some literature suggests using batch renormalization when using small-batch sizes[47]. One interesting point is that the batch normalized parameters are learned from batches of different iid slices, while in deployment, the batches are in sequence over the same MRI volume, changing the dynamic of how the standardization works. In training, we used entropy to remove images with little information. Therefore the parameters in batch normalization (and in the network in general) are learned by only looking at images with a decent amount of information.

For the batch normalization parameters, the mean and standard deviation is from feature maps with a lot of information, while during inference, the feature maps in the first and last parts of the MRI sequence do not contain any or very little information. One other interesting thing to note is that, when using the pre-trained weights, the batch normalization parameters are learned from the pre-trained data, which might cause some problems during transfer-learning as the data could be very different.

9.4 Generalization problems

In the first set of experiments, the data from one machine was used. Different MRI machines can have different noise properties, and image contrasts even for the same type of image sequence (e.g. FLAIR). By only training using data from one machine, the model could develop a bias towards these properties. As shown in the entropy figures in previous sections, the entropy distributions are very different in the two training datasets. The entropy figures are also shown together to compare in Figure 9.1.

We also performed an experiment where we used the model trained on the small training dataset, which contained only MRI volumes from one machine, to test on the large validation dataset. The lesion metrics are shown in Figure 9.2. As we can see in this figure, the model did not perform all that well. Some examples of large false negatives from this experiment are shown in Figure 9.5. In this figure, the model has problems with many large lesions, even though they er fairly bright.

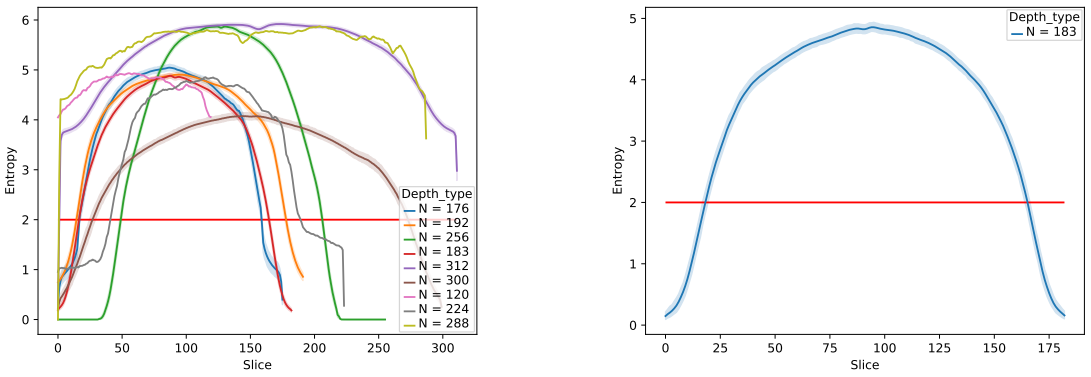
False positives are shown in Figure 9.6. Here the prediction was not in the brain at all. The reason for this is, not only that the small dataset is very small, but since they are all from the same machine, the resolution difference will have a strong impact. The voxel size in the small dataset is the same for all volumes and is larger than the average voxel size in the large dataset. I think much of this issue would be resolved with proper scale augmentation.

We also did one experiment where we inferred on the small validation dataset with the model trained with the large dataset in Chapter 8.15. In this experiment, we see that the model performs really well as it finds almost all the lesions. The lesion metrics are shown in Figure 9.3. In this figure, we can observe that the large lesions are predicted really well.

An important thing to note is that the small dataset, only containing one resolution, will overfit the receptive field for this resolution. For instance, if the model is only trained on MRI volumes with a given resolution, the model will only learn the receptive field containing this given resolution. If the resolution is low, the context within the receptive field will look at a broader region but with fewer details.

If the resolution is high, then only finer details are within the receptive field. Since the model trained on the larger dataset contains different resolutions, it will generalize more for different resolutions. As mentioned, UNet has skip-connection, which will add a range of receptive fields, causing the model to have the ability to learn and understand many different resolutions.

Dropout layers were not used in any of these experiments because one of the earliest experiments did not perform well when applied dropout to all layers. Because of various implications in the code at that time and that Tversky focal loss also helps in avoiding overfitting, made us not include the experiment. In retrospect, we should have looked more into the use of dropout layers.



(a) *Entropy from the large training dataset.*

(b) *Entropy from the small training dataset.*

Figure 9.1: *Entropy for the two training datasets used in experiments. There are many different entropy distributions in the dataset with samples from many different MRI machines.*

9.5 Dataset difference (introduction of larger dataset)

The large validation dataset is almost 3x the size of the smaller training dataset. In Figure 9.7 we can see the Kullback-Leibler divergence between the small training data and the large validation data from the two datasets. In Figure 9.8 the large training data and the large validation for the variance distributions $N = 60$,

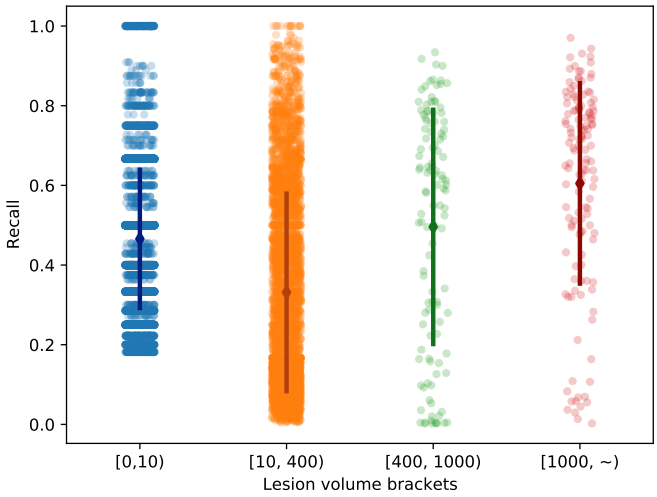


Figure 9.2: *Lesion metrics from the model in Chapter 8.9). This is the model trained on the small training data and inferred on the large validation data. The score was* $F_3 = 0.40$. *The mean recall for all except the largest bracket is below 0.5.)*

