Nerve fiber tracing in bright-field images of human skin using deep learning

Author:
Herman Bergwerf
Student TU Delft, Erasmus MC
hermanbergwerf@gmail.com

Supervisor:
Dr. ir. Erik Meijering
Biomedical Imaging Group Rotterdam
meijering@imagescience.org

June 28, 2018

Abstract

The goal of this thesis is to find an automated method that can trace all nerve fibers in bright-field images of skin tissue. This is an important step towards the automated quantification of intra-epidermal nerve fiber density, an important biomarker in the diagnosis of small-fiber neuropathy. Deep learning is a popular new field of research in computer vision. In recent years it has been successfully applied to a lot of computer vision problems. Here we try to use deep learning for nerve fiber segmentation. It will be shown how a convolutional neural network can be implemented and trained to produce nerve fiber segmentation maps. This involves the optimization of many layers of computations (summing up to tens of millions of parameters), which we did using a modern machine learning toolkit. Statistical analysis of the obtained results show that the neural network has a performance that is comparable to a human control, and out-competes an earlier method that was developed using conventional image analysis tools, by a big margin. A number of improvements are proposed to further increase the neural network performance.

Contents

Chapter 1

Introduction

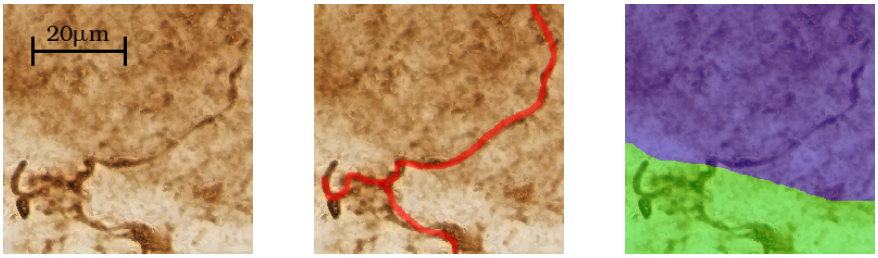
1.1 Neuropathology

Neuropathology studies the nervous system tissue, and in particular diseases that may occur with it, such as Alzheimer. The class of diseases that is relevant for this thesis is called peripheral neuropathy. This class includes damage or disease that affects the peripheral nervous system. The National Institute of Neurological Disorders and Stroke has summarized the symptoms of peripheral neuropathy on their website:[1]

> Symptoms can range from numbness or tingling, to pricking sensations (paresthesia), or muscle weakness. Areas of the body may become abnormally sensitive leading to an exaggeratedly intense or distorted experience of touch (allodynia). In such cases, pain may occur in response to a stimulus that does not normally provoke pain. Severe symptoms may include burning pain (especially at night), muscle wasting, paralysis, or organ or gland dysfunction. Damage to nerves that supply internal organs may impair digestion, sweating, sexual function, and urination. In the most extreme cases, breathing may become difficult, or organ failure may occur.

1.2 Intra-Epidermal Nerve Fiber Density

The specific peripheral neuropathy conditions relevant for this thesis are small-fiber neuropathies (SFNs). SFNs are relatively common, affecting an estimated 15 to 20 million people over 40 in the United States alone[2]. It is



(a) Original image (b) Nerve (c) Junction

Figure 1.1: Example of an intra-epidermal nerve fiber. All nerve crossings have to be counted in selected regions of the imaged tissue in order to determine the IENFD. In (c) the dermis and epidermis are highlighted in green and blue.

characterized by the predominant involvement of somatic unmyelinated C-fibers and thinly myelinated Aδ-fibers that convey thermal and nociceptive stimuli. Symptoms typically start with burning feet and numb toes. Skin biopsy is an important tool for investigating the small nerve fibers involved in these conditions[3]. A widely used biomarker is the nerve fiber density between the dermis and epidermis, commonly referred to as the intra-epidermal nerve fiber density (and often abbreviated as IENFD). To measure this quantity the number of nerve fibers that cross the dermis/epidermis boundary is counted and divided by the skin length. This involves significant manual work: extracting a skin sample from the patient; slicing, staining and imaging; and finally identification/counting of nerve fibers. For this reason a diagnosis can take a long time, and is expensive. Figure 1.1 shows an example of one intra-epidermal nerve fiber.

Both bright-field and immunofluorescence microscopy are used for tissue imaging. The data for this study has been made available by the neurology department at Erasmus MC where only bright-field microscopy is used for skin biopsies[1], therefore this thesis is focused on bright-field images.

1.3 Automated counting

In order to reduce the time and cost of measuring, and to improve the accuracy with respect to manual counting[2], we would like to develop a computer

[1]One of the reasons is that immunofluorescence samples cannot be physically archived.

[2]One study showed a correlation coefficient between sites of 0.94, and a variability of 25.5% between laboratories[4].

program that can determine the intra-epidermal nerve fiber density. Four high-level steps the automated procedure could follow based on the manual counting process are:

1. **Select regions of interest** The raw data from the microscope contains an array of samples, some are not usable for counting because of artifacts such as folds. During the manual process a human expert will select regions that look suitable enough for counting. A computer program would need a similar ability to select regions of interest.

2. **Determine location of nerves** A human expert must learn to find nerves in the imaged tissue in order to correctly count the number of crossings. It is not necessary to trace the nerves, as long as all crossing nerve fibers are identified.

3. **Determine junction** It is often not clear which part of the tissue belongs to the epidermis, and which part to the dermis. By looking at the location and orientation of nerve fibers, and the tissue staining color and texture, an educated guess can be done[3]. This is sufficiently accurate for counting the number of crossings[4].

4. **Measure skin surface length** This is more of a formality since in most cases the junction length is quite similar to that of the skin surface. However, the European guidelines for determining the IENFD require it to be computed with respect to the skin surface[5].

1.4 Previous work

In the past years there have been some attempts at automating the nerve counting. Some interesting publications that are closely related to this topic, or that were pointed out as relevant by previous publications, are listed in Table 1.1. According to the literature research that was conducted for this thesis there are only two publications that implement automated nerve fiber detection in bright-field skin samples:

1. Shanon, Manuel, Kathrin, *et al.* [6] implemented a special two-level edge detection algorithm using the Orbit image analysis program, and

[3]Unfortunately I am not an expert on this topic. I have seen a number of examples but I cannot tell with confidence how to identify the epidermis/dermis junction in many.

[4]Guidelines for counting exist, such as the minimal nerve length after the crossing[5].

used several pre- and post-processing steps to reconnect nerves. Unfortunately no additional details are given about which steps are taken exactly or how the method could be reimplemented. In their paper they point out that for bright-field images only 40% of the data available to them could be automatically processed due to a high variability in straining. The data from some labs was unsuitable due to a low contrast between nerves and the epidermis.

2. Pontenagel [7] implemented a nerve detection algorithm using a Frangi vesselness filter, tensor voting with steerable filters, and small object removal. Section 2.3.1 describes this approach in more detail. This method is also quantitatively compared to one developed in this thesis.

1.5 Challenges

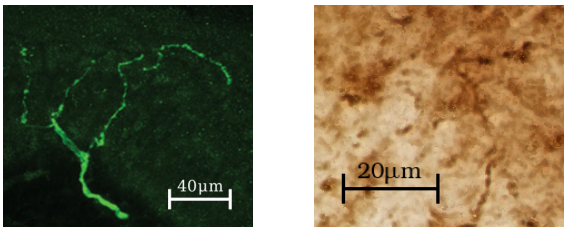
From previous research and from expert experience we know that nerve detection in bright-field images is not a straightforward task. An example of this is illustrated in Figure 1.2. It is difficult to tune conventional image processing algorithms such that the low contrast and high variability can be accounted for. As shall later be demonstrated, counting nerves is not a trivial task even for humans[5].

Because the imaging method requires the samples to be thin slices, nerves or nerve segments that do not run approximately parallel to the slicing plane are not (or only partially) visible. Humans are experts at interpolating information however, as is demonstrated in Figure 1.3. Imaged nerve fibers often contain numerous gaps, have parts with low and high contrast, and go from thick to extremely thin. All this poses significant challenges for automated detection.

[5]See Figure 3.12. where the performance of a human expert is compared to two novices.

Table 1.1: A few recent publications involving nerve counting or nerve density computation.

Year	**Author**	**Topic/method**
2011	Casanova-Molla, Morales, Solà-Valls, *et al.* [8]	Test correlation between PGP9.5 immunoreactive fluorescence and skin innervation (IENFD is counted manually). It is established there is a positive correlation.
2012	Sathyanesan, Ogura, and Lin [9]	Nerve fiber density measurement using Hessian-based feature extraction (replacing edge detection filters to reduce noise) and line intensity scan analysis. This publication deals with tissue from the central nervous system.
2014	Vincenzo, Maria, Annamaria, *et al.* [10]	Comparison between manual counting of nerves in confocal 3D images using Neurolucida software, and manual counting directly through the oculars of an epifluorescence microscope. It is established both methods are in agreement.
2015	Shanon, Manuel, Kathrin, *et al.* [6]	Nerve detection was based on special two-level edge detection in combination with several pre-processing and post-processing steps to connect nerve fragments. The epidermis/dermis junction is drawn manually.
2017	Pontenagel [7]	Automatic IENFD computation using local variance focus stacking, Frangi vesselness filtering, tensor voting with steerable filters, thresholding, and morphological operations.



(a) Immunofluorescence (b) Bright-field

Figure 1.2: An example of a nerve image obtained through immunofluorescence (left) and bright-field microscopy (right). Bright-field microscopy generally produces images with much less contrast between nerves and background, and much more variation in brightness. This makes it harder to develop a consistent nerve detection algorithm. The left image was copied from Casanova-Molla, Morales, Solà-Valls, *et al.* [8] Fig. 6A

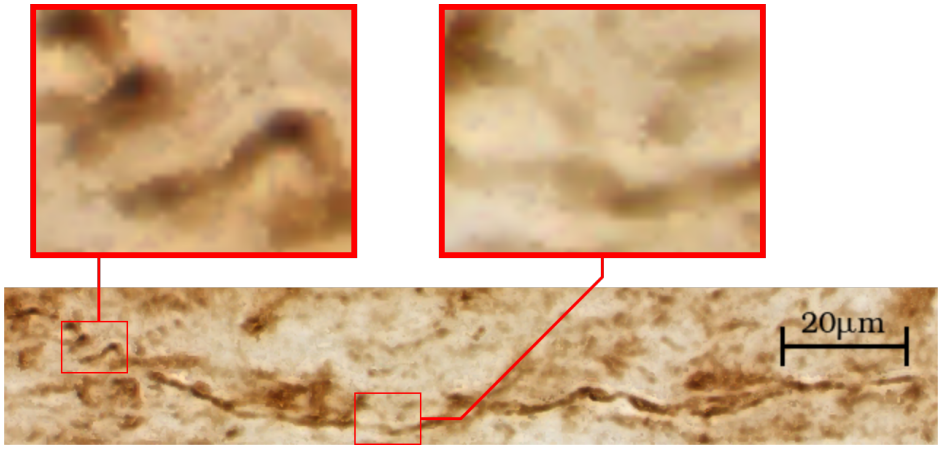


Figure 1.3: The human brain is very good at extracting continuous structures from visual information such as the nerve fiber that is shown here running from the left to the right side of the image. On close inspection it turns out there are several gaps in the nerve staining. This makes tracing similar nerve fibers a challenging problem for computers.

1.6 Deep learning in biomedical imaging

In 2012 AlexNet[11], a Convolutional Neural Network (CNN), won the ImageNet image classification competition by a considerable margin. Since then CNNs have quickly become the technique of choice for many computer vision problems such as classification, detection, segmentation, and enhancement[12]. Thanks to the development of hardware and software, increasingly large CNNs can be developed (often called deep convolutional neural networks for their high number of stacked layers). Large CNNs, together with other methods that involve building large neural networks, are often referred to as deep learning.

The medical image analysis community has also picked up CNNs in order to solve problems that were previously much harder or impossible to solve. During 2015 and 2016 the number of papers on this subject grew strongly. An important paper that is often cited in the context of segmentation problems is *U-Net: Convolutional Networks for Biomedical Image Segmentation*[13] which was published in 2015 and has been cited almost two thousand times[6]. U-Net is a CNN that contains a symmetric contracting and expanding path, and can be trained with a small number of images[7] by using strong augmentation (such as smooth deformations) and drop-out layers in the neural network. U-Net out-computed rivals at the 2015 ISBI cell tracking challenge[14].

So far there seem to be no publications on applying CNNs to the segmentation of nerves in bright-field images. However, for a somewhat similar problem, the segmentation (and analysis) of blood vessels in retinal images, a number of CNN based approaches have been published in the past few years[12]. The automated assessment of retinal images for detecting or predicting a range of diseases has also been subject to study[8]. An example of this is diabetic retinopathy, one of the fastest growing causes of blindness. In 2016 it was shown that a CNN (Google Inception v3) can identify diabetic retinopathy with a performance comparable to a panel of human experts. More recently it was shown that deep learning can also extract certain risk factors of cardiovascular disease from retinal images[15].

The problem of retinal vessel segmentation has been extensively stud-

[6] 1951 times according to Google Scholar as of July 22th 2018.

[7] The paper presents a cell segmentation problem with a training set of 30 images of 512×512 pixels.

[8] Big companies like Google have shown interest in this, possibly because of the big potential for future health-care.

ied. Many supervised and unsupervised methods have been developed in the past. However, deep CNN models have already outperformed all other published methods by a significant margin[16]. Altogether this demonstrates the tremendous potential of CNNs in biomedical imaging.

1.7 Goal

We have seen that automated counting of nerve fibers is desirable, but very challenging. This thesis focuses on one step of this process: nerve detection. Several conventional methods have been developed for this purpose but none has demonstrated a satisfying performance. We believe that a neural network has the potential to solve this problem by learning from existing annotations. The goal is to develop a convolutional neural network that can produce nerve fiber segmentation maps. In the following chapters a neural network and a training procedure are described and evaluated.

Chapter 2

Methods

2.1 Data

The skin biopsy protocol used for a comparable SFN study is described in detail by Lauria, Hsieh, Johansson, *et al.* [5]. A disk-shaped piece of skin tissue is taken from a subject's lower back or the distal part of the leg using a 3mm disposable punch. The sample is immediately cooled to around 4 C°. Each biopsy is cut into about thirty $50\mu m$ slices (three such slices are shown in Figure 2.1). The tissue samples are stained with PGP9.5 to increase the nerve visibility, and scanned using a bright-field microscope[1]. The available dataset contains slices from 4 volunteers (not affected by small-fiber neuropathy) containing a total of 96 manually picked regions of 640×448 pixels. Figure 2.2 shows an example.

[1]The images that were provided to us were obtained using the Hamamatsu NanoZoomer at a 40× magnification.

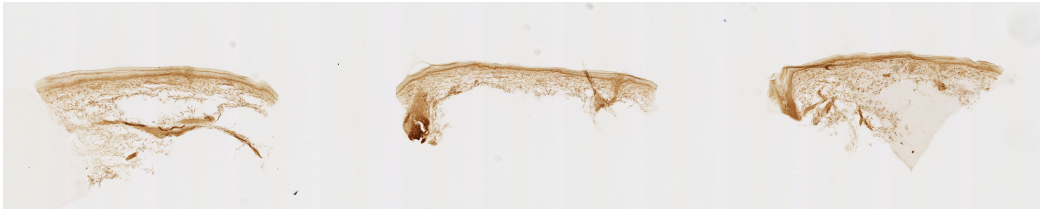


Figure 2.1: Cropped slide showing three slices.

