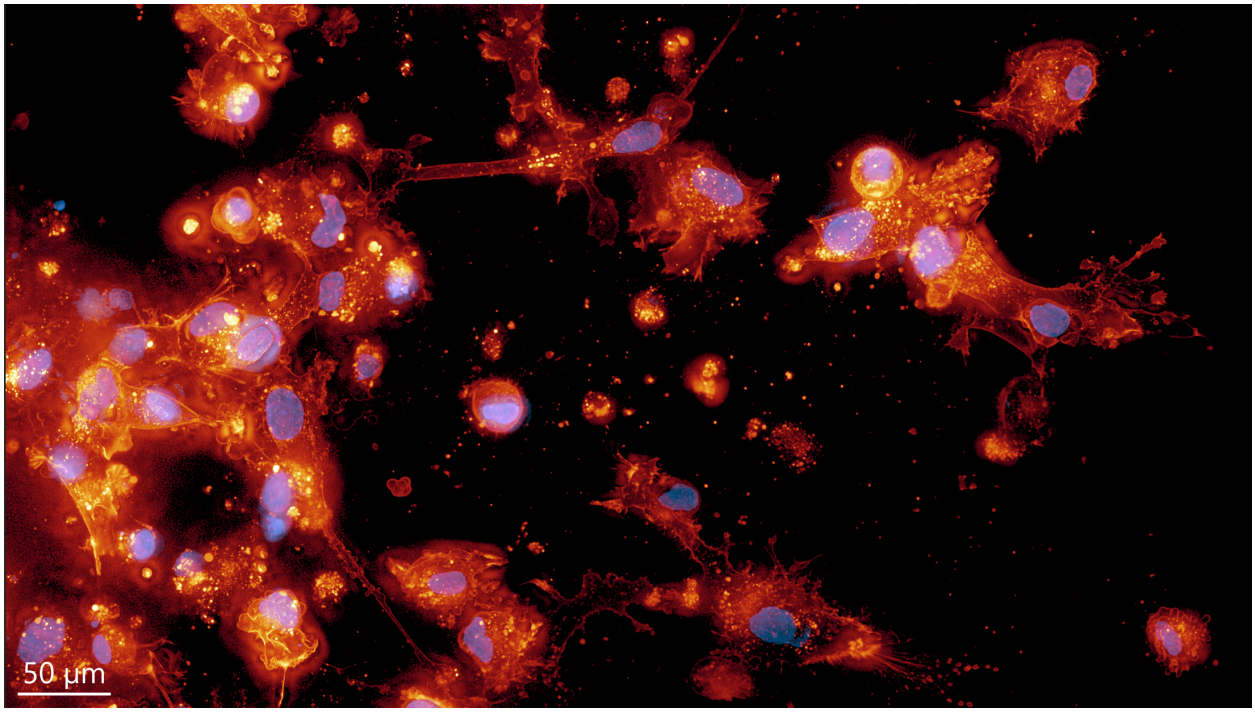
Utrecht University




# Deep learning-based segmentation of cell protrusions

## A method towards enabling better understanding of the tumor microenvironment of Diffuse Midline Glioma

Matthijs Kemp - 5299890

Utrecht University

FACULTY OF SCIENCE
Graduate School of Natural Sciences
www.uu.nl

Master's thesis 2022

# Master's thesis in Artificial Intelligence commissioned by the Princess Maxima Center

Matthijs Kemp


Faculty of Science
*Graduate School of Natural Sciences*
Utrecht University
Utrecht, the Netherlands 2022

Supervisor: Michiel Kleinnijenhuis, Rios Group, Princess Maxima Center
First examiner: Tim Ophelders, Utrecht University
Second examiner: Alex Telea, Utrecht University

Faculty of Science
Graduate School of Natural Sciences
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
Telephone +31 (0)30 253 35 50

Commissioned by:
Princess Maxima Center
Research division
Rios Group
Heidelberglaan 25
3584 CS Utrecht
Telephone +31 (0)88 972 72 72

Cover: Diffuse Midline Glioma cells under a microscope, scalebar 50 µm. Imaged on the Leica Thunder microscope, using widefield microscopy with objective 40x.

Typeset in LaTeX

# Abstract

Recently it was discovered that brain tumor cells exhibit tumor microtubes, long membranous tubes that provide a physical connection between neighboring tumor cells which can be harnessed to communicate or facilitate invasion. Cell segmentation and tracking is essential to better study the role of tumor microtubes in the invasiveness of certain brain malignancies. This research aims to develop a deep learning model to segment tumor microtubes and track their nuclei. A supervised training method was used to train several models. The performance of these models is compared to state-of-the-art models using widely-used metrics. The method developed in this work enables higher throughput research into cell-to-cell interactions and vastly speeds up behavioral studies on tumor microtubes.