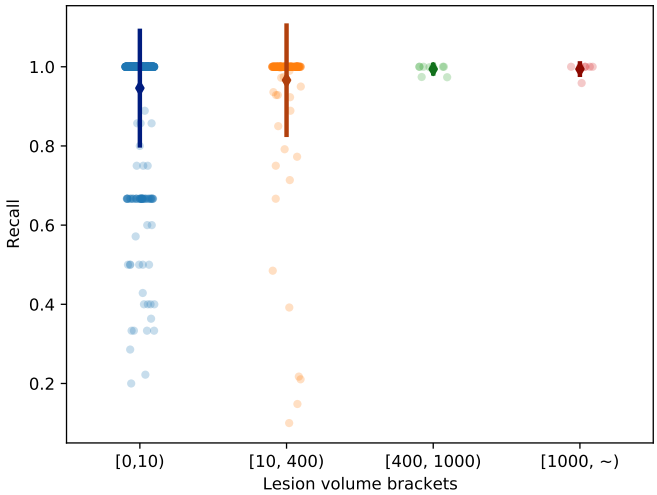


Figure 9.3: *Lesion metric from the model in Chapter 8.15. This is the model trained on the large training data and inferred on the small validation data. The model performs well on the small validation. The score was* $F_3 = 0.64$. *The score is low compared to the test data because this model seems to over segment the lesions in the small validation dataset. Examples of over-segmentation is shown in Figure 9.4.*

except the old validation data which only had $N = 9$ volumes is shown.

In Figure 9.9) and Figure 9.10, we can see the same but with the small validation data instead. In the two first figures, we can see that the small training data lacks a lot of the information contained in the new validation data, with emphasis on the second-order moment. The difference in datasets can also be observed in the 2D slice entropy plots in Figure 8.18 and Figure 7.1. What these divergence distributions can tell us is

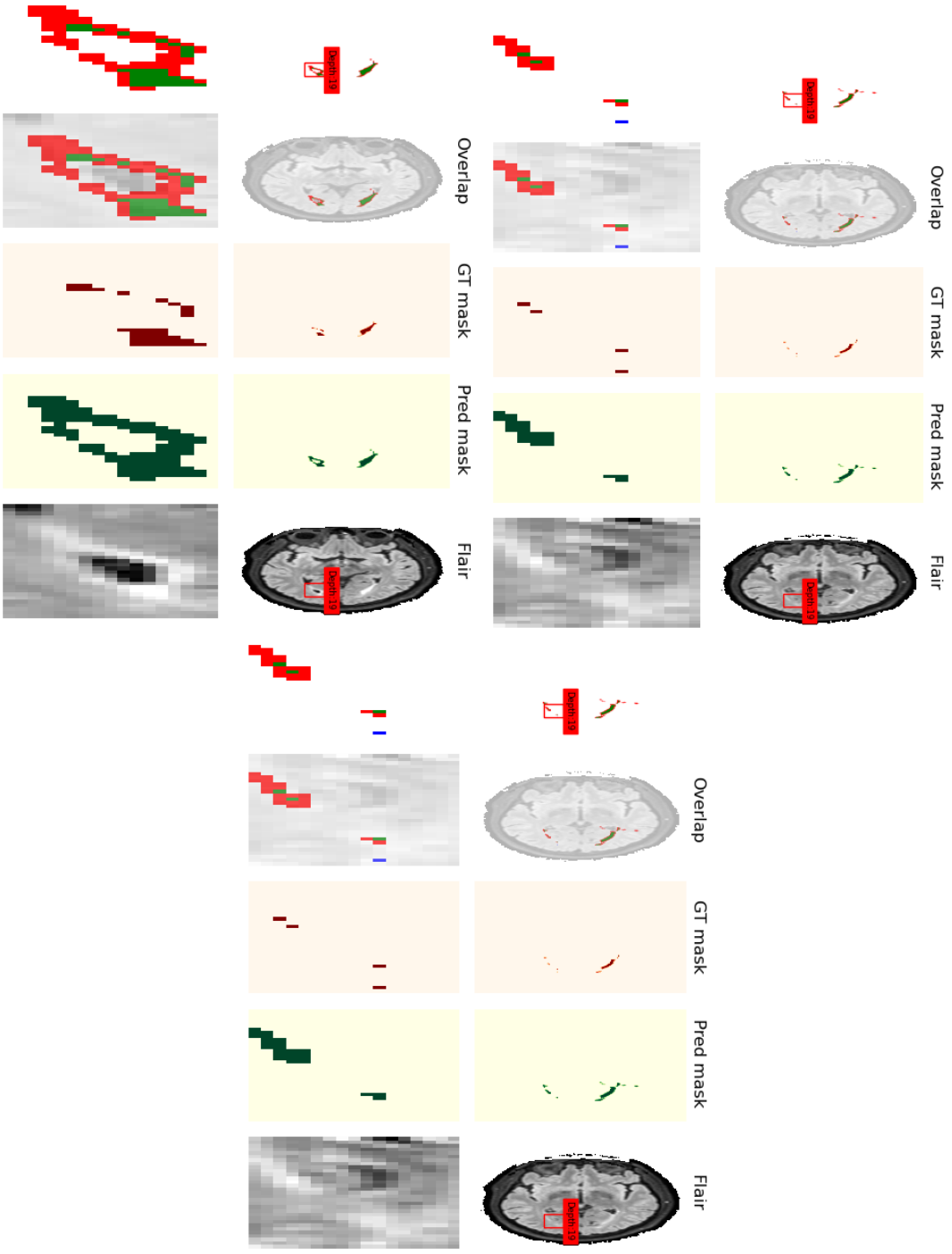


Figure 9.4: *Examples from the small validation data predicted by the model from the large training data. It seems that there are many cases of bad ground truths as the model seems to segment more of the bright regions than the ground truth. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

that the small dataset seems to lack the information to understand the larger dataset. We also see the entropy difference in Figure 9.1.

