Universiteit Leiden

Master Computer Science

Performance evaluation of CNN's used for image registration on serial section images

Name:	Casper Carton
Student ID:	s2584999
Date:	September 27, 2024
Specialisation:	Bioinformatics
1st supervisor:	Prof.dr.ir. F.J. Verbeek
2nd supervisor:	Dr. L. Cao

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Approximately 70% of the human genome has a zebrafish ortholog. Besides their genetic similarities, zebrafish also exhibit physiological similarities to humans, making the zebrafish model very useful in many biomedical researches. Using three-dimensional models of the zebrafish facilitates studies into spatial structure and morphology. To obtain these 3D models the 2D serial section images of the model have to be stacked on top of each other. During preparation of the sections the structural integrity in the third dimension is compromised. To resolve this problem rigid image registration is crucial. Four neural networks have been trained on the entire section images to find the required rigid transformations. These networks have difficulty registering the entire section images. So, there is room for improvement. We propose a novel approach of training these networks on solely the edges of the section images instead. Both methods are able to retrieve simulated transformations to a certain extend. However, our proposed method demonstrates more accurate results. The trained models show stable qualities, as application on a pre-aligned series does not result in deterioration of the 3D alignment. Therefore, the proposed approach looks promising and might be interesting to continue experimenting with non-rigid registration as well.

Contents

1 Introduction

Model organisms such as mice, fruit flies and zebrafish are used to study the human body genetically and physiologically. These three are considered popular model organisms due to over 70% of the human genome shared and having a homolog for over 75% of disease causing genes in humans. Besides sharing many genes with humans, these model organisms also show remarkable anatomical and physiological similarities. Therefore, the model organisms are used for research before application on humans [PN11, HCT+13].

To gain insights into the internal structures of model organisms, serial sectioning is applied. Serial sectioning is the physical process of acquiring a series of thin layers from biological tissue [Ver99]. The obtained slices allow detailed analysis within the object. However, the information gained from the two-dimensional slices is limited to the tissue cutting plane. Reconstruction of the three-dimensional model adds the possibility of studying spatial structure, morphology and organization of the biological tissue in three-dimensional space. A reconstructed three-dimensional model results in the combined information of both the internal and spatial structures.

By layering the consecutive sections on top of each other, a three-dimensional reconstruction of the model is created. However, during physically preparing the serial sections and scanning the section images under the microscope the structural integrity of the object in the third dimension can be compromised. Therefore, simply layering the section images on top of each other will not result in a correctly aligned three-dimensional model. To resolve this problem, image registration is essential [Ver96].

Image registration is the process of aligning two or more images of the same scene taken at different times, from different sensors or different viewpoints [ZF03]. During the registration process one of the images to be registered is used as a reference image. This reference image will not be transformed during the registration. In case of aligning two images, this means one image is used as reference on which the second image will be geometrically aligned. In case of registering multiple images there are two options. The first option is to assign one reference image on which all other images will be aligned. In case the images have a specified order, the second option is to align them serially. When aligning serially, the images are registered using the previous image in the dataset as reference.

Image registration can be divided into two approaches differing in the type of transformations used, rigid and non-rigid registration [ZDGZ20]. Rigid registration is bound to using linear transformations to modify the images. This entails the use of rotation and translation of the images. In contrast, non-rigid image registration, also known as elastic/deformable registration, allows for a non-uniform alignment of images, which consequently alters their shapes. This method results in changes to the pixel-wise relationships within the images.

As mentioned, image registration is crucial for three-dimensional reconstruction of the model using serial section images [Ver96]. When directly applying non-rigid image registration, the possibly large differences between two scanned section images can cause unwanted deformations. Because of that, it is necessary to first apply rigid image registration to create an initial alignment of the section images. Rigid image registration is a step in the alignment of serial section images that can not be skipped. Therefore, this project will focus on rigid registration of serial section images.

Some approaches to rigid registration of serial section images have been developed over time. The very first approach was to register the images by hand. However, the quality of the alignment is largely impacted by the expertise of the clinician. A second approach to the alignment task is the mechanical use of Chamfer matching. It optimizes the Chamfer distance between two serial sections directly during scanning in the microscope [Bor88].

Convolutional neural networks (CNN's) are known for their success in image classification. Theses CNN's are able to recognise and learn features in images allowing them to also learn distinction between classes. The ability to learn images makes CNN's not only successful in image classification, but image recognition tasks in general. CNN's have already shown to be successful in multi-modal image registration. This entails registering two images of the same object acquired with different imaging modalities. This shows that CNN's are able to learn the transformations in order to optimize the image alignment. Therefore, this research will investigate CNN's ability to learn to realign serial section images.

1.1 Research questions

In this research the performance of convolutional neural networks used for rigid image registration on serial section images will be evaluated. Four CNN's known for their success in image recognition tasks are trained and evaluated. To simplify the problem en reduce the complexity, the networks will be trained on lower resolution serial section images. The results obtained during this research by training on the entire serial section images indicates room for improvement. Therefore, we introduce an alternative approach to train the networks on solely the edges of the serial section images instead. In order to evaluate the performance of the networks a measure is created for determining the alignment quality. This study aims to compare both approaches to training and thereby answer the following research questions:

> *RQ1: How well are convolutional neural networks able to register serial section images?*
>
> *RQ2: Does applying edge detection on data support neural networks during their training?*
>
> *RQ3: Can we establish a measure for image registration quality that corresponds with the CNN operation?*

1.2 Thesis overview

The objective of this study is to apply convolutional neural networks to register serial section images. Moreover a measurement for the registration quality will be elaborated. The remainder of this thesis is structured as follows.

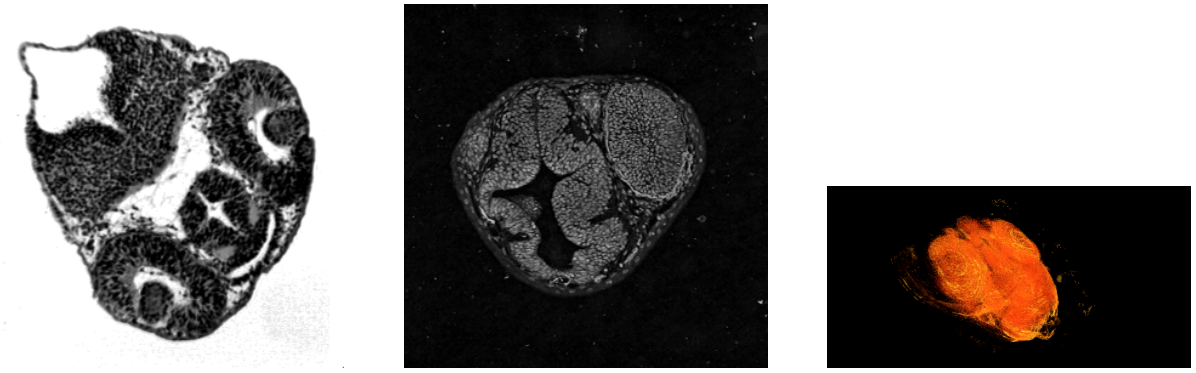
In chapter 2 the serial section images used for training the networks will be elaborated. Moreover, it addresses the networks and hardware used. A detailed explanation of the proposed implementation is given in chapter 3. Next, chapter 4 focuses on the experiments and discusses the obtained results to assess the performance of the networks. Lastly, in chapter 5 we answer the research questions and discuss the outcome with regard to limitations and future possibilities.

2 Material & methods

This section contains an overview of the datasets used during this project. Also the software, neural networks and hardware that have been used will be further specified.

2.1 Zebrafish data

The zebrafish model has been introduced for the first time by Streisinger et al. [SWD+81] in 1981 and is ever since used in many biomedical researches [CCL+21]. In particular human genetic studies advanced with the help of the zebrafish model, due to approximately 70% of human genes shared [HCT+13]. Although mice contain orthologs for 80% of the human genome, zebrafish have certain characteristics making them more advantageous. They are able to reproduce in large quantities and their transparent body allows in vivo imaging [SF02, LC07]. Because of these characteristics zebrafish are very useful in biomedical research.



(a) (b) (c)

Figure 1: The zebrafish data used during this project. (a) Example in the dataset of a serial section image resulting from Light Microscopy. (b) The serial section image used as start point to this project with dark background. Created from (a) by applying a Top-hat operation in order to get rid of some noise/debris present in the image. (c) The three dimensional model resulting from stacking one dataset of serial section images.

During this research three datasets are used for training and evaluation of the networks. The datasets contain Light Microscopy serial section images of the head of 48 hour old zebrafish models. These serial section images have a size of 300×300 pixels. Figure 1a shows one of the Light Microscopy serial section images in the dataset. Some of these images still contain noise/debris in the image, which can cause problems during registration. To reduce this problem a Top-hat transformation is executed. Furthermore, it is necessary the images contain a dark background so they are inverted. The result as shown in figure 1b is the state in which the images were received at the beginning of this research. This state is considered to be the "entire serial section image" on which the networks will be trained. Figure 1c shows the 3D model obtained after stacking the serial section images within a dataset.

Table 1 contains some information on the three datasets used. This entails the amount of images per dataset and what datasets are used for training and during evaluation. Two of the three datasets are used for training, the third is used to evaluate the trained networks. All three datasets have already been aligned using a mechanical Chamfer matching method [Bor88].

dataset	#slices	used for
1	52	training
2	99	training
3	125	evaluation

Table 1: Information of the three datasets used during this research.

2.2 Software

Python version 3.10.12 was used for the implementation. This allowed usage of several useful libraries supported by Python. Such as the Pillow package, part of the Python Imaging Library for reading images into arrays, and the scikit-image package for image processing. Moreover it allows the use of Tensorflow to implement and train the convolutional neural networks. Besides importing Python supported libraries, using Python allows usage of the Spatial Transformer.