Keywords: segmentation, tracking, cancer, DMG, DIPG, deep learning.

# Acknowledgements

First off I would like to extend my sincere thanks to Michiel Kleinnijenhuis and Tim Ophelders for the daily and weekly supervision and feedback. I am also thankful for colleagues Sam de Blank and Raphaël Collot who provided me with (annotated) data and domain expertise. Thanks should also go to Ravian van Ineveld for setting up an imaging experiment on the microscope which led to the cover image. Lastly, I would like to express my gratitude towards Alex Telea, who took time out of his holiday to examine this work as a 2$^{nd}$ examiner.

Matthijs Kemp, Utrecht, July 2022

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order. Letters that are capitalized in the acronyms are also capitalized in the definitions. Plural forms of acronyms will be suffixed with a lower capital s, for example CNNs means Convolutional Neural Networks.

| | |
|---|---|
| CNN | Convolutional Neural Network |
| DIPG | Diffuse Intrinsic Midline Glioma |
| DL | Deep Learning |
| DMG | Diffuse Midline Glioma |
| ELU | Exponential Linear Unit |
| EX | mouse Embryo, X days old. |
| GPU | Graphical Processing Unit |
| GUI | Graphical User Interface |
| HO | Hyperparameter Optimization |
| IoU | Intersection over Union |
| LAP | Linear Assignment Problem |
| S mLSR-3D | multispectral, Large-Scale Single-cell Resolution 3D |
| OBS | Organotypic Brain Slice |
| OS | Overall Survivial |
| PX | Post-natal mouse, X days old |
| ReLU | Rectified Linear Unit |
| SGD | Stochastic Gradient Descent |
| SNR | Signal-to-Noise ratio |
| SOTA | State-Of-The-Art |
| STAPL-3D | SegmenTation Analysis by ParaLlelization of 3D Datasets |
| TM | Tumor Microtube |

# Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis. Single-use variables, sets and parameters are defined in the location they are used, and therefore not included in the nomenclature.

## Indices

| | |
|---|---|
| $i$ | General index |
| $j$ | Index for the layers in a network |
| $s$ | Index for iterative steps |
| $t$ | Index for time steps |

## Sets

| | |
|---|---|
| $H$ | Set of the history of iterates in optimizer |
| $H_t$ | Set of the history of iterates up to and including $t$ |
| $\mathbb{N}$ | Set of natural numbers |
| $W$ | Set of weights |
| $W_j$ | Set of weights at layer $j$ |
| $X$ | Set of input variables |
| $Y$ | Set of ground truth |
| $\hat{Y}$ | Set of predictions |
| $\theta$ | Set of model parameters |
| $\phi$ | Set of hyperparameters |
| $\nabla\ell(\theta_s)$ | Set of first partial derivatives of loss $\ell$ over parameters $\theta$ at step $s$ |

## Distributions

| | |
|---|---|
| $\mathcal{U}$ | Uniform distribution |
| $\mathcal{N}$ | Normal distribution |

## Parameters

| | |
|---|---|
| $l$ | Atrous rate |
| $p$ | Parameter for probability |
| $\alpha$ | Learning rate |
| $\gamma$ | Momentum parameter or focal parameter |
| $\epsilon$ | Parameter for numerical stability, i.e. preventing division by zero |
| $\lambda$ | Parameter for the measure of deformation |

## Variables

| | |
|---|---|
| $x$ | Input |
| $y$ | True output |
| $\hat{y}$ | Predicted output |

# Contents

# List of Figures

# List of Tables

# List of Equations

# 1

# Introduction

Diffuse midline gliomas (DMG), such as diffuse intrinsic pontine glioma (DIPG), are a rare and highly lethal type of brain malignancy in children. They are very difficult to treat, mostly due to the tumor's highly infiltrative spread. Therefore, targeting the invasive nature of DMG is a promising way to treat these children. Understanding the cellular and molecular mechanisms responsible for invasiveness is essential to achieve this.

Recently, brain tumor malignancies have been found to exhibit long membranous tubes that provide a physical connection between neighboring tumor cells which can be used for invasion. These tubes have been named tumor microtubes (TM) [1, 2, 3] and add a new level of complexity to brain tumors. TMs have been shown to form a functional network that can extend over a long distance. This network is responsible for the tumor's plasticity, resistance and heterogeneity. However, little is known about the role of TMs in defining tumor cell behavioral patterns. DMG cells exhibit heterogeneous behaviors [4], but it is not known if these behaviors are dictated by the integration of tumor cells in the TM-network.