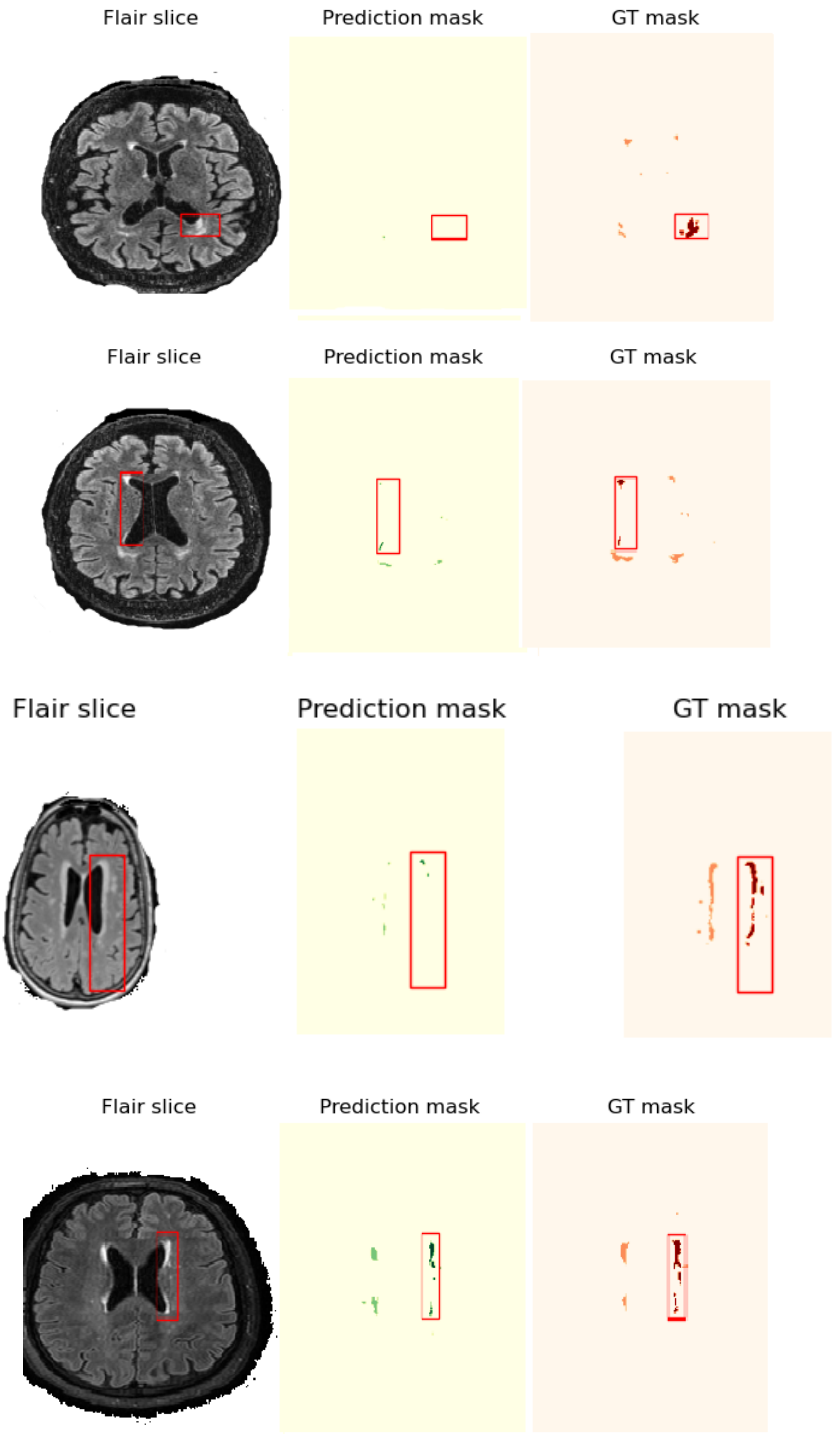


Figure 9.5: *Example predictions for the model from Chapter 8.9 (best results from small dataset experiment) on the large validation data. The model seems to have a hard time with some of the less bright regions. The second last image had almost zero predictions in the 2D slice. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

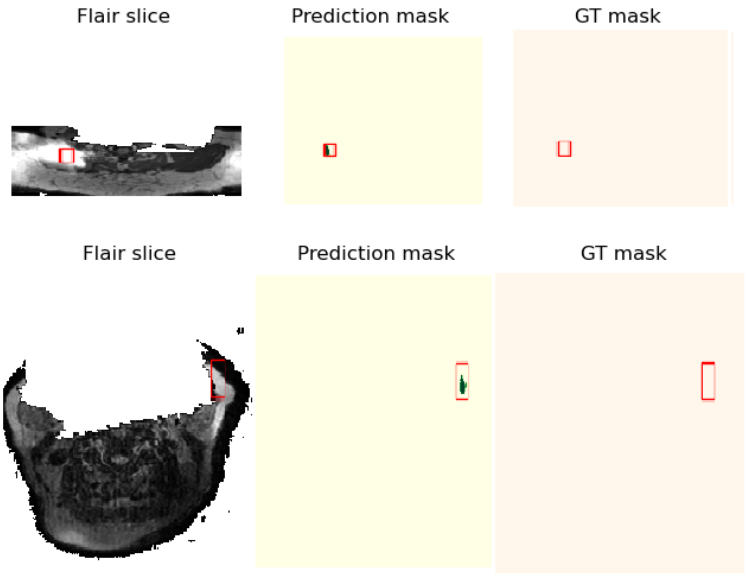


Figure 9.6: *In this example the false positives are completely off. The very bright region caused the model to predict lesions outside the brain. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

9.6 Difference between dice metric and lesion metrics

The dice score is found by taking the harmonic mean between precision and recall, which is based on the per-slice input (in these experiments). If a slice contains two positive pixels and none are predicted, the score will be zero if we use the non-smoothed version. In practice, finding only a two-pixel wide lesion is not very important. Hence, looking at the lesion metric by volume size is more important. One good finding of the model is that it is much better at segmenting the large lesions. Dice score also weights the recall and precision the same, which is clinically not as important. This is why the F_3 scoring is a better metric, but it does not say anything about the size of the lesion, which is even more important. A lesion volume size metric-based loss would be the best choice, although this would require a 3D-based segmentation model. It is important to note that there is some resolution difference for the voxels in the MRI volumes. This is good for the generalization of the model but impacts the understanding of the lesion metric to some degree. Hence, the average lesion voxels can be a little bit different in SI length units.

9.7 Errors in dataset

In Table 8.14 and Table 8.15 one large false-positive lesion with volume of 600 was predicted. The slice is shown in Figure 9.11. As we can see in this figure, the model predicted fairly large lesions which are not in the ground truth. After consultation with a professional, this was shown to be a lesion. The ground truth mask was also faulty in general. A well-trained model over a large training data can therefore be used to find outliers, which can later be inspected to help find an error in the ground truth data.

The experiment in Chapter 8.12 and Chapter 8.13 predicted the false negative, which is falsely annotated, which might suggest that these models are more generalized.

