UNIVERSITY OF COPENHAGEN

COMPUTER SCIENCE

Punctae Segmentation for Measuring Transcytosis Across the Blood–brain Barrier

Author:
Casper BRESDAHL

Teacher:
Martin LILLHOLM

June 27, 2021

1 Motivation

The blood-brain barrier is an interface separating the brain form the circulatory system [4]. The very restrictive nature of the blood-brain barrier, only allowing certain small molecules to pass, makes it a big obstacle in central nervous system drug delivery [4]. Defects in the blood-brain barrier have also been reported in Alzheimer's and Parkinson's disease patients, connecting the two diseases to the blood-brain barrier [5]. For these reasons research in the blood-brain barrier and its permeability is important. When studying the permeability of the blood-brain barrier, tracer molecules are injected into the brain of mice which then accumulate around the blood-brain barrier interface. With two-photon microscopy we can see and measure the concentration of these tracer molecules. However, quantifying these accumulations is very subjective, non-trivial and time consuming, even for experts, and for these reasons this project aims to explore the possibilities of training and validating a deep learning model for automatic segmentation of accumulation of these tracer molecules.

2 Theory

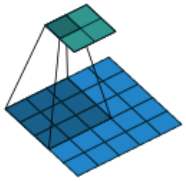
2.1 Two-photon microscopy

Two-photon microscopy is a fluorescence imaging technique, where two photons are brought to an excitation state at the same time and then absorbed to emit a photon of fluorescence. As the wavelength of the two excited photons is required to be longer than the emitted light, near-infrared excitation light is often used. Because of the multi photon absorption, the background signal is strongly suppressed, and because using infrared light minimizes scattering in tissue, two-photon microscopy gets an increased penetration depth over confocal microscopy, which uses a spatial pinhole technique to block out-of-focus light [6]. Due to this increased penetration depth, two-photon microscopy can be used to obtain images of living tissue up to one millimetre in thickness [1]. For this project, this means two-photon microscopy can be used to measure transcytosis through the blood-brain barrier. This is done in [3] where the change in blood-brain barrier permeability of small molecules is measured after injection of fluorophore, a compound which emits photons upon excitation. In more detail this was done by using two-photon microscopy in vivo in mice, where the anesthetized mice's brains were imaged through a craniotomy over the somatosensory cortex. The light emitting fluorophore signal can then be measured in both the blood vessels and brain parenchyma. The used two-photon microscopy allows for imaging at different depths in the tissue and can be compiled to a three dimensional reconstruction of the brain microvasculature. Figure 1 shows a cross section of such reconstructions and thus how the transcytosis in the blood-brain barrier has been affected. With time, the fluorophore coalesce into numerous puncta of vescicles. In Figure 1 we see these puncta of vesicles which accumulate around the blood-brain barrier interface. The three dimensional reconstruction can be sliced in different ways to give different perspectives of the results. Imagining the reconstruction as a block Figure 1 is likely images where the block has been sliced from top to bottom. One could also slice the block from left to right giving a view of the 'insides' of the arteries where we would then see the puncta coalesce in a circle around the blood-brain barrier interface.

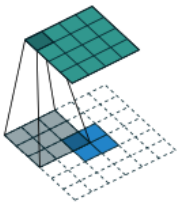
Figure 1: Cross section images of reconstruction taken from the training data and overlaid with the true masks indicating puncta we are interested in finding.

2.2 Convolutional layers

In the convolution layers a (discrete) convolution is performed. A (discrete) convolution is an operation in which a kernel is moved over a signal and the output is the sum of products between the kernel and the overlapping elements. In our concrete case we are dealing with images, which can be thought of as an array of pixel values. If we simplify the discussion to greyscale images, we simply have an array of same dimension as the image filled with pixel intensities. We can now overlay the input image with our kernel and compute the sum of products. We can now move the kernel over the image, and each time we move the kernel we get an output. The number of pixels we move the kernel is called the *stride*. After having moved the kernel over the entire image, the output is a new image which dimensions are smaller than the input. How much smaller depends on the stride and the kernel size. If a certain output dimension is needed after a convolution, the input images might need to be padded with 'extra' pixels. Several strategies exists for padding, the most common is to pad with zeros. An illustration of a convolution can be seen in Figure 2a [11].



(a) Convolution of a 5x5 input image with a 3x3 kernel using no padding and 1 stride, producing a 2x2 output image.



(b) Transposed convolution of a 2x2 input image with a 3x3 kernel using 2 'layers' of padding all around and 1 stride, producing a 4x4 output image.

Figure 2: Examples of a convolution and a transposed convolution. Blue layer denotes input image, cyan layer denotes output image, shaded squares denotes kernel and dotted squares denotes padding. Images are from [11].

2.3 Max pooling layers

In the max pooling layers a kernel is defined and moved around the input image. The output is then the largest value inside the kernel. The number of pixels it is moved is again determined by a stride. A max pooling layer using a 2x2 kernel and a stride of 2 thus halves both input image dimensions as a 2x2 area in the input image becomes a single pixel in the output image [11].

2.4 Transposed convolutions

In the upsampling layers we want to perform a 'reversed' convolution where we go from a small image to a larger image while still keeping the connectivity pattern. The 'up-conv' layer used in Figure 3 is also called a *transposed convolution*, and we perform these instead of simply upscaling the image because we can learn an optimal way of upscaling the images. To perform a transposed convolution we compute the output shape of a convolution with a given input shape and then we invert the input and output shapes. The input would then be padded with zeroes to comply with the kernel size. As an example we would like to perform a

convolution with a 3x3 kernel on a 4x4 input image with no padding and 1 stride. This would produce a 2x2 output image. Inverting the input and output, we would then have to convolve a 2x2 image with a 3x3 kernel to get a 4x4 output. As the 2x2 image does not comply with the 3x3 kernel, we pad it with 2 'layers' of zeroes all around. An illustration of a transposed convolution can be seen in Figure 2b. Although this is not the most efficient way of performing a transposed convolution, it gives an intuition of what is happening [11].