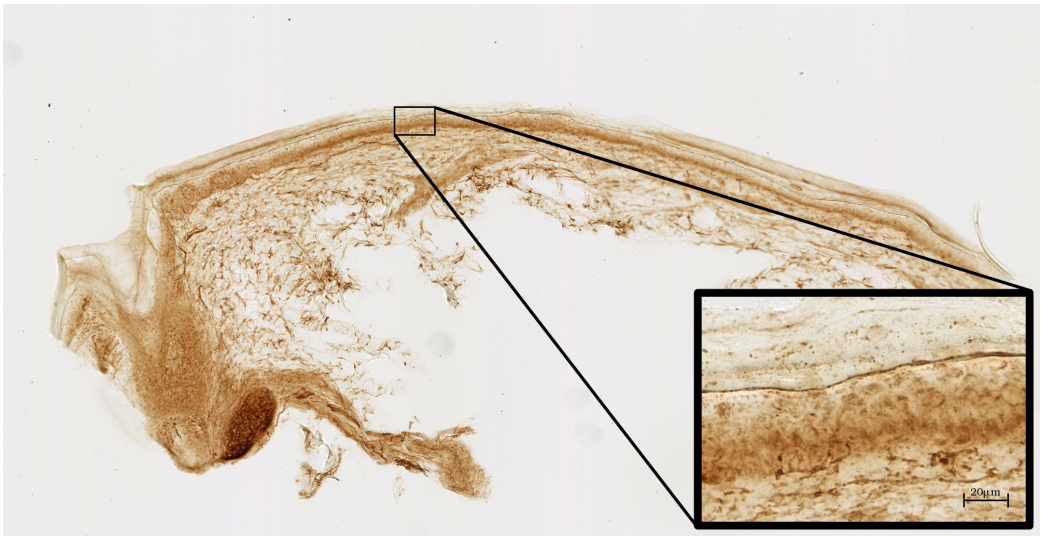


Figure 2.2: Small regions have been manually selected from the scanned slices such as the one shown here.

2.1.1 Focus stacking

The slices are imaged at 31 focal planes in the range $[15, -15]\mu m$. In order to simplify processing we have decided to compute an extended depth of field image (also referred to as z-projection, focus stacking, and focus merging). By doing so the neural network will not have to figure out how to deal with multiple focal planes. On top of that the data size is reduced by an order of magnitude which is very important since the GPU memory that is available for training the neural network is limited. The Laplacian-of-Gaussian (LoG) is used to determine which slide is in focus for each pixel. The LoG function is visualized in Figure 2.3. It can be used as edge/noise detector by convolving with an image. The Gaussian will smooth the image while the Laplace operator will detect edges. Because there is a high response for parts of the image that are in focus (since edges are more crisp there), the in best focal plane can be determined for each pixel by finding the maximum LoG value. Figure 2.4 shows two focal planes and a merged image.

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

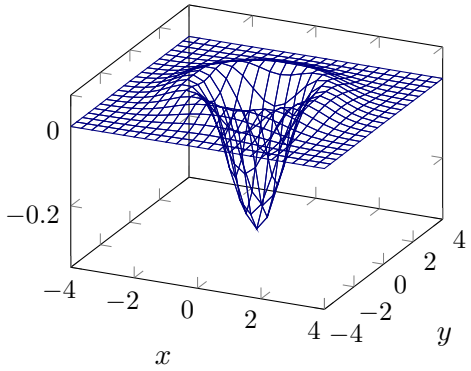
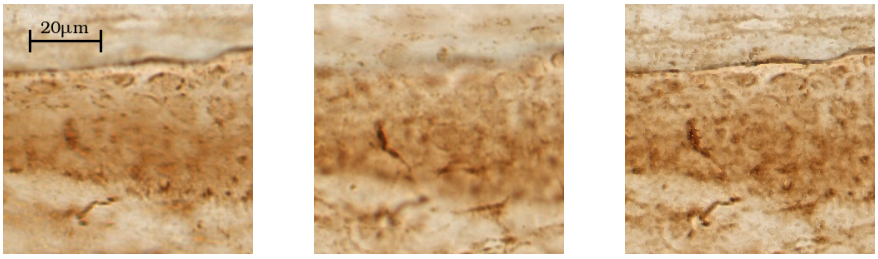


Figure 2.3: The Laplacian-of-Gaussian with $\sigma = 1$.



(a) $z = -4\mu m$ (b) $z = 1\mu m$ (c) Merged image

Figure 2.4: Here (a) and (b) show two of the 31 focal planes, (c) shows the result of merging all 31 focal planes.

2.1.2 Generated patches

To train, validate and test the neural network, 448×448 patches are cropped from the focus stacked images. To prevent over-fitting it is very important to use good image augmentation. By applying random transformations to the patches the number of different training patches that can be generated is much larger. The pipeline for generating training patches contains the following steps[2]. Figure 2.5 shows some examples of unnormalized, augmented patches.

1. **Random cropping** Each 448×448 patch is cropped from a randomly selected region with a uniform random horizontal offset ($x_{\text{offset}} \in [0, 640 - 448]$). The cropped patches are also down-scaled to 224×224 (this will be explained in the next section).

2. **Affine transformation** A relatively fast way to deform the image is using affine transformations. A transformation is applied to each patch that is described by $M = M_1 \cdot M_2$, where M_1 is obtained by computing the mapping between random translations of three corners, and M_2 is a transformation that rotates the patch around its center with a random angle $-0.8\pi \leq \alpha \leq 0.8\pi$ (radians). To describe M_1 more fully we will look at a few equations. M_1 it determined by OpenCV by solving Equation 2.1 for three points (v_1, v_2, and v_3) that are all randomly translated to three new points ($v\acute{}_1$, $v\acute{}_2$, and $v\acute{}_3$). Basically OpenCV computes the transformation that fits the random displacement of these three points, this transformation is M_1.

[2] In practice new patches are constantly generated while the neural network is training.

$$\begin{bmatrix} \acute{x_i} \\ \acute{y_i} \end{bmatrix} = M_1 \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{2.1}$$

$$v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{2.2}$$

$$v_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ S_p \end{pmatrix} \quad v_3 = \begin{pmatrix} S_p \\ S_p \end{pmatrix} \tag{2.3}$$

$$\acute{v_i} = v_i + \frac{S_p}{5} \begin{pmatrix} R_{i,0} \\ R_{i,1} \end{pmatrix} \tag{2.4}$$

where:

$$i \in \{1, 2, 3\} \tag{2.5}$$

$$S_p = \text{Patch size} \tag{2.6}$$

$$R_{i,d} \in [-1, 1] \text{ is a uniform random number} \tag{2.7}$$

Some projects use more sophisticated image distortion techniques that stretch and squeeze multiple parts of the patch[3].

3. **Horizontal flipping (reverse rows)** Because in this project information along the vertical axis has an important meaning (transition from dermal to epidermal tissue), only horizontal flipping is applied with a 50% probability[4].

4. **Color saturation/value shifting and hue rotation** By converting the RGB color data of the patch to HSV[5], it is easy to apply a few color transformations. In this project the hue, saturation, and value are all offset by a random number. For the saturation and value this offset is

[3] An example of this can be found in the source code of the winning solution for the MICCAI 2017 Endoscopic Vision Sub-Challenge: Robotic Instrument Segmentation. See `https://github.com/ternaus/robot-surgery-segmentation`[17].

[4] Patches are generated on the fly. This means that, when a new patch is generated, a uniform random number generator is used to determine if flipping will be applied

[5] In the equations that follow each channel has a floating point value between 0 and 1.

computed by finding the minimal shift that would cause overflow for each pixel and multiplying this by a uniform random number in the range $[-1, 1]$ (Equation 2.8). The hue is offset by a random number as well, but wrapped around to prevent overflow (Equation 2.10)[6].

$$R_{[x,y]} \text{ is a uniform random number in the range } [x, y] \tag{2.8}$$

$$\text{Shift}(x, R_{[-1,1]}) = \min\{x, 1 - x\} \cdot R_{[-1,1]} \tag{2.9}$$

$$\text{Rotate}(x, R_{[0,1]}) = (x + R_{[0,1]}) \bmod 1.0 \tag{2.10}$$

5. **Normalization** The last transformation is image normalization. The normalized image is computed by subtracting the mean and dividing by the standard deviation (Equation 2.11). This kind of normalization is very common in image processing tasks because it standardizes the data to a set of numbers that is always similarly distributed around 0.[7]

$$N(P, x, y) = \frac{P[x, y] - \text{mean}(P)}{\text{stdev}(P)} \tag{2.11}$$

2.1.3 Ground-truth

The ground-truth data contains traced nerves and traced skin surfaces. From this data a number of masks are generated that will be used to train the neural network. The skin surface annotation is used to generate a skin mask that is used to ignore all pixels that do not belong to the normal skin (also excluding the layer of dead skin). An example of this mask is shown in Figure 2.6.

For reasons that are explained in the next section, three other masks are extracted from the nerve annotations: a background tissue mask, a nerve

[6]The saturation/value shifts and hue rotation that are used here are rather extreme. In our case it turned out to work reasonably well but that is no guarantee that this is a good augmentation strategy. More trials with varying augmentation parameters would have to be conducted to find this out. It could be argued that by introducing these more extreme augmentations, the validation/testing set appear as a small subset of the data that was used for training. This can be a good thing because the network is forced to learn the features from topologies, rather than from colors.

[7]It is now also popular to normalize the training batch between some network layers. This is called batch normalization[18]. Batch normalization is not used in this project.

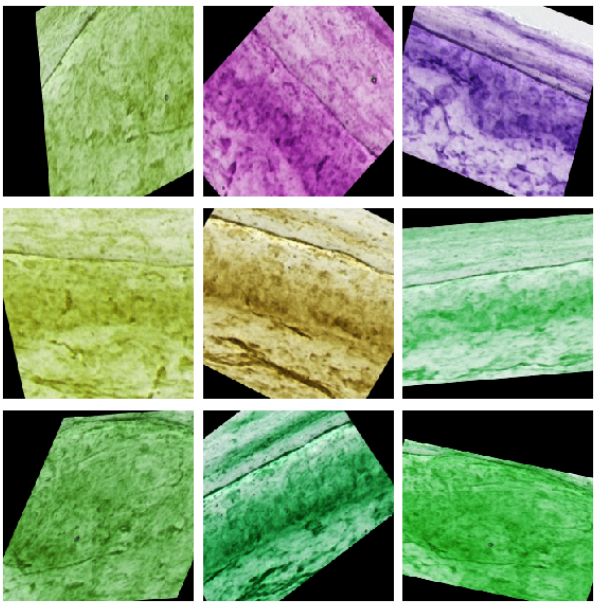
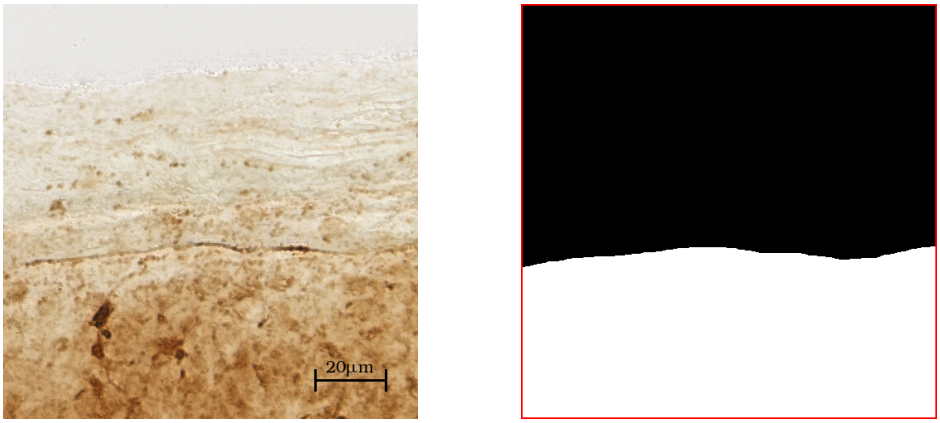
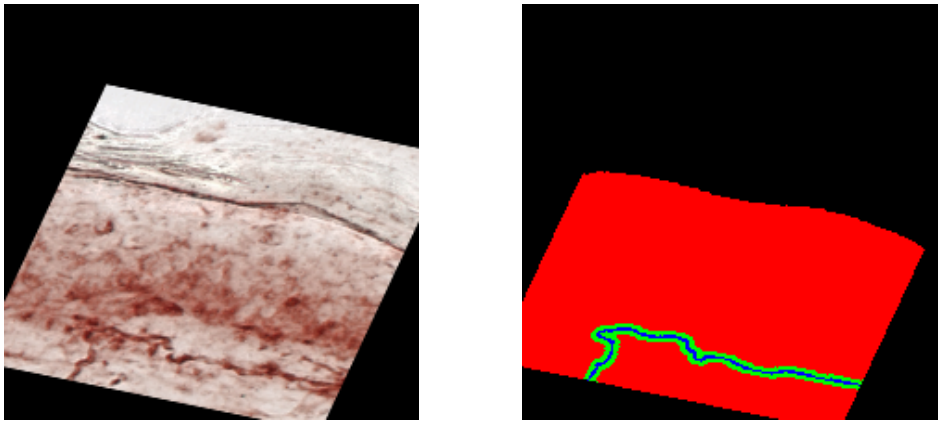


Figure 2.5: A few examples of augmented image patches before normalization.



(a) Input image (b) Mask

Figure 2.6: Skin mask example (red border is not part of the mask).



(a) Image patch (b) Ground-truth

Figure 2.7: An augmented patch and the corresponding transformed ground-truth. The background tissue, nerve boundary, and nerve class are colored red, green, and blue, respectively.

boundary mask, and a nerve mask. The nerve mask is generated by drawing 4 pixel thick lines along the traced nerve fiber coordinates. The nerve boundary mask covers an area of 6 pixels around each line in the nerve mask. Finally the background tissue mask is generated by subtracting the nerve and nerve boundary mask from the skin mask. The coordinate transformations that are applied to the image patches are also applied to the corresponding ground-truth. An example of this is shown in Figure 2.7.

2.2 Neural network

2.2.1 Architecture

The neural network architecture (or 'model') that is used in this project is based on the U-Net model[13]. The first part of the neural network uses the same design as the first five blocks of VGG16[19] (a CNN). Pre-trained weights are used for this part, which helps improve the training results. This is a strategy that can also be found in literature and has been shown to produce better results for some cases[8][20]. The network[9] input is a 3-channel (RGB) image patch with size 448×448[10]. Because the original images have a relatively good resolution, the patch is first down-scaled to 224×224 using linear interpolation[11].

At the network 'bottom' there is a residual block, a type of connection that was suggested in 2015 to allow training much deeper neural networks[21]. It is included in this model because is has been used successfully in other segmentation models such as LinkNet-34[22][17]. The up-scaling blocks in the model use an equivalent number of filters as the corresponding down-scaling blocks from VGG16[12]. Figure 2.8 depicts the entire neural network as a simplified flowchart. A detailed description can be found in Appendix A.