In Figure 9.12 we show some of the **large false-negative lesions**. Here we can see that the ground truth

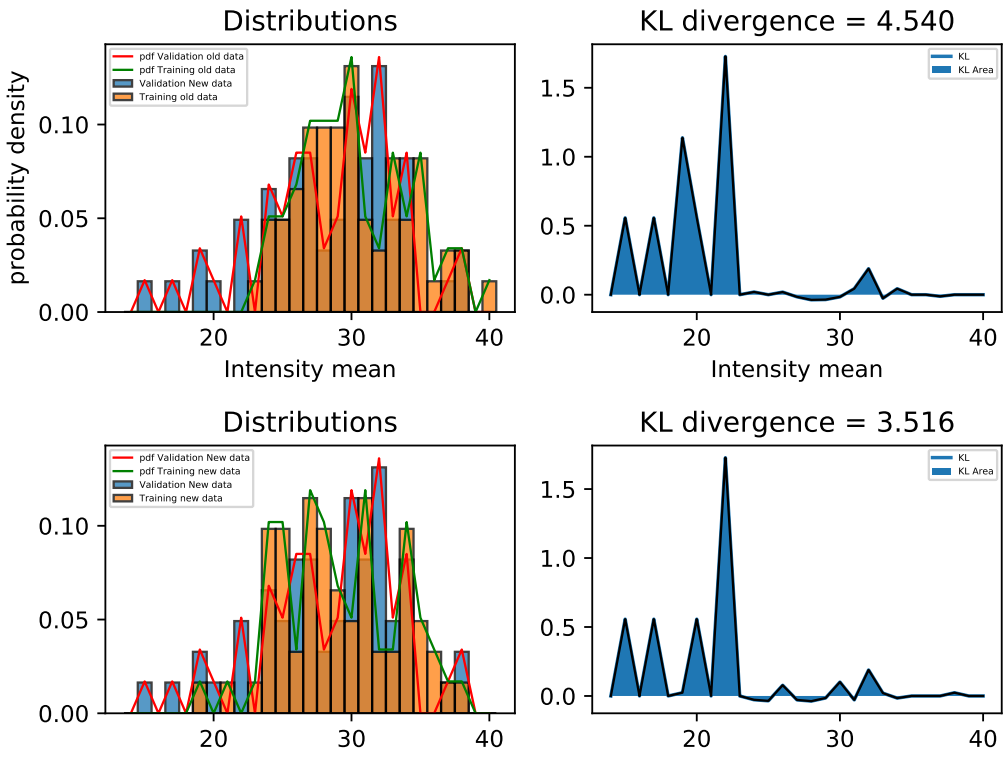


Figure 9.7: *Distribution of 60 patient volume sample means. KullbackLeibler divergence between the distributions. We can observe a larger difference between the small training data and the large validation data(first row). The large training data is much larger and, therefore, expect the large training data to overlap the distributions much more than shown (last row).*

masks are over-segmenting and not correctly placed according to the FLAIR 2D slice. The model predicted the correct places very well, but the ground truth annotation was wrong. This is an interesting example of how we can use the model to detect wrong annotations. If we assume that most of the training data is correct, the wrong annotated examples will end up as outlier predictions under validation/testing.

It is expected to have such errors when datasets are very large. What is most important is that after the data has been split into training, validation, and test that the test data is checked as much as possible before testing. In this thesis, this was problematic as the data must be inspected by a domain expert, which can be expensive and time-limited.

Some **bad results in testdata** was expected as we could not check for bad ground truths. As seen in the lesion metric table, we had some false negatives and false positives. This was expected as the errors which were in the whole dataset leaked during the dataset split. One slice of the false-positive seen in Table 8.17 is shown in Figure 9.13.

9.8 Training and prediction for different image orientations

During training and evaluation, the sagittal orientation was used. In the sagittal orientation, the throat is always inside the slice when most of the brain is visible, as seen in Figure 8.7. This could act as noise to the network, which the network has to learn and understand. Looking at the sagittal orientation as shown in Figure

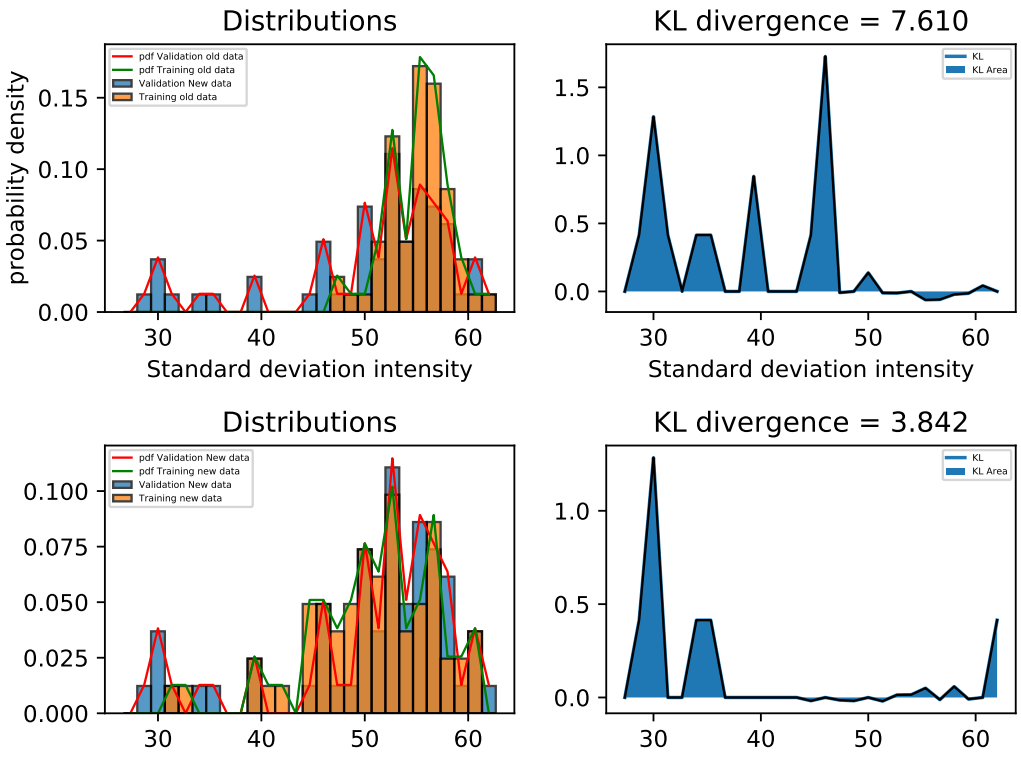


Figure 9.8: *Distribution of 60 patient volume sample standard deviations. KullbackLeibler divergence between the standard deviation distributions. The small training dataset has a very high divergence from the large validation data. The seconder-order moment has a larger divergence than the first-order moment. This indicates some fundamental statistical differences between the datasets.*

9.11, we see it has less noise surrounding the brain. The model might perform better if we zero pad the uneven dimension and use a sagittal orientation instead for both training and prediction.

