university of groningen / faculty of science and engineering

PET Tumor Segmentation Using a Deep Residual CNN with Dilated Convolutions

Bachelor's Project Thesis

Werner van der Veen, s2667088, w.k.van.der.veen.2@student.rug.nl,
Supervisors: Dr M.A. Wiering & E.A.G. Pfaehler, MSc.

Abstract: This research examines the effectiveness of using deep learning for medical image segmentation. Here, tumors are segmented from a dataset of three-dimensional PET-scans. A deep convolutional neural network is trained on this dataset, and approaches from natural image segmentation are tested, particularly residual connections and dilated convolution. Its segmentation performance is compared to that of a simple thresholding algorithm that segments based on voxel intensity values. Using a weighted Jaccard index metric and loss and a positive predictive value and sensitivity metric, the neural network appears to outperform the simple thresholding algorithm slightly but consistently. However, additional varied research is needed to corroborate the findings and the advantages of deep learning for medical image segmentation in general.

1 Introduction

Semantic segmentation is the process of partitioning meaningful objects from natural images. This technique is often used to filter out relevant information from an image. In recent years, semantic segmentation has found practical use in applications that are in active development, including autonomous navigation [1], augmented reality systems [2], and video surveillance [3]. This growing relevancy is linked to an increasing performance of segmentation methods, resulting from the development of more effective machine learning techniques. Starting in 2012, deep learning in particular became an increasingly popular approach to perform image segmentation, generally outperforming established techniques such as Random Forest–based classifiers [4][5]. In subsequent years, the techniques in deep learning–based semantic segmentation have improved considerably—and with it, the popularity and performance of its applications [6].

One paper that has proved an important contribution to the early successes of deep learning in semantic segmentation is named "Fully Convolutional Networks for Semantic Segmentation" (FCN) [7]. Its key contribution is the use of an end-to-end convolutional network that incorporates deconvolutional upsampling and residual connections to negate information loss from pooling. Since then, other papers have improved upon the innovation of FCN. Among them is a paper named "Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation" [5], which reduced the memory complexity of the residual connections, and "Multi-Scale Context Aggregation by Dilated Convolutions", [8], which replaced the pooling layers by dilated (or *atrous*) convolution layers. Later, a paper named "Pyramid Scene Parsing Network" (PSPNet) was published [9], which uses pyramid pooling and auxiliary loss.

These contributions have all improved the contemporary state-of-the-art performance on semantic segmentation datasets such as Pascal VOC2012 and MSCOCO. FCN reached an average precision of 62.2% on VOC2012 in 2014, whereas the best performing system at the moment of writing is "DeepLab v3" [10], scoring 89.0% on VOC2012 in February 2018 when pretrained on the JFT-300M dataset*.

However, the research on the use of image segmentation for medical purposes is still less matured than the research on segmentation in natural images. At the moment, the question whether the methods and results from semantic segmentation in natural images can be reliably extrapolated to medical imaging is being actively researched. For instance, a paper named V-Net explored the possibility of segmenting three-dimensional MRI scans using an implementation of FCN [11]. In another

*http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6

paper, bladders are segmented using deep convolutional neural networks on CT-scans [12]. In a comparative study by Hatt et al. [13], a method based on convolutional neural networks outperformed 12 other automatic functional volume segmentation methods in PET-scans. The aim of this paper is to analyze the effectiveness of using existing deep learning segmentation methods to segment tumors from three-dimensional PET-scans. PET-scans are constructed by introducing a positron-emitting tracer onto a biologically active molecule. By detecting the emitted gamma rays from these tracers, certain metabolic processes can be observed. The PET-scans used in this report depict regional glucose uptake, because the scans are acquired with the tracer fludeoxyglucose (FDG). Since tumors have a high metabolic activity, this tracer is commonly used to detect cancer tissue in the human body.

In radiation therapy—a common cancer treatment in oncology—a tumor is targeted with a high dosage of radiation to eliminate it, while the volume of surrounding healthy tissue that is also exposed to the radiation is kept to a minimum. As tumor segmentation from PET-scans becomes more precise, it is easier to expose more of the tumor, and less healthy tissue, to the radiation.

Recent advances in PET imaging have increased demand for better segmentation techniques to improve diagnostics and treatment [14]. The rapidly improving image quality of PET-scans is driven by better detection material and new reconstruction algorithms. Consequentially, PET-scans are used more often for diagnostic and treatment purposes, and have to be segmented more frequently as well.

If the recent advancements in two-dimensional semantic segmentation from deep learning turn out to extrapolate well to tumor segmentation in three-dimensional PET-scans, and outperform other medical segmentation methods such as manual or thresholding-based methods, the efficiency of the diagnosis and the quality of radiation therapy can be improved.

The primary research question of this paper is therefore: "Can deep learning be used for more accurate tumor segmentation in three-dimensional PET-scans than simple thresholding methods?"

To answer this question, a deep convolutional neural network with residual connections and dilated convolutions is trained on a diverse set of PET-scans, which have been manually segmented by an expert. Its performance is then compared to the performance of a thresholding algorithm.

2 Method

2.1 Data preprocessing

Before the simple threshold and neural network model algorithms can be tested on the data, the PET-scans must first be processed. This section will describe how the data is transformed from a set of PET-scans in their native file format to a representation that can be properly used by both algorithms. This series of data transformation steps—the preprocessing pipeline—is necessary to ensure that the neural network model can train efficiently.

The original data set contains 75 PET-scans and corresponding ground truth segmentations. There are varying types of cancer tissue in the PET-scans, including lung tumors, lymph node tumors, melanomas, and sarcomas.