2.2.1 Networks

Deep convolutional neural networks have proven to be capable of outstanding results in image recognition tasks [PK17]. On top of that, they have also proven to be applicable for several medical imaging purposes [SWS17, KK23]. Due to the success of deep CNN's in medical image recognition, three well known deep CNN's are used during this project. In addition, a non-deep, fully connected network has been implemented. To train these networks the Adam optimizer is used for backpropagation of the gradients. The four implemented networks are stated below.

1. The **VGG16** model is one of the two best performing VGG networks presented in the "Very deep Convolutional Networks for Large-Scale Image Recognition" paper [Sim14]. The authors were able to create a much deeper network than achieved up until then. This was due to the very small 3×3 receptive field of their kernels used in every convolutional layer. As shown in figure 2 the architecture of the network consists of 16 weighted layers, 13 convolutional and three dense layers. After each convolution block there is a max pooling layer. The three fully connected layers with 4096, 4096 and 1000 features at the end are followed by SoftMax activation.

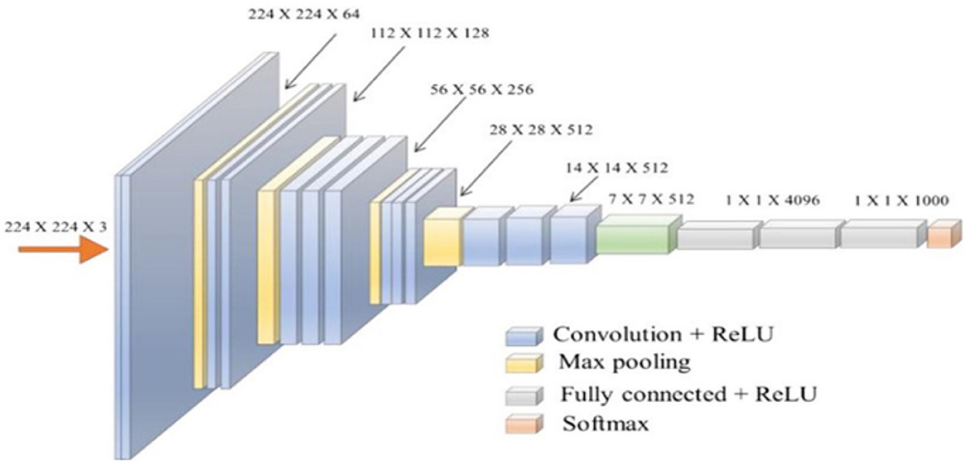


Figure 2: The VGG16 architecture by [Sim14], containing 16 weighted layers (13 convolutional and 3 dense layers).

2. The **VGG19** model is the other best performing VGG network made publically available [Sim14]. As shown in figure 3, the structure is the same as for the VGG16 network with one extra convolutional layer in the last three convolutional blocks. Resulting in 19 weighted layers in total, 16 convolutional and three dense layers. As in the VGG16 network, there are max pooling layers after each convolutional block and SoftMax activation after the three dense layers.

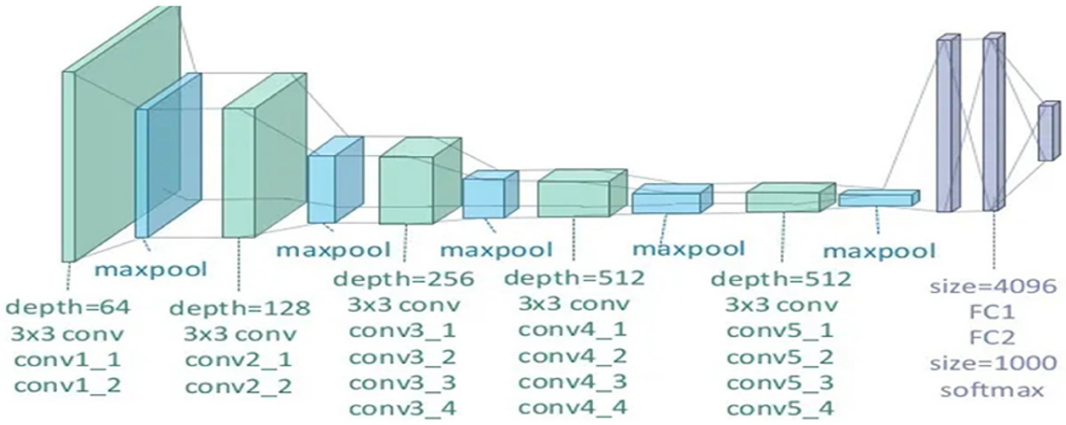


Figure 3: The VGG19 architecture by [Sim14], containing 19 weighted layers (16 convolutional and 3 dense layers).

3. The last deep CNN implemented in this project is the **ResNet50** model, a version of the Residual Network presented in the "Deep Residual Learning for Image Recognition" paper [HZRS16]. To create even deeper networks than the VGG networks for example but without a degrading accuracy, the authors proposed the use of a residual/identity block. This means, skip connections are introduced such that the gradient can bypass layers in

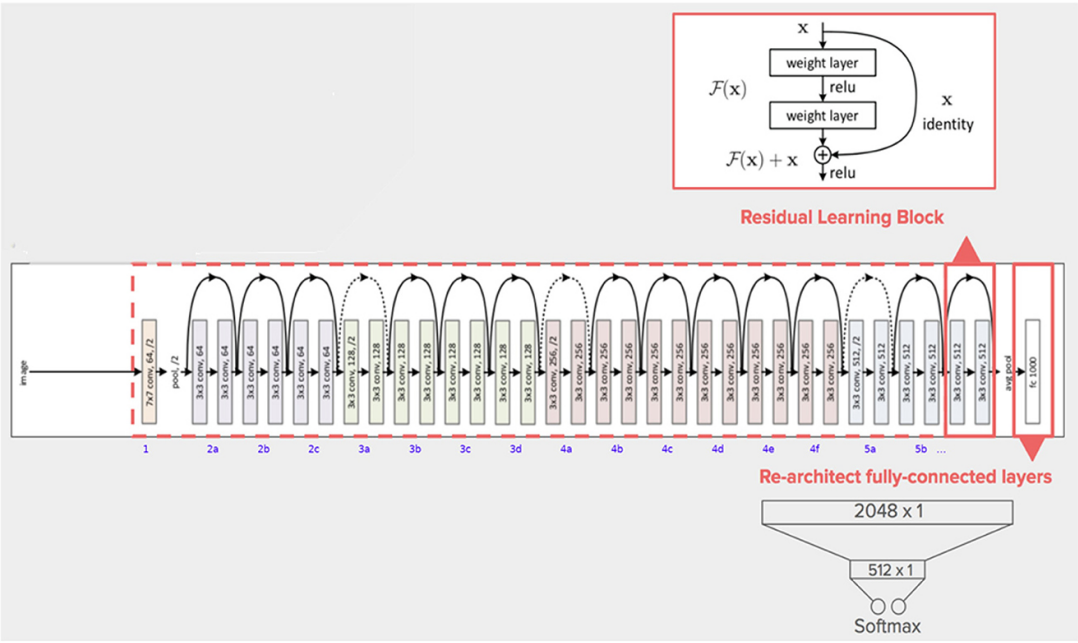


Figure 4: The ResNet50 architecture by [HZRS16] containing fifty weighted layers.

the network. In the top right of figure 4 such a residual learning block is visualised. The middle of figure 4 shows the entire architecture with at the beginning a convolutional block with a max pooling layer and in the end a global average pooling layer with one dense layer. In between the two pooling layers several identity and projection blocks are located. As mentioned the identity block skips convolutional layers by adding the input of the block directly to the activation function at the end of the block. Within a projection block the filter size is doubled and the first convolutional layer uses a 2×2 stride. This results in a different dimension compared to the input, meaning a skip connection is not possible. To resolve this an extra 1×1 convolution with the 2×2 stride is applied to the input. Ultimately, the network contains 12 identity blocks and four projection blocks, resulting in a total of 50 weighted layers.

4. Lastly, a simpler fully convolutional network (**FCN**) is implemented, as presented in the "Learning Rigid Image Registration: Utilizing Convolutional Neural Networks for Medical Image Registration" paper [SGS18]. The network contains an input for both the reference image and the target image. The two images are concatenated over the channel axis before entering a series of strided convolutions. Every convolutional layer consists of seven 5×5 kernels with ReLu activation and 2×2 strides. The amount of layers necessary depends on the size of the input images. This series of layers continues until the output of the final layer has the correct dimension for the three transformation parameters. As opposed to the other layers, the final layer contains three 3×3 kernels with 2×2 strides and linear activation to allow for negative transformation values.

   What stands out compared to the previous three deep networks, is the absence of pooling layers in the fully convolutional network. The authors decided to leave the pooling layers out due to their *local shift invariance* property. This means that small transformations in the input feature map do not affect the output feature map. Since the goal of the network is to align two images as accurately as possible, applying a pooling layer would most likely result in higher errors.

2.2.2 Transformer

The Spatial Transformer module has been presented to create networks that are spatially invariant to the input data [JSZ+15]. This means the module allows networks to make the distinction between differently positioned and different texture or shape. For every input sample separately the necessary transformation is determined during training for the greater task at hand, making it a very dynamic mechanism.

The Transformer module consists of three main components, a localisation network followed by a grid generator and lastly a differentiable sampler that produces the output feature map. An overview of the Transformer module is shown in figure 5. The first component is the localisation network, which outputs transformation parameters θ after the input feature map is subjected to a series of hidden layers. The architecture of the localisation network may vary, it can be a fully connected or convolutional network. It should however end with a regression layer to output θ with the possibility of negative values.