2.5 Convolutional Neural Networks

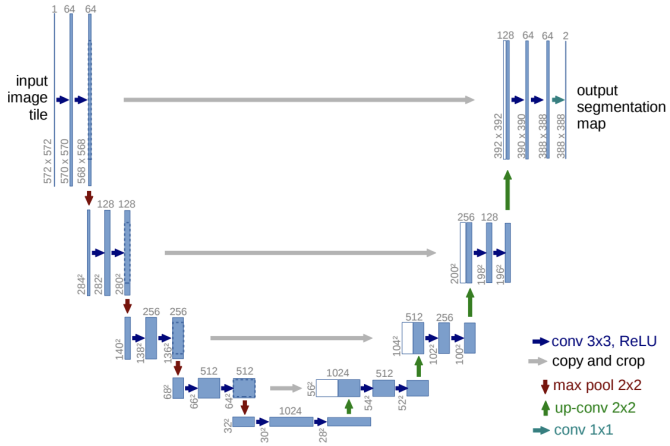


Figure 3: U-Net architecture example for input image of size 572x572 pixels. Image taken from [7]

Deep neural networks (DNN's) are neural networks with several layers between the input and the output which allows them to learn complex non-linear relationships in the data. Each layer holds a number of neurons which usually are fully connected, meaning each output from the neurons in the previous layer goes into the neurons of the next layer. DNN's have a lot of parameters which are getting trained as each neuron have their own weight. A convolutional neural network (CNN) is a special class of DNN's which mainly uses convolutional layers. This makes it possible to process images as it is the weights of a kernel which are trained rather than processing each individual pixel in the images. CNN's are usually used for image segmentation or in general tasks which requires processing of images. A concrete example of a CNN could be a modified U-Net model where the encoder is replaced by a ResNet-18 model. The original U-Net architecture can be seen in Figure 3.
The U-Net architecture can be split in two parts, a left part which downscales the processed image and a right half which upscales the image again. Replacing the encoder of the U-Net model means this left part is replaced with the ResNet-18 model. ResNet-18 begins by performing a 7x7 convolution with a stride of two followed by a 3x3 max pooling with a stride of two. It then follows a pattern of doing two 3x3 convolutions each followed by a rectified linear unit operation before a skip connection, skipping two convolutions. After three convolutions the fourth uses a stride of two to downsample the image. This results in the feature map getting halved in size and the number of the feature channels getting doubled after each downsampling step. Three downsamplings are done this way before we reach the right half of the U-Net model [8]. For the right half, an upsampling followed by a 2x2 convolution is performed. This halves the number of feature channels and doubles the size of the feature map. Then a concatenation from the corresponding left half feature map is done. These skips (along with whose in the ResNet-18 model) allows the model to recover lost spatial information and alleviates the vanishing gradient problem, which results in improved performance. After the skips, two 3x3 convolutions are performed each followed by a rectified linear unit operation. In the end a 1x1 convolution is performed to map the final feature map to the correct number of prediction classes [7]. Another example of a CNN is LinkNet which in a similar fashion as to U-Net has a left part which downsamples images and a right path which then upsamples them with skip connection in between, but

instead of concatenating the feature maps after these skips, in LinkNet they are added. This reduces the number of parameters the network needs to learn [9].
The clear advantages of CNN's are how accurate and how fast the models can make predictions. This allows for fast automatisation of suited tasks. However, it might take weeks to train state of the art models without any guarantees if it will perform better than its predecessor, and so it is a time consuming process to develop good models. It is also taxing on power consumption and the environment to train for these long periods of time. Often times the most limiting factor in model development is gathering varied data or annotating this data.

2.6 Transfer learning

Transfer learning is a common technique where a model developed for some task is reused as starting point for another model. This allows the model to get a 'head start' as it has already learnt a set of features to detect. The ImageNet project is a large database consisting of more than 14 million images in over 20 thousand categories [2]. It is very common to use transfer learning with a model trained on the ImageNet data as this gives a broad foundation, and further training then 'specializes' the model to the specific task at hand.

2.7 Optimization

For optimization, the stochastic gradient descend (SGD) based learning algorithm *Adam* is often used. SGD is an iterative method for optimization and can intuitively be thought of as having a ball rolling on a hill landscape, SGD finds the direction of steepest descend and takes a step in this direction. Adam implements three important improvements to traditional SGD. The first being each parameter has its own adaptive learning rate. The second being each learning rate can obtain momentum. That is, if we move down a steep slope, we should begin to move faster as we probably are far from a minima. At the same time we should begin to move slower when the slope is flat to avoid 'overshooting' the minima. The third change is learning rates should change faster at the beginning of training, and slower towards the end [12].

2.8 Data augmentation

For all DNN's overfitting is a concern. Data augmentation is a technique often employed when working with CNN's which aims to reduce overfitting during training. It works by taking already existing training images and alter them by for instance to rotate, shear, crop or in some way distort the motive. The aim is that adding augmented images to the training set will make it represent a more comprehensive set of possible data points and thus minimize the distance between the training, validation and test set [10].

2.9 Measurement Metrics

To evaluate the performance of our models we will use two metrics, DICE and circle count. DICE is a known measure where the overlap of two sets is measured. In our case we will use two masks, a predicted mask and a true mask. If we define the predicted mask as X and the true mask as Y the actual formula is: DICE $= \frac{2|X \cap Y|}{|X|+|Y|}$ where $|X|$ and $|Y|$ denotes the number of dots in a mask. Because DICE will give a measure of 0 if there are no dots in the true mask, it can be hard to report an accurate average DICE. Therefore, we will also use a metric we will call circle count. As several dots can constitute a single punctae, we will count the number of coherent 'elements' and call these circles. This can be done through connected component labelling where the connectivity of the dots are assessed, and if the dots are connected they will be counted as one circle.

We can then compare the number of circles in the predicted mask and in the true mask to see whether we predict less, the same or more circles / puncta.

3 Methods

The goal of this project is to train a model which as input takes the two-photon microscopy images of mice brains and outputs the locations of vescicles puncta. To train the model an annotated dataset is needed, where each puncta has been marked by hand by an expert in the field. The CNN is then trained by supervised learning to predict the location of the puncta. This section describes the methods used to achieve this.