Finally three 1×1 convolutions are used to produce a 3-channel prediction, each channel representing one segmentation class. In the same order as the output channels these classes are: *background tissue*, *nerve boundary*, and *nerves*. The nerve boundary class is included to help the training converge by allowing the network to predict something between nerve and tissue. A softmax function is applied to the output layer to determine the winning class

[8] Pre-training can be particularly effective when a neural network is first trained on a similar (not identical) problem for which there is much more training data available. By pre-training the neural network it can learn filters that may also be useful for the target problem.

[9] For convenience the term *neural network* is often referred to as *network* in this thesis.

[10] It is important that the input patch is relatively big in order for the neural network to get enough context information for detecting nerves.

[11] Manual inspection of down-scaled images showed that nerves are still visible to the human eye at a 50% scale. It was suggested to me to use down-scaling because it reduces the number of parameters in the network (less down-scaling blocks are necessary to find similar global features). This should make it easier to train the network.

[12] There is no direct reason to pick the same number of up-scaling filters as down-scaling filters. As a general rule the number of filters should increase when removing localized information (in this case by max-pooling), and decrease when adding localized information (in this case by bilinear upscaling and concatenation).

for each pixel. The similarly named masks that are described in Section 2.1.3 are used to train the network.

2.2.2 Loss function

The loss function that is used to train this network is a mix between the multi-class Jaccard coefficient and categorical cross-entropy, both with pre-computed weights per class. As described in the previous section, the network outputs a three-dimensional result for each input pixel representing the probability that it belongs to one of the pre-defined classes (numbered 1 to 3 in the equations that follow). Since the images contain a disproportionate amount of tissue, weights are introduced into the loss function to make all three classes equally important. The weight for each class is computed using Equation 2.13 where X_i indicates the total area of class i in the entire dataset (in number of pixels).

$$W_{net} = 1 = \sum_{i=1}^{3} W_i X_i \tag{2.12}$$

$$W_i X_i = \frac{1}{3}, \quad W_i = \frac{1}{3X_i} \tag{2.13}$$

Both cross-entropy (2.14) and the Jaccard coefficient (2.15) are used because together they are expected to produce a better landscape for converging[13]. Cross-entropy is suitable for per-pixel classification, and the Jaccard coefficient measures the similarity between predicted and actual nerve from the intersection to union ratio. Experiments showed that training using the Jaccard coefficient alone does not converge (data not shown). Therefore cross-entropy is added to create a good initial training gradient[14].

[13]There are various examples of mixing cross-entropy and the Jaccard coefficient. One loss that is used is $L = H - \log J$[17][23].

[14]There is no rigorous argument for using cross-entropy as secondary loss. I have been using it because of its popularity in deep learning applications including image segmentation.

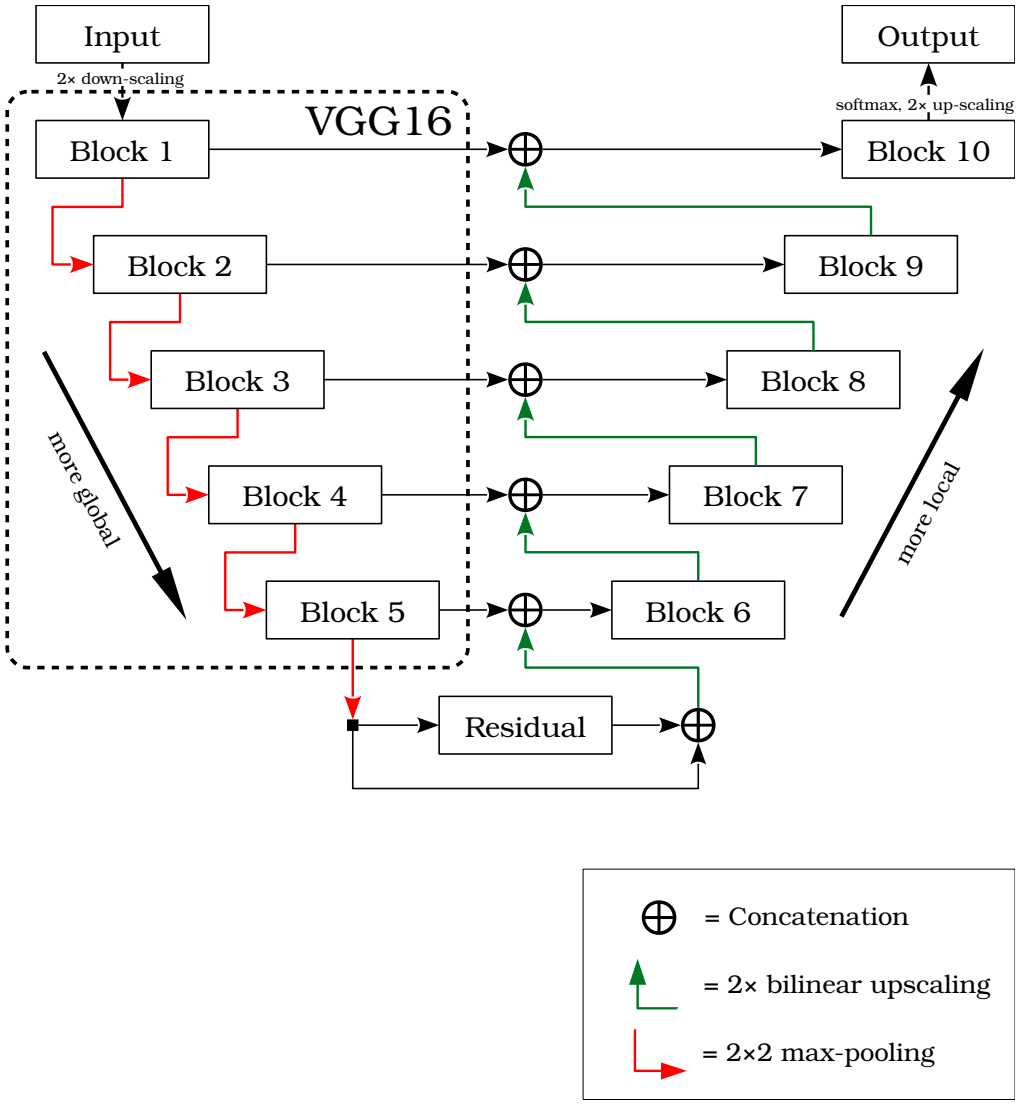


Figure 2.8: Flowchart of the neural network presented in this thesis. Each box represents several convolutional layers. The dashed box contains the part of the network that corresponds to VGG16.

$$H(Y_{\text{true},i}, Y_{\text{pred},i}) = -\left\langle Y_{\text{true},i} \cdot \ln(Y_{\text{pred},i}) \right\rangle \tag{2.14}$$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.15}$$

$$H_{net}(Y_{\text{true}}, Y_{\text{pred}}) = \sum_{i=1}^{3} W_i \cdot H(Y_{\text{true},i}, Y_{\text{pred},i}) \tag{2.16}$$

$$J_{net}(Y_{\text{true}}, Y_{\text{pred}}) = \sum_{i=1}^{3} W_i \left(1 - J(Y_{\text{true},i}, Y_{\text{pred},i})\right) \tag{2.17}$$

Because early experiments in this study showed that the cross-entropy strongly diverges again when the Jaccard coefficient is still steadily converging, a pre-factor that depends on the Jaccard coefficient is added to the cross-entropy in order to decay its influence after the training progresses. This pre-factor is plotted in Figure 2.9. The final loss is computed as follows:

$$L(Y_{\text{true}}, Y_{\text{pred}}) = 25\left[S(J - 85)\right]^{\frac{1}{10}} \cdot H + J \tag{2.18}$$

where:

$$J = J_{net}(Y_{\text{true}}, Y_{\text{pred}}) \tag{2.19}$$

$$H = H_{net}(Y_{\text{true}}, Y_{\text{pred}}) \tag{2.20}$$

$$S(x) = \frac{e^x}{e^x + 1} \quad \text{(Sigmoid function)} \tag{2.21}$$

2.2.3 Training procedure

The VGG16 part in the neural network is initialized with weights that have been pre-trained on the ImageNet challenge[19]. All other weights are initialized with a random initializer[15].

[15]The default weight initializer in Keras is used. At the time of writing, this is the Glorot uniform initializer[24].

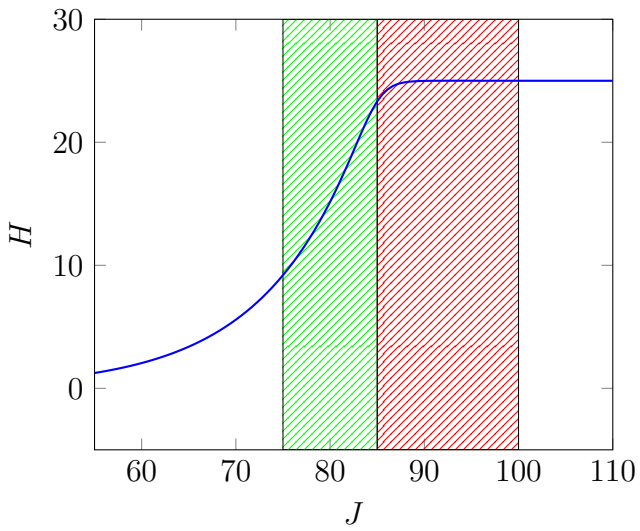


Figure 2.9: Curve of the heuristic cross-entropy pre-factor: $25\,[S(J-85)]^{\frac{1}{10}}$. The red region roughly covers the value of J during the steep decline in the first 100 epochs. The green region represents the rest of the optimization (the Jaccard coefficient converges roughly at 80-75 while the cross-entropy is already diverging at this point).

The 96 images are randomly shuffled and split into 6 sets of 16 images. Each set is labeled D_i where $1 \leq i \leq 6$. A total of 5 networks[16] is trained using the same initialization weights to cross-validate the performance and reproducibility. The training, validation, and testing data for each network are defined by (2.22), (2.23) and (2.24) where n denotes the network number ($1 \leq n \leq 5$). So one sixth of the data is put aside for testing, the rest is used to create 5 different training/validation combinations.

$$Tr_n = \left[\bigcup_{i=1}^{5} D_i \right] - D_n \tag{2.22}$$

$$Vl_n = D_n \tag{2.23}$$

$$Ts_n = D_6 \tag{2.24}$$

[16]The different trained weights are referred to as separate networks despite sharing the same architecture. In some cases they are referred to as CNN (convolutional neural network) to distinguish from other methods.

Each network is trained for 1500 epochs, each epoch containing 64 'batches' of only one patch[17]. After each epoch the validation loss is computed based on 8 randomly cropped and flipped patches from the validation set. The final set of weights for each network (after 1500 epochs of training) is picked based on the lowest validation loss.

Learning rate

In this project the Adam optimizer is used[18][25]. It is important to start with the highest possible learning rate[19]. In this project the initial rate is set to 1×10^{-4}.

One method to slowly reduce the learning rate is to decrease it as function of the epoch index. Another method is to reduce it when the validation loss encounters a plateau. This is the case when no improvements are found after n epochs (the patience). The learning rate is then multiplied by a factor f. In this project the second method is used with $n = 100$ and $f = 0.5$. The learning rate is not reduced below 1×10^{-6} because at this point it has practically converged[20]. Since the cross-entropy shows quite unstable progress, the learning rate is reduced according to the validation Jaccard coefficient instead[21] (see Equation 2.15).

2.3 Evaluation

To compute a prediction for each test image, all horizontally shifted 448×448 patches in the test images are extracted in order to average these predictions. The prediction quality near the edge of a patch can be less accurate than in

[17]Usually the batch size is much larger, like 32 patches, in order get a more stable gradient. Unfortunately this was not possible for this project since the GPU that was used did not have sufficient memory.

[18]This was recommended as a good choice in general. I think that, if the optimizer can converge, it is not likely that much better results will be obtained with another optimizer. The learning rate is the important parameter because it determines the step size in the parameter space.

[19]The learning rate can be thought of as the step size along the network gradient. When this step size is too big the iterations practically step to arbitrary points in the parameter space. However, when the step size is too small the network will converge to a local minimum too quickly.

[20]This was also confirmed in experiments (data not shown).

[21]This choice is based on heuristics. Limited time has made this approach the one that was tried for the final comparison, although it is certainly not claimed as effective one.

the center because there is less contextual information[22]. Since the images are 640 pixels wide, for each image $640 - 447 = 193$ shifted patches can be extracted. The final prediction for each pixel in the test images is the average of all overlapping shifted patches (so pixels on the left and right edge are predicted only once and so on).

In order to determine the network performance we compute the spatial distance between the predicted and annotated nerves. This measure is also used in the *V3D* software to score traced neurons[27].

Given the nerve fibers A and B containing n_A and n_B equidistant (with respect to neighboring points) vertices, the directed divergence $D_{DIV}(A, B)$ is defined as the average distance from each vertex on A to the nearest one on B. The spatial distance (SD) is the average of $D_{DIV}(A, B)$ and $D_{DIV}(B, A)$. The mathematical description of this formula is given below where the coordinate vector of the n-th vertex on a nerve[23] X is given by X_i.

$$D_{DIV}(A, B) = \frac{1}{n_A} \sum_{i=1}^{n_A} \min\{||A_i - B_j||_2 \ : \ 0 < j < n_B\} \tag{2.25}$$

$$SD(A, B) = \frac{D_{DIV}(A, B) + D_{DIV}(B, A)}{2} \tag{2.26}$$

The 3-channel prediction produced by the neural network is processed into a nerve distance map[24] using the steps described below. Such a pixel-wise distance map is also computed for the annotations, and can be used to easily extract the distances that are required to compute the spatial distance.

1. **Thresholding** The third channel of the prediction[25] is thresholded to obtain a binary nerve mask.

2. **Skeletonization** A nerve skeleton is computed to get rid of the varying width of the nerve prediction.

[22]When regarding the convolution filters as indicating certain features, like demonstrated for ZF-Net[26], then zero-padding in the convolution layers essentially turns off all contextual features. It is not hard to imagine this might be a problem for the detection of nerves which strongly depends on contextual information as described in Section 1.5.

[23]Here we are talking about a single nerve fiber. However, eventually the measure is applied to a whole image that may contain multiple nerve fibers.

[24]In the distance map the value of each pixel contains the distance to the closest nerve.

[25]The third channel holds the nerve segmentation, see Section 2.2.1.

3. **Remove small nerve segments** Small nerve segments are removed[26] using the `skimage.morphology.remove_small_objects` algorithm from scikit-image (Python library)[28].

4. **Distance transform** Finally the Euclidean distance map is approximated with OpenCV using a 5×5 mask[27]

In order to obtain comparable results, the spatial distance is always computed for patches that correspond to 448 × 448 squares in the original test images. When the conventional method and the neural network are evaluated, such a patch is cropped from the left and right side of each test image. When a patch contains no nerve predictions, or no ground-truth nerves, the spatial distance cannot be computed (an empty array has no mean). These patches are ignored when computing the mean or median on the entire test set.

2.3.1 Conventional reference method

The neural network performance will be compared to human performance, and to the performance of a conventional method developed by Pontenagel [7] in 2017. His approach contains 4 main steps:[28]

1. **Select blue channel** By analyzing the histogram of some images it is determined that the blue channel *'is the best channel to use for contrast based thresholding for the epidermis segmentation'*[29]. It is used for all further processing, including the nerve segmentation.