2.1.1 File format conversion

Both the PET-scans and the label segmentations were encoded in the NIfTI-1 file format (file extension `.nii`). This file format allows metadata to be encoded with a PET-scan itself, in a type of dictionary inside the file format. One of the entries in this dictionary is a matrix of the intensity values of the voxels of the PET-scan. Other examples of the metadata are the date on which the PET-scan was made, the anonymized patient details, and other information that relates to the PET-scan. Since the segmentation algorithms only require the matrix of voxel intensity values, and not the related metadata, the NIfTI scans were converted to NumPy format (file extension `.npy`), by using the NiBabel package in Python. This conversion retained only the matrices of the PET-scans in a three-dimensional NumPy array, allowing for highly efficient data processing in a conventional and flexible way. The other scan metadata was discarded. The resulting PET-scans had shapes that varied considerably in size, ranging from $144 \times 144 \times 213$ to $256 \times 256 \times 991$. Their voxel values had a floating point number data type, and ranged from 0 up to roughly 1.6×10^5. The label segmentations had the

same shape as the corresponding PET-scan, but their data types were integer numbers. A value of 0 indicated no tumor on the corresponding PET-scan voxel, and vice versa for a value of 1.

The input scans were individually normalized to floating point values between 0 and 1. This normalization has a number of advantages. First, the network can train more efficiently on values in this range. Second, the scans are now all equal in terms of intensity values, reducing the chances of overfitting on scans with exceptionally high intensity values. Third, the simple thresholding algorithm can only work when the scans are in a uniform range of values.

2.1.2 Distribution into sets

After the scans are converted to NumPy arrays and normalized, they are randomly distributed into a training set (55 scans), validation set (10 scans), and test set (10 scans). The training set is used to train the weights of the network (see Section 2.4.4). The validation set is used to adjust and optimize the hyperparameters of the network. The test set is used to determine the final performance of the segmentation algorithms (see Section 2.2).

2.1.3 Scan slicing

The scans are inadequate to serve directly as data points due to their large size and limited amount; accepting a full scan at every single epoch in training the network would lead to a poor space complexity and to inefficient training. Therefore, the sets of three-dimensional training and validation scans (as well as their label segmentations) are converted to a larger set of smaller slices from these scans. These slices are obtained by iterating over the coronal plane of the scan and resized to a shape of $128 \times 128 \times 1$. Hence, four large sets of slices are created: the training images, the training labels, the validation images, and the validation labels. It is not yet necessary to convert the test scans to slices, because the test scans are only used during the test phase of the system, and converting the scans to slices at this point would merely result in memory usage that might hinder the training of the neural network. In fact, the test scan data is not loaded into RAM either. The system only saves the file names of the test scans, so that the random distribution is still adhered to in the testing phase.

2.1.4 Training data balancing

Roughly 90% of the training slices have empty label segmentation slices. The slices that are not empty generally only have a relatively small area of positively labeled pixels. Experimental testing indicated that this would make it very likely that the network would find one of the local minima where all output would be empty. This was circumvented by balancing the training data with regard to the segmentation labels. In this step, 98% of the training slices that have empty segmentation labels were removed from the training dataset.

2.1.5 Data augmentation

After removing a large majority of the slices that have empty labels, the size of the training set was decreased substantially. To restore it to roughly its original size, data augmentation is performed on each of the remaining training slices. Seventeen transformations of every training slice were appended to the training dataset. These transformations were horizontally flipping the slice, rotating it clockwise or counterclockwise by 10 degrees, zooming in or out by 30%, or any combination of these three transformations. This increased the size of the dataset with a factor of eighteen, close to its size before the data balancing step.

2.2 Performance metric

The performance of the algorithms is determined by the similarity of their segmentations to the labels. Two metrics are used to measure this similarity. Both of these metrics use the intersection of the output and label, the relative complement of the label with respect to the output, and the relative complement of the output with respect to the label. Respectively, the sizes of these sets are referred to as the number of true positives (TP), the number of false positives (FP), and the number of false negatives (FN).

The first of these metrics M_1 is the weighted sum of the positive predictive value (PPV) and the sensitivity (SE) (see Equation 2.1). The PPV is calculated as the number of true positives divided by the sum of true positives and false positives. The SE is

calculated as the number of true positives divided by the sum of true positives and false negatives. In other words, PPV is the fraction of the voxels in the output segmentation that are indeed voxels labeled as tumor tissue in the corresponding label scan. In contrast, SE is the fraction of voxels in the label segmentation that was positively identified in the output segmentation.

Here, S is the set of positives in the algorithm segmentation and L is the set of voxels that is labeled as tumor tissue in the label scans. The reason for using this metric is that it is widely used, and accounts for both false positives and false negatives.

$$M_1(S, L) = 0.5 \cdot \frac{I}{I + S_{\mathrm{RC}}} + 0.5 \cdot \frac{I}{I + L_{\mathrm{RC}}} \quad (2.1)$$

where

$$\begin{aligned} I &= |S \cap L| \\ L_{\mathrm{RC}} &= |L \setminus S| \\ S_{\mathrm{RC}} &= |S \setminus L| \end{aligned}$$

An exceptional case to this metric is when both the scan and label are empty sets (i.e., have no tumors). In that case the metric is 1 (see Equation 2.2).

$$M_1(\varnothing, \varnothing) = 1 \quad (2.2)$$

However, false positives and false negatives are not always equally harmful in medicine. For instance, in the case of tumor detection, false negatives can be considered as more harmful, because there are usually many more negatives than positives. A second assessment of all positives by a human expert is realistic, but not of all negatives. For that reason, a second metric M_2 is used (see Equation 2.3), which is a weighted variant of the Jaccard index (i.e., intersection over union). The standard Jaccard index is calculated as the number of true positives divided by the sum of true positives, false positives, and false negatives. This weighted variation includes a parameter γ, which is a value between 0 and 1, and acts as a multiplier to set the importance of false negatives in the Jaccard index. When $\gamma = 0$, only false positives are counted towards M_2, and contrariwise, when $\gamma = 1$, only false negatives are counted.