3.1 Preprocessing

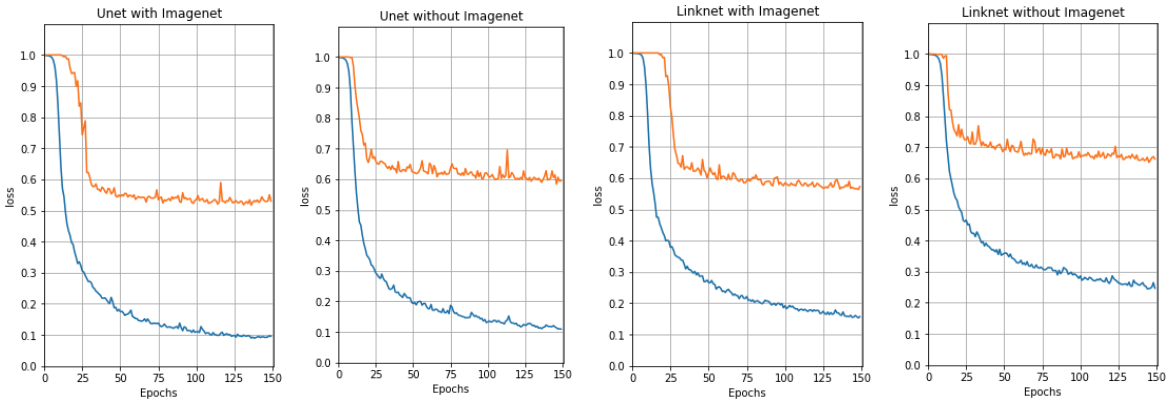
As the two-photon microscopy images comes in many different resolution depending on equipment used, preprocessing is needed to standardize the images. First all images have been scaled to the same resolution. Then, by finding the largest image, all other images has been padded with zeros until they achieve the same size. The images have then been resized to a size of 1024 by 1088. To ensure the masks still correspond to the images, the same scaling, padding and resizing have been applied to the masks. The images only have one colour channel with intensities between 0 and 4095. To make this match the input layer of the modified U-Net model, the intensities are normalized to be between 0 and 1 by first clamping all pixel intensities to be between 0 and 4095 and then dividing each pixel in the images with 4095. As the masks are binary to mark the location of puncta, this last preprocessing step has not been applied to the masks. The, in total, 722 images are now split into a training set of 506 images, a validation set of 104 images and a test set of 112 images. After the initial preprocessing the images have been resized to 512 by 544, that is the images have been halved. A bicubic interpolation has been used for the images and a nearest neighbour interpolation has been used for the masks in all datasets. Regardless of image sizes, in order to make the images comply with the modified U-Net model, three colour channels are needed. As our images only have one colour channel, this channel has simply been replicated twice.

3.2 Models

The first model developed was a modified U-Net model where the encoder part has been replaced by a ResNet-18 model, transposed convolutions have been used for the upsampling layers to learn how the images are upsampled optimally, a sigmoid function have been used as activation function for the final output layer as the masks are binary, Adam have been used for optimization with an initial learning rate of 0.0001, soft DICE have been used for loss function and its weights have been initialized with the ImageNet weights. Very similarly, the second model uses the same configuration except it is not initialized with ImageNet weights. The last two models are LinkNet models using the same configurations as the previous two models.

3.3 Testing

To measure the models unbiased performance on unknown data, the test set is used. As all predictions and true masks have image sizes 524 x 544 they are resized back to original sizes using the same interpolation rules. The 'immediate' output of the models are confidence maps, and to make these binary, we threshold them. This is done by thresholding all predicted masks from the training set and then computing the average DICE. The threshold that achieves the highest average DICE is then chosen, and all predicted masks from



(a) U-Net model whose weights are initialized with ImageNet weights.

(b) U-Net model whose weights are not initialized with ImageNet weights.

(c) LinkNet model whose weights are initialized with ImageNet weights.

(d) LinkNet model whose weights are not initialized with ImageNet weights.

Figure 4: All models are trained on the original and halved training data. The blue graphs indicates training loss measured with DICE, and the orange graphs indicates validation loss also measured with DICE.

the test set are then thresholded by this value, and its unbiased performance can then be measured. There exists different ways of measuring DICE on the test set. The test set can be split into three separate volumes, one for the single series images, one for the five series images and one for the six series images and the average DICE can then be measured across each. The three volumes can also further be split into 'time intervals' where the DICE is measured across all the images coming first in the series, second in the series and so on.

3.4 Augmentations

The first augmentation used is large random rotations. Here each of the original images have been rotated between -45 and 45 degrees. Another dataset of small random rotations between -30 and 30 degrees have also been created. The next augmentations flips each image horizontally and vertically giving us two more datasets. Two datasets have also been created for small and large random shearing in both the x and y direction. A *mix* dataset has also been created where the original images were first sheared then rotated and finally flipped horizontally. A dataset was also created where the original images were 'bend' around the middle to form an arc shape. The last dataset created used the python package *imgaug* to randomly apply one or more of the following augmentations to each image: horizontal flipping, vertical flipping, $80-120\%$ scaling, small translations, -45 to 45 degrees rotations, -16 to 16 degrees shearing, randomly drop $1-10\%$ of the pixels in the image, locally move some pixels around and lastly, perform local distortions.