2. **Frangi vesselness filter** A 2D Frangi vesselness filter is used to find vessel like structures in the grayscale image. The result of this step is shown in Figure 2.10c.

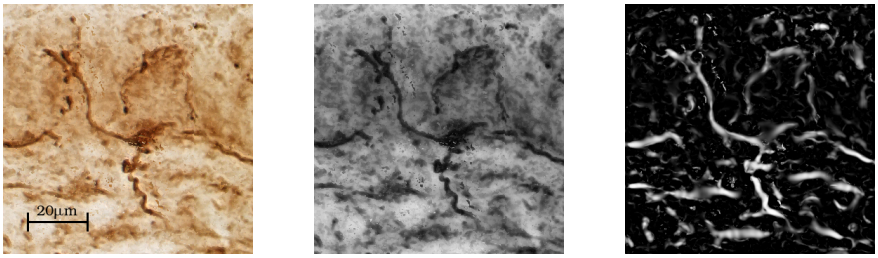
3. **Tensor voting** Tensor voting with steerable filters is used to emulate the perceptual grouping that is happening in the human brain when

[26] In practice is appears that short nerve segments have often not been annotated, causing some trouble for the training and evaluation.

[27] The algorithm that is used by OpenCV has been described by Borgefors [29].

[28] Unfortunately the original thesis does not appear to be available online. The *Biomedical Imaging Group Rotterdam* could be contacted to request the thesis documents.

[29] Apart from segmenting nerves this method also attempts to segment the epidermis so that the nerve crossings can be determined.



(a) Input image (b) Blue channel (c) Frangi vesselness

(d) Tensor voting (e) Thresholding (f) Filtered skeleton

Figure 2.10: Some intermediate images of the conventional nerve detection method.

looking for line like structures[30]. The intention is to find continuous nerve structures (connecting segments that point along the same direction), and create a good basis for thresholding. The result of this step is shown in Figure 2.10d.

4. **Thresholding and binary processing** A binary nerve mask is produced by thresholding on a fixed value (Figure 2.10e). From this mask a nerve skeleton is computed. Nerve segments that contain less than 50 pixels are removed (Figure 2.10f).

2.3.2 Human performance

To measure the human annotation accuracy, test subjects (expert and non-expert) will be asked to annotate nerves in a set of 100 random patches (generated as described in Section 2.1.2 excluding affine transformations, color augmentation and normalization). These annotations will be compared

[30]When looking for nerves the human brain is able to interpolate a lot of information. When parts of the nerve cannot be seen, perhaps because it is not part of that particular slice, a human can still reconstruct the presence of a line.

to the ground-truth in the same way as the neural network (e.g. the spatial distance is computed).

2.4 Implementation details

This section contains references to some of the software that was used to implement the method described here for anyone attempting to do a similar project. Everything is implemented in Python 3. NumPy is used to deal with most matrix data[30]. OpenCV and skikit-image are used for advanced image manipulation[31][28]. Keras is used to build and train the neural network[32]. A Python library called deepdish is used to deal with the large volumes of hierarchical data (processed annotations, images, masks, predictions, etc.). The NDPITools software is used to extract the regions of interest from the NDPI data files produced by the Hamamatsu NanoZoomer[33]. The neural network was trained using an NVIDIA GeForce GTX 980 GPU which has roughly 4GB of memory.

Chapter 3

Results

3.1 Training

The running average in Figure 3.1 shows that there is a converging and diverging trend, although the validation loss varies strongly between epochs. A potential reason for this high variability could be that the validation score is computed based on only 8 randomly cropped and flipped patches from the training set. If the segmentation difficulty varies strongly between patches, then picking one 'unlucky' patch will have a strong negative effect on the validation loss. In Section 3.5 a nerve is shown that is much more difficult to see than most other nerves. Such difficult nerves may account for some of the noise in both training and evaluation.

The neural network training shows some peculiar behaviour. It is common that the validation loss slightly diverges when the neural network starts to over-fit on the training set[1]. However, the cross-entropy diverges so strongly that it even exceeds the converged value after the first epoch (see Figure 3.2). This also strongly affects the mixed loss, despite the Jaccard coefficient dependent pre-factor that was added to prevent this[2]. It is unclear why this happens, but some ideas are discussed in Section 4.1.

The learning rate was intended to be reduced only when the validation Jaccard coefficient reaches a plateau. However, when plotting the training curves it turned out that this is not quite what happened. Instead the learn-

[1] A way to think of this is that instead of learning universal patterns, the neural network starts to remember patterns that are specific to the training images. This would cause worse performance on the validation images.

[2] The parameters for this pre-factor were no more than an educated guess. It is certainly possible that another pre-factor performs much better.

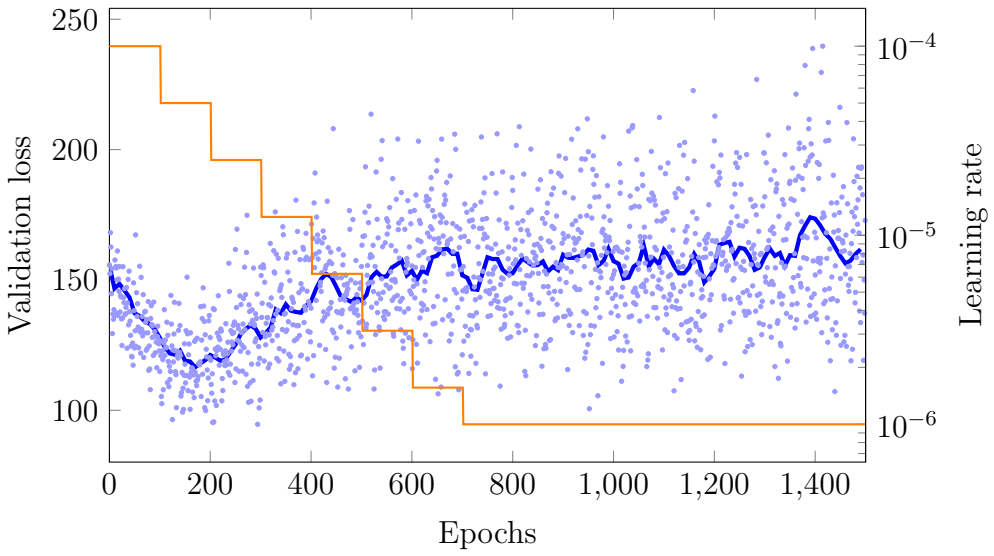


Figure 3.1: Validation loss of the first network ($n = 1$, see Section 2.2.3). The solid blue line represents a running average of the raw data points (also blue). The orange line corresponds to the right axis and represents the learning rate.

ing rate has been reduced every time the patience threshold of 100 epochs was reached (as can be seen in Figure 3.3). Further inspection showed that this was caused by an implementation error which is discussed in Section 4.2.

3.2 Post-processing parameter space

The post-processing steps that are applied in order to evaluate the neural network performance have two parameters: a thresholding level and a minimum object size (for filtering). Figure 3.4 shows the median spatial distance score for a range of parameters. Roughly it can be said that a lower threshold value can be compensated with a higher object size filter. The first optimum is at $L = 0.25$ and $O_{min} = 60$. These settings are used for all further evaluation of the neural network. The exact values of the first 10 optima are listed in Table 3.1.

3.3 Cross-validation

Five networks were trained using a different training/validation data split in accordance with the method. An important question is now how large the

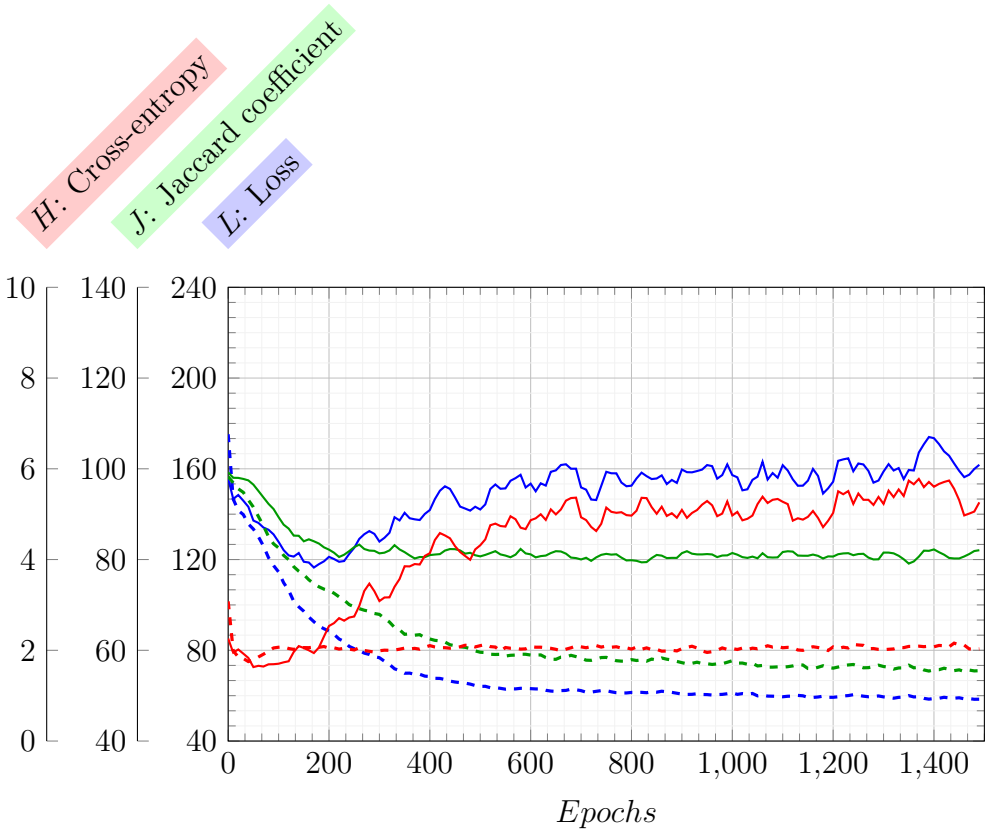


Figure 3.2: Loss, weighted categorical cross-entropy, and Jaccard coefficient of the first network. The solid lines represent the running average of these measures for the validation images, and the dashed lines for the training images.

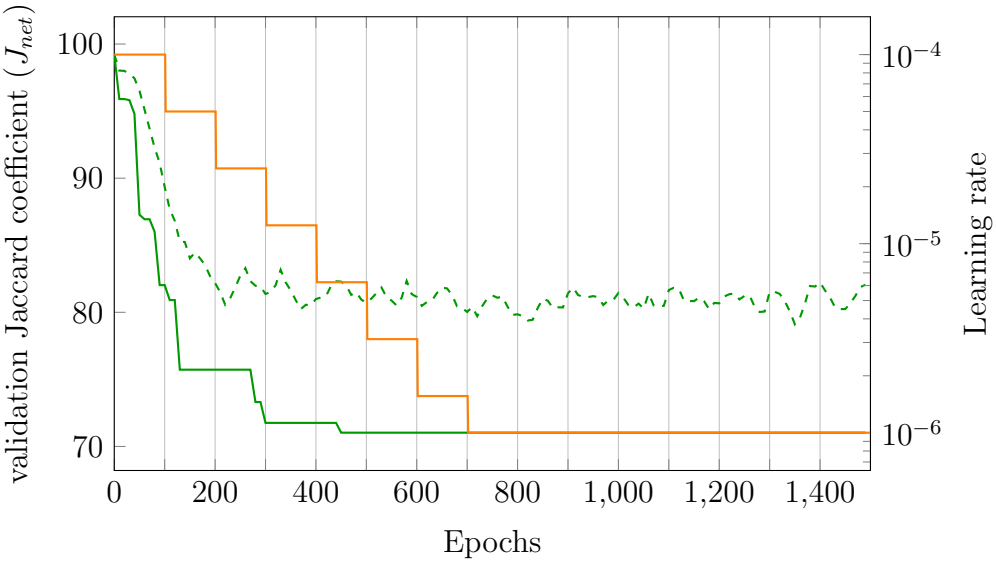


Figure 3.3: Learning rate of the first network (orange) together with the validation Jaccard coefficient (the solid green line represents the minimum Jaccard coefficient since the first epoch, and the dashed green line the running average) which is used for automatically dropping the learning rate. It is apparent that the learning rate has been reduced every time the patience threshold of 100 epochs was reached instead of when a plateau is reached.

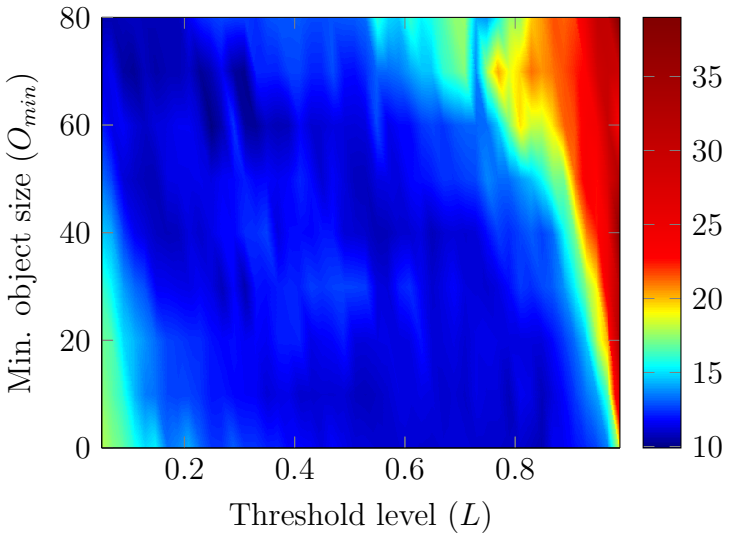


Figure 3.4: Median spatial distance between ground-truth and prediction for different post-processing settings.

Table 3.1: The top 10 optimal settings for threshold level and minimum object size for minimizing the spatial distance median. The 4th column contains the total number of patches that contained any nerve predictions after the corresponding settings were applied.

Threshold	Min. size	SD median	Patch count
0.25	60	9.94	145
0.31	70	10.14	141
0.23	70	10.26	144
0.11	70	10.26	144
0.33	60	10.27	145
0.31	60	10.38	145
0.29	70	10.4	141
0.09	70	10.47	145
0.15	70	10.51	144
0.31	30	10.6	146

variation is between the predictions by these networks. We compared three factors:

1. **The optimized validation loss value** The validation loss improvements during the first 500 epochs are plotted in Figure 3.5. The final validation loss and median spatial distance of each network is listed in Table 3.2. Figure 3.6 shows that the correlation between validation loss and median spatial distance is not very strong (not enough data is available to make any rigorous statements). Potential reasons for this include: (1) the loss function or evaluation measure is unsuitable, or (2) the training set is not very representative and therefore the test set gives strongly varied results.

2. **The spatial distances of each network** The distribution of spatial distance scores for each network, and for the median and average predictions of all five networks combined, is shown in Figure 3.7. There are some quite extreme outliers such as a value of 234 for network 2 which is not shown in the plot[3].

[3]Given that the spatial distance can be thought of as an average distance between the prediction and ground-truth in pixels, 234 is a huge deviation. This outlier could have been caused by an entire nerve that is not segmented (maybe because the network has not learned certain subtle edge features).