$$M_2 = J_w(S, L, \gamma) = \frac{I}{I + 2\gamma \cdot L_{\mathrm{RC}} + 2(1 - \gamma)S_{\mathrm{RC}}} \quad (2.3)$$

where

$$\begin{aligned} I &= |S \cap L| \\ L_{\mathrm{RC}} &= |L \setminus S| \\ S_{\mathrm{RC}} &= |S \setminus L| \end{aligned}$$

An exceptional case to this metric is when both the scan and label are empty sets (i.e., have no tumors). In that case the weighted Jaccard index is 1 (see Equation 2.4).

$$J_w(\varnothing, \varnothing, \gamma) = 1 \quad (2.4)$$

The loss function of the neural network (see Section 2.4.4) is closely based on M_2, and so the network can be trained toward a particular balance of false positive and false negative errors. The two metrics are similar, in the sense that they both indicate the error based on the number of false positives and false negatives. However, the way in which the metrics are presented offers different insight into the results. M_1 is plotted against the values of the binary threshold value θ, which classifies the images and network output into positive and negative pixels. On the other hand, M_2 is plotted against the values of γ, which indicates the desired trade-off between false positives and false negatives. Therefore, M_1 can be used to see how the threshold value influences the performance, and M_2 can be used as a complement to see which of the two algorithms performs better for which error type trade-off.

2.3 Simple thresholding

The simple threshold algorithm segments the image slices in a relatively straightforward fashion. This algorithm requires no training phase, so it is only used in the testing phase of the system. The set of normalized testing scans is used to find the performance of the simple thresholding algorithm (see Section 2.1.2). For every possible threshold value θ (0 to 1 exclusive, with a step size of 0.01), the set of test scans is segmented. The test scan voxels are segmented if and only if their intensity value exceeds this threshold value. Then, the segmentation is compared to the label segmentation. Hence, for

every threshold value the mean values of TP, FP, and FN are obtained.

The first metric M_1 can be directly calculated from these values. The sum of the positive predictive value and sensitivity is calculated (following Equation 2.1) for the mean values of TP, TN, and FN for every value of θ. For the second metric M_2, however, the mean values of the error types are not enough to determine the performance, because the third parameter γ is also used. For all possible values of γ (in the range between 0 and 1 with a step size of 0.01), the threshold value with the highest weighted Jaccard index is saved, as well as its corresponding performance.

2.4 Convolutional neural network

The neural network model operates very similarly to the simple thresholding algorithm in terms of calculating the performance metrics, but with a crucial additional step: whereas the simple thresholding algorithm takes the input scan, the neural network model takes the network output of the input scan. That means that in this research, we essentially see if this additional step improves the performance of the segmentation. It is necessary to threshold the output of the neural network, because its last layer is a sigmoidal activation function that yields a continuous value between 0 and 1 for every voxel, although a binary value (as in the label scans) is required. Only then can the values for TP, FP, and FN be calculated.

The neural network needs to be trained before it can give meaningful output segmentations. Put concisely, the neural network is a series of layers. The original input slices are given to the input layer, which passes it to the other layers. Every subsequent layer has a number of convolutional weights that transform the image (Sections 2.4.1–2.4.3). The images in the output layer are compared to the labels that correspond to the input slices, and the similarity is calculated using a loss function (Section 2.4.4). Then, the loss is backpropagated into the network, and an optimizer function adjusts the weights in such a way that the output will be more similar to the labels (Section 2.4.5). In the following subsections, these various aspects of the network model will be explained in more detail.

In Figure 2.1, a diagram of the architecture of the network is presented. This architecture is loosely based on that of SegNet [5] and Multi-Scale Context Aggregation Using Dilated Convolutions [8]. It is a simple type of an encoder-decoder network that uses dilated convolutions. More specifically, it is a series of blocks, each of which consists of a number of convolution + leaky ReLU layers, as well as one dilated convolution layer (in the first, "encoding" half) or transposed convolution layer (in the second, "decoding" half). In this report, a depth of 1 dilation, and 1 convolution + leaky ReLU layer in the two resulting blocks is used. During parameter optimization, deeper networks appeared more likely to overfit on the training data, compared to when only one dilation layer was used. The total size of the dataset is 103,500 training slices from 55 scans, and 7,002 validation slices from 10 scans.

Leaky ReLU is a simple activation function that returns the input if it is positive, and returns a fraction of the input if it is negative.

2.4.1 Convolutional layers

Most of the network layers are convolutional layers, each of which has a set of kernels. When an image is given as input to such a convolutional layer, the weights of the kernels are convoluted with this image, and the resulting set of convolutional output images is passed on to the next layer. Every standard convolutional layer has kernels with size 10×10. The number of filters increases linearly throughout the network: the first block's convolutional layers have 96 filters, and for every subsequent block, this number is incremented by another 96 filters.