When performing a prediction in deployment, this zero padding has to be removed to fit the original volume so that it can be overlayed correctly later. After experimentation changing the orientation did not seem to change the results. It is very important to keep the input consistent because prediction on the sagittal orientation when trained on axial orientation changes the prediction results considerably.

We also experimented with a 3-slice approach to give the model more context for prediction. In these experiments, we only used three slices for *one* orientation. We could also experiment with three slices where each is from their unique orientation to get 3D context.

Future work also includes data from more machines to further increase the generalization.

9.9 Future work

On the topic of medical segmentation, there is a lot of important research. The question of **explainability** is one of these. In Figure 9.15 an occlusion experiment was done. Occlusion testing is using some geometry to block an area of the image before prediction to see what the model weights the most during inference on a given input. Occlusion can also be used as an augmentation method[48]. In Figure 9.15 the blue areas are regions that decreased the number of lesions predicted from the original prediction. A positive value in red is

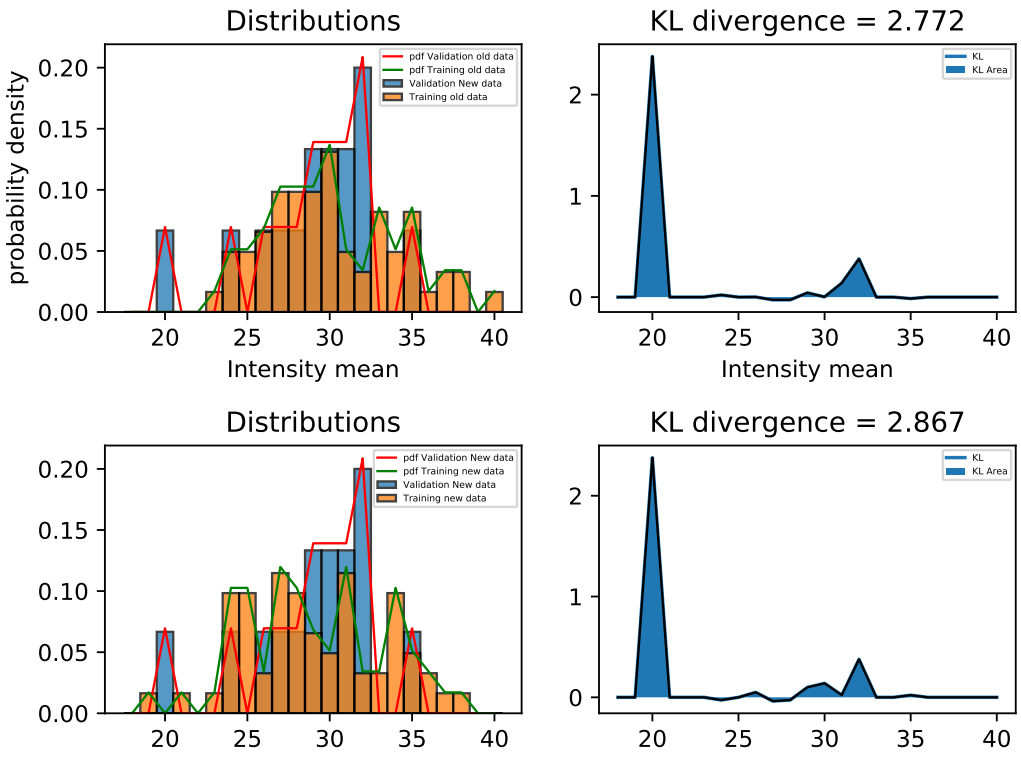


Figure 9.9: *Distribution of 60 patient volume sample first-order moments. KullbackLeibler divergence between the distributions. Here the validation data is from the small dataset. We can observe that the large training dataset is very close to the small validation data. The training data from the large dataset is much larger as well. Hence, the distribution will cover much more than shown in this figure. The small validation data does only have about 9 MRI volumes, therefore, this comparison is not that good. The divergence is mostly the same, we could expect the large training data to perform the same or better on the small validation data as the small training data.*

the regions that add more lesions than the original prediction. An example where more lesions are predicted is shown in Figure 9.16.

Another work is to make the models could be better in general. The second smallest lesion bracket as seen in Figure 8.17 from the test results could need some tuning. The recall score is generally unstable and should be increased. One problem with the small lesions is more easily lost in the cascading convolutions as they are so small. One idea is to have two different architectures, one for larger lesions and one for the smaller ones. Newer deep learning segmentation architectures can also be tested. The augmentation can also be changed, as it was done in [4], where they used sheering, rotation and scaling. One problem with augmentation like rotation, sheering, and scaling is that it has to do some interpolation which could cause errors in the ground truth.

In these experiments, we did not tune hyperparameters to the maximum either. By using new deep learning libraries that can help in hyperparameter tuning, such as Tune[49], we could enhance the model.

