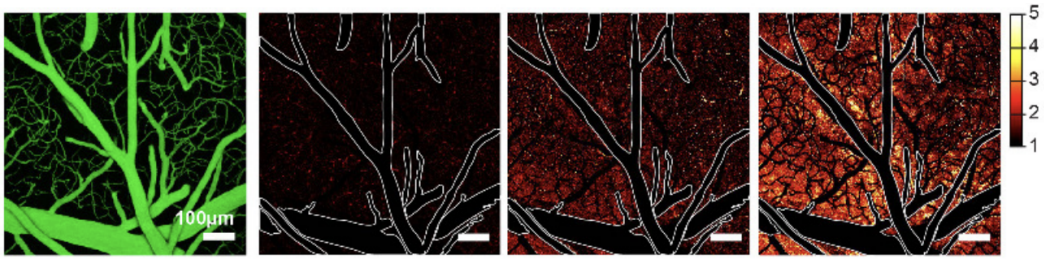
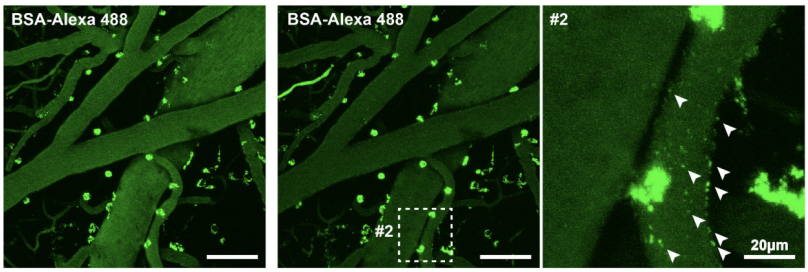


(a) Cross section time lapse images with injection of NaFluo. We see an accumulation of NaFluo in the brain parenchyma over a period of 30 minutes. Image taken from [2].



(b) Coalescence of BSA-Alexa 488 after 60 and 120 minutes. White arrows indicate puncta we are interested in finding. Image taken from [2].

2.2 Convolutional Neural Networks

A convolutional neural network (CNN) is a special class of deep neural networks which consists of several layers, but mainly convolutional layers. CNN's are usually used for image and video recognition, image classification, image segmentation or in general tasks which requires processing of images, and so the input to a CNN is usually an image. A concrete example of a CNN is U-Net which we will use to train our models later. U-Net consists of 46 layers of which 23 are convolutional layers and its architecture can be seen in Figure 2.

Is this accurate?

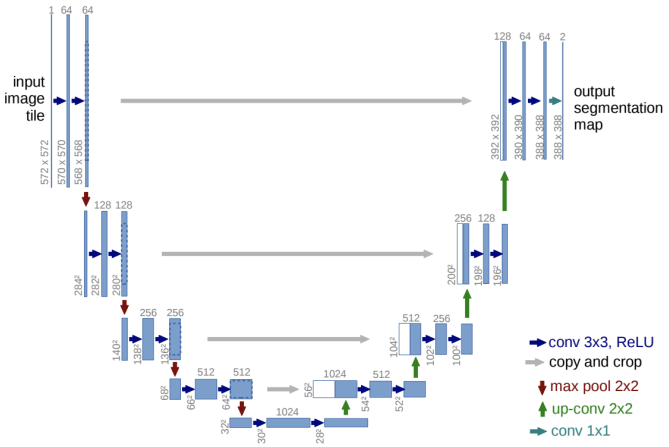
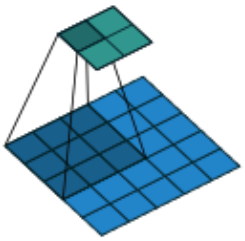


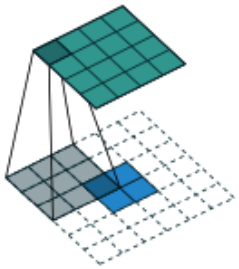
Figure 2: U-Net architecture example for input image of size 572x572 pixels. Image taken from [5]

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. The left half follows a pattern of doing two 3x3 convolutions each followed by a rectified linear unit and then a 2x2 max pooling operation with 2 stride to do the downsampling. This results in the feature channels getting doubled after each downsampling step. For the right half an upsampling followed by a 2x2 convolution is performed which halves the number of feature channels. Then a concatenation from the corresponding cropped left half feature map and two 3x3 convolutions each followed by a rectified linear unit are performed. In the end a 1x1 convolution is performed to map the final vector to the correct number of prediction classes [5].

Unsurprisingly, 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. In case of an RBG image we would have a stack of three arrays each encoding the pixel values for each colour. 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 are to pad with zeros, mirror the edges or 'wrap around the image' and use the same intensities as on the other side of the image. An illustration of a convolution can be seen in Figure 3a [6].



(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 3: 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 [6].

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 [6].

In the upsampling layers we want to perform a somewhat '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 2 is also called a *transposed convolution*, and it works by swapping the forward and backward passes of the convolution. 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 3b. Although this is not the most efficient way of performing a transposed convolution, it gives an intuition of what is happening [6].

When we in this project want to train a model, we want its input to be the two-photon microscopy images of mice brains and we want the outputs to be the locations of vescicles puncta. To train the model we thus

need an annotated dataset, 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. Our data is split into a training, validation and test set such that we can compare models and tune hyper-parameters on unknown data (the validation set) and we can measure the models' unbiased performance (the test set). For optimization we will use the gradient descend based learning algorithm *adaptive moment estimation* also known as *Adam*. Adam implements three important improvements to traditional gradient descend. 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 [7].

2.3 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 images, the prediction we make and the mask to the predicted image. If we define image 1 as X and image 2 as Y the actual formula is: DICE $= \frac{2|X \cap Y|}{|X|+|Y|}$ where $|X|$ and $|Y|$ denotes the number of dots in an image. Because DICE will give a measure of 0 if there are no dots in the mask, and thus will make it hard to report an accurate average DICE, we will also use a metric we will call circle count where we will count the number of dots in our prediction and the number of dots in the mask. We can then compare the two values to see whether we predict less, the same or more dots than in the mask.

3 References

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

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

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

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

[5] 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.

[6] 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.

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