2.4.2 Dilated convolution layers

In dilated convolution layers, the resolution is halved by using dilated kernels. By using valid padding and a kernel size of 1 plus 1/4th of the width and height of the input image (so $128/4+1 = 33$), the output image is downsampled by a factor of 2. Valid padding means that no zero-padding is used, but instead only the positions where the kernel fully fits inside of the input image are used in the convolution. The advantage of using dilated convolutions over the more traditionally used pooling layers is that dilated convolutions can preserve pixel-level patterns, because the dilated convolution kernels can be trained to detect those. In pool-

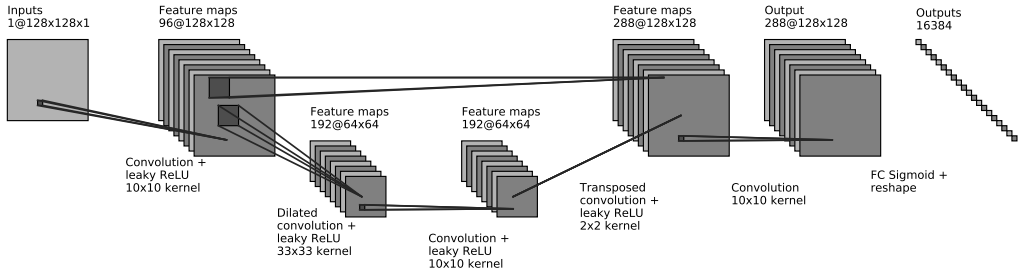

Figure 2.1: The architecture of the neural network model.

ing, however, a rectangular grid of pixels is reduced to one single pixel by the use of a simple and constant function. That means that potentially important patterns and information will be lost in the process. Using dilated convolution is a way to downsample the image while retaining this information.

2.4.3 Transposed convolution layers

The transposed convolution layers upsample the images back again to their original size. Here, the input image is dilated, and zero-padding is inserted in-between the dilated input pixels. By using a kernel size of 2, the image is upsampled by a factor of two.

2.4.4 Loss function

The loss function reflects the weighted Jaccard index metric as closely as possible. The weighted Jaccard index operates on the sets of voxels that are true positives, false positives, and false negatives after the network output was thresholded. However, this means that this metric is discontinuous and therefore non-differentiable. Because the network loss function should be differentiable, we take the direct output of the neural network at the output layer and calculate the loss before applying the thresholds. The weighted Jaccard index is imitated by using linear algebra as a substitute for set operations. For instance, in the weighted Jaccard index, the intersection of the segmentation S and label L was defined by $|S \cap L|$. In the loss function, we use the continuously-valued output prediction matrix P and label matrix L. The substitute for the intersection, for instance, can be calculated by the sum of all the pixels in the Hadamard product (i.e., elementwise multiplication) of S and L. Equation 2.5 gives the full equation for the loss function, given the network output P and label L.

$$\text{loss}(P, L, \gamma) = 1 - \frac{S(I) + \epsilon}{S(I) + S(L_\gamma) + S(P_\gamma) + \epsilon} \tag{2.5}$$

where

$$\begin{aligned} S(I) &= \sum_{i=1}^{\text{width}} \sum_{j=1}^{\text{height}} (I)_{ij} \\ I &= L \circ P \quad \text{(Hadamard product)} \\ L_\gamma &= 2\gamma(L - I) \\ P_\gamma &= 2(1-\gamma)(P - I) \\ \epsilon &= 1\text{e}^{-10} \quad \text{(arbitrary small value)} \end{aligned}$$

Here, the loss is a value between 0 and 1. The small constant ϵ prevents division by zero, which would otherwise happen when the union is empty, in case of an empty label and an empty prediction. This constant ensures that in this case, the loss is $1 - \frac{0+\epsilon}{0+\epsilon} = 0$.

This loss function was selected because it performed better than the mean squared error (MSE) loss function. One of the reasons that MSE did not perform as well is the fact that the positively-labeled voxels are relatively sporadic—many of the label slices are empty, even after the data balancing step, and the slices that are not empty contain only

a few positively labeled pixels. When MSE is used, the network would almost certainly converge toward one of the many local minima where the output is consistently empty. This is because an empty output yields a very low loss in the MSE function, but not in the weighted Jaccard loss function. For instance, in the extreme but not rare case where the label slice contains only one pixel that is a tumor value, the MSE loss would be $128^{-2} = 6.1\text{e}^{-5}$ for an empty output prediction. The weighted Jaccard loss, on the other hand, would be $1 - \frac{0+\epsilon}{1+\epsilon} \approx 1$. This makes the network far less likely to converge to giving empty output.

Another more obvious advantage of the weighted Jaccard index loss function is that its parameter γ can be used to train towards any desired balance between false positive errors and false negative errors, which is important in medical diagnostics. Symmetrical loss functions such as MSE do not have this parameter.

2.4.5 Optimizer

The loss is used to adjust the weights to better performing values that result in a lower loss in later batches of images. If the weights change, then the loss will change correspondingly. To optimize the weights, the optimization algorithm takes the gradient of the error in the direction where the loss decreases the fastest. In this network, the Adam optimizer is used [15]. This optimizer computes individual adaptive learning rates for the weights of the neural network, based on the first and second moments of the gradient. It calculates an exponential moving average of the gradient and squared gradient, and uses two decay parameters β_1 and β_2 to control the adaptive learning rate. The parameters for a time step θ_t are updated by using the first and second raw moment estimates: $\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$. Here, α is the step size parameter, and $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second raw moment estimates, respectively: $\hat{m}_t = m_t/(1-\beta_1^t)$ and $\hat{v}_t = v_t/(1-\beta_2^t)$ These are in turn based on the gradient on the corresponding time step: $m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ and $v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$. This optimizer will continue to update the parameters until they converge at some time step.