With the generated transformation parameters θ a sampling grid can be generated. First a standard grid G is created using normalised coordinates. Meaning in case of the x-coordinates the far most left pixels are assigned -1 and the far most right pixels assigned 1 with gradually increasing values in between. The same is done for the y-coordinates with the upper pixels

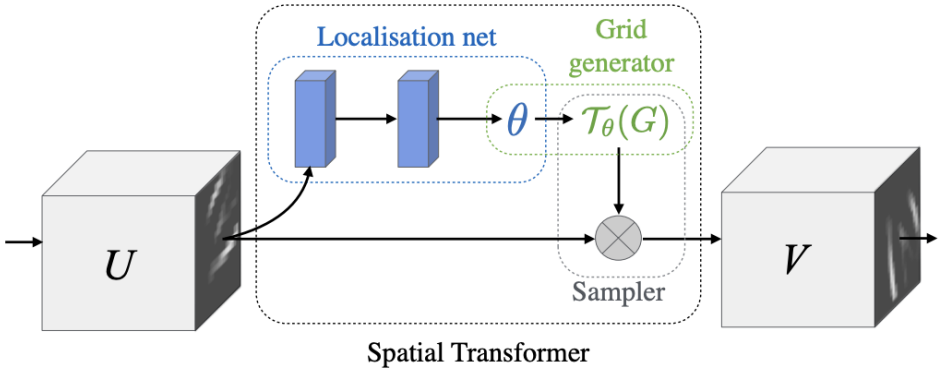


Figure 5: The Spatial Transform Network architecture containing a localisation network, grid generator and differentiable sampler to produce the output feature map. [JSZ+15]

assigned -1 and gradually increasing till 1 for the bottom pixels. Next the created grid is multiplicated with the transformation matrix A_θ, resulting in the transformed sampling grid $T_\theta(G)$.

Lastly a sampler will combine transformed sampling grid $T_\theta(G)$ with the input feature map to create the output feature map. Each coordinate in $T_\theta(G)$ represents a location on the input feature map. By using a sampling kernel at a location in the input map the value for it's corresponding location in the output map is determined. The sampling kernel can contain any form of interpolation, as long as the gradients can be determined with respect to the grid coordinates. This sampling technique results in the ability of back-propagation of the loss gradients through the sampling grid, transformation parameters to the localisation network.

2.3 Hardware

For the computing tasks a CPU/GPU desktop was used. This desktop runs a Ubuntu operating system on a 'Ryzen 7' CPU with eight processor cores and 64GB RAM. The desktop has two additional RTX 2080 GPU's with 8GB memory each. Data preparation, as further explained in section 3.1, is executed on the CPU, due to the memory needed for the training data. Both training the networks and running the experiments was performed using the GPU's to reduce runtime. For each run a GPU is used. However, training the networks could still be altered for distributed computing.

3 Implementation

The four networks as listed in section 2.2.1 will be trained and evaluated on their alignment of serial section images. An important first step is to preprocess the data. The neural networks can then be trained on the preprocessed data. After training the models are evaluated on their ability to retrieve simulated transformations. Lastly the models are applied to try to improve the alignment of the serial section images in the test set.

3.1 Data preparation

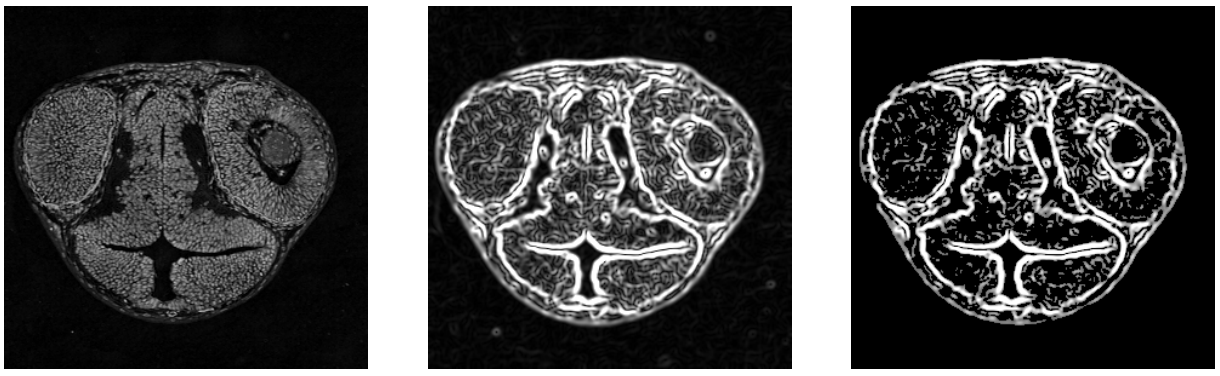
As mentioned in section 2.1, the datasets received at the beginning of this research contain preprocessed serial section images with a dark background. These images are read into a separate array for each dataset. From here the different approaches of training data separate.

3.1.1 Data variations

The neural networks are trained using three different approaches, visualized in figure 6.

1. The neural networks will be trained on the received serial section images. As shown in figure 6a, this is the entire image containing all information within the object.

2. The networks will be trained solely on the edges of the images. The entire serial section images used in the previous approach are complex and very detailed. Therefore, the neural networks could improve their training by simplifying the images. To retrieve the edge data, first a Gaussian filter with default kernel size of 9×9 is applied. This smooths the image and will result in thicker edges instead of a 1-pixel wide edge. After applying the filter, the Sobel edge detection algorithm is applied. The Sobel algorithm combines the gradients in both the x- and y-direction to obtain only the edges of the object. The resulting image is shown in figure 6b.



(a) (b) (c)

Figure 6: The three different variations of the serial section images the neural networks will be trained on. (a) The standard received serial section image, referred to as the entire image. (b) The serial section image after applying Sobel edge detection on figure 6a. (c) The serial section image after removing all values lower than the acquired Otsu threshold on figure 6b.

3. The networks will be trained on the most important outlines of the serial section images. The images resulting from the edge detection contain differences in intensity between the edges, as some are softer edges than others. Assuming the edges with a lower intensity contain less important information on the object's structure, the networks might improve by further simplifying the images. To remove the less important edges thresholding is applied. Otsu's method returns a threshold value that distinguishes the important edges from the less informative ones. All intensity values below the threshold value are removed from the image. This results in an image, as shown in figure 6c, containing only the most informative outlines of the object.

3.1.2 Data extension

Since the goal is to learn a neural network to align two section images, a second so called target image is necessary for the network to train. So for all images in the dataset a new target image has to be generated. These targets can be created through two distinct approaches.

1. The first option is to simply generate random rigid transformations and apply these to the images. This results in combinations of the reference image with a translated and/or rotated target image. With this method both the reference as the target image contain the same object and could therefore be perfectly realigned.

2. Target images will be retrieved from a window around the image in it's original dataset series. The ultimate goal is not only to align two of the same section images, but to align two consecutive serial section images. These images will thus not contain the exact same object. Therefore, it could be beneficial for the networks to train on such reference-target image combinations. With an image located at index i and a window size of x, the images within range $[i - x : i + x]$ will all be selected as target images. Because of this method, the datasets had to be stored in separate arrays. As the received datasets were already attempted to align, there are very few differences between the reference and target images for the network to train on. Therefore, random rigid transformations are also applied to these target images.

Both approaches to generate the target images apply random rigid transformations. A uniform distribution of the random transformations can be used and will cause an equal amount of smaller and larger transformations. However, assuming the series that eventually will have to be aligned will only need small transformations, a normal distribution might be more suitable. This will result in mostly smaller transformations, as expected is needed for prediction, and some relatively larger transformations. These larger transformations are needed to challenge the network not to overfit on the smaller transformations.

Depending on the approach used for generating the target images, there are either one or $2x + 1$ target images. This means a relatively small dataset for the network to train on. Training on this will result in overfitting on the generated reference-target image combinations and therefore poor results when predicting on unseen data. Even aligning the same reference image to a new target image will cause trouble. To prevent overfitting on the generated image combinations, multiple random rigid transformations are applied to generate more target images for the same reference image. However, this only partly solves the problem. With these generated datasets, the network will learn to align from always the same starting position, most likely a centered object. To overcome this, also multiple random rigid transformations are

applied to the reference image while the target image stays the same. Eventually this results in a dataset containing multiple transformed target images combined with the same reference image and multiple transformed reference images combined with the same target image.

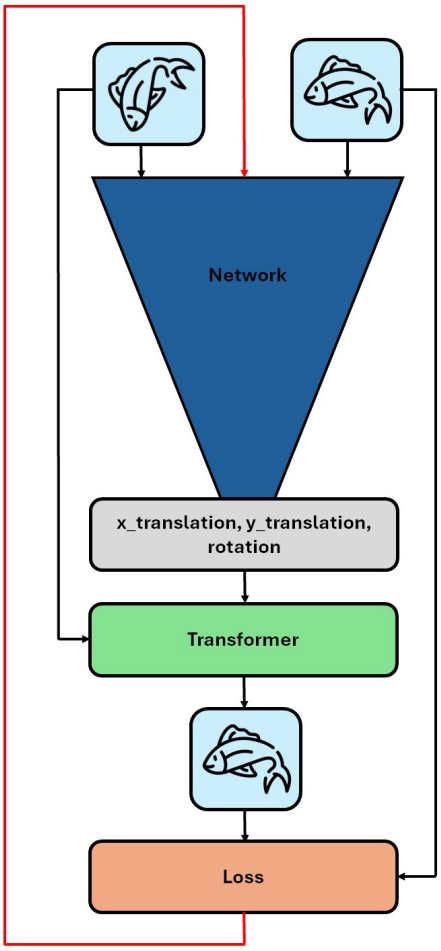


Figure 7: Flow-chart of training the convolutional neural networks on the prepared serial section image datasets.

3.2 Network training

Figure 7 shows a flow-chart of a training iteration of the convolutional neural networks on the prepared serial section image datasets. Before the networks listed in section 2.2.1 can actually start training, the architectures of the three deep CNN's have to be slightly altered. For image classification tasks these networks use one input image and 1000 output nodes with softmax activation. However, as mentioned before and shown in figure 7, the network uses the reference-target image combination as input. So the networks will have two input images instead of one. Furthermore, three output nodes are implemented with linear activation to allow both negative and positive predictions. The three output nodes, as shown in grey in figure 7 and equation 1, will contain the predicted translation over the x-axis and y-axis and

the rotation. Lastly, the amount of filters for each convolutional layer has been reduced by a factor 4, due to memory limitations.

$$output = [rotation, x\_translation, y\_translation] \tag{1}$$

The altered networks having the transformation values as output leads to an extra challenge. Since the true transformation values are unknown, the predicted values can not be compared to determine the loss value. Because the true prediction values do not exist, a manual training step has been implemented instead of using the build-in model.fit() method. A gradient tape is used to record the training iterations as shown in figure 7 and automatically differentiates to retrieve the gradients that will be back propagated to the network.