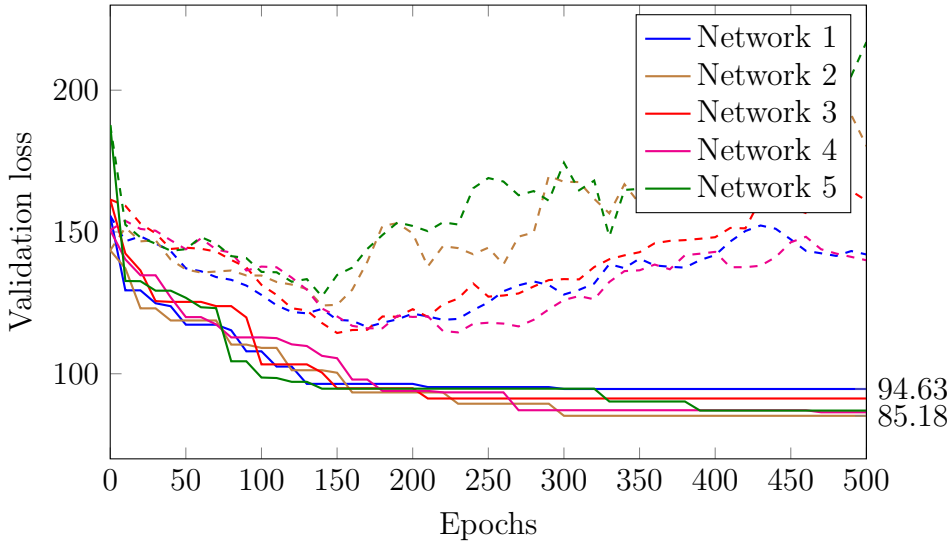


Figure 3.5: Running average (dashed) and minimum (solid) of the validation loss of all five network for the first 500 epochs (all networks converge in this interval). On the right axis the values of the smallest and largest final minimum loss values are shown.

3. **The spatial distances between networks** The spatial distance between each network and every other one was computed. All possible combinations are plotted in Figure 3.8. It is clear that the networks do not compute very consistent predictions. This is an important indication that the training sets are not representative enough to get reproducible training results. Assigning different training images could cause each network to learn different features than the other networks (this is why taking the median of the prediction of each network is effective). Another explanation is that the training converges to a local minimum too quickly, and that each network has converged to a different, bad local minimum[4].

[4]This is made less likely by the fact that the same initialization weights were used for each network. So each network started at the same point in the parameter space.

Table 3.2: Validation loss minimum for each network.

Network	Validation loss	SD median
1	94.63	16.56
2	85.18	12.08
3	91.28	12.06
4	86.42	12.57
5	87.02	9.88

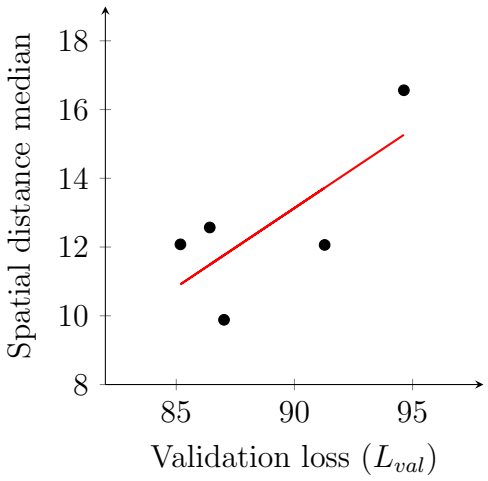


Figure 3.6: Linear regression between validation loss and spatial distance median (Table 3.2).

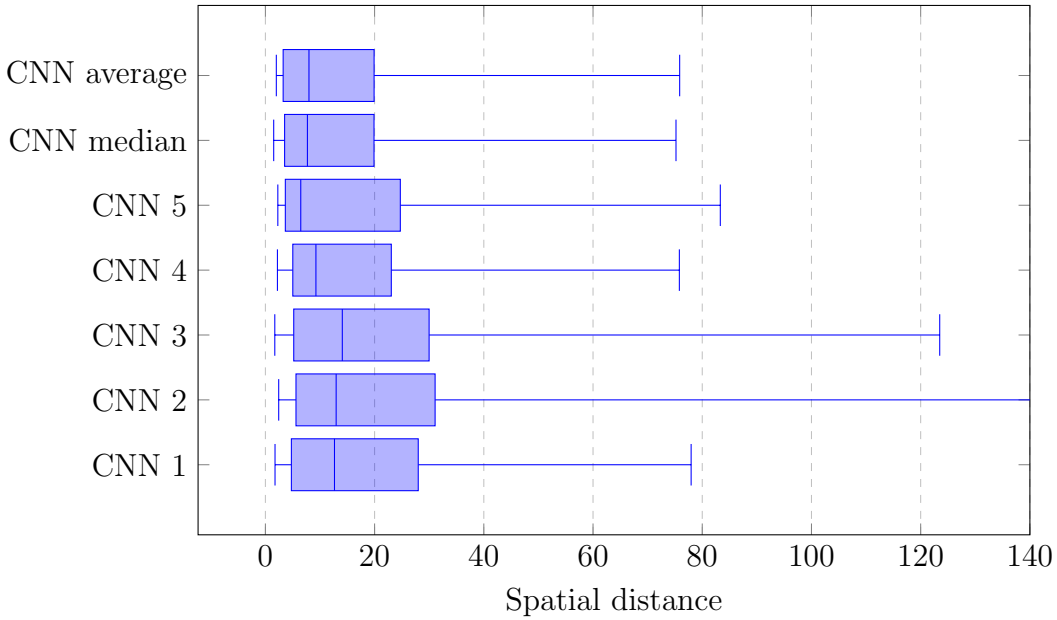


Figure 3.7: Distribution of the spatial distance scores for each network and the median and average prediction of all networks combined. The whisker top and bottom of the box-plot represent the smallest and largest value. Network 2 has one outlier at 233.52 that is not shown.

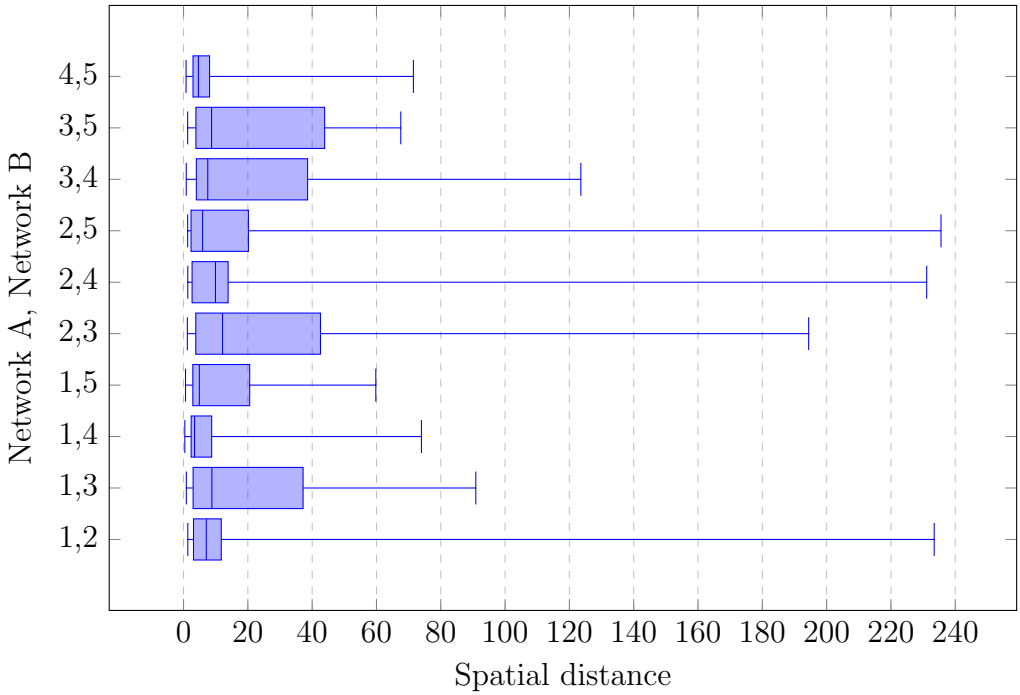
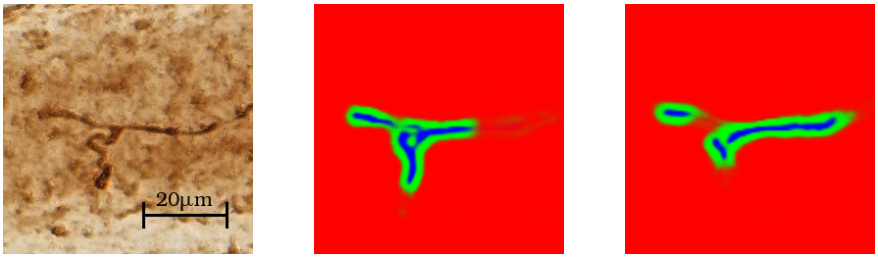
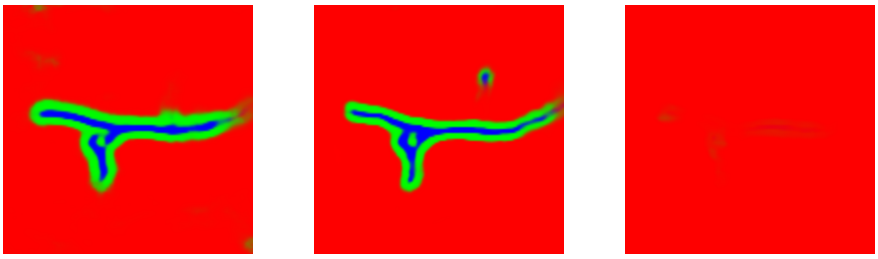


Figure 3.8: Distribution of the spatial distance between the trained networks.



(a) Input image (b) CNN 1 (c) CNN 2

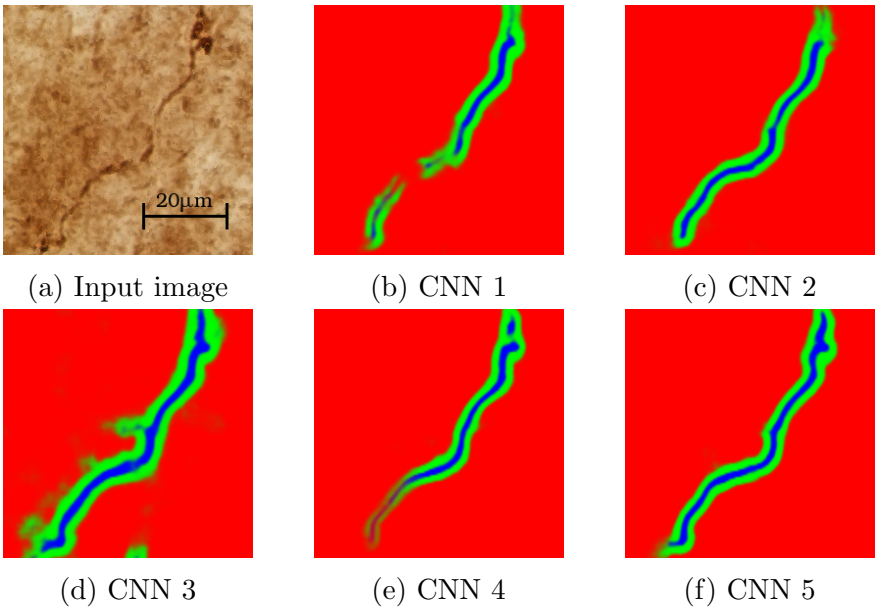


(d) CNN 3 (e) CNN 4 (f) CNN 5

Figure 3.9: Example prediction from each network. Substantial differences between the predictions are visible. The fifth network fails to predict the nerve.

The cross-validation factors that we have looked at show that there are significant differences between each network. Although this can be attributed to different reasons (such as an unsuitable loss function causing bad convergence), it is quite reasonable to conclude that the training images are not representative enough for the trained networks to learn very similar image features. To obtain a more reproducible result the test set has to be larger. Figure 3.9 shows an example segmentation from each network with substantial differences between each other[5]. Fortunately there are also examples where there the differences are a bit more subtle, such as the one in Figure 3.10. Small differences between the five networks are almost universally noticeable across the test set. Using the median or average of all predictions is a powerful way to obtain better results. To get a more complete impression of the neural network predictions, the average prediction for all test images is included in Appendix B.

[5]I think that, when looking at this figure, it is not so hard to imagine the large number of different edges and nerve structures that must be learned.



(a) Input image (b) CNN 1 (c) CNN 2

(d) CNN 3 (e) CNN 4 (f) CNN 5

Figure 3.10: Another set of example predictions.

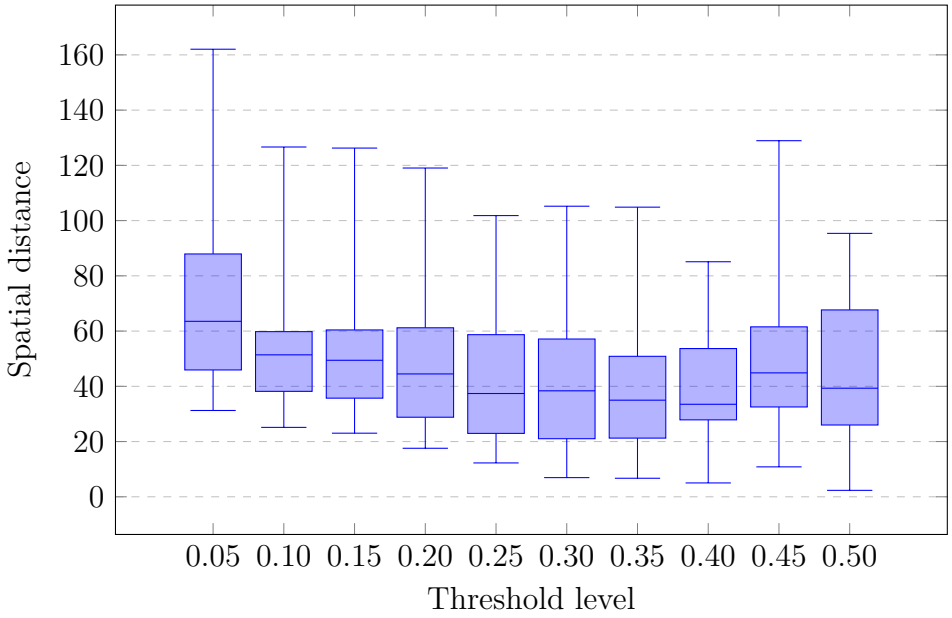


Figure 3.11: Distribution of the spatial distances computed for the 16 test images using the conventional method when using different values to threshold the tensor voting output. The results for a threshold value larger than 0.50 are omitted because they are very bad.

3.4 Comparison to other methods

3.4.1 Optimizing the conventional method

The conventional method is especially sensitive to the threshold level that is applied after tensor voting. To get a fair comparison this value is optimized for our testing set by evaluating all threshold values in $\{0.05, 0.10, ..., 0.95\}$. The result of this analysis is plotted in Figure 3.11. The best performing threshold value 0.40 is picked for the final comparison.