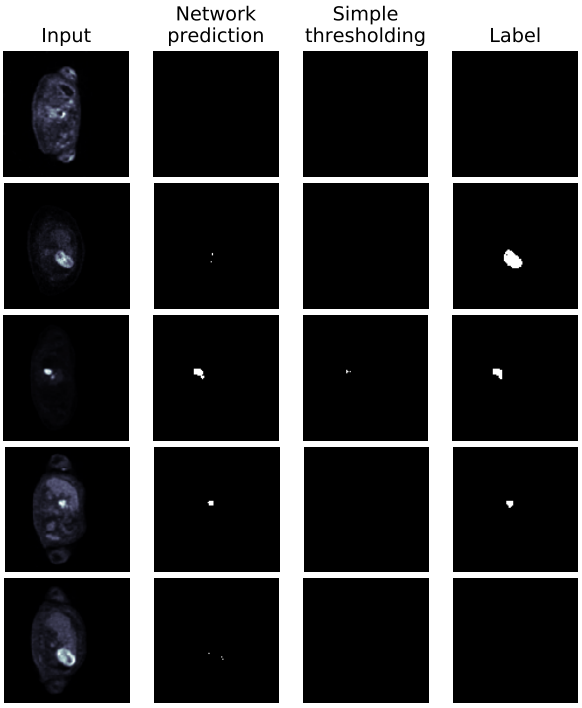

Figure 3.1: Some example input-prediction-label pairs from the validation set.

3 Results

In this section, the results of the two segmentation algorithms are presented. First, in Figure 3.1, some example results from the simple thresholding algorithm and the neural network model are displayed. Then, in Figures 3.2 and 3.3, the PPV+SE and the weighted Jaccard metrics (as explained in Section 2.2) are plotted.

Figure 3.1 illustrates a small number of example segmentations. These are examples from the validation set, because the test set is not converted to slices, and these are for exploratory purposes only. During the search for well-performing hyperparameter settings, these validation slice predictions were used to examine how the various settings affect the output.

Figure 3.2 illustrates the PPV+SE metric M_1 (see Equation 2.1). The continuous lines indicate the weighted sums of the positive predictive values (dotted lines) and sensitivity (dashed lines). The blue lines represent the model performance, and the orange lines represent the simple threshold per-

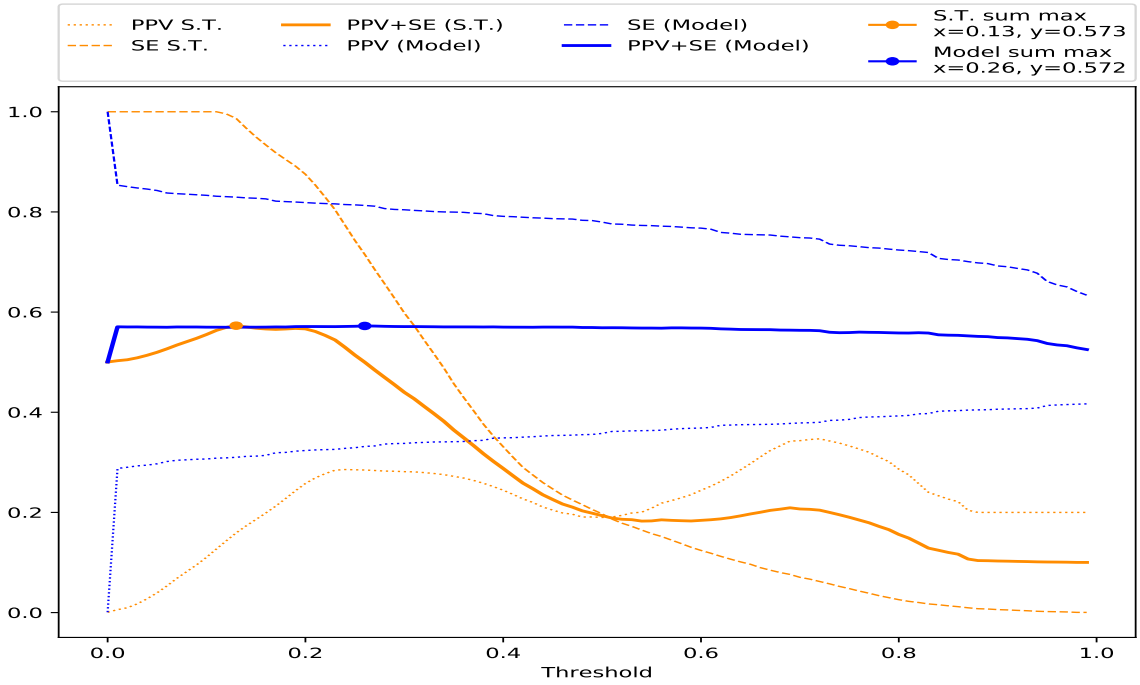

Figure 3.2: This graph shows the performance of the simple thresholding (referred to as "S.T.") and the neural network model segmentation algorithms, according to the PPV+SE metric. On the horizontal axis are the possible thresholds in the range [0,1]. On the vertical axis are the scores for positive predictive value, sensitivity, and their sum. The highest metric scores are 0.573 for the simple thresholding (with $\theta = 0.13$) and 0.572 for the neural network model (with $\theta = 0.26$).

formance. The horizontal axis indicates the value that is used for the binary threshold. Any values in the output (for the neural network model) or in the scans (for the simple thresholding algorithm) smaller than this value will be classified as having no tumor, and vice versa. This graph is generated for a balanced false positive and false negative trade-off (i.e., $\gamma = 0.5$). It appears that the neural network model performs relatively consistent over the full range of possible threshold values: increasing the threshold value only slightly reduces the SE and increases the PPV. This effect is caused by the fact that network output is not evenly distributed, but polarized to values near 0 or 1. The binary threshold value appears to affect the simple threshold algorithm more strongly. Two performance maxima are observed in Figure 3.2—the highest at $\theta = 0.13$, and a lower maximum around $\theta = 0.7$. The sensitivity monotonically decreases as the binary threshold increases, but the positive predictive value fluctuates between 0 and 0.38.