The networks can be trained with an arbitrary batch size. A larger batch size will result in a shorter training time, while a smaller batch size will provide more robust learning. To simplify the explanation of a training iteration, a batch size of one is assumed. The first step in a training iteration is to enter a reference-target image combination into the network. Based on the input images, the network predicts the three transformation parameters. These three values are converted to transformation matrix $theta$, as shown in equation 2. Using $theta$, the reference image can be transformed.

The Spatial Transformer module is used to apply transformation matrix $theta$ on the reference image. Usage of the Spatial Transformer module allows the gradient tape to follow the transformation and eventually determine the gradients. Resulting from the Transformer module is the transformed reference image, hopefully better aligned to the target image. The loss is then determined by measuring the similarity between the transformed reference image and the target image. The similarity is calculated by either the mean squared error (mse) or the normalised cross correlation (ncc) between the two images. Based on the loss the gradient tape calculates the gradients for all trainable variables in the network. As indicated by the red line in figure 7, the Adam optimizer applies the gradients with a certain learning rate to the network for the next iteration. The steps as shown in figure 7 are executed for the entire training set. For the network to learn how to align two input images, it is trained on the training set for multiple iterations.

$$theta = \begin{bmatrix} cos(output[0]) & -sin(output[0]) & output[1] \\ sin(output[0]) & cos(output[0]) & output[2] \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

To prevent the network to get stuck in local optima, a simulated annealing technique is applied on the learning rate of the Adam optimizer [BT93]. Three decreasing learning rates are applied equally divided over the amount of iterations. This will result in smaller gradients in the later stages of training compared to the earlier epochs and therefore a better convergence of the network.

3.3 Network evaluation

Once the networks are trained, the models are tested on their accuracy in retrieving applied transformations. Uniformly distributed random rigid transformations are applied to a new unseen dataset by the model. For each image in the dataset one transformed target image is generated. The model then tries to predict the applied transformation for each image in an iterative fashion. As long as the similarity score between the predicted transformed image and the target image improves compared to the previous prediction, the model will predict again in

an attempt to come closer to the target image. As soon as the similarity score is worse than the previous, the model continues with the next image in the dataset. These iterations are executed for every image until the entire dataset is attempted to be aligned with it's generated target images. The similarity scores are calculated by both the mse and ncc. Whether the similarity score improves, is based on the ncc-scores. Once the entire test set is attempted to align, an average similarity score is calculated over the entire test set, to indicate the quality of the network. In case of the models trained on the edges of the serial section images, Sobel edge detection is first applied to the dataset before generating the transformed targets. This means, the network will determine a prediction based on the edge-based reference and target image. The predicted transformation is then executed on the entire, not edge-based, serial section image to determine the similarity scores. This way, the quality of the edge-based models can be compared to the standard models. The evaluation algorithm on the edge-based models is shown in Algorithm 1.

Algorithm 1 Evaluating trained models, retrieving random transformations

Input: unseen dataset $eval\_data$ with image $height$ and $width$, the Sobel edge detected models $sobel\_models$ and maximum number of iterations max_iter

$sobel\_data \leftarrow sobel\_edge(eval\_data, height, width)$
$transformations \leftarrow gen\_transformations()$
$targets \leftarrow transform\_data(sobel\_data, transformations)$
for $model$ in $sobel\_models$ **do**
    $registrated\_imgs \leftarrow eval\_data.copy()$
    for x in $range(len(eval\_data))$ **do**
        $counter \leftarrow 0$
        $repeats \leftarrow True$
        $prev\_score \leftarrow sim\_score(eval\_data, targets)$
        while $(repeats == True)$ and $(counter < max\_iter)$ **do**
            $current\_img \leftarrow sobel\_edge(registrated\_imgs[x], height, width)$
            $output \leftarrow model.predict(current\_img, targets[x])$
            $theta \leftarrow gen\_matrix(output)$ (Eq.2)
            $transformed \leftarrow warp(registrated\_imgs[x], theta)$
            if $sim\_score(transformed, targets[x]) > prev\_score$ **then**
                $repeats \leftarrow False$
            else
                $counter \leftarrow counter + 1$
                $registrated\_imgs[x] \leftarrow transformed$
                $prev\_score \leftarrow sim\_score(transformed, targets[x])$
            end if
        end while
    end for
    $final\_score \leftarrow sim\_score(registrated\_imgs, targets)$
end for

3.4 Three dimensional alignment

Assuming the trained models are able to improve the similarity scores during the evaluation, the models will also be tested to align a test set of serial section images. To align a dataset of serial section images the algorithm will start at the middle of the dataset. This image is most likely to contain a large and centered object, so aligning towards this object is most likely to keep the following objects within the image frame. Having selected the starting point, the algorithm starts aligning outwards in both directions one image at a time. The previously registered image is then used as target for the next image. Once the entire series is aligned, a score representing the quality of the alignment is calculated. In case the quality of the alignment has improved compared to the quality beforehand, the alignment process is repeated on the just aligned test set. This keeps iterating until the quality does not improve anymore, assuming the best alignment is achieved. The algorithm will also stop iterating when a fixed maximum number of iterations is reached, assuming the model will not significantly improve any further.

As well as in section 3.3, the algorithm is slightly altered for the edge-based models. The edge-based images of the test set are entered into the network, while the predicted transformations are applied to the entire serial section images. This way the alignment quality of the edge-based models can be compared to the standard models. Algorithm 2 shows an overview of the alignment process using the edge-based models.

3.4.1 Alignment quality score

As mentioned above, a quality score of the 3D alignment of serial section images has to be calculated. This quality score is similarly calculated as the similarity score used in section 3.3. The similarity is determined between an image and it's target. In case of determining the alignment quality, the target is the next image in the series. So a similarity score, both mse and ncc, is calculated for every adjoining pair of serial section images. An average of these scores results in the overall quality score of the alignment. The lower the mse score the more similar two images are, with two equal images having a mse score of zero. The ncc score on the other hand is ranged between -1 and 1. With -1 representing the best possible similarity and becoming worse the closer it comes to 1.

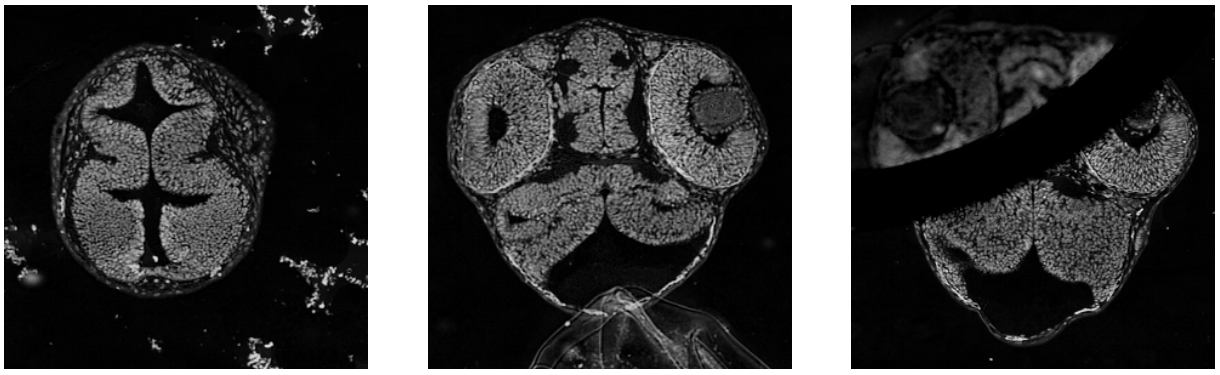


Figure 8: Three examples of images that will be considered as outliers. Left: too much noise surrounding the object. Middle: some sort of contamination of the slide, will result in a higher error. Right: an air bubble on the slide during scanning, results in missing data compared to the previous and next image in the series.

A problem to this approach however, is the fact that some images in the series might contain

flaws. Three examples are shown in figure 8. These flaws can cause the model to be unable to correctly align these images and their two adjoining ones, resulting in worse similarity scores and even alignment quality. However, these worse scores are not representative of the models alignment abilities due to flaws in the data. Therefore, these similarity scores are considered to be outliers and withheld from calculating the overall alignment quality. Determining which similarity scores are considered outliers is not executed visually based on the serial section images. This is determined based on the similarity scores. A threshold based on the distribution of the similarity scores is calculated. Every score worse than the threshold are considered to be outliers. Based on this, the outliers are assumed to contain flaws in their images.