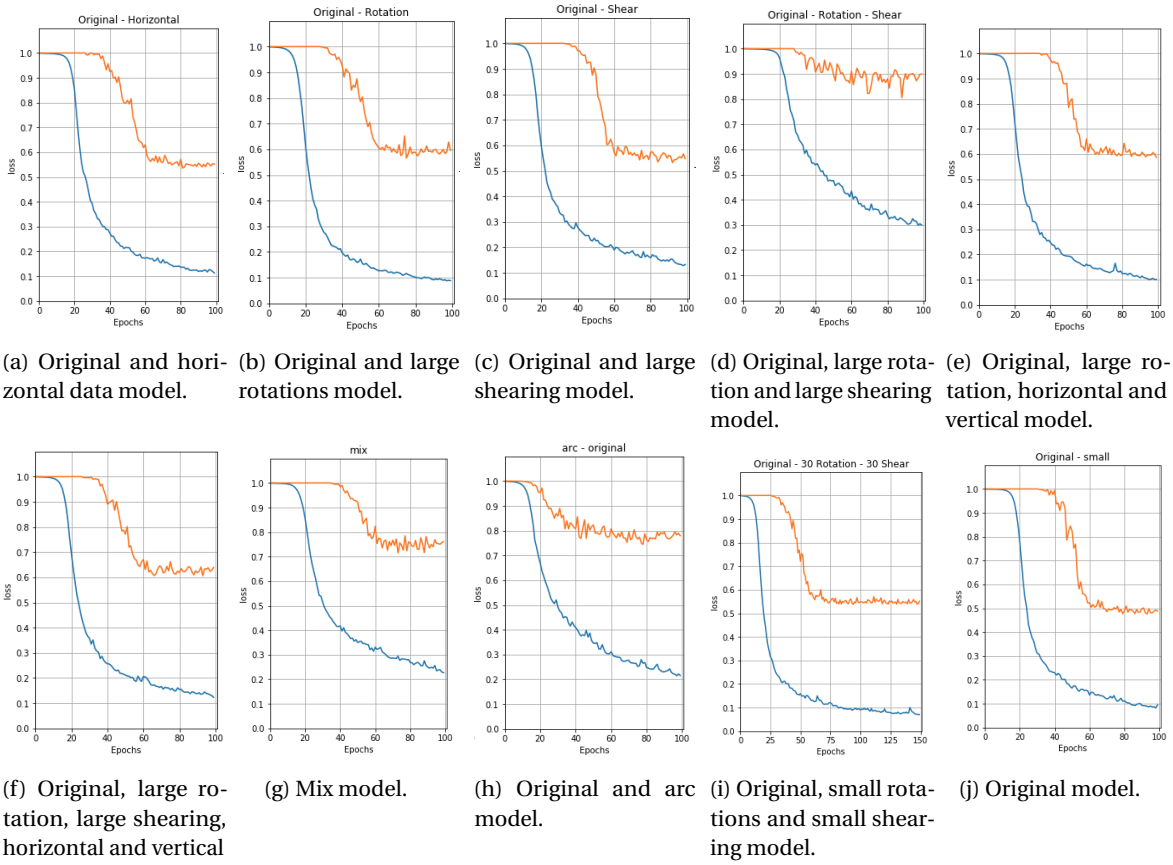
3.5 Construction of augmented training sets

With the augmented datasets, several training sets were constructed and used to train models. Each training set consists of 506 training images, and is constructed by randomly sample from the involved datasets without replacement, ensuring an even amount of images are taken from each dataset. An *original* training set is also created consisting of the un-augmented images.

4 Results

4.1 Initial results

Early in the project the models described in subsection 3.2 were trained on the original training set. In Figure 4 we see the training and validation loss measured in DICE for each of the models. We see that all models could probably still achieve lower training loss, but after 150 epochs both U-Net models achieve lower training loss than the LinkNet model using Imagenet weights, and a lot lower training loss than the LinkNet model not using ImageNet weights. Looking at the validation loss, the two models using ImageNet weights shows the lowest loss with the U-Net model using ImageNet weights having slightly lower loss than the LinkNet model using ImageNet weights.



(a) Original and horizontal data model.

(b) Original and large rotations model.

(c) Original and large shearing model.

(d) Original, large rotation and large shearing model.

(e) Original, large rotation, horizontal and vertical model.

(f) Original, large rotation, large shearing, horizontal and vertical model.

(g) Mix model.

(h) Original and arc model.

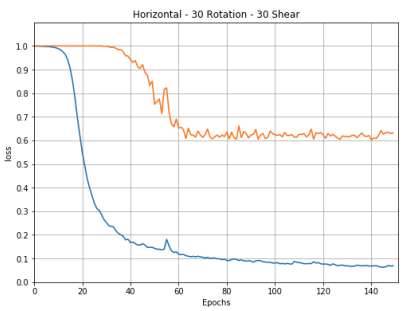
(i) Original, small rotations and small shearing model.

(j) Original model.

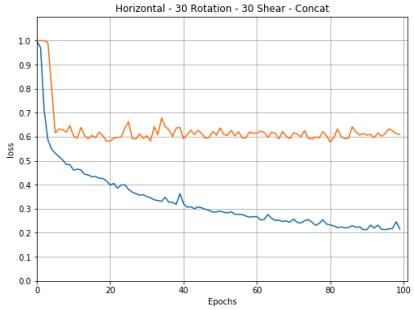
Figure 5: Results of training the modified U-Net model initialized with the ImageNet weights on different augmented datasets. The blue graphs indicates training loss measured with DICE, and the orange graphs indicates validation loss also measured with DICE.

4.2 Augmentation results

In Figure 5 are the results of training the modified U-Net model initialized with the ImageNet weights on different augmented training sets. We note the model achieving the lowest validation loss is the model trained on the original training set, achieving an average DICE around 0.5. The model trained on augmented data which achieves the lowest validation loss is the model trained on the original, the small rotations and the small shearing data sets, achieving an average DICE around 0.46. Although this model has been trained on 150 epochs it seems to give the same results after 100 epochs. The model trained on the original and horizontal data along with the model trained on the original and large shearing data also seems to come



(a) Modified U-Net model trained on the horizontal, small rotations and small shearing datasets.



(b) Modified U-Net model trained on the horizontal, small rotations and small shearing datasets concatenated and shuffled.

Figure 6: Experiment showing concatenating the data sets rather than picking 506 images to train on yields same validation results.

close to an average DICE of 0.45. Examples of prediction results can be seen in the appendix.