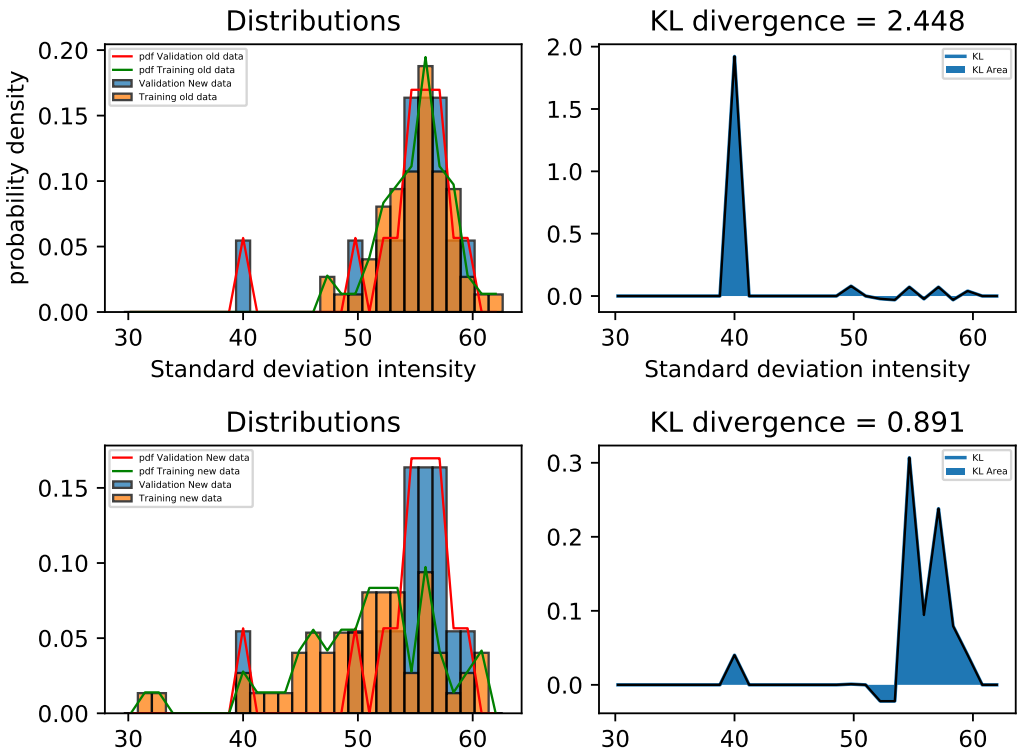


Figure 9.10: *Distribution of 60 patient volume sample standard deviations. KullbackLeibler divergence between the distributions. Here the validation data is from the small dataset. We can observe that the large training data is very close to the old validation data. The training data from the large dataset is much larger as well. Hence, the distribution will cover much more than shown in this figure. The small validation dataset does only have about 9 patients, therefore, this comparison is not very good but could give an expectation in performance. The divergence between the second-moment distributions is much lower with the large dataset, and since the small dataset mean divergence is about the same, we could expect the same or better performance on the small validation after training on the new data.*

Figure 9.11: *This false positive was later discovered to be a true positive by a domain expert. The bounding box shows the size of the lesion at the maximum. This also fits our suspicion discussed in Figure 9.11. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

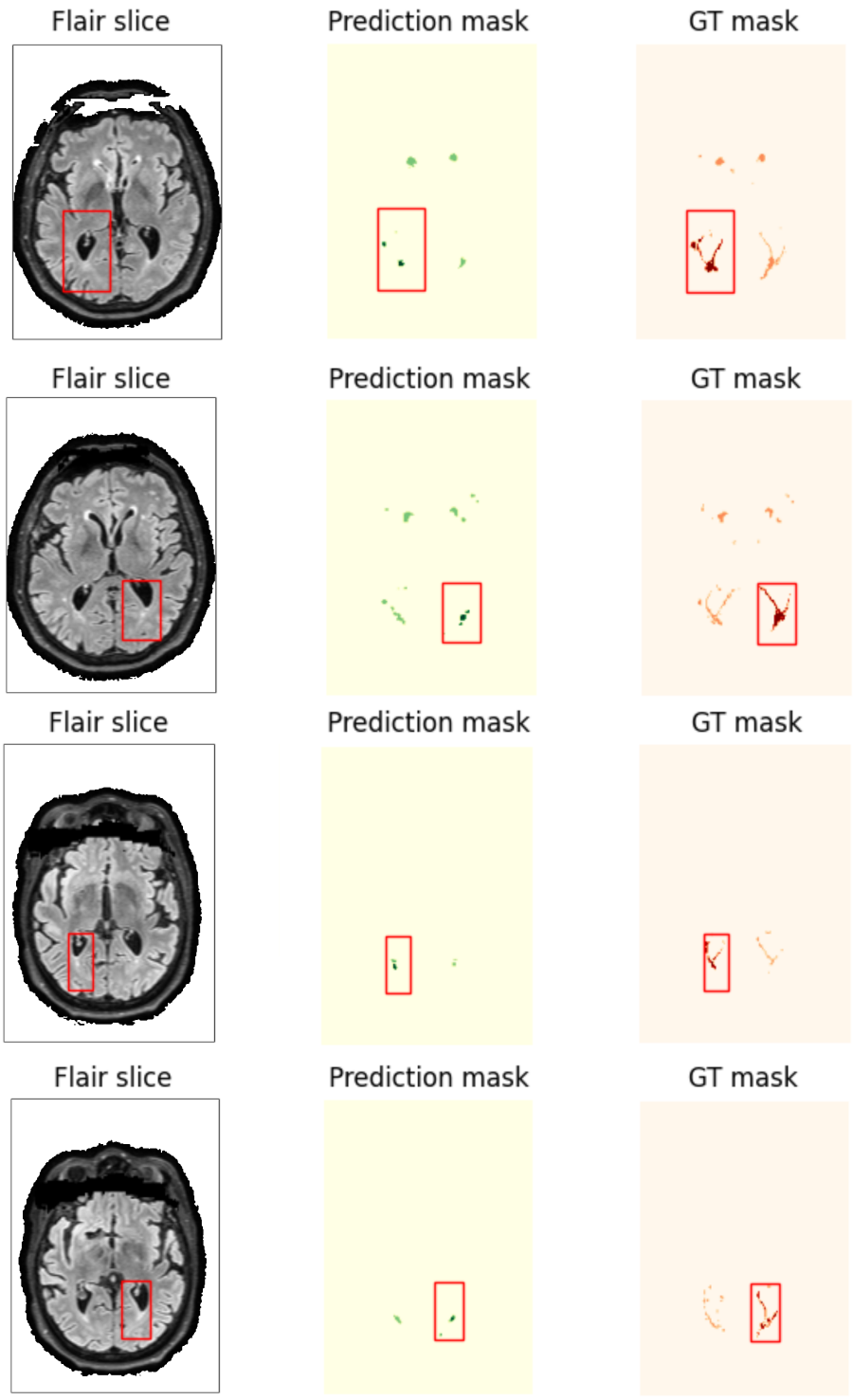


Figure 9.12: *In many of the experiments, the model did well but some large lesions ended up as a false negative since some of the ground truth masks were not correctly mapped to the volume. We did the same lesions metric analysis on the training data to see if we could find errors there as well, and we did. After consultation with an expert, we removed some of these examples from the dataset. After removal of these examples, the model was retrained with a dataset labeled* fixed large dataset*. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

Figure 9.13: *The model had one large false positive as seen in Table 8.17. In this figure, we see one slice of the large lesion. We can observe that the ground truth is corrupted/wrong. After consultation with a domain expert, this was found to be lesions predicted correctly by the model. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