Figure 3.3 illustrates the weighted Jaccard index metric M_2 (see Equation 2.3). The horizontal axis indicates the value of γ. The continuous lines indicate the weighted Jaccard index, calculated using the highest-scoring binary threshold value, which is indicated by the dashed line, for every value of γ. For both the neural network model (indicated by blue lines) and the simple thresholding algorithm (indicated by orange lines), the weighted Jaccard index is generally higher for higher values of γ. The optimal threshold logically decreases as γ increases, because at higher values of γ, false negatives become increasingly more important than false positives. Hence, as γ increases, the number of negatives is decreased by lowering the binary threshold value.

4 Conclusion

A number of conclusions can be drawn from the results in Figures 3.2 and 3.3. First, for all balances

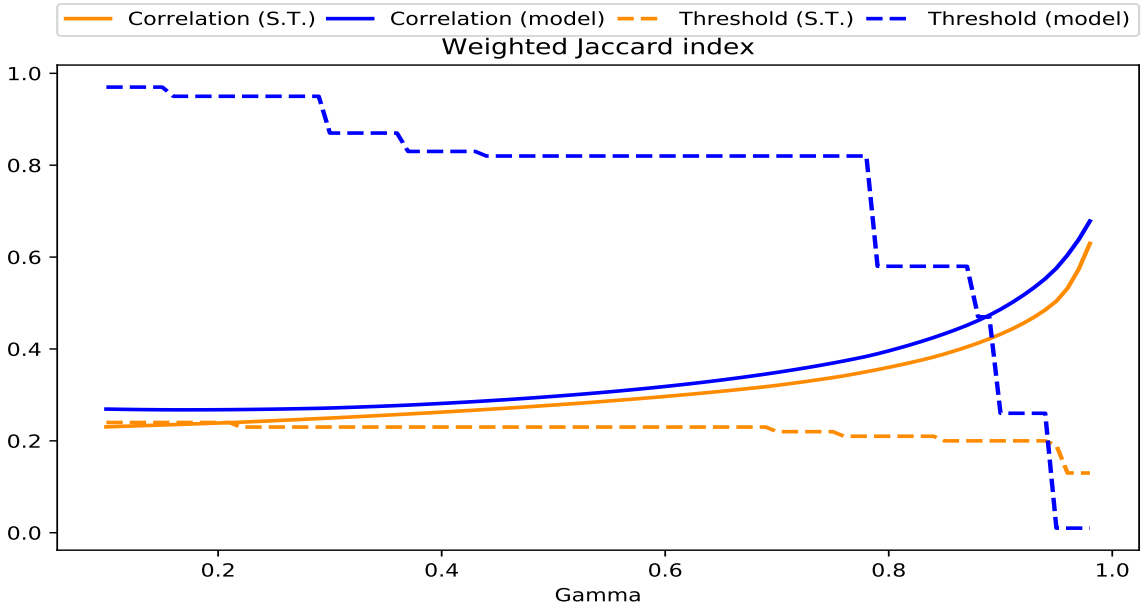

Figure 3.3: This graph shows the performance of the simple thresholding (referred to as "S.T.") and the neural network model segmentation algorithms, according to the weighted Jaccard index metric. On the horizontal axis are the possible FN/FP-balances (γ) in the range [0.1, 0.99]. On the vertical axis are the highest weighted Jaccard index for the optimal threshold (continuous line), and the optimal thresholds themselves (dashed line).

between false positive and false negative errors, the neural network model outperforms the simple thresholding algorithm in the weighted Jaccard index metric (see Figure 3.3). Therefore, we can answer the research question posed in Section 1. Deep learning can indeed be used for more accurate tumor segmentation in three-dimensional PET-scans than simple thresholding methods. However, this conclusion involves some important considerations and side notes, which are set out in Section 5.

Figure 3.3 also shows that for higher values of γ, the weighted Jaccard index is also higher, for both algorithms. This indicates that the number of false positives in the segmentation output is higher than the number of false negatives, possibly because of the metastasis in the scans, that yields similar voxel intensity values but is not labeled in the ground truths. Another observation that can be made, is that the optimal threshold values are not centered around one value, but display rather unpredictable behavior.

Although the neural network consistently outperforms the simple threshold algorithm on the weighted Jaccard index, this is not always the case in the other PPV+SE metric (see Figure 3.2). The main reason for this is that this metric indicates the performance for all possible threshold values, rather than for all values of γ. The simple threshold algorithm performs better when lower threshold values are used (specifically, only for the threshold values of 0.13 and 0.14), but its performance is poor for higher threshold values.

Some other interesting conclusions can be drawn from the PPV+SE graph as well. The positive predictive value of the neural network is considerably higher than that of the simple thresholding algorithm, especially for higher threshold values. This means that the number of false positives on the simple thresholding algorithm is relatively high. However, for the neural network model, the PPV and SE are fairly consistent over the range of thresholds. This indicates that the mean of the nonzero pre-thresholded output voxels is close to 0 (as can also be seen in the weighted Jaccard index graph, because the optimal threshold is also close to 0). Both metrics show that for the simple threshold al-

gorithm, the optimal threshold value is 0.13 when there is no custom balance between false positives and false negatives (i.e., $\gamma = 0.5$). The optimal threshold for the different values of γ for the neural network model varies more widely, but has a median value of roughly 0.8.