Algorithm 2 Registration of serial image data

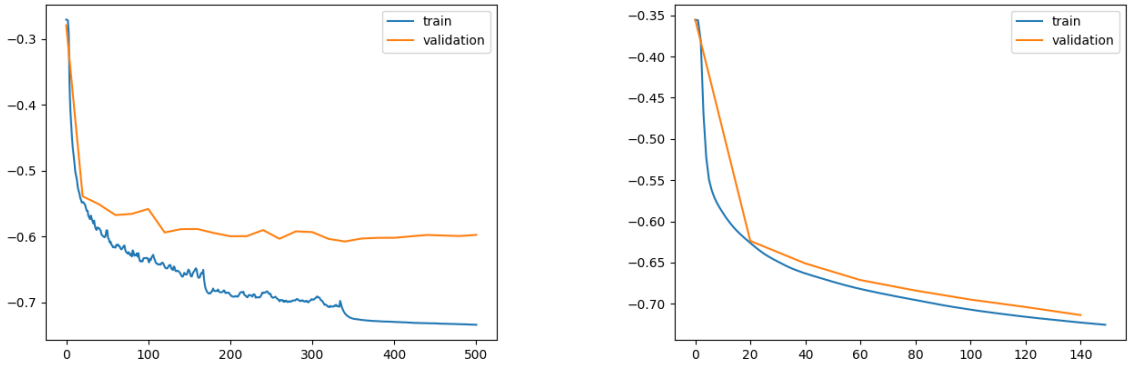
Input: unseen dataset reg_data with image $height$ and $width$, the Sobel edge detected models $sobel\_models$ and maximum number of iterations max_iter

$start\_idx \leftarrow int(len(reg\_data)/2)$
for $model$ in $sobel\_models$ **do**
    $counter \leftarrow 0$
    $repeats \leftarrow True$
    $registrated\_imgs, prev\_registrated\_imgs \leftarrow reg\_data.copy()$
    $prev\_loss, prev\_loss\_before\_outliers \leftarrow reg\_score(registrated\_imgs)$
    while $(repeats == True)$ and $(counter < max\_iter)$ **do**
        $edge\_data \leftarrow sobel\_edge(registrated\_imgs, height, width)$
        $target \leftarrow edge\_data[start\_idx]$
        for x in $range(start\_idx - 1, -1, -1)$ **do**
            $output \leftarrow model.predict(edge\_data[x], target)$
            $theta \leftarrow gen\_matrix(output)$ (Eq.2)
            $registrated\_imgs[x] \leftarrow warp(registrated\_imgs[x], theta)$
            $target \leftarrow warp(edge\_data[x], theta)$
        end for
        $target \leftarrow sobel\_edge(registrated\_imgs[start\_idx - 1], height, width)$
        for x in $range(start\_idx, len(edge\_data)$ **do**
            $output \leftarrow model.predict(edge\_data[x], target)$
            $theta \leftarrow gen\_matrix(output)$ (Eq.2)
            $registrated\_imgs[x] \leftarrow warp(registrated\_imgs[x], theta)$
            $target \leftarrow warp(edge\_data[x], theta)$
        end for
        $loss, loss\_before\_outliers \leftarrow reg\_score(registrated\_imgs)$
        $counter \leftarrow counter + 1$
        if $loss < prev\_loss$ or $loss\_before\_outliers < prev\_loss\_before\_outliers$ **then**
            $prev\_registrated\_imgs \leftarrow registrated\_imgs.copy()$
            $prev\_loss, prev\_loss\_before\_outliers \leftarrow loss, loss\_before\_outliers$
        else
            $repeats \leftarrow False$
        end if
    end while
    $save\_registrated\_imgs(prev\_registrated\_imgs, model)$
end for

4 Experiments & results

Several variations of models have been trained for 500 epochs in batches of size 50. The descending learning rates used were $1e-5, 5e-6$ and $1e-6$ respectively. Datasets $1\&2$ were used for training the models. The VGG16, VGG19 and FCN network have been trained as standard model and edge-based models with different settings. Due to longer training time, the ResNet50 network has only be trained for it's standard and edge-based model, no extra settings have been experimented.

In general the trained models showed to converge over the 500 epochs, such as the edge-based ResNet50 model shown in figure 9a. However, there is an exception, the learning curve of the standard FCN model as shown in figure 9b. After 150 epochs only NaN results were reported for both the training as the validation set. This can be caused by extreme network output values from which the network is unable to recover.



(a) (b)

Figure 9: The learning curves of two trained models. (a) The learning curve of the edge-based ResNet50 model. It shows convergence within the 500 epochs, but also a gap between train and validation data. (b) The learning curve of the standard FCN model. It shows no results after epoch 150, while in fact NaN results were returned. These are caused by exploding gradients resulting in extreme network output values.

Dataset 3 will be used for experimentation. Tables 1 and 2 in appendix A show respectively the results of the retrieval of random rigid transformations and the 3D alignment for all trained models. In this chapter these two tables will be analysed and explained in parts.

The same random rigid transformations were applied to the test set for all models to retrieve. These transformations resulted in an average mse-score of 1967.284 and -0.191 ncc-score. As for the 3D alignment, the original test set had a quality mse-score of 836.119 without any outliers present. The quality ncc-score beforehand was -0.697 and -0.712 with five outliers removed.

4.1 Standard vs edge-based models

To get insight in the performance of CNN models on registration of serial section images the standard models are compared against both edge-based models. For both the VGG16 and FCN network the standard, edge-based and edge-based+threshold model have been trained. For the

VGG19 and ResNet50 network only the standard and edge-based model have been trained. All these models have been trained on data that generated the target images using the window approach. Furthermore, they have been trained using the ncc loss function.

4.1.1 Evaluation

Table 2 shows the results of retrieving the applied random rigid transformations by the four networks trained on the entire section, edge-based and edge-based+threshold images. Compared to the scores mentioned above after the random transformations, all trained models show better scores after registration. With as worst result still an mse-score of 526.423 and -0.741 ncc-score. While the best result improves up to a mse-score of 266.560 and -0.862 ncc-score. In both cases these scores are before removing outliers.

data	model	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
standard	VGG16	407.498	398.023	1	-0.789	**-0.805**	4
edge-based	VGG16	**377.094**	**369.412**	1	**-0.799**	-0.804	1
+ threshold	VGG16	462.202	462.202	0	-0.766	-0.804	7
standard	VGG19	**425.856**	**425.856**	0	**-0.786**	**-0.799**	3
edge-based	VGG19	526.423	526.423	0	-0.741	-0.754	3
standard	Resnet50	499.102	499.102	0	-0.752	-0.763	2
edge-based	ResNet50	**378.059**	**378.059**	0	**-0.794**	**-0.808**	3
standard	FCN	NaN	NaN	-	NaN	NaN	-
edge-based	FCN	**266.560**	**159.961**	12	**-0.862**	**-0.927**	16
+ threshold	FCN	319.128	281.217	4	-0.817	-0.867	10

Table 2: Similarity scores of the standard, edge-based and edge-based+threshold models after registration of random rigid transformations. This table is a subsection of table 10. Both mse and ncc scores after registration of simulated transformations are shown, both with and without outliers. Before registration the dataset had a mse-score of 1967.284 and -0.191 ncc-score. Bold scores are the best results for that network.

Four aspects stand out from the results in table 2. First are the Not a Number (NaN) results for the standard FCN model. When inspecting the training data as shown in learning curve 9b, NaN results also occurred from the epoch where the learning rates are lowered for the first time, and onward. As explained above, these NaN results can occur due to exploding gradients causing extreme network output values.

The second notable aspect is the fact that only the VGG19 network shows better evaluation scores for the standard model. While both the VGG16 and ResNet50 network have a better performing edge-based model. This indicates that a smaller VGG network architecture works better on the images with less information compared to the deeper VGG architectures. While the ResNet50 network has even more layers than the VGG19 network, the use of residual blocks seems to help the network to deal with the less informative images.

Thirdly, removing the edges with an intensity below the Otsu threshold from the edge-based images does not seem to improve the model for both the VGG16 and FCN network. In both cases worse scores are achieved compared to the edge-based models without the threshold. This indicates that by applying the threshold on the edge data too much information is lost for a accurate registration.

Lastly the amount of outliers for the FCN models stands out compared to the other models. This suggests that for the FCN models most similarity scores are close to a very good mse or ncc score with the outliers being significantly worse, but no results in between. While the results of other models are most likely more spread out. Inspecting the histograms of the edge-based FCN and edge-based VGG16 models, with 16&1 outlier in ncc score respectively, in figure 10 these suggestions are confirmed. The FCN model has a similar worst similarity score as the VGG16 model, however less scores in the middle.

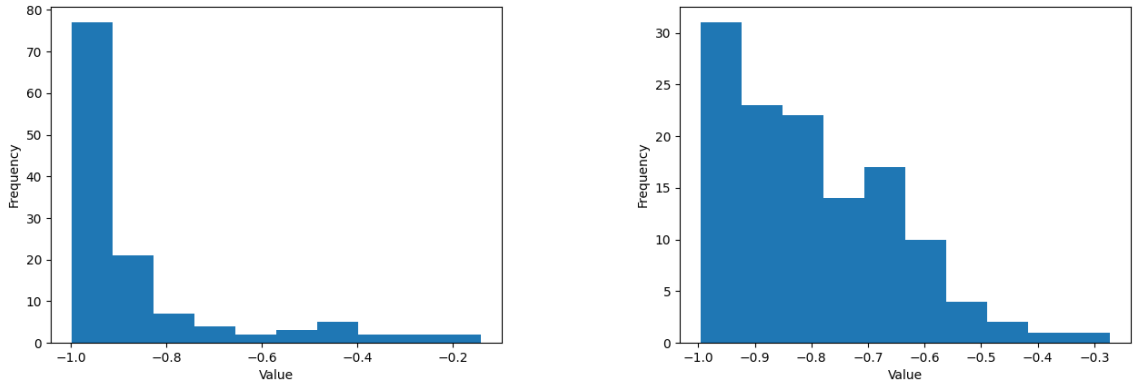


Figure 10: Histograms of the ncc similarity scores resulting from the registration of random rigid transformations by two edge-based models. Left: edge-based FCN model, similarity scores resulted in 16 outliers being removed, due to high amount of similarity scores close to -1.0. Right: edge-based VGG16 model, similarity scores resulted in only one outlier being removed, due to a more spread out distribution of the similarity scores.

4.1.2 3D alignment

All trained models were able to retrieve the simulated transformations to a certain extend. To assess the ability of CNN's to align serial section images, the models are evaluated on dataset 3. As mentioned in section 2.1, this dataset has already been aligned by a mechanical chamfer matching approach. This experiment will show if the CNN models are able to maintain the alignment or might even be able to slightly improve the quality. Table 3 shows the results of the four networks trained on the entire section, edge-based and edge-based+threshold images.

What mainly stands out from the results is the different directions the mse and ncc quality scores progressed compared to the quality beforehand. When assessing the alignment quality of the models based on the ncc scores all models were able to more or less maintain the alignment executed with the Chamfer method, showing similar quality scores. After removing outliers, the standard VGG16 and VGG19 models even recorded a slight improvement. On the other hand, when assessing the alignment quality based on the mse scores the standard and both edge-based models were able to further improve the original alignment. This difference between the mse and ncc quality might be caused by the interpolation happening during the transformation. The interpolation can cause the mse scores to improve but has no impact on the correlation between the intensity values. Therefore this also will not impact the ncc quality.