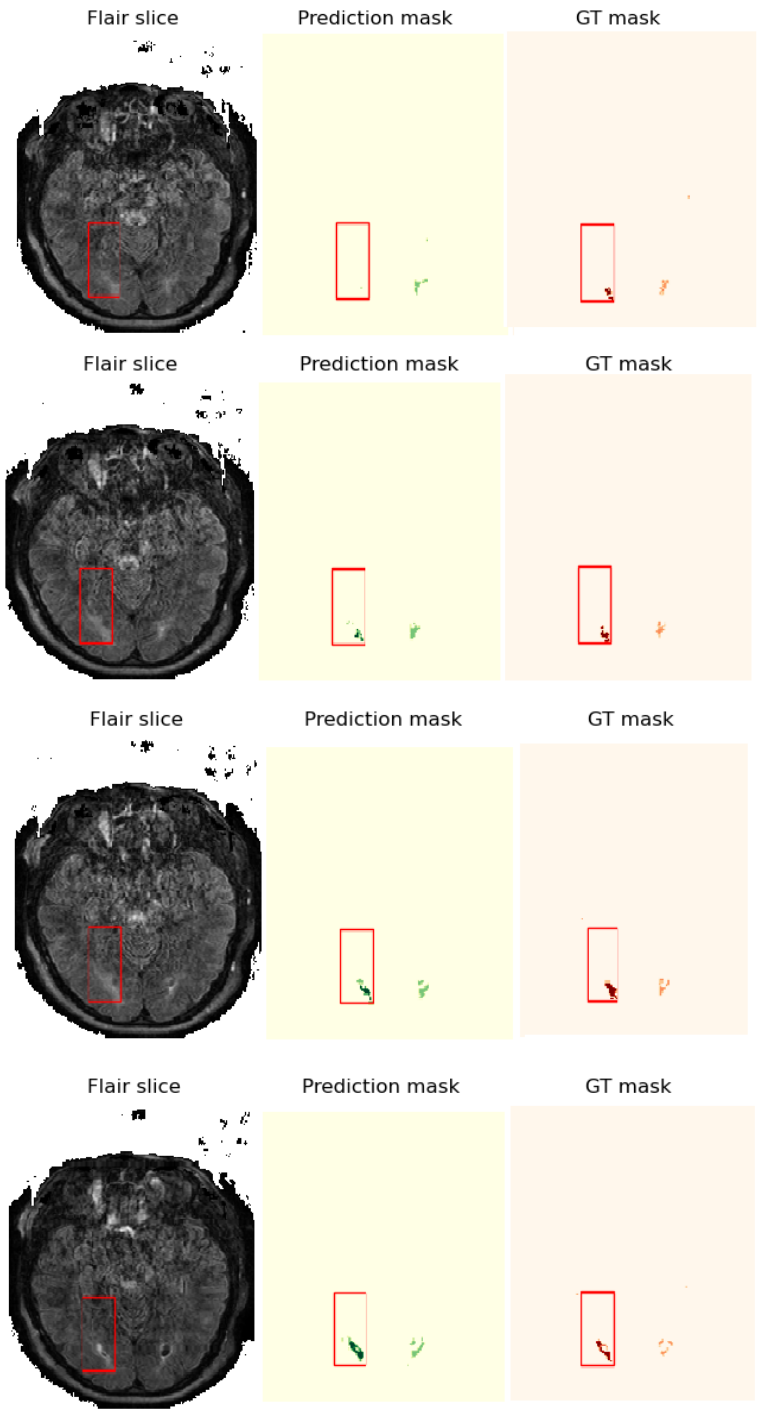


Figure 9.14: *Example of large false-negative lesions. This is a sequence in the MRI volume. In the top example, we can see that there were no predictions. Faint areas like these made the lesions predicted as false negative as only parts of the ground truth lesion was detected. In the sequence, we can observe that the lesion is still fairly well predicted. Using the a priori knowledge of errors in the data, it can be assumed that some of these large lesions have faulty ground truth, causing the prediction to be classified as false negatives. The aspect ratio is a bit off, causing some stretching effects on the slice images.*

Figure 9.15: *Ablation test for the whole image. Parts in blue show negative predictions from the baseline. The baseline is the number of lesions predicted on the original image. In this test, we used a square by the size of 25x25 to block out an area of the image before prediction. The values in the square are the minimum values of the image (values outside the head).*

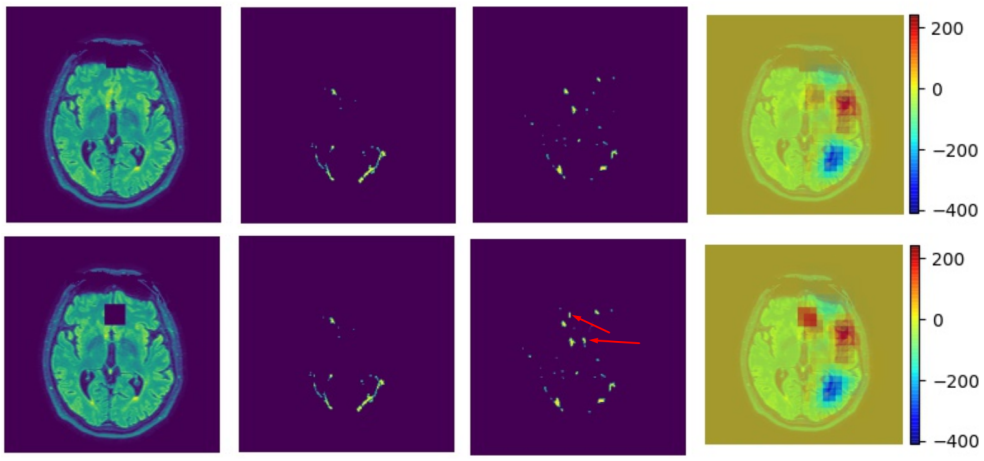


Figure 9.16: *This figure shows as the box is moving inside the brain tissue. In the image, to the left, the box is outside the brain. In the figure to the right, we can see the blocking of the brain leads to more predictions. Most of the lesions are around the ventricle, which is mostly the dark areas in the brain. When blocking one region and there are bright spots around the blocked square the model seems to have high confidence that there are lesions.*

10 Conclusions

In this thesis, we have experimented with different UNet architectures for white matter hyperintensity segmentation on MRI slices. By leveraging the power of Tversky focal loss, the models obtained a good generalized F_3 score. Two main UNet architectures were tested, Regular UNet and Attention UNet, with and without pyramid input. The use of pre-trained weights, the use of Mish activation function, and horizontal and vertical flip data augmentation were tested. By using F_3 as the main metric and the lesion recalls, we found the best model to be *data augmented, mish activation with three slices in axial depth as input.* In our experiments, we only used 50 epochs as the maximum, which in some experiments showed to be too little as the *best model* found during the experiments were very close to 50 epochs, which might suggest that a better model were still ahead. The experiments showed that it was not a very big difference between all the models.