5 Discussion

The research in this report sought to explore the possibility of using deep learning for semantic segmentation in non-natural images—specifically, the segmentation of tumor tissue in PET-scans. A relatively novel neural network architecture was selected that performed well in natural image segmentation, and some adaptations were made to fit this new type of data. The performance of the neural network model was compared to a simple thresholding algorithm, and the main result was that the neural network model consistently outperformed the simple thresholding. However, this does not necessarily mean that the neural network model is a good alternative; to comprehensively appraise this model and its possible implementation, some careful considerations must be made about this particular model, and about the use of deep learning for semantic medical image segmentation in general.

5.1 Considerations

The results of this report are unambiguous: on the testing data, the trained neural network model performs better than the simple thresholding algorithm on the weighted Jaccard index metric. The other metric indicates that for unweighted error types, the positive predictive value of the neural network is higher than that of the simple thresholding algorithm. The sensitivity of the simple thresholding algorithm is higher for low thresholds, but the merits of this high sensitivity remain questionable, considering its associated low positive predictive value. The results indicate (and output segmentations show) that both algorithms segment with a relatively high number of false positives, compared to false negatives. A factor that leads to this might be found in the data: metastasis is not labeled, even though it looks very much like the labeled tumor tissue in the scan slices. Therefore, it might be the case that the network has learned to distinguish cancerous tissue from healthy tissue, but not labeled tumors from metastasis. It can be conjectured that this distinction might be learned when more data is used, which would improve the network performance substantially.

It is important to note that the clarity of the outcome in this report does not mean that deep learning outperforms all other segmentation techniques. Many other segmentation techniques exist, and simple thresholding is only one of them—and a very trivial one as well. Examples of such techniques include fuzzy local adaptive Bayesian (FLAB) methods [16], fuzzy C-means [17], and basic region growings [18]. Like deep learning and simple thresholding, each of these techniques has its own particular advantages and disadvantages. For instance, in FLAB methods, a bounding box has to be drawn around the tumor, and the algorithm will segment the tumor inside this box. However, the segmentation is highly dependent on the exact position of the bounding box, introducing an undesirable factor of arbitrariness to the segmentation. This often results in the bounding box having to be drawn over and over again, until a satisfactory segmentation is obtained. Fuzzy C-means classifies and segments voxels on an individual basis, without regarding spatial context of neighboring voxels. This generally leads to a less than optimal segmentation. Basic region growing methods depend on initial seeds that are set by the operator. If the seed points are badly placed, the segmentation will also be relatively poor. This means that other well-known segmentation algorithms that require some user interaction, like the FLAB method, are more contingent and arbitrary than both simple thresholding and deep learning methods, which do not require such interaction. Still, despite these disadvantages, these conventional methods might still outperform simple thresholding methods—and indeed deep learning methods as well. A more exhaustive comparison is needed to accurately gauge the relative performance of deep learning methods in the broad field of non-natural image segmentation methods. This goes for evaluating various types of conventional methods, such as the ones mentioned above, but also for the many different types and implementations of deep learning.

In this research specifically, the performance of residual connections and dilated convolution in

an encoder-decoder network for non-natural image segmentation was explored. The idea behind the residual connections is that the network learning speed might be positively enhanced: the network is both shallow and deep at the same time, depending on the series of connections that is taken. The shallow route of connections improves the learning speed in the early stages of training, because clear patterns are learned quickly. In later stages, the deeper connections allow the model to be more detailed in its output, because they have the capacity to learn deeper and more abstract underlying patterns of the data. The dilated convolutions offer a way to increase the receptive field of the intermediate convolution layers, without compromising on pixel-scale information (when using pooling layers) or on space or time complexity (when using larger kernels). By effectively increasing the receptive field this way, larger patterns may be detected through a single convolutional filter that would otherwise require a complex combination of multiple smaller filters and layers. For instance, if a tumor in one part of the slice would reliably predict another tumor elsewhere, a receptive field that encompasses both areas could be more effective than multiple small receptive fields that work together through connections between multiple layers.

However, many other parameter settings and implementations of residual connections and dilated convolution exist. Although the hyperparameters of this model were optimized reasonably well, it cannot be guaranteed that these are the optimal parameters. A large variety of different network settings was used, including deeper networks with multiple parallel residual connections and dilated convolutions, but these quickly caused the network to converge toward giving only empty output—despite using the Jaccard index loss function and data augmentation. Furthermore, encoding scan slices with a depth of 3, so that the three channels comprise the slice itself, and its two adjacent slices, did not result in a higher accuracy. However, on a subset of the dataset it did. The reason for trying out this approach is that the adjacent slices might contain additional information about the middle slice, and effectively provide some three-dimensional information about the scan. Another approach was to reconstruct the prediction scan not from the coronal plane only, but also from the sagittal and transverse planes. This did not work as well as reconstruction from a single direction, presumably because the network would need to generalize scan readings from multiple directions. Larger image sizes and more filters per layer reliably resulted in a higher performance, but because of memory constraints, these parameters could not be increased to higher values (the model was trained on a GTX 1070 GPU with 8GB memory). The architecture presented in this report (in Figure 2.1) performed the best out of all tested architectures. Searching for the optimal set of hyperparameters is a non-trivial task, and there are not much algorithms available yet to find very good settings. The hyperparameters in this model were found by using a combination of random search and grid search heuristics, in a way that is reminiscent of simulated annealing. However, when handling other types of data, or more data, the optimal hyperparameters and architecture might be wildly different from those used in this research.

There are also different ways in which the data could be encoded. For example, as small three-dimensional cubes instead of two-dimensional slices. However, this method has the drawback that convolution and downsampling would be more difficult.