3.4.2 Final results

The performance of the neural network is compared to the conventional method developed by Pontenagel [7], and to the performance of a few human test subjects. Two human novices and an expert were asked to trace nerves manually. They were given unlimited time. The test subjects have different experience levels with this type of task that are interesting when interpreting the results:

- **Expert** This is the same person who created the ground-truth annotations 1-2 months earlier. It should therefore be reasonable to expect that this person can perform very well with respect to the ground-truth[6].
- **Novice 1** The first novice is me (the author of this thesis). I have no specialist knowledge to tell me what a nerve should look like. However, I have seen plenty examples of nerves over the past months, and discussed some considerations related to tracing nerves with the expert.
- **Novice 2** The second novice is someone with no prior training at all. Only a few examples of nerve fibers in the test images have been shown before the test.

The results of this experiment are plotted in Figure 3.12 together with the conventional method and the results of the neural network median (because it performed best). Only the human expert performs better than the deep convolutional neural network presented in this thesis. The neural network defeats the conventional method by a big margin, and performs similar to a human novice. This shows that deep CNNs are a promising and suitable technique for nerve segmentation[7]. In the next chapter a number of improvements to the method presented here are discussed that could result in even better results from the neural network.

[6]It should be noted that in this test the images are focus-stacked, while the ground-truth was annotated in Aperio ImageScope (by Leica) where you can scroll through the focal planes manually. Whether or not this is an advantage is discussed in Section 4.5.1.

[7]If you find this quantitative analysis unconvincing, or a bit limited, perhaps the images in Appendix B can convince you instead.

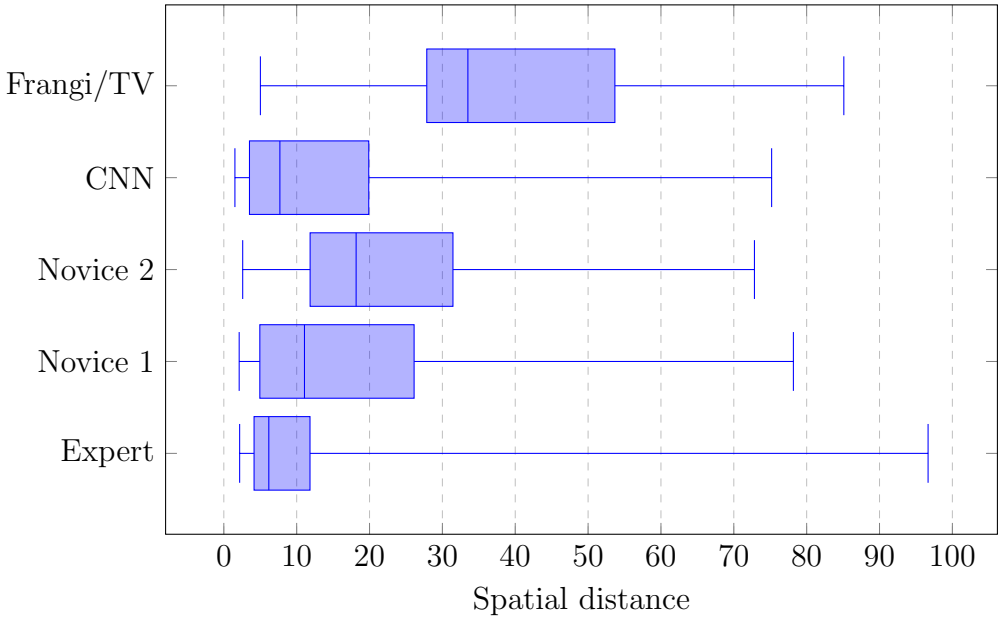
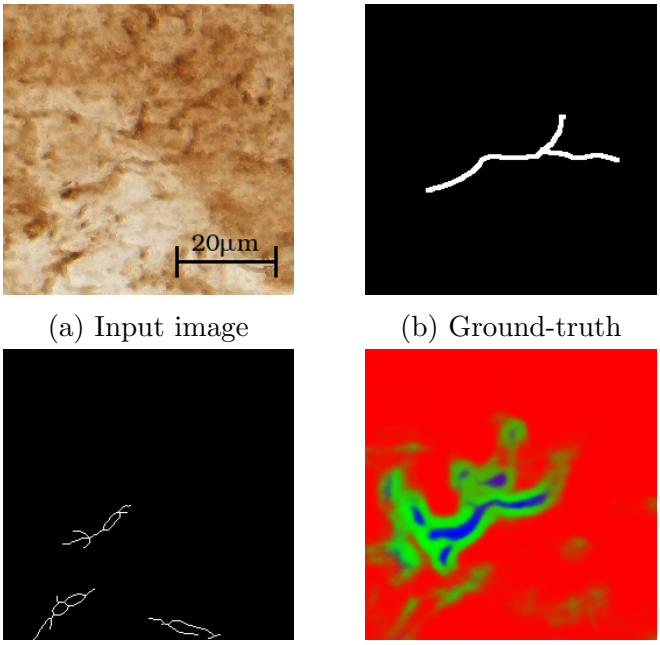


Figure 3.12: Distribution of the spatial distances computed for the 16 test images by different methods. Exact values can be found in Table 3.3.

Table 3.3: Data points that are presented in Figure 3.7 and 3.12. Q1 and Q3 are the first (25% of the data is better) and third (75% of the data is better) quartile.

Method	Q1	Median	Q3	Min.	Max.
Expert	4.15	6.18	11.83	2.17	96.67
Novice 1	4.94	11.07	26.12	2.11	78.18
Novice 2	11.84	18.16	31.45	2.59	72.83
CNN 1	4.76	12.65	27.99	1.76	77.96
CNN 2	5.6	12.96	31.08	2.44	233.52
CNN 3	5.19	14.09	29.98	1.72	123.49
CNN 4	5.02	9.25	23.06	2.2	75.81
CNN 5	3.65	6.48	24.71	2.28	83.31
CNN median	3.52	7.69	19.89	1.52	75.19
CNN average	3.25	7.98	19.91	2	75.87
Conventional	26.61	35.01	45.7	19.08	63.95



(a) Input image (b) Ground-truth

(c) Conventional method (d) CNN 3

Figure 3.13: A nerve that causes significant trouble for the neural network. The nerve here is hard to see, but in the ground-truth it is annotated as a nerve.

3.5 Outliers

In the results there are some large outliers. To illustrate where these may come from we look at one example image[8] that, according to the ground-truth, shows a nerve fiber. This nerve fiber is particularly hard to see, even for a human observer, and unsurprisingly all networks except for the third one fail to recognize any nerve in this area[9]. The conventional method does detect a part of this nerve along with some false positives.

[8]See image 6 in Appendix B.

[9]When looking at other predictions of network 3 it is apparent that it is much more sensitive for subtle nerve fiber features (data not shown).

Chapter 4

Discussion

This chapter touches on several problems that were encountered during the development and evaluation of the neural network, and makes some recommendations for further work.

4.1 Cross-entropy divergence

It was found that the cross-entropy diverges very strongly after a minimum (see for example Figure 3.1). To prevent this from affecting the composite loss too much the pre-factor could be shifted, although it is questionable if the pre-factor is a good idea at all. I think the more important and interesting question is why the cross entropy shows this behaviour. I can think of one reason: the class weights. They could be causing this because they create a huge difference between the way false negatives are treated by the categorical cross-entropy and the Jaccard coefficient.

The categorical cross-entropy penalizes false negatives for each class, but not false positives. Instead, false positives are penalized as false negative for another class (note that the softmax function will make sure the sum of all classes per pixel is one). This would not be a major problem if we weren't using per-class weights. However, the weight that is applied to the nerve class is almost 73 times that of the background class because there are much more background pixels than nerve pixels in the ground-truth. This means that for the nerve class, cross-entropy highly penalizes false negatives, but not false-positives. The Jaccard coefficient however does not suffer from this problem. As training goes on the Jaccard coefficient forces the network to make sharper predictions in order to improve the intersection-over-union. This could help create false negatives that are highly penalized by the nerve cross-entropy.

So the suspicion is that cross-entropy would like the prediction to be more spread out (more 'speculative'), while the Jaccard coefficient wants the prediction to be more sharp in order to obtain a better intersection-over-union ratio. It would be interesting to see what happens when the weighted categorical cross-entropy is replaced with multi-class binary cross-entropy.

4.2 Learning rate implementation error

Inspecting the Keras source code revealed why the implementation that was supposed to reduce the learning rate on plateaus (like illustrated in Figure 4.1) did not work properly. The Keras 'callback', a function that is executed after every epoch, that was used for automatic learning rate reduction (`ReduceLROnPlateau`) takes the name of the metric that should be checked after each epoch as input in order to determine if a plateau has been reached. The metric name I used for the multi-class Jaccard coefficient is `'val_jaccard_multi_coeff'`. It turns out that by default Keras checks if the metric name contains the string `'acc'` to determine if it should be minimized or maximized[1]. Keras thought I wanted to maximize the Jaccard coefficient (only because the metric name we picked happened to contain `'acc'`) and was looking for plateaus in the positive direction, which did not occur. I found this unexpected because elsewhere the default is always to minimize (unfortunately the API documentation does not specifically warns for this special behaviour). There was not enough time to compute new results.

4.3 Spatial distance shortcomings

Although the spatial distance does a relatively good job at producing easy to understand numbers, there are some serious drawbacks. For example, a small false positive could benefit the overall score when it is close to a false negative. In this case $D_{DIV}(Y_{\text{pred}},\ Y_{\text{true}})$ would not be much worse (see Section 2.3 for the equations), but $D_{DIV}(Y_{\text{true}},\ Y_{\text{pred}})$ would be much lower. I think that a future comparison should look for a different way to measure the similarity between prediction and ground-truth. I looked at the histograms of the individual distances that are averaged in D_{DIV}, but I did not manage to process this into something that could be presented in this report. Perhaps

[1]The string `'acc'` is commonly used in Keras examples to denote accuracy, a property that should be maximized.

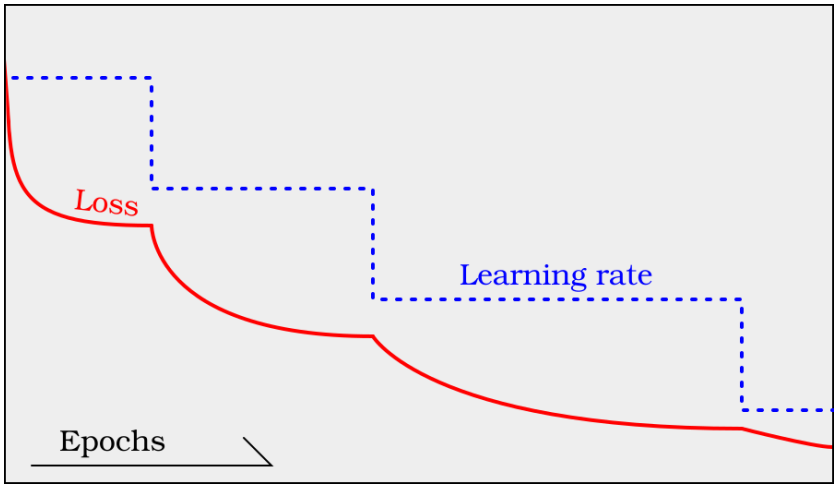


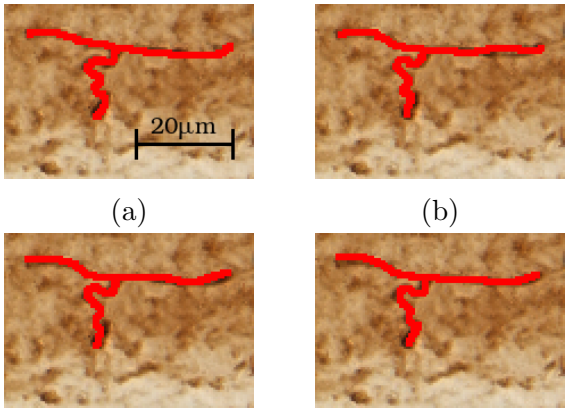
Figure 4.1: Artist impression of idealized loss plateaus and learning rate reduction.

a future study can come up with another similarity measurement. Recent work on developing a metric for scoring retinal vessel segmentations, such as [34], could be interesting to look at.

4.4 Ground-truth improvements

Earlier we concluded that the training sets are not representative enough for highly reproducible results, and that therefore more data (in the form of annotated images) is required. Another problem with the data is its low precision. Figure 4.2 shows a few duplicate expert annotations. At the pixel level quite some variation can be observed between them. While it cannot be expected of a single human annotator to reproduce annotations with 100% agreement, I suspect this uncertainty does create a problem for training the neural network. If there were enough training examples (and sufficient GPU memory to allow larger batches), this could probably be overcome. But this is not the case, and therefore I suspect the network has a hard time to learn definite features in some cases. An edge might belong to a nerve in one example, but not in the next one.

It would certainly be a good idea to gather more data, and achieve a higher precision (for example by having a group of experts produce annotations, and computing the median). More information-rich annotations that do not only indicate the nerve fiber centerline, but also the thickness, could



(a) (b)

(c) (d)

Figure 4.2: The same nerve from three annotated patches in the human evaluation test as annotated by the expert. This is the same nerve as in Figure 3.9.

improve the performance as well[2].

Another aspect that has not been discussed so far is the origin of the training data. In this study only data from volunteers was used. However, it is not a given that the nerve fibers in SFN patients look similar enough to those in healthy volunteers, that volunteer data is sufficient for training a neural network that can detect nerves in patients with the required accuracy. Further studies should consider using a more diverse set of data for training to ensure reproducible results across different targets.

4.5 Further work

4.5.1 Focus stacking

As described in the methods, a Laplacian-of-Gaussian was used for z-projection of the focal planes recorded by the microscope. Other methods such as local variance might produce better looking results (although that doesn't mean the neural network will automatically be able to perform better). The network could also be designed to deal with the different focal planes by adding a z dimension to the input tensor. This could even allow the network to

[2]Since in the current ground-truth all nerves have the same width, the network does not only have to learn to detect nerves, but also to filter this information in such a way that it can compute a constant-width segmentation in the end.

extract more information; while discussing the annotation routine I observed that scrolling through focal planes is often done to see if a line structure linearly propagates through the focal planes (if adjacent parts are in focus in adjacent focal planes, this can hint at a nerve fiber). It could certainly be the case that this makes it easier for the neural network to detect nerve fibers, and given the high amount of subjectivity involved in this task it could be critical for further improvement.

4.5.2 Secondary network

In order to count nerves and further automate the IENFD measurement, a secondary neural network could be developed that takes the nerve segmentation that is computed by this network (or an analog), along with the input image, as input. This network could learn to segment the dermis and epidermis based on both the tissue texture, and the location of nerve fibers. If this is successful the next step could be to automatically count the intra-epidermal nerve fiber crossings.

4.5.3 Different architectures

While developing the neural network presented here a lot of layers were added in an attempt to get better results. Although I am quite sure that the max-pooling layers are very important (to find global features), some of the convolution layers can probably be removed. There are a lot of large convolution layers since the VGG16 architecture was copied for the first half of the network. Apart from trying to shrink the model, there are also newer segmentation CNN models (such as LinkNet[22]) that would be interesting to evaluate.