The best model received a $F_3 = 0.74$ on the test data, where the large and most important lesions had an average recall score of 0.93. The smaller lesions were harder to segment correctly due to annotation errors, and that the region of interest is very small. The model could most likely be further enhanced by using more proper augmentation methods like scaling, rotation and use a more up-to-date segmentation model, and of course, more data.

Some flaws in the ground truth were discovered during experimentation as some large lesions were predicted as false positives, even though they were true positives. This was discovered after consultation with a professional where the false positives were shown. Hence, more flaws in the dataset are expected. Experiments on different orientations show little changes in scoring, and since 2D slices were used for prediction, the volume has to be built up during prediction to be viewed from all orientations.

Medical data have problems with ground truth since professionals have to annotate the data, and even professionals can disagree. There must exist a GT (either there is a lesion or not a lesion) - but the GT may not be known. This makes the labels in medical data something called a **gold standard**, which can be problematic for a model to understand as the gold standards might deviate from an underlying global function/ GT. The experiments performed showed promising results with the use of UNet architecture for the segmentation of WMH lesions.

Appendix

Most used frameworks:
Deep learning framework Pytorch[50]
Numerical computations Numpy[51]
Plotting[52][53]
Classical image processing Scikit-image[42]

References

[1] C. Jack, D. Knopman, W. Jagust, R. Petersen, M. Weiner, P. Aisen, L. Shaw, P. Vemuri, H. Wiste, S. Weigand, T. Lesnick, V. Pankratz, M. Donohue, and J. Trojanowski, "Tracking pathophysiological processes in alzheimer's disease: an updated hypothetical model of dynamic biomarkers," *Lancet neurology*, vol. 12, pp. 207–216, 02 2013.

[2] A. C. Birdsill, R. L. Koscik, E. M. Jonaitis, S. C. Johnson, O. C. Okonkwo, B. P. Hermann, A. Larue, M. A. Sager, and B. B. Bendlin, "Regional white matter hyperintensities: aging, alzheimer's disease risk, and cognitive function," Apr 2014.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[4] H. Li, G. Jiang, J. Zhang, R. Wang, Z. Wang, W.-S. Zheng, and B. Menze, "Fully convolutional network ensembles for white matter hyperintensities segmentation in mr images," *NeuroImage*, vol. 183, p. 650665, Dec 2018.

[5] N. Sharma and L. Aggarwal, "Automated medical image segmentation techniques," *Journal of medical physics / Association of Medical Physicists of India*, vol. 35, pp. 3–14, 04 2010.

[6] B. M. Dale, M. A. Brown, and R. C. Semelka, *Pulse Sequences*, pp. 80, 81. 5 ed.

[7] R. C. Gonzalez and R. E. Woods, *Digital image processing.* 330 Hudson Street, New York, NY 10013: Pearson, 2008.

[8] E. Bendersky, "Convolution figures," 2021.

[9] H. Le and A. Borji, "What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks?," *CoRR*, vol. abs/1705.07049, 2017.

[10] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," 2017.

[11] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," 2019.

[12] Y. Wu and K. He, "Group normalization," 2018.

[13] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2017.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[15] Wikipedia, "Relu activation."

[16] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Neural Networks: Tricks of the Trade (2nd ed.)*, vol. 7700. 330 Hudson Street, New York, NY 10013: Springer, 2012.

[17] Wikipedia, "Sigmoid activation."

[18] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *CoRR*, vol. abs/1908.08681, 2019.

[19] E. Alpaydin, *Introduction to machine learning*. The MIT Press, third ed., 2014.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[21] "Adam optimizer at papers with code," March. 11, 2021. [Online].

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

[23] M. Hon and N. Khan, "Towards alzheimer's disease classification through transfer learning," 2017.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[26] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," *CoRR*, vol. abs/1810.07842, 2018.

[27] M. Yap, G. Pons, J. Martí, S. Ganau, M. Sentís, R. Zwiggelaar, A. Davison, and R. Martí, "Automated breast ultrasound lesions detection using convolutional neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, pp. 1218–1226, July 2018.

[28] Q. Yu, Y. Xia, L. Xie, E. K. Fishman, and A. L. Yuille, "Thickened 2d networks for efficient 3d medical image segmentation," 2019.

[29] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[31] A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. van Ginneken, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers, P. Bilic, P. F. Christ, R. K. G. Do, M. Gollub, J. Golia-Pernicka, S. H. Heckers, W. R. Jarnagin, M. K. McHugo, S. Napel, E. Vorontsov, L. Maier-Hein, and M. J. Cardoso, "A large annotated medical image dataset for the development and evaluation of segmentation algorithms," 2019.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[33] P. R. C.D. Manning and H. Schütze, *Introduction to Information Retrieval*. No. 260, 2008.

[34] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," 2016.

[35] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.

[36] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," 2017.

[37] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[38] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[39] B. Alsallakh, N. Kokhlikyan, V. Miglani, J. Yuan, and O. Reblitz-Richardson, "Mind the pad – cnns can develop blind spots," 2020.

[40] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[41] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018.

[42] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[43] M. Buda, A. Saha, and M. A. Mazurowski, "Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm," *Computers in Biology and Medicine*, vol. 109, 2019.

[44] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," 2018.

[45] A. Farooq, S. Anwar, M. Awais, and S. Rehman, "A deep cnn based multi-class classification of alzheimer's disease using mri," in *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–6, 2017.

[46] J. Nalepa, M. Marcinkiewicz, and M. Kawulok, "Data augmentation for brain-tumor segmentation: A review," *Frontiers in Computational Neuroscience*, vol. 13, p. 83, 2019.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[48] R. Fong and A. Vedaldi, "Occlusions for effective data augmentation in image classification," 2019.

[49] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[51] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[52] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[53] M. Waskom and the seaborn development team, "mwaskom/seaborn," Sept. 2020.