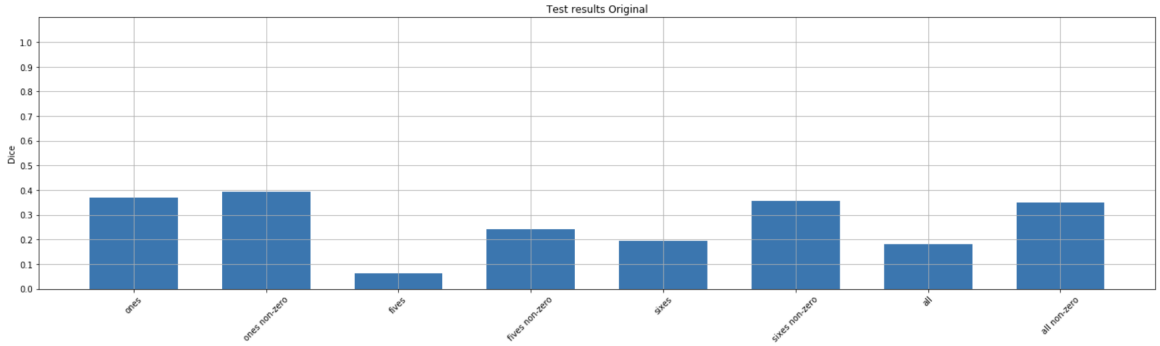
An experiment was conducted to see if it would make a difference in model performance to concatenate the datasets together rather than the fixed 506 images. In Figure 6 we see two models trained on the horizontal, small rotation and small shearing data sets, but one have been trained on 506 images whereas the other have had the three datasets concatenated and then shuffled. The most significant change is how fast the validation loss converges in the model trained on the concatenated dataset, although the validation loss ends being the same. We also note the training loss ends a lot higher in model trained on the concatenated data sets.

4.3 Test results

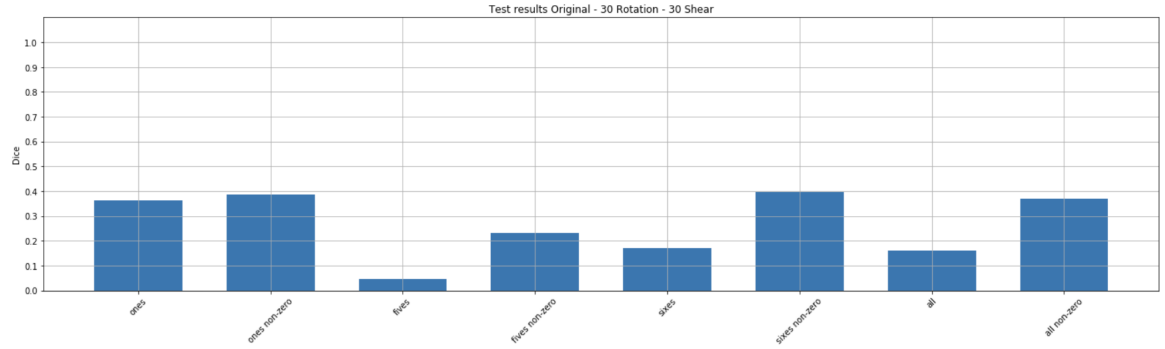
		t_1	t_2	t_3	t_4	t_5	t_6
Fives data	Zero augmented	0.0	0.0	0.055	0.038	0.138	
	Non-zero augmented	0.0	0.0	0.387	0.132	0.240	
	Relative change	0.0	0.0	0.332	0.094	0.102	
	Zero original	0.0	0.0	0.105	0.082	0.123	
	Non-zero original	0.0	0.0	0.366	0.191	0.216	
	Relative change	0.0	0.0	0.261	0.109	0.093	
Sixes data	Zero augmented	0.0	0.0	0.067	0.217	0.318	0.427
	Non-zero augmented	0.0	0.0	0.667	0.309	0.398	0.427
	Relative change	0.0	0.0	0.600	0.092	0.080	0.0
	Zero original	0.0	0.059	0.147	0.239	0.289	0.439
	Non-zero original	0.0	0.198	0.367	0.299	0.362	0.439
	Relative change	0.0	0.139	0.220	0.060	0.073	0.0

Table 1: Average DICE of each 'time interval' of the five series and six series test data.

In Figure 7 the test set has been split into three separate volumes. Each of the results have been reported with and without having zero results in the average. The model trained on augmented data seems to score slightly higher or the same averages on all volumes when we do not count in zero results whereas when we do count zero results the model trained on the original training set seems to score higher. The overall results seem to be quite similar. Test results for the rest of the models can be seen in the appendix. In Table 1 the volumes have further been split into 'time intervals'. Here we see when removing zero results the relative increase in DICE is greatest for the model trained on augmented data, and the relative increase diminishes as we move 'up in time'. In Figure 8 the performance have been measured with the circle count metric. We note for both models the regression line is quite close to have slope 1. We also note for the model trained on the original training set, that in images where the true mask count is low the predicted



(a) Results for the model trained on the original training set.



(b) Results for the model trained on the original, small rotations and small shearing data sets.

Figure 7: Average DICE on test set split into three separate volumes consisting of the single series images, five series images and six series images.

number of circles is high, whereas when the true mask count is high, the predicted number of circles is close to be the true number of circles. For the model trained on the augmented data we see the opposite trend.

5 Discussion of results

5.1 Initial results

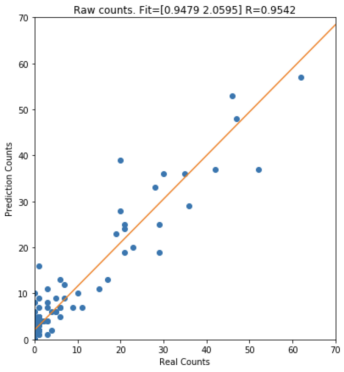
Early in the project an experiment was conducted to determine which model to keep working with. The choice was between the four models in Figure 4 where the modified U-Net model initialized with ImageNet weights showed the lowest validation loss after 150 epochs, and because of this the model was chosen to see if the performance could be further improved by augmentation.