4.6 General recommendations

I thought it would be useful to sum up some findings that lead to good results when developing a convolutional neural network for image segmentation.

1. **Enough pooling** If a neural network fails to learn features that can only be constructed from pixels that are far apart, it is critical that more pooling layers are added in order to allow the network to collect global information, before computing the final localized prediction. It can be hard to find out if this is indeed the limiting factor. Fortunately too much pooling is not immediately harmful.

2. **Good augmentation** One of the central challenges of training a neural network is over-fitting. Initially I tried to prevent this by adding dropout layers and reducing the number of parameters using less pooling. However, this approach is not very productive. It was advised to me not to use dropout in CNNs, and certainly not to compromise the network flexibility/depth. Instead stronger augmentation, such as contrast and hue shifts, and affine transformations, proved a more successful method.

3. **Stick to standard models first** During my study I had a tendency to try specialized approaches, such as introducing pixel based weights into the loss function. Implementing such approaches can be fun (and time consuming), but according to my limited experience they do generally not create huge improvements. Instead it is more productive to first modify more simple aspects of the network design, such as the amount of pooling, or the type of loss function, in order to find an approach that roughly solves the problem. If this doesn't work there are also more sophisticated models for segmentation problems that have been published (such as the aforementioned LinkNet). When there are specific problems, such as the incorrect segmentation of two adjacent cells as one (this was the case in the original U-Net paper), then it is a good time to start trying specialized solutions such as per-pixel weighted cross-entropy.

4. **Start with a high learning rate** When I noticed that my network was showing very chaotic training behaviour I decided to remedy this by reducing the initial learning rate. This does indeed produce a more stable training curve because smaller steps are taken through the parameter space. However, with small steps the network will also more quickly converge to a local minimum. For good results the learning

rate should start as high as possible so the gradient descent can make big steps through the parameter space until it reaches some hyper-dimensional 'area' where jumping around in big steps doesn't improve the loss anymore. At this 'plateau' the learning rate should be reduced so that a new plateau can be found within this area, and so on (illustrated in Figure 4.1).

Acknowledgment

For technical advice and discussions, I thank my supervisor Erik Meijering, as well as Ihor Smal (postdoc at the Biomedical Imaging Group Rotterdam) and Aleksei Tiulpin (data scientist in the field of Medical Technology and PhD candidate at Oulu University, Finland). Through a couple of discussions Aleksei helped me to find a working neural network architecture and suitable loss function. I also thank the Erasmus MC neurology department for providing us data for this study. And finally I thank Malik Bechakra (PhD candidate at Erasmus MC Rotterdam) for his work on data acquisition and collection, annotation, and discussions.

Appendices

Appendix A

Architecture

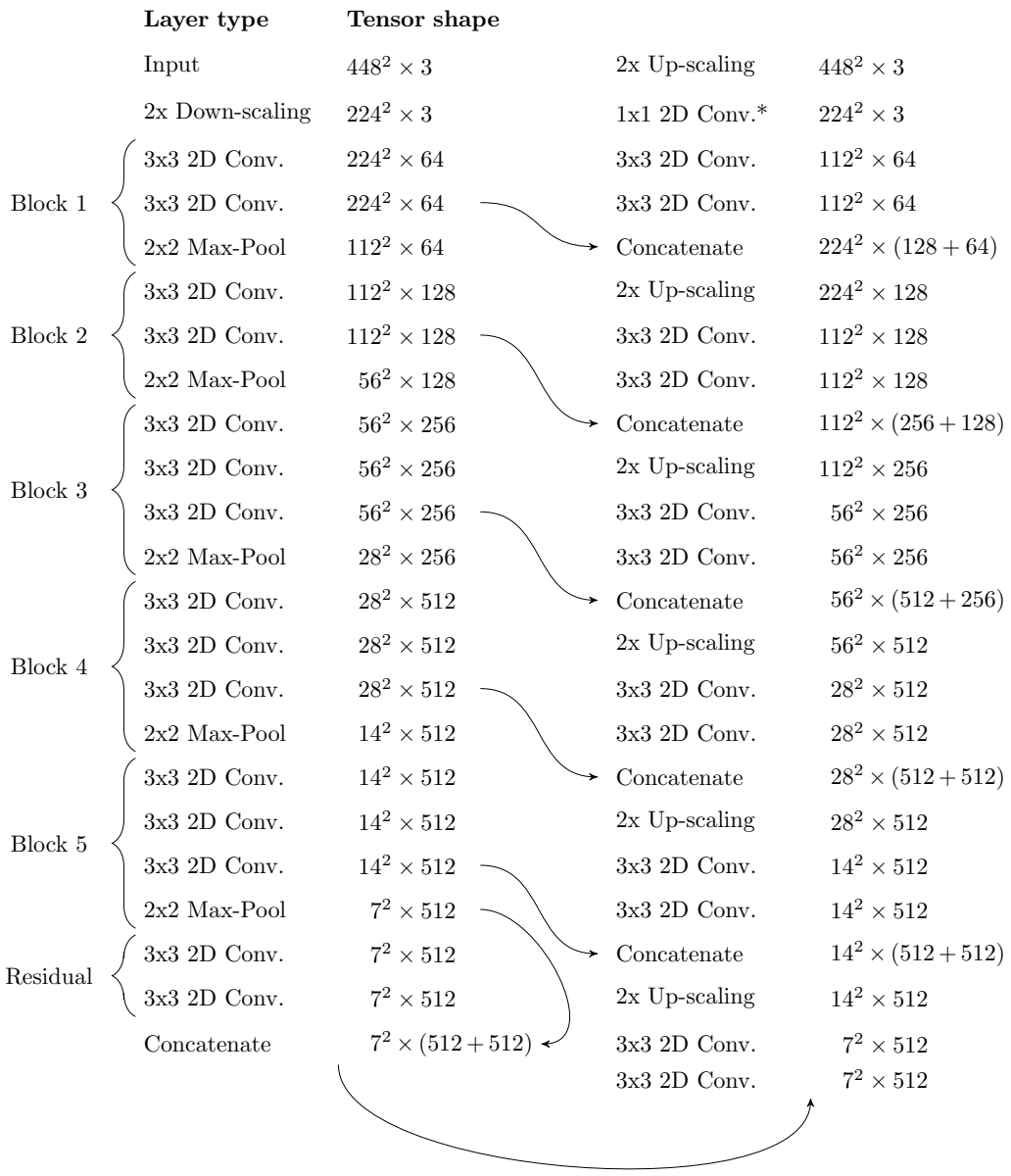


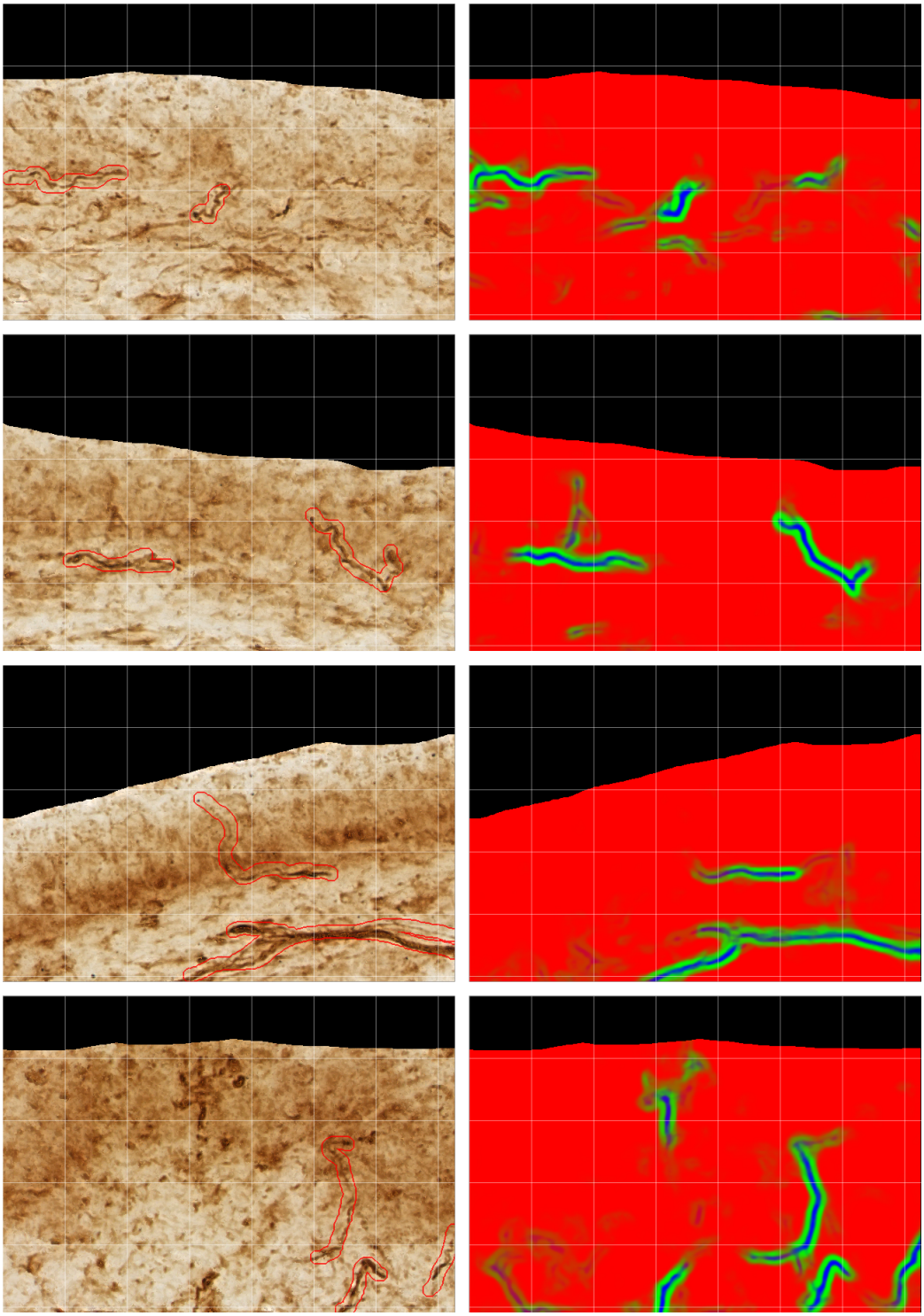
Figure A.1: Detailed network architecture diagram containing the tensor size at each layer (excluding the batch size dimension). Block 1-5 correspond to the five convolution blocks in VGG16. The number of convolutions that is applied in a convolution layer can be determined from the last tensor shape dimension. All convolution layers use zero padding and apply the ReLU function to the convolution output.

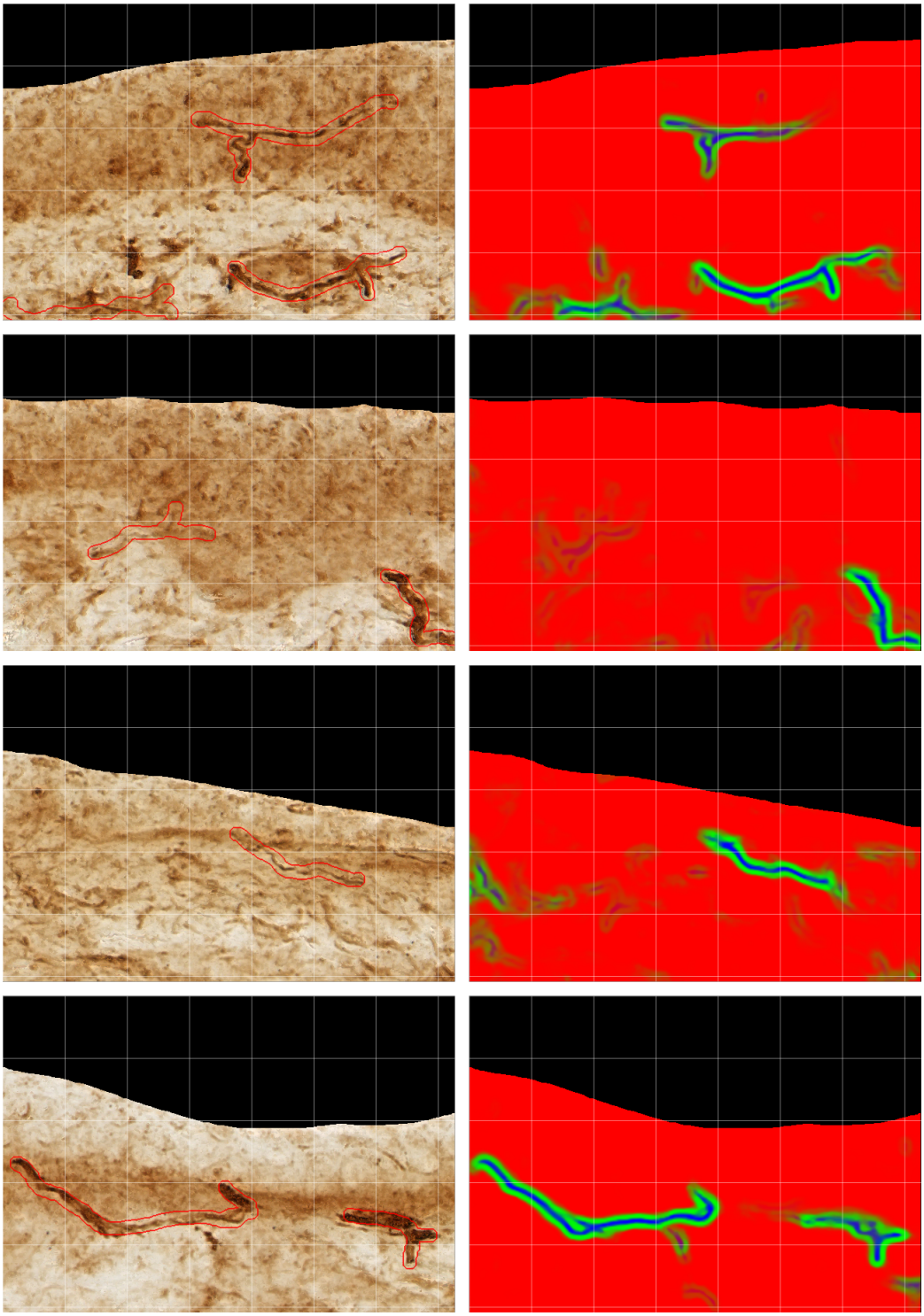
*Uses softmax activation.