The network model architecture is most closely based on FCN [7], SegNet [5], and U-Net [19], which are all designed to do semantic segmentation on natural images. However, in optimizing the results on this non-natural dataset, some adaptations were made to this architecture to increase the performance. A factor that might have necessitated these adaptations was the limited size of the data set, especially compared to the very large datasets used in natural image segmentation. Particularly the shallowness of the network, compared to networks used in the FCN, SegNet, and U-Net papers, was necessary to prevent overfitting on the limited number of training slices. The smaller size of datasets could be considered as an intrinsic factor of non-natural image datasets—it is more expensive to make a PET-scan than a natural image, and therefore accumulating a very large dataset of medical images is more burdensome. Data protection regulation and medical privacy law are other factors that hinder large-scale data collection of these types of data, especially in Western countries. For these reasons, non-natural image datasets might almost intrinsically necessitate shallower networks because

of their size, compared to the network models in natural image segmentation that grow deeper as larger datasets become available and better data augmentation methods are developed.

The model in this research explored the possibility of a general neural network model for various types of tumors. However, what remains to be explored is the extent to which more specialized models might improve the segmentation accuracy. This specialization can be understood in terms of tumor type (i.e., training a model only on specific types of tumors), but also in terms of PET-scanner configurations. Simple thresholding methods are not so generalizable, and this is one of the main limitations of simple thresholding. This drawback should be kept in mind when interpreting the metric results—the optimal threshold for a desired value of γ might not hold for a different test set. There exists no threshold that works for every image. But it is possible that more specialized models and standardized PET-scan data helps to improve the reliability of simple thresholding. However, this requires more specialized research and thus more data.

5.2 Deep learning: drawbacks and benefits

At the moment, deep learning has some drawbacks and limitations, compared to other, more conventional methods. First, since deep learning models are essentially "black-box" models, where the output prediction cannot be clearly traced and explained, there is an ethical and social implication when using deep learning for medical applications, where critical life-and-death decisions are made based on the network model. Deep learning models are not perfect, and although on average they might potentially be highly accurate, it is possible that in exceptional cases a person might die because of complacency in second assessment or overreliance on deep learning models. Second, training deep learning models requires a large body of medical data. This data is relatively difficult to collect, either because it is expensive to make, or because patient records are confidential and sharing data discretely and safely is difficult—not only in gathering data for a dataset, but also for peer-reviewing existing research. Additionally, training a neural network on such a dataset requires label images as well. Because these label images have to be segmented manually by an expert, it is tedious and time-consuming to collect a substantial dataset. Finally, the performance of deep learning methods depends heavily on the degree of specialization that is used. General models are applicable in a lot of cases, but require much more data than specialized models. The extent to which a model is trained on generalized or specialized data is quite arbitrary and it is not always clear in advance which degree of specialization performs best, or which is most desired.

However, there are also a lot of benefits and advantages to deep learning methods. For instance, whereas the performance of methods such as simple thresholding or FLAB methods is static, the performance of deep learning methods grows indefinitely as more data is collected. To be sure, there are complications in the collection of this data, but there might be ways to overcome these. For instance, the discretion and safety of patient records can be safeguarded by using anonymization or encryption. That way, researchers can share confidential patient records more easily. The possibility of online learning can also help in this regard: second assessments by a doctor can be fed to the model as new training examples, allowing the model to improve indefinitely and while it is being deployed. This might be made even more effective if hospitals would share the same trained neural network model (but not patient data). Since it is very difficult, if not impossible, to retrieve patient records from the weights of a complex trained model, large collections of patient data might hence still be used confidentially and securely. Another reason why deep learning is promising is the fact that at the moment, it is a field that is rapidly developing and improving. Medical areas for which deep learning is currently difficult to implement might have great potential for improvement with the techniques and approaches that will be developed in the future, especially as those medical areas continue to improve as well.

To overcome the main drawbacks of using deep learning in medical imaging, and to maximize its utility, more research is required. Interdisciplinary research in particular remains important as the different disciplines develop—not only for integration of findings and sharing of results, but also for a mutual understanding of the leading techniques and operational frameworks in medical imaging and

machine learning.

References

[1] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[2] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *British Machine Vision Conference*, volume 1, page 2, 2017.

[3] Bahram Lavi, Mehdi Fatan Serj, and Ihsan Ullah. Survey on deep learning techniques for person re-identification task. *arXiv preprint arXiv:1807.05284*, 2018.

[4] Chenxi Zhang, Liang Wang, and Ruigang Yang. Semantic segmentation of urban scenes using dense depth maps. In *European Conference on Computer Vision*, pages 708–721. Springer, 2010.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[8] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[9] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[11] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 565–571. IEEE, 2016.

[12] Kenny H Cha, Lubomir Hadjiiski, Ravi K Samala, Heang-Ping Chan, Elaine M Caoili, and Richard H Cohan. Urinary bladder segmentation in ct urography using deep-learning convolutional neural network and level sets. *Medical physics*, 43(4):1882–1896, 2016.

[13] Mathieu Hatt, Baptiste Laurent, Anouar Ouahabi, Hadi Fayad, Shan Tan, Laquan Li, Wei Lu, Vincent Jaouen, Clovis Tauber, Jakub Czakon, et al. The first miccai challenge on pet tumor segmentation. *Medical image analysis*, 44:177–195, 2018.

[14] Brent Foster, Ulas Bagci, Awais Mansoor, Ziyue Xu, and Daniel J Mollura. A review on segmentation of positron emission tomography images. *Computers in biology and medicine*, 50:76–96, 2014.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Mathieu Hatt, Catherine Cheze Le Rest, Alexandre Turzo, Christian Roux, and Dimitris Visvikis. A fuzzy locally adaptive bayesian segmentation approach for volume determination in pet. *IEEE transactions on medical imaging*, 28(6):881–893, 2009.

[17] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[18] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.