5.2 Augmentation results

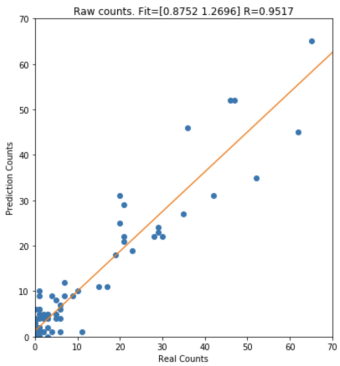
Our expectation would be the model performance would increase if the model would be trained on augmented data. However, as seen in Figure 5 training on augmented data makes the modified U-Net model perform worse, and in some cases a lot worse. In some cases this seem to stem from over augmentation, as when we compare the results of training on the original, large rotation and large shearing data compared to training on the original, small rotations and small shearing data. Simply making smaller rotations and smaller shearing gives us significantly better results. Over augmentation is likely also the reason the mix model perform worse. For the model trained on the original and arc data, the arcing effect might alter the image too much resulting in the model learning suboptimal features. Because CNN's are translation and rotation invariant it would be expected that the original and horizontal model and the original and large rotations model would perform as good as simply training on the original dataset, however, this is not the

case. Although both of these models achieve a good validation loss it is still worse than the validation loss of the model trained on the original training set. This would indicate that substituting augmented data into the training set makes the model overfit relatively more than it does on the original data alone. As we remove some data to add the augmented data this points in the direction of our model not having enough data to train on or at least it only have a few images with some quite important features which seems to be replaced by less important features after augmentation. This second idea seem to be strengthened by Figure 6 where although triple the amount of data is used to train the model, the validation loss does not improve which would indeed indicate the augmented data does not 'catch' the 'rare' features of the original dataset and thus does not increase the variety of data but instead adds more of the same 'kind' of data.
One could have conducted experiments by increasing or decreasing the number original images in the training sets to see if this would influence the assumed overfitting in the models trained on augmented data. An experiment have however been conducted to see the importance of image sizes, however the results are inconclusive and can be seen in the appendix.

5.3 Test results



(a) Results for the model trained on the original dataset.



(b) Results for the model trained on the original, small rotations and small shearing dataset.

Figure 8: Results of the circle count metric. A regression line has been fitted and plotted to show how close the results are to lie on a line with slope 1.

From Table 1 we note when we remove all the results which gives a DICE of 0 the average DICE increases relatively more for the model trained on augmented data than the model trained on the original data. Getting an increase in DICE when removing zero results means the predictions made are closer to the true mask as predicting too many or too few dots results in a lower DICE which would lower the average. This relative increase in DICE equalizes between the models as we increase the 'time intervals'. This might be due to the model trained on original data performs better on later time intervals than the model trained on augmented data or that predicting too many or too few dots becomes relatively less impactful the more dots that are present. This indicates the model trained on augmented data performs better on data with only a few dots than the model trained on the original data does. In Figure 8 we see this trend again as the model trained on the original data seemingly over predicts a lot of circles / puncta in images with a true count below 15 which the other model does not. However, we also see the model trained on the original data seem to predict the number of circles more accurately on images with a true count of more than 30 circles, whereas the model trained on augmented data seem to either over predict or under predict quite symmetrically.

5.4 Automatic tool

One goal of this project was to explore the possibility of automatic segmentation of two-photon microscopy images. Based on the results in Figure 7 it seems this is not quite possible as the best models developed achieve an average DICE around 0.4. However, this does not mean these

models can not be used when working with two-photon microscopy images. One could use the models to initially find some of the dots. Depending on if it is easier to find the dots or verify that a dot is placed correctly, this might save significant time. Based on the results in Figure 8 one can get an initial count of the circles in the images significantly faster than having to count manually. Although this might introduce some uncertainty, if one is only interested in an indication about the number of circles this might speed up work significantly.

6 Conclusion

In conclusion we have seen how transfer learning helps improve the performance of models and conclude based on the validation losses in Figure 4 that the modified U-Net model performs best for our task. We have also seen how training on augmented data only makes this modified U-Net model perform worse. We conclude this is likely due to the original training set only containing a few images with some important features, and substituting in augmented images adds more of the same features instead of adding a more comprehensive set of possible features. We also conclude based on the results in Figure 8 that training on augmented data, although not visible in the losses, does improve performance on images with only a few dots, whereas training on the original training set increases model performance on image with a lot of dots. To end with, we conclude these models can not be used as an automatic tool to annotate two-photon microscopy images, but they can be used to give an initial count of puncta in the images although with some uncertainty.

7 References

[1] Wikipedia, 'Two-photon excitation microscopy', https://en.wikipedia.org/wiki/Two-photon_excitation_microscopy, Visited: June 11, 2021.

[2] Wikipedia, 'ImageNet', https://en.wikipedia.org/wiki/ImageNet, Visited: June 26, 2021.

[3] Mathiesen Janiurek et al, 'Apolipoprotein M-bound sphingosine-1-phosphate regulates blood–brain barrier paracellular permeability and transcytosis', eLife 2019, Published: 25 November 2019.

[4] Swathi Ayloo and Chenghua Gu, 'Transcytosis at the blood–brain barrier', Current Opinion in Neurobiology, Published: 2019.

[5] Zhen Zhao et al, 'Establishment and Dysfunction of the Blood-Brain Barrier', Cell 163, Published: November 19, 2015.

[6] Wikipedia, 'Confocal microscopy', https://en.wikipedia.org/wiki/Confocal_microscopy, Visited: June 22, 2021.

[7] Olaf Ronneberger et al, 'U-Net: Convolutional Networks for Biomedical Image Segmentation', Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, Published: 2015.

[8] Kaiming He et al, 'Deep Residual Learning for Image Recognition', Microsoft Research, Published: December 10, 2015.

[9] A. Chaurasia and E. Culurciello, 'LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation', Purdue University, Published: June 14, 2015

[10] C. Shorten and T. Khoshgoftaar, 'A survey on Image Data Augmentation for Deep Learning', Journal of Big Data, Published: 2019.