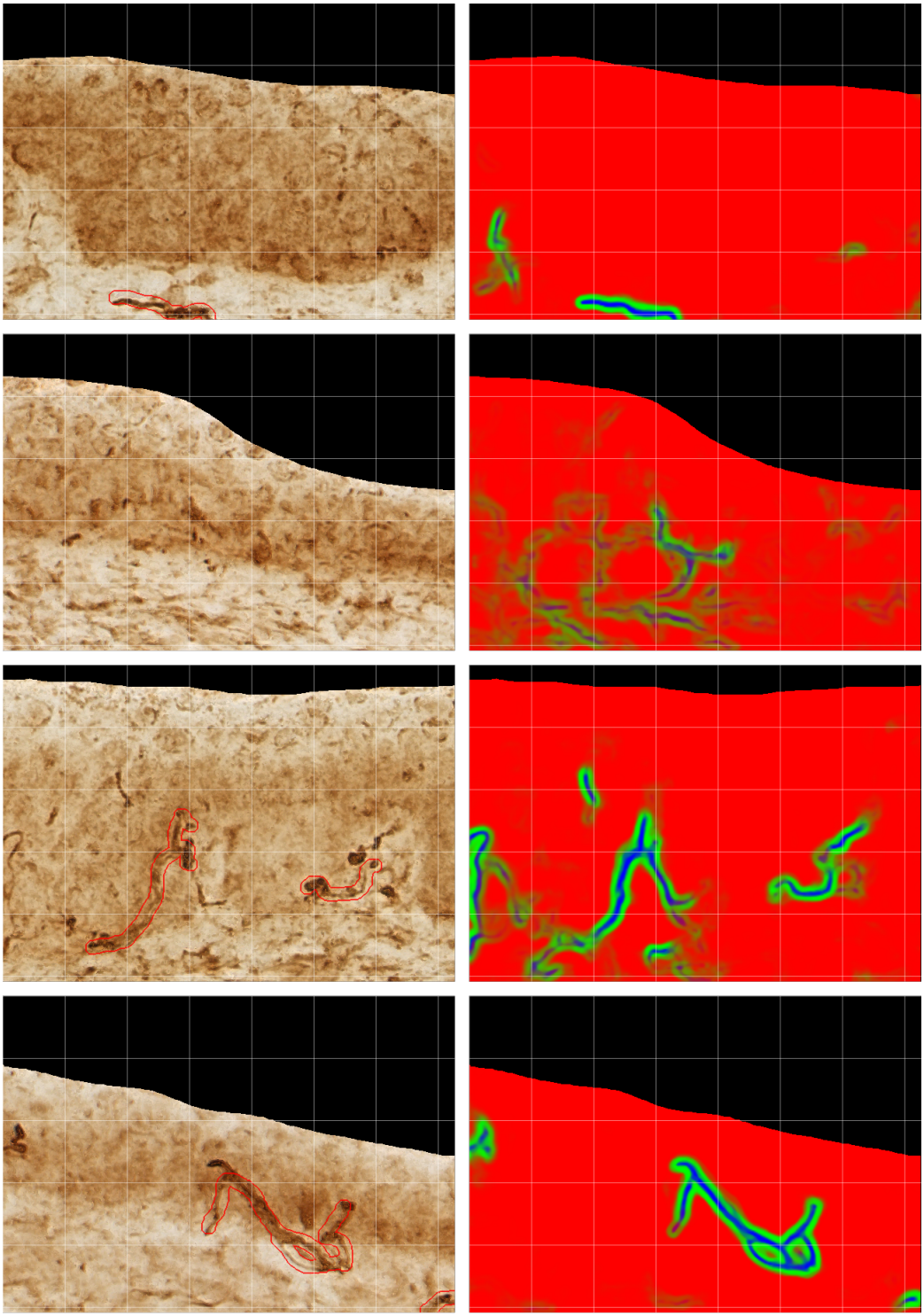
Appendix B

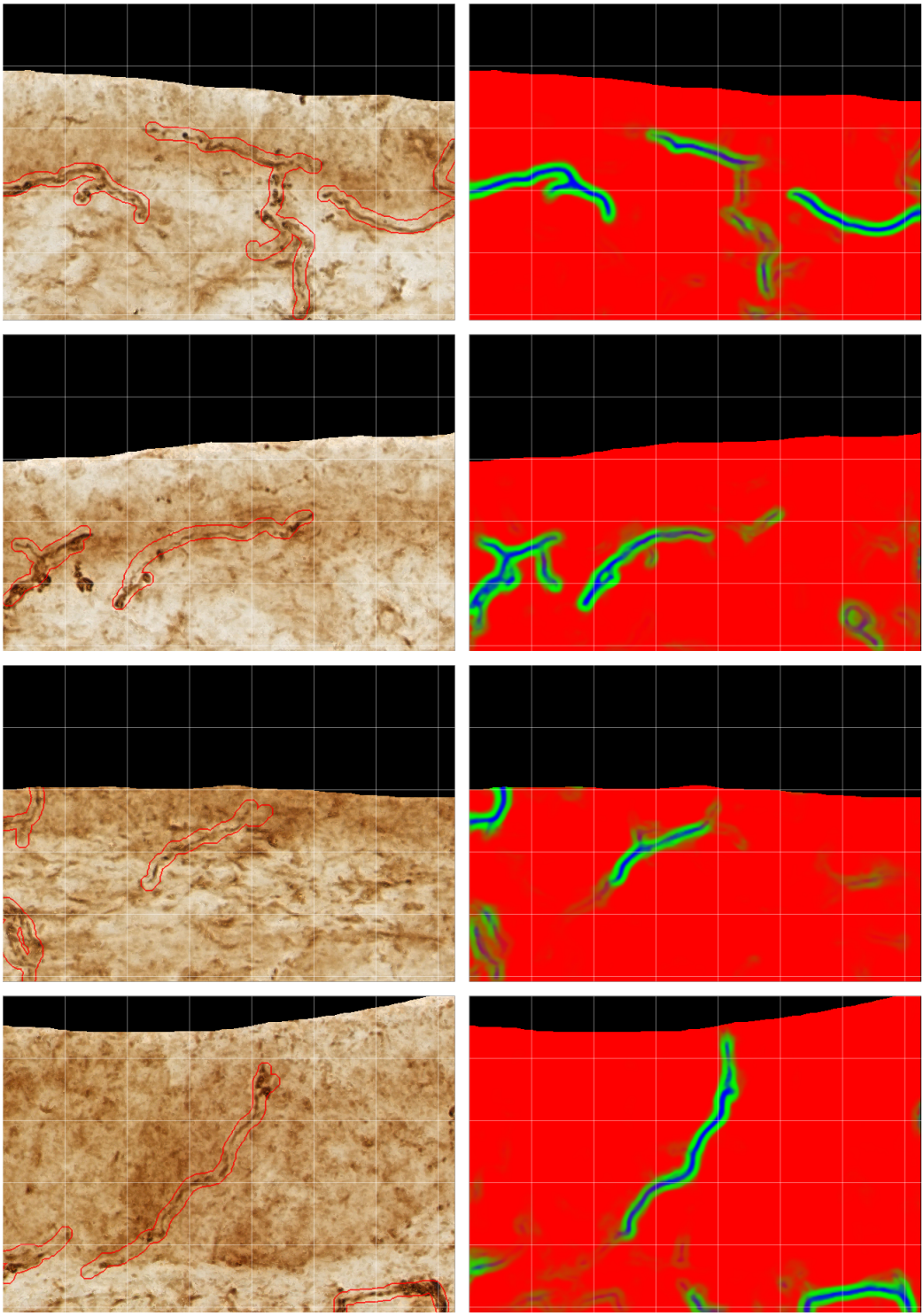
Predictions

The following images are the averaged predictions for all 16 test images. I decided to include the average predictions because they reveal a bit of the variation between the five trained networks (i.e. faded out predictions were produced by only one). In the input images (left) the ground-truth annotations are circled by a red line (the enclosed area exactly represents the nerve and boundary class). I think these images demonstrate the variability in the ground-truth. Some structures I would classify as nerve are not annotated as such (and I believe there is a good change some nerves are missing in the ground-truth). However, I also find the predictions of the neural network stunningly accurate, in particular considering the total training set contained only 80 images. A grid is drawn on top of all images with a step size of $20\mu m$.









Bibliography

[1] National Institute of Neurological Disorders and Stroke, *Peripheral neuropathy fact sheet*, [Online; accessed 20. Jun. 2018].

[2] L. Z. Jinny Tavee, "Small fiber neuropathy: A burning problem," *Cleveland Clinic Journal of Medicine*, pp. 297–305, May 2009.

[3] G. Lauria, R. Lombardi, F. Camozzi, and G. Devigili, "Skin biopsy for the diagnosis of peripheral neuropathy," *Histopathology*, vol. 54, no. 3, pp. 273–285, Jul. 2008, ISSN: 1365-2559. DOI: 10.1111/j.1365-2559.2008.03096.x.

[4] W. Kennedy, G. Wendelschafer-Crabb, M. Polydefkis, and J. McArthur, "Pathology and quantitation of cutaneous innervation," English (US), in *Peripheral Neuropathy*. Elsevier Inc., Dec. 2005, vol. 1, pp. 869–895, ISBN: 9780721694917. DOI: 10.1016/B978-0-7216-9491-7.50037-5.

[5] G. Lauria, S. T. Hsieh, O. Johansson, W. R. Kennedy, J. M. Leger, S. I. Mellgren, M. Nolano, I. S. J. Merkies, M. Polydefkis, A. G. Smith, C. Sommer, and J. Valls-Solé, "European Federation of Neurological Societies/Peripheral Nerve Society Guideline on the use of skin biopsy in the diagnosis of small fiber neuropathy. Report of a joint task force of the European Federation of Neurological Societies and the Peripheral Nerve Society," *European Journal of Neurology*, vol. 17, no. 7, pp. 903–912, Jul. 2010, ISSN: 1468-1331. DOI: 10.1111/j.1468-1331.2010.03023.x.

[6] S. Shanon, S. Manuel, D. Kathrin, F. Stephan, P. Adrian, K. Thierry, S. Andreas, P. Axel, S. Claudia, and S. A. K, "A semi-automated method to assess intraepidermal nerve fibre density in human skin biopsies," *Histopathology*, vol. 68, no. 5, pp. 657–665, Aug. 2015. DOI: 10.1111/his.12794.

[7] P. Pontenagel, "Towards automated quantification of intra-epidermal nerve fibers for diagnosis of small-fiber neuropathy in brightfield microscopy," Master's thesis, Erasmus MC, TU Delft, Feb. 2017.

[8] J. Casanova-Molla, M. Morales, N. Solà-Valls, A. Bosch, M. Calvo, J. M. Grau-Junyent, and J. Valls-Solé, "Axonal fluorescence quantitation provides a new approach to assess cutaneous innervation," *Journal of Neuroscience Methods*, vol. 200, no. 2, pp. 190–198, 2011, ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2011.06.022.

[9] A. Sathyanesan, T. Ogura, and W. Lin, "Automated measurement of nerve fiber density using line intensity scan analysis," *Journal of Neuroscience Methods*, vol. 206, no. 2, pp. 165–175, 2012, ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2012.02.019.

[10] P. Vincenzo, N. Maria, S. Annamaria, C. Giuseppe, V. D. F., and S. Lucio, "Intraepidermal nerve fiber analysis using immunofluorescence with and without confocal microscopy," *Muscle & Nerve*, vol. 51, no. 4, pp. 501–504, Jul. 2014. DOI: 10.1002/mus.24338. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mus.24338.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 0001-0782. DOI: 10.1145/3065386.

[12] Litjens Geert, Kooi Thijs, Bejnordi Babak Ehteshami, Setio Arnaud Arindra Adiyoso, Ciompi Francesco, Ghafoorian Mohsen, van der Laak Jeroen A.W.M., van Ginneken Bram, and Sánchez Clara I., "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017, ISSN: 1361-8415. DOI: http://dx.doi.org/10.1016/j.media.2017.07.005.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[14] Ulman Vladimír, Maška Martin, Magnusson Klas E G, Ronneberger Olaf, Haubold Carsten, Harder Nathalie, Matula Pavel, Matula Petr, Svoboda David, Radojevic Miroslav, Smal Ihor, Rohr Karl, Jaldén Joakim, Blau Helen M, Dzyubachyk Oleh, Lelieveldt Boudewijn, Xiao Pengdong, Li Yuexiang, Cho Siu-Yeung, Dufour Alexandre C, Olivo-Marin Jean-Christophe, Reyes-Aldasoro Constantino C, Solis-Lemus Jose A, Bensch Robert, Brox Thomas, Stegmaier Johannes, Mikut Ralf, Wolf Steffen, Hamprecht Fred A, Esteves Tiago, Quelhas Pedro, Demirel Ömer, Malmström Lars, Jug Florian, Tomancak Pavel, Meijering Erik, Muñoz-Barrutia Arrate, Kozubek Michal, and Ortiz-de-Solorzano Carlos, "An objective comparison of cell-tracking algorithms," *Nature Methods*, vol. 14, p. 1141, Oct. 2017. DOI: http://dx.doi.org/10.1038/nmeth.447310.1038/nmeth.4473.

[15] Poplin Ryan, Varadarajan Avinash V., Blumer Katy, Liu Yun, McConnell Michael V., Corrado Greg S., Peng Lily, and Webster Dale R., "Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning," *Nature Biomedical Engineering*, vol. 2, no. 3, pp. 158–164, 2018, ISSN: 2157-846X. DOI: https://doi.org/10.1038/s41551-018-0195-0.

[16] C. L Srinidhi, P. Aparna, and J. Rajan, "Recent advancements in retinal vessel segmentation," *Journal of Medical Systems*, vol. 41, no. 4, p. 70, Mar. 2017, ISSN: 1573-689X. DOI: 10.1007/s10916-017-0719-2.

[17] A. Shvets, A. Rakhlin, A. A. Kalinin, and V. Iglovikov, "Automatic instrument segmentation in robot-assisted surgery using deep learning," *bioRxiv*, 2018. DOI: 10.1101/275867. eprint: https://www.biorxiv.org/content/early/2018/06/20/275867.full.pdf.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computing Research Repository*, vol. abs/1409.1556, 2014. arXiv: 1409.1556.

[20] V. Iglovikov and A. Shvets, "TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation," *Computing Research Repository*, vol. abs/1801.05746, 2018. arXiv: 1801.05746.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[22] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," *Computing Research Repository*, vol. abs/1707.03718, 2017. arXiv: 1707.03718.

[23] V. Iglovikov, S. Mushinskiy, and V. Osin, "Satellite imagery feature detection using deep convolutional neural network: A kaggle competition," *Computing Research Repository*, vol. abs/1706.06169, 2017. arXiv: 1706.06169.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations 2015*, 2015. arXiv: 1412.6980.

[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 818–833, ISBN: 978-3-319-10590-1.

[27] Peng Hanchuan, Ruan Zongcai, Long Fuhui, Simpson Julie H, and Myers Eugene W, "V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets," *Nature Biotechnology*, vol. 28, p. 348, Mar. 2010. DOI: http://dx.doi.org/10.1038/nbt.161210.1038/nbt.1612.

[28] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. 453, Jun. 2014, ISSN: 2167-8359. DOI: 10.7717/peerj.453.

[29] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, pp. 344–371, 1986.

[30] T. E. Oliphant, *Guide to NumPy*, 2nd. USA: CreateSpace Independent Publishing Platform, 2015, ISBN: 9781517300074.

[31] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

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

[33] C. Deroulers, D. Ameisen, M. Badoual, C. Gerin, A. Granier, and M. Lartaud, "Analyzing huge pathology images with open source software," *Diagnostic Pathology*, vol. 8, no. 1, p. 92, Jun. 2013, ISSN: 1746-1596. DOI: 10.1186/1746-1596-8-92.

[34] Z. Yan, X. Yang, and K. T. Cheng, "A skeletal similarity metric for quality evaluation of retinal vessel segmentation," *IEEE Transactions on Medical Imaging*, vol. 37, no. 4, pp. 1045–1057, Apr. 2018, ISSN: 0278-0062. DOI: 10.1109/TMI.2017.2778748.