When comparing the standard model with the edge-based models, the VGG19 network shows to perform better trained on the entire section images. This corresponds with the results reported in the previous section. As for the VGG16 and ResNet50 networks, the standard

data	model	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
standard	VGG16	747.465	723.050	2	**-0.690**	**-0.719**	10
edge-based	VGG16	**744.892**	**722.233**	2	-0.687	-0.712	8
+ threshold	VGG16	762.703	728.895	3	-0.684	-0.712	7
standard	VGG19	**733.365**	**709.262**	2	**-0.690**	**-0.718**	10
edge-based	VGG19	765.193	727.296	5	-0.676	-0.687	3
standard	Resnet50	811.957	**750.427**	7	-0.663	-0.683	7
edge-based	ResNet50	**790.255**	753.050	4	**-0.672**	**-0.691**	6
standard	FCN	NaN	NaN	-	NaN	NaN	-
edge-based	FCN	**796.780**	768.088	2	**-0.669**	-0.689	6
+ threshold	FCN	799.768	**725.562**	7	-0.664	**-0.694**	9

Table 3: 3D alignment quality scores of the standard, edge-based and edge-based+threshold models. This table is a subsection of table 11. Both mse and ncc quality scores after alignment of the serial section images in the test set are shown, both with and without outliers. Before alignment the test set had a mse quality score of 836.119 without any outliers present. The ncc quality score beforehand was -0.697 and -0.712 with five outliers removed. Bold scores are the best results for that network.

and edge-based model show very similar quality scores, neither model clearly shows a better performance. As all models were able to maintain but not improve the original alignment, no difference in performance between the standard and edge-based models is visible in these results.

There is however a difference in performance quality between the networks. The difference in quality is not visible in table 3, but comes to light when visually inspecting the alignment. Figure 11 shows a cross-section along the y-axis of the alignments created by the edge-based VGG16, VGG19 and ResNet50 models from left to right. Even though all three models showed relatively similar quality scores and amount of outliers, a clear difference in how the network handles the outliers is visible in figure 11. The edge-based VGG16 model on the left image showed unable to correctly align outlier images, causing the following section images also to fall out of the alignment. Both the edge-based VGG19 and ResNet50 models however, will see the same images as outliers based on their similarity scores, but are still able to properly align them.

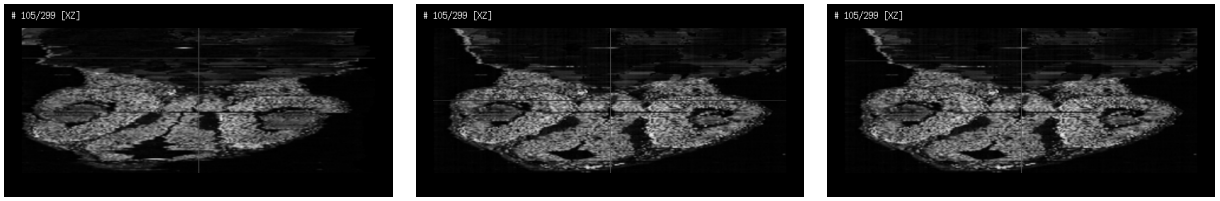


Figure 11: Cross-section images along the y-axis of the alignments created by the edge-based VGG16, VGG19 and ResNet50 models from left to right. The edge-based VGG16 model on the left shows shifts in the alignment caused by outlier images. While on the other hand both the edge-based VGG19 and ResNet50 models were able to properly align these outlier images.

model	loss_function	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	ncc	377.094	369.412	1	-0.799	-0.804	1
VGG16	mse	**298.616**	**298.616**	0	**-0.839**	**-0.855**	4
VGG19	ncc	526.423	526.423	0	-0.741	-0.754	3
VGG19	mse	**366.074**	**366.074**	0	**-0.809**	**-0.822**	3
FCN	ncc	**266.560**	**159.961**	12	**-0.862**	**-0.927**	16
FCN	mse	469.017	469.017	0	-0.764	-0.784	4

Table 4: Average similarity scores after registration of random rigid transformed section images by the edge-based models trained with mse and ncc loss function. Furthermore, these models were trained on data that generated target images using the window approach. This table is a subsection of table 10. Mse and ncc similarity scores after registration of random rigid transformations are shown, both with and without outliers. Before registration the dataset had an overall mse-score of 1967.284 and -0.191 ncc-score. Bold scores are the best results for that network.

4.2 Hyperparameters

Besides comparing the standard models with the edge-based models, some additional settings on the edge-based models have been experimented with. For example varying the loss function, adding kernel regularization and two different approaches to generating the target images. Due to longer training time no additional settings have been experimented for the ResNet50 network. Additional settings will only be experimented for the VGG16, VGG19 and FCN networks.

4.2.1 ncc vs mse

Firstly, two different variants of loss function used during training are tested. In table 4 the average similarity scores after registering the random rigid transformed section images are shown for the edge-based models trained with either mse or ncc loss. Furthermore, these models have been trained on data that generated the target images using the window approach. For both the VGG16 and VGG19 model training with the mse loss function results in better similarity scores than training with the ncc loss function. In case of the FCN model however, the opposite is the case. What does stand out for the FCN model, is the fact that the usage of mse loss during training results in significantly less outliers.

The quality of aligning the test set by edge-based models trained with either mse or ncc loss, as shown in table 5, does not extend the trend shown in table 4. The VGG16 network shows better results when trained using the ncc loss function, while the FCN networks seem to perform better when trained with the mse loss function. As for the VGG19 network, neither loss function is clearly outperforming the other. Instead, training with the mse loss function results in better mse quality scores and training with the ncc loss function results in better ncc quality scores.

Considering the improvement of mse quality scores is caused by the interpolation during the transformation, the use of neither loss function during training results in an improved alignment. On the other hand, also neither loss function causes the alignment to deteriorate. Combining the results from tables 4 & 5, no conclusion can be drawn on which loss function is best used to train the models.

model	loss_function	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	ncc	**744.892**	**722.233**	2	**-0.687**	**-0.712**	8
VGG16	mse	763.388	738.288	2	-0.677	-0.706	9
VGG19	ncc	765.193	727.296	5	**-0.676**	**-0.687**	3
VGG19	mse	**687.868**	**663.266**	2	-0.666	-0.683	5
FCN	ncc	796.780	768.088	2	-0.669	-0.689	6
FCN	mse	**773.634**	**746.498**	3	**-0.686**	**-0.704**	6

Table 5: 3D alignment quality scores of the edge-based models trained with mse and ncc loss function. Both these models were trained with target images generated using the window approach. This table is a subsection of table 11. Mse and ncc quality scores after alignment of the serial section images in the test set are shown, both with and without outliers. Before alignment by the models the test set had a quality mse-score of 836.119 without any outliers present. The quality ncc-score beforehand was -0.697 and -0.712 with five outliers removed. Bold scores are the best results for that network.

4.2.2 Regularization

Secondly was experimented with adding regularization to each convolutional layer in the networks. Regularization is used to generalize the data and therefore reduce overfitting of the model. This should help the model to perform better on unseen data. In this experiment L1, L2 and L1L2 combined regularization were compared to the models without regularization. The models evaluated in the previous experiments were all trained without regularization.

model	regularization	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	L1	**349.916**	349.916	0	**-0.812**	-0.816	1
VGG16	L2	353.513	**333.636**	3	-0.811	**-0.820**	2
VGG16	L1L2	382.302	382.302	0	-0.805	-0.812	2
VGG16	None	377.094	369.412	1	-0.799	-0.804	1
VGG19	L1	**392.482**	**384.833**	1	-0.801	-0.801	0
VGG19	L2	400.690	400.690	0	**-0.806**	**-0.814**	3
VGG19	L1L2	398.804	398.804	0	-0.801	-0.806	1
VGG19	None	526.423	526.423	0	-0.741	-0.754	3
FCN	L1	299.893	257.198	5	-0.848	-0.902	14
FCN	L2	641.235	641.235	0	-0.676	-0.676	0
FCN	L1L2	604.204	604.204	0	-0.682	-0.682	0
FCN	None	**266.560**	**159.961**	12	**-0.862**	**-0.927**	16

Table 6: Average similarity scores after registration of random rigid transformed section images by the edge-based models trained with L1, L2, L1L2 or no regularization added to the convolutional layers. These models were all trained on target images generated using the window approach and using the ncc loss function. This table is a subsection of table 10. Mse and ncc similarity scores after registration of random rigid transformations are shown, both with and without outliers. Before registration the dataset had an overall mse-score of 1967.284 and -0.191 ncc-score. Bold scores are the best results for that network.

In table 6 the average similarity scores after registering the random rigid transformed section images are shown for the edge-based models trained with four variations of regularization.

Furthermore, these models have been trained using the ncc loss function and with target images generated by the window approach. The results show that adding regularization does improve the results for the VGG16 and VGG19 models. For both networks the L1 regularized network shows the best similarity scores, but closely followed by the L2 regularization. As for the FCN network, without any regularization shows the best results closely followed by L1 regularization. Applying L2 and L1L2-combined regularization however does not turn out as successful as intended, resulting in the worst scores in this table.

model	regularization	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	L1	753.161	727.397	2	-0.686	**-0.717**	10
VGG16	L2	**695.575**	**672.545**	2	-0.671	-0.684	4
VGG16	L1L2	750.501	727.731	2	-0.680	-0.709	9
VGG16	None	744.892	722.233	2	**-0.687**	-0.712	8
VGG19	L1	756.840	734.404	2	**-0.685**	**-0.718**	12
VGG19	L2	700.378	679.178	2	-0.676	-0.693	5
VGG19	L1L2	**652.853**	**625.942**	2	-0.669	-0.690	7
VGG19	None	765.193	727.296	5	-0.676	-0.687	3
FCN	L1	827.918	806.719	2	-0.653	-0.687	12
FCN	L2	**778.123**	748.265	2	-0.672	-0.700	9
FCN	L1L2	779.634	**720.755**	5	**-0.687**	**-0.712**	7
FCN	None	796.780	768.088	2	-0.669	-0.689	6

Table 7: 3D alignment quality scores of the edge-based models trained with L1, L2, L1L2 or no regularization added to the convolutional layers. All models were trained using the ncc loss function and with target images generated by the window approach. This table is a subsection of table 11. Mse and ncc scores after alignment of the serial section images in the test set are shown, both with and without outliers. Before alignment the test set had a quality mse-score of 836.119 without any outliers present. The quality ncc-score beforehand was -0.697 and -0.712 with five outliers removed. Bold scores are the best results for that network.