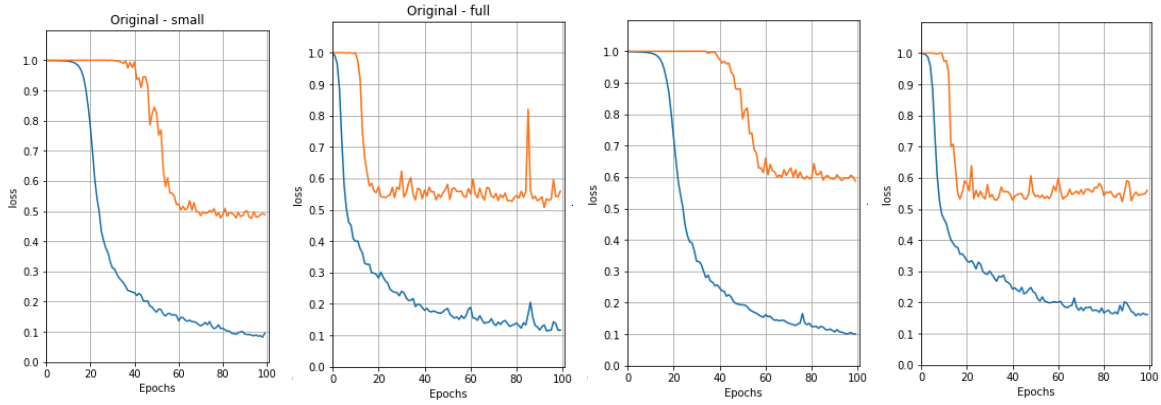
[11] Vincent Dumoulin and Francesco Visin, 'A guide to convolution arithmetic for deep learning', MILA, Université de Montréal and AIRLab, Politecnico di Milano, March 24, 2016.

[12] D.Kingma, J. Ba, 'ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION', Published as a conference paper at ICLR, 2015

8 Appendix

8.1 Full sized images vs. halved sized images

In another experiment it was investigated whether the model performances would change if the full sized images were used rather than the half sized images. In Figure 9 four modified U-Net models initialized with the ImageNet weights were trained first on a dataset where the images were halved and then where the images kept their full size. On both datasets using the full sized images seems to make the models converge considerably faster. For the models trained on the original dataset the validation loss seems to be lower when the images are halved while the training loss seems to be about the same. However, when using data from the original, large rotation, horizontal and vertical datasets the validation loss seem to be higher for the halved images, but the training loss seems be lower.



(a) Model trained on original dataset with halved sizes.

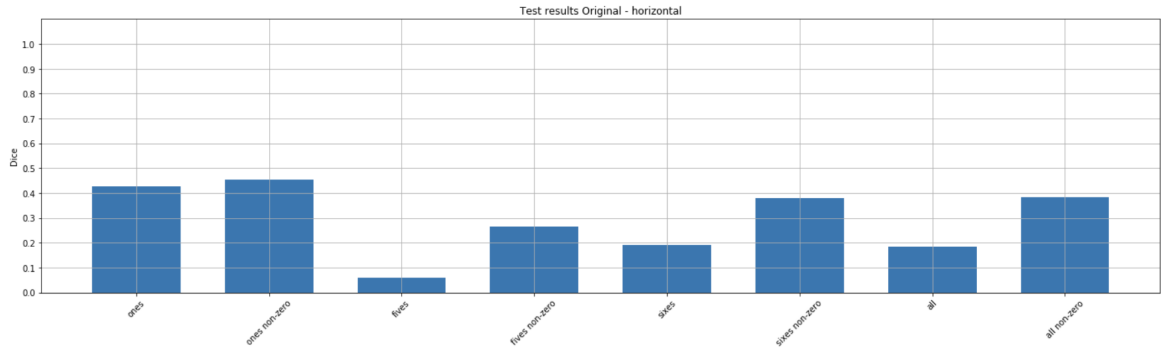
(b) Model trained on original dataset with full sizes.

(c) Model trained on original, large rotations, horizontal and vertical dataset with halved sizes.

(d) Model trained on original, large rotations, horizontal and vertical dataset with full sizes.

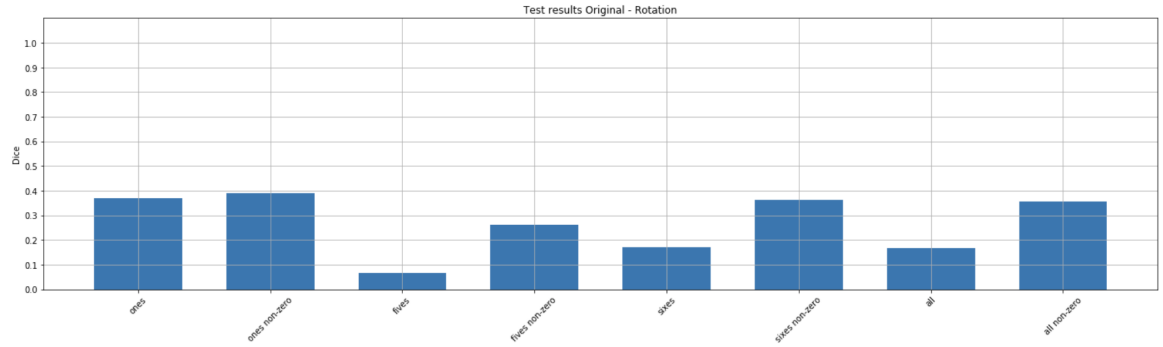
Figure 9: Experiment showing the effect of training on full sized images vs. halved sized images.

8.2 Test results

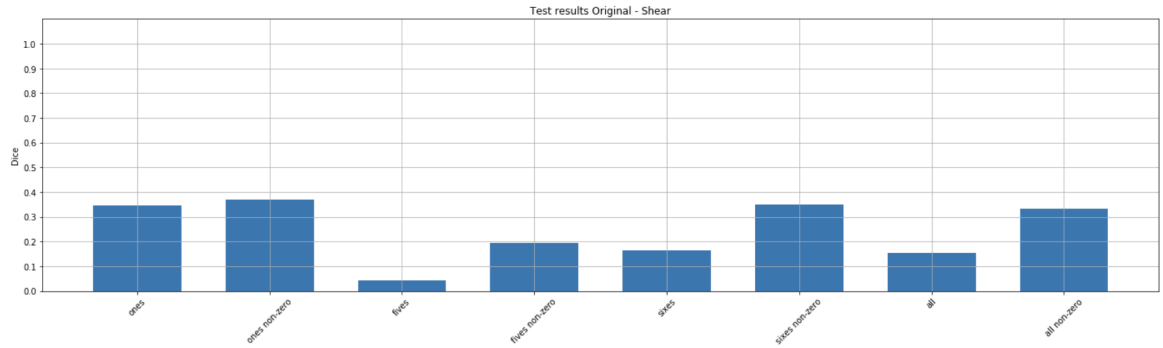


(a) Results for the model trained on the original and horizontal data sets.

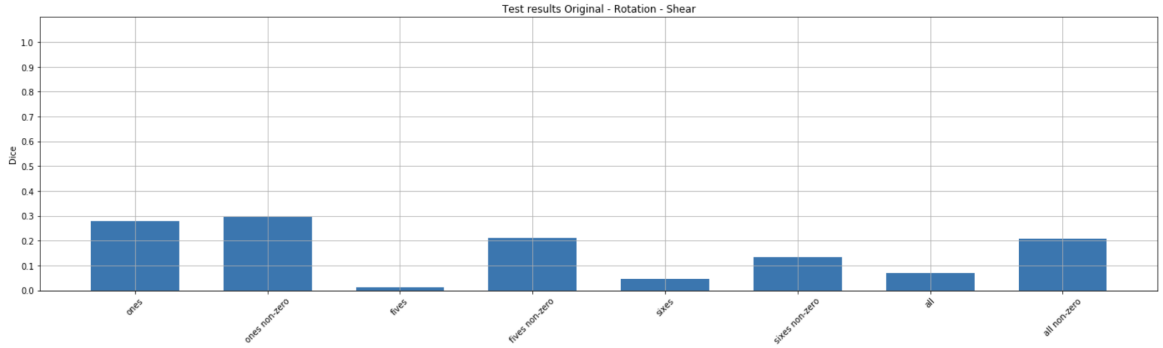
Figure 10: Average DICE on test set split into three separate volumes consisting of the single series images, five series images and six series images.



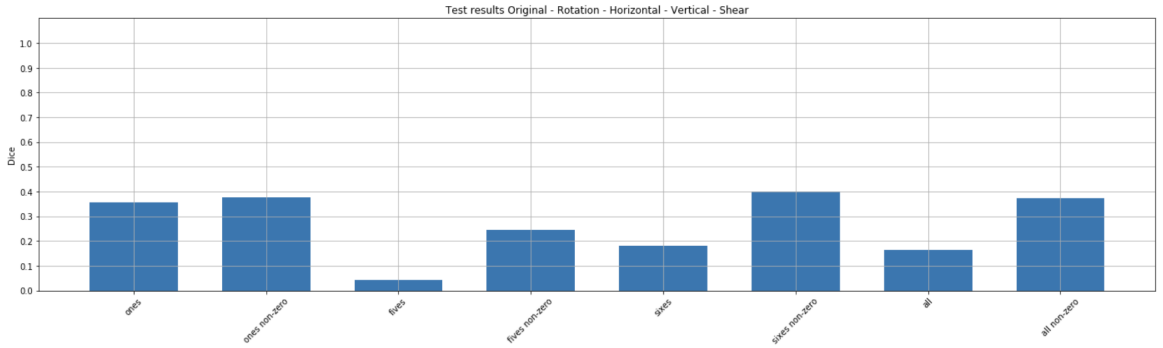
(a) Results for the model trained on the original and large rotations data sets.



(b) Results for the model trained on the original and large shearing data sets.

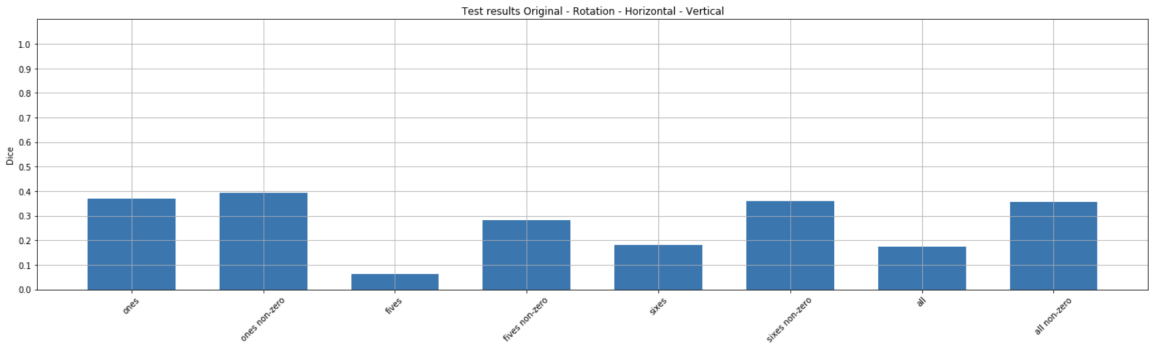


(c) Results for the model trained on the original, large rotations and large shearing data sets.



(d) Results for the model trained on the original, large rotations, horizontal and vertical data sets.

Figure 11: Average DICE on test set split into three separate volumes consisting of the single series images, five series images and six series images.



(a) Results for the model trained on the original, large rotations, horizontal, vertical and large shearing data sets.

Figure 12: Average DICE on test set split into three separate volumes consisting of the single series images, five series images and six series images.

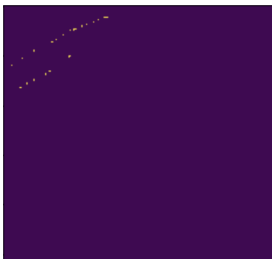
8.3 Prediction examples

Here we see predicted masks versus the corresponding true mask for different models and images in the training set.

(a) Predicted mask from model trained on original, small rotations and small shearing. 0.93 DICE.

(b) Predicted mask from imgaug model. 0.87 DICE.

(c) Predicted mask from model trained on original training set. 0.93 DICE.

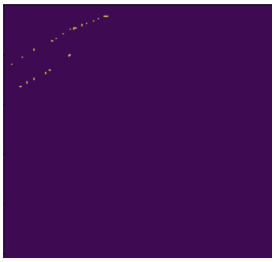


(d) Predicted mask from model trained on original and horizontal data sets. 0.90 DICE.

(e) True mask from model trained on original, small rotations and small shearing.

(f) True mask from imgaug model.

(g) True mask from model trained on original training set.



(h) True mask from model trained on original and horizontal data sets.