As for these models' results on 3D alignment of the test set, see table 7, they do not seem to show whether applying any form of regularization improves the model. As was the case for the previous two experiments, the mse quality scores for all models improve on the quality beforehand while the ncc scores become slightly worse. Foccussing on the ncc scores and it's amount of outliers, no patterns of the models with regularization performing better than the models without are visible. As for the mse scores, two models stand out. VGG16 with L2 regularization and VGG19 with L1L2-combined regularization show better alignment quality scores than the models without added regularization. However, as these two models did not stand out for the ncc scores and in table 6, these mse results might not be that meaningful.

4.2.3 Window vs transformed

In section 3.1.2 two approaches to generating the target images were explained. The first using random rigid transformations of the reference image, resulting in a transformed version of the same object. The second approach also applies random rigid transformations, but on all images within a window around the reference image. Therefore multiple targets are generated, containing a transformed slightly different object. Both approaches have been tested with

edge-based models trained using the mse loss function and no regularization added to the network.

model	data_shape	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	window	**298.616**	**298.616**	0	**-0.839**	**-0.855**	4
VGG16	transformed	565.056	555.352	1	-0.746	-0.759	3
VGG19	window	**366.074**	**366.074**	0	**-0.809**	**-0.822**	3
VGG19	transformed	582.355	568.470	1	-0.692	-0.709	3
FCN	window	**469.017**	**469.017**	0	**-0.764**	**-0.784**	4
FCN	transformed	741.645	741.645	0	-0.617	-0.617	0

Table 8: Average similarity scores after registration of random rigid transformed section images by the edge-based models trained with either the transformed reference image or window approach to generating the target images. Both models were trained without regularization added and using the mse loss function. This table is a subsection of table 10. Mse and ncc similarity scores after registration of random rigid transformations are shown, both with and without outliers. Before registration the dataset had an overall mse-score of 1967.284 and -0.191 ncc-score. Bold scores are the best results for that network.

Table 8 shows the average similarity scores after registering the random rigid transformed section images for the edge-based models with the transformed reference image and window approach to generating the target images. The models using the window approach are better able to register the random transformed section images, showing better similarity scores for all three networks. This is against expectations as the transformed reference image approach would allow the network to exactly retrieve the applied transformations during training, as the reference and target object are the same. With the window approach on the other hand, there will always be an error because the objects are different. However, with the reference image also being within the window this technique might benefit from both approaches, resulting in more generalization and therefore better results.

model	data_shape	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
VGG16	window	**763.388**	**738.288**	2	**-0.677**	**-0.706**	9
VGG16	transformed	806.144	766.249	4	-0.665	-0.684	5
VGG19	window	**687.868**	**663.266**	2	-0.666	-0.683	5
VGG19	transformed	718.201	689.300	2	**-0.711**	**-0.729**	5
FCN	window	773.634	746.498	3	-0.686	-0.704	6
FCN	transformed	**657.590**	**636.344**	2	**-0.739**	**-0.758**	8

Table 9: 3D alignment quality scores of the edge-based models trained with either the transformed reference image or window approach to generating the target images. Both models were trained without regularization added and using the mse loss function. This table is a subsection of table 11. Mse and ncc scores after alignment of the serial section images in the test set are shown, both with and without outliers. Before alignment the test set had a quality mse-score of 836.119 without any outliers present. The quality ncc-score beforehand was -0.697 and -0.712 with five outliers removed. Bold scores are the best results for that network.

As for the quality of the alignment executed by these models, shown in table 9, there is no approach clearly outperforming the other over all three networks. The window approach shows better quality of alignment for the VGG16 network and the VGG19 network, if solely focusing on the mse scores. The transformed reference image approach shows better quality of alignment for the VGG19 network when focusing on the ncc scores and for the FCN network. Furthermore, it stands out that the transformed reference image approach for the VGG19 and FCN model are the first models able to improve quality of the original alignment based on the ncc scores. As the ncc similarity scores of these two models in table 8 were unable to exceed -0.7, this improved alignment quality is unexpected. On top of that, the mse quality scores from these two models have been achieved by other models seen in the previous experiments (ex. VGG19 with L1L2 regularization in table 7), however their ncc quality scores did not improve. To better understand this difference in results, an alternative quality measure might be needed.

5 Conclusion & Discussion

In this final section the findings from this research will be summarized and used to answer the research questions stated in chapter 1. Within the discussion section the conclusions drawn will be discussed with regard to challenges and limitations encountered throughout the project. Lastly, possible future work as continuation of this research will be suggested.

5.1 Conclusion

During this research the performance of convolutional neural networks used for rigid image registration on serial section images has been evaluated. Four CNN's known for their success in image recognition tasks have been trained on either the entire serial section images or solely the edges. Several variation of models have been trained and experimented in order to evaluate their performance. Based on these results as shown in chapter 4, the stated research questions can be answered.

1. **How well are convolutional neural networks able to register serial section images?**
   The CNN models have shown to be able to register a random rigid transformed section image to it's original reference image. Although, the models are able to align the two images only up to a certain extend and not exactly realign the two. Considering the models did not deteriorate the alignment already executed using the Chamfer method, it seems the networks were able to learn when nearly no transformation is necessary. This indicates convolutional neural networks show great potential but leave room for improvement.

2. **Does applying edge detection on data support neural networks during their training?**
   In order to state whether applying edge detection on data significantly supports neural networks during their training statistical analysis is necessary. This analysis can not be executed over our results, as three networks showed results for both approaches on only one test set. To be able to perform the statistical analysis, experiments on more datasets will have to be performed. However, based on the results shown in table 2, the edge-based models do look promising.

3. **Can we establish a measure for image registration quality that corresponds with the CNN operation?**
   The results show that the proposed measure using an average of pairwise similarity scores contains some flaws. The measure is very useful to evaluate the registration of two images, such as during the experiment of retrieving the random rigid transformations. However, the measure was misleading in case of the 3D alignment of the serial section images. As the measure solely focuses on the pairwise similarity, the global alignment could be bad due to outlier images. Therefore, the measure is not recommended for the quality assessment of 3D alignment of serial section images.

5.2 Discussion

As stated in the conclusion, the performance of convolutional neural networks used for rigid registration of serial section images left room for improvement. However, during this research

multiple challenges were faced that could have prevented a better performance.

Firstly, NaN values occurred on multiple occasions during this project. Not only the standard FCN model resulted in NaN scores, but during training also some variations of the VGG16 and VGG19 reported NaN values. As mentioned in section 4, these NaN values can occur due to extreme network output values. These extreme values are caused by vanishing or exploding gradients, a notorious problem with neural networks [BSF94, GB10]. This problem can be resolved by adding Batch Normalization layers to the network. Applying Batch Normalization also works as a regularizer and allows the use of higher learning rates [IS15]. This corresponds with the experiences during this research, as the relatively small learning rates ultimately used during training are based on the many NaN occurrences with higher learning rates. As the possibly vanishing gradients would cause the network to stop learning, application of Batch Normalization might improve the training of these three networks. This could eventually result in better performance of the network on rigid image registration. Lastly, Batch Normalization results in quicker convergence as 14 times less epochs are needed to reach the same accuracy [IS15], meaning the training process would take far less time.

Secondly, the CNN models would improve by training on more datasets. Currently the models are trained on only two datasets causing overfitting and lack of performance on unseen data. Upon inspecting the learning curves of the models the overfitting becomes clear. First is the fact that there is a significant gap between the training data scores and validation scores, as is visible in figure 9a. This showcases the difficulty with unseen data. Secondly, the learning curves of the VGG networks show great fluctuation between epochs. Training on more different datasets will cause generalization and therefore a smoother learning curve and better results on unseen data.

Furthermore, the results showed the CNN models were unable to improve the quality of alignment. As the test set is already aligned with a previous approach, there will be very few images that need any extra rigid transformation. Not every image in the dataset will need as many iterations as others. By realigning the entire dataset as long the quality improves, two things can happen that limits the ultimate alignment quality. Either a perfect registration between two section images becomes worse by continuing or a registration is not further improved where needed as the overall quality did not improve. Therefore, it might be beneficial to apply an approach where the images are registered one by one instead of per series.

Lastly, the intensity values of the edge-based images shown in figures 6b & 6c are enhanced for visibility in this thesis. These values were actually hardly visible, but this was not noticed until creating the figure for this paper. During this research the pyplot module in the Matplotlib library was used for visualization of the images. However, pyplot displays the images ranged on the intensities present in the image and not between 0 and 255. Resulting in the supposition the edge detection resulted in high intensity values for the edges. The actual low intensity values might cause inaccurate similarity scores, as the similarity between zero and a low intensity might be higher than between zero and a higher intensity from the original image. This inaccurate score might push a model in the wrong direction as it has the idea it comes closer to the original image, while this is not the case. Also with regard to the interpolation during the transformation, the difference between object and background becomes less clear. Regardless of the low intensity values, the edge-based models were still able to get representative scores during evaluation and for some networks outperform the base model. Scaling the detected edges

to a higher intensity range before training might improve the edge-based models. Assuming this does result in an improvement, removing the less informative edges based on a threshold might also become more relevant.

5.3 Future work

During the preparation of the edge-based section images a Gaussian filter with default kernel size of 9×9 is applied to create thicker edges. In future work it would be interesting to experiment with the kernel size of the Gaussian filter. This way can be tested what kind of impact the thickness of the edges have on the performance of the neural networks.

Additionally, during this research the models were trained on a low and fixed resolution. An interesting next step would be to experiment with higher resolutions. Investigate whether these results with lower resolution directly translate to images with higher resolutions. To accommodate this, a tool to either up- or downscale images to the required resolution might be beneficial. Continuing this, it would be interesting to investigate the accuracy of predicting on a down-scaled image while applying the transformation on the original scale.

Furthermore, this research focused on rigid image registration of serial section images. However, rigid registration is not able to correct for any flexible deformations applied on the 2D sections during preparation. To get an even more accurate overall alignment, non-rigid image registration can be applied. It would be interesting to explore if training on solely the edge data has the same impact on non-rigid registration as it did on rigid registration of serial section images.

Finally, the results showed that our proposed quality measure is not able to provide a good score on the quality of the overall alignment. The measure is not able to take shifts caused by outliers in to account. Therefore, in future research it would be beneficial to create a quality measure based on the overall alignment in addition to the pairwise quality score.

Acknowledgements

The author would like to thank responsible supervisor Dr. F.J. Verbeek (LIACS) and secondary supervisor Dr. Lu Cao (LIACS) for their guidance through the research and their feedback on the report.

References

[Bor88] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 10(6):849–865, 1988.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[BT93] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[CCL+21] Tae-Young Choi, Tae-Ik Choi, Yu-Ri Lee, Seong-Kyu Choe, and Cheol-Hee Kim. Zebrafish as an animal model for biomedical research. *Experimental & Molecular Medicine*, 53(3):310–317, 2021.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[HCT+13] Kerstin Howe, Matthew D Clark, Carlos F Torroja, James Torrance, Camille Berthelot, Matthieu Muffato, John E Collins, Sean Humphray, Karen McLaren, Lucy Matthews, et al. The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446):498–503, 2013.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[JSZ+15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[KK23] Srigiri Krishnapriya and Yepuganti Karuna. Pre-trained deep learning models for brain mri image classification. *Frontiers in Human Neuroscience*, 17:1150120, 2023.

[LC07] Graham J Lieschke and Peter D Currie. Animal models of human disease: zebrafish swim into view. *Nature Reviews Genetics*, 8(5):353–367, 2007.

[PK17] Myeongsuk Pak and Sanghoon Kim. A review of deep learning in image recognition. In *2017 4th international conference on computer applications and information processing technology (CAIPT)*, pages 1–3. IEEE, 2017.

[PN11] Udai Bhan Pandey and Charles D Nichols. Human disease models in drosophila melanogaster and the role of the fly in therapeutic drug discovery. *Pharmacological reviews*, 63(2):411–436, 2011.

[SF02] Jordan T Shin and Mark C Fishman. From zebrafish to human: modular medical models. *Annual Review of Genomics and Human Genetics*, 3(1):311–340, 2002.

[SGS18] James M Sloan, Keith A Goatman, and J Paul Siebert. Learning rigid image registration-utilizing convolutional neural networks for medical image registration. 2018.

[Sim14] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[SWD+81] George Streisinger, Charline Walker, Nancy Dower, Donna Knauber, and Fred Singer. Production of clones of homozygous diploid zebra fish (brachydanio rerio). *Nature*, 291(5813):293–296, 1981.

[SWS17] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19(1):221–248, 2017.

[Ver96] FJ Verbeek. 3-d reconstruction from serial sections: applications and limitations. *Microscopy and Analysis*, pages 33–36, 1996.

[Ver99] FJ Verbeek. Theory & practice of 3d-reconstructions from serial sections. *Image processing, a practical approach*, pages 153–195, 1999.

[ZDGZ20] Xiaoran Zhang, Hexiang Dong, Di Gao, and Xiao Zhao. A comparative study for non-rigid image registration and rigid image registration. *arXiv preprint arXiv:2001.03831*, 2020.

[ZF03] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

A Appendix

data	model	loss_function	regularization	data_shape	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
standard	VGG16	ncc	None	window	407.498	398.023	1	-0.789	-0.805	4
edge-based	VGG16	ncc	L1	window	349.916	349.916	0	-0.812	-0.816	1
edge-based	VGG16	ncc	L2	window	353.513	333.636	3	-0.811	-0.820	2
edge-based	VGG16	ncc	L1L2	window	382.302	382.302	0	-0.805	-0.812	2
edge-based	VGG16	ncc	None	window	377.094	369.412	1	-0.799	-0.804	1
+ threshold	VGG16	ncc	None	window	462.202	462.202	0	-0.766	-0.804	7
edge-based	VGG16	mse	None	window	**298.616**	**298.616**	0	**-0.839**	**-0.855**	4
edge-based	VGG16	mse	None	transformed	565.056	555.352	1	-0.746	-0.759	3
standard	VGG19	ncc	None	window	425.856	425.856	0	-0.786	-0.799	3
edge-based	VGG19	ncc	L1	window	392.482	384.833	1	-0.801	-0.801	0
edge-based	VGG19	ncc	L2	window	400.690	400.690	0	-0.806	-0.814	3
edge-based	VGG19	ncc	L1L2	window	398.804	398.804	0	-0.801	-0.806	1
edge-based	VGG19	ncc	None	window	526.423	526.423	0	-0.741	-0.754	3
edge-based	VGG19	mse	None	window	**366.074**	**366.074**	0	**-0.809**	**-0.822**	3
edge-based	VGG19	mse	None	transformed	582.355	568.470	1	-0.692	-0.709	3
standard	Resnet50	ncc	None	window	499.102	499.102	0	-0.752	-0.763	2
edge-based	ResNet50	ncc	None	window	**378.059**	**378.059**	0	**-0.794**	**-0.808**	3
standard	FCN	ncc	None	window	NaN	NaN	-	NaN	NaN	-
edge-based	FCN	ncc	L1	window	299.893	257.198	5	-0.848	-0.902	14
edge-based	FCN	ncc	L2	window	641.235	641.235	0	-0.676	-0.676	0
edge-based	FCN	ncc	L1L2	window	604.204	604.204	0	-0.682	-0.682	0
edge-based	FCN	ncc	None	window	**266.560**	**159.961**	12	**-0.862**	**-0.927**	16
+ threshold	FCN	ncc	None	window	319.128	281.217	4	-0.817	-0.867	10
edge-based	FCN	mse	None	window	469.017	469.017	0	-0.764	-0.784	4
edge-based	FCN	mse	None	transformed	741.645	741.645	0	-0.617	-0.617	0

Table 10: Evaluation of all trained networks, accuracy in retrieval of simulated transformations. All models had the same random rigid transformations, resulting in an average similarity mse-score of 1967.284 and ncc-score of -0.191. Resulting mse and ncc scores achieved after registration are shown, both with and without outliers. Scores in bold are the best results of that particular model. The green coloured cells are the overall best scores.

data	model	loss_function	regularization	data_shape	mse	mse-outliers	outliers	ncc	ncc-outliers	outliers
standard	VGG16	ncc	None	window	747.465	723.050	2	**-0.690**	**-0.719**	10
edge-base	VGG16	ncc	L1	window	753.161	727.397	2	-0.686	-0.717	10
edge-base	VGG16	ncc	L2	window	**695.575**	**672.545**	2	-0.671	-0.684	4
edge-base	VGG16	ncc	L1L2	window	750.501	727.731	2	-0.680	-0.709	9
edge-base	VGG16	ncc	None	window	744.892	722.233	2	-0.687	-0.712	8
+ threshold	VGG16	ncc	None	window	762.703	728.895	3	-0.684	-0.712	7
edge-base	VGG16	mse	None	window	763.388	738.288	2	-0.677	-0.706	9
edge-base	VGG16	mse	None	transformed	806.144	766.249	4	-0.665	-0.684	5
standard	VGG19	ncc	None	window	733.365	709.262	2	-0.690	-0.718	10
edge-base	VGG19	ncc	L1	window	756.840	734.404	2	-0.685	-0.718	12
edge-base	VGG19	ncc	L2	window	700.378	679.178	2	-0.676	-0.693	5
edge-base	VGG19	ncc	L1L2	window	**652.853**	**625.942**	2	-0.669	-0.690	7
edge-base	VGG19	ncc	None	window	765.193	727.296	5	-0.676	-0.687	3
edge-base	VGG19	mse	None	window	687.868	663.266	2	-0.666	-0.683	5
edge-base	VGG19	mse	None	transformed	718.201	689.300	2	**-0.711**	**-0.729**	5
standard	Resnet50	ncc	None	window	811.957	750.427	7	-0.663	-0.683	7
edge-base	ResNet50	ncc	None	window	790.255	753.050	4	-0.672	-0.691	6
standard	FCN	ncc	None	window	NaN	NaN	-	NaN	NaN	-
edge-base	FCN	ncc	L1	window	827.918	806.719	2	-0.653	-0.687	12
edge-base	FCN	ncc	L2	window	778.123	748.265	2	-0.672	-0.700	9
edge-base	FCN	ncc	L1L2	window	779.634	720.755	5	-0.687	-0.712	7
edge-base	FCN	ncc	None	window	796.780	768.088	2	-0.669	-0.689	6
+ threshold	FCN	ncc	None	window	799.768	725.562	7	-0.664	-0.694	9
edge-base	FCN	mse	None	window	773.634	746.498	3	-0.686	-0.704	6
edge-base	FCN	mse	None	transformed	**657.590**	**636.344**	2	**-0.739**	**-0.758**	8

Table 11: 3D alignment performed by all trained models. Before alignment the test data series had a quality mse-score of 836.119 without any outliers present. The ncc-score beforehand was -0.697 and -0.712 with five outliers removed. Both ncc and mse quality scores before and after eliminating outliers are shown. Scores in bold are the best results of that particular model. The green coloured cells are the overall best scores.