Visualization is key to characterize the dynamic nature of living cells and pinpoint the features that facilitate invasiveness. Particularly, the methods of single-cell resolution 3D imaging enable visualization of tumor cell behavior in its native environment [5]. However, the extraction of the desired information from the resulting imaging data is often restricted by the lack of adequate computational methods for this specific purpose. Studying tumor cell dynamic behavior requires tools to reliably identify all individual tumor cells (instance segmentation) and follow all these objects over different timepoints (tracking). Although tracking the tumor cell nuclei is already a demanding yet worthwhile task, the biggest challenge lies in segmenting the protrusions surrounding the nuclei. Accurate identification of cell-to-cell connections through segmentation of protrusions would enable in-depth characterization of the structure and dynamic properties of TM-networks, allowing better understanding of the infiltrative capacities of DMG tumors.

DMG patients have a poor prognosis. Although clinical trials have been going on for years, no therapeutic strategies other than radiotherapy exist. Radiotherapy is not a very successful treatment of DMG, with patients having a median overall survival (OS) of 11.2 months [6].

In this research we develop deep learning (DL)-based tools for accurate segmentation and tracking of tumor cells exhibiting TM over long distances, with the final goal of creating a dynamic graph of tumor cell connectivity that can be related to different behavioral patterns exhibited by the tumor cells.

The Rios lab[1] has recently developed 8-color, multispectral, large-scale single-cell resolution 3D (mLSR-3D) imaging and image analysis software for the parallelized, DL–based segmentation of large numbers of single cells in tissues, called Segmen-Tation Analysis by ParaLlelization of 3D Datasets (STAPL-3D) [5]. However, algorithms geared towards segmenting extensive protrusions are not yet featured in this pipeline. This research aims to apply and adapt state-of-the-art (SOTA) DL approaches for astrocyte segmentation and methods for single-cell tracking and make them suitable for brain cell segmentation integrated into the STAPL-3D pipeline.

[1] `https://research.prinsesmaximacentrum.nl/en/research-groups/rios-group`

# 2
# Related work

In 2006, 29 clinical studies performed in DMG patients were reviewed, ranging from 1984 to 2005, including a total of 973 patients [7]. Most of these studies were non-randomized. This makes for a difficult comparison as the inclusion criteria were ill defined. The authors did find that the use of hyperfractionated radiotherapy, pre-irradiation chemotherapy, concurrent chemo-radiotherapy, adjuvant chemotherapy, high-dose chemotherapy regimens and radiosensitizers did not increase long-term OS in DMG patients.

Since then, immunotherapy approaches have been, or are being, implemented in clinical trials [8, 9, 10], with limited success. Immunotherapy treatment poses a number of challenges as the DMG tumor microenvironment is immunosuppressive and has a debilitated immune surveillance. The immunosuppressive property of DMG is due to the absence of antigen presenting cells and downregulation of the histocompatibility complex dampening the anti-tumor response [11], which essentially renders the tumor unresponsive to the use of immunotherapies alone. One way to circumvent this problem is to combine adjuvant therapies with immunotherapies. This combination has the potential to elicit anti-tumor responses [12].

To date, little research on the dynamic tumor microenvironment has been conducted, even though new insights are essential to developing successful therapeutic strategies. Research into the role of TM in defining tumor cell behavioral patterns is, amongst others, needed to better understand this microenvironment. To facilitate such research, tracking and segmentation of cells exhibiting TM is needed so that their behavior can be logged and studied.

In recent years, various algorithms have been created to segment and/or track cells [13]. These algorithms were made in response to the need for quantification of the large amounts of image data generated by microscopy. Quantification of image data would previously require counting and segmenting cells manually.

Specialized software would be written on a case-by-case basis, as different laboratories focus of different aspects of cellular data. Learning-based methods are often used to eliminate the need for specialized software. Common solutions of this type include ilastik [14] and WEKA [15]. In more recent times, deep learning as a learning-based method has gained more traction. Its main advantage being the automatic extraction of optimal image features and therefore eliminating the need for human expert feature design [13, 16, 17].

This brought upon a breakthrough in biomedical image analysis through deep convolutional neural networks (CNNs), specifically the U-Net architecture [13]. Deep learning-based image processing is the fastest available method. Its adaptive nature further improves the performance and robustness.

However, to study the dynamic tumor cell behavior, cell segmentation alone is not sufficient. Behavior is exhibited over time, not in a single timepoint. Tracking of segmented cells is required to study this dynamic cell behavior. To do so, conventional object tracking algorithms exist [18, 19, 20]. These algorithms have been successfully applied to live-cell imaging data. However, limitations exist. Cell tracking is an arduous task, as cell division and cells entering and leaving the field of view often lead to erroneous tracks. In addition, a single segmentation error can render its tracking unfit for analysis.
Recent works [21, 22] proved the capability of DL-based methods to track single dividing or non-dividing cells over time after segmentation. These methods are far more generalizable than traditional tracking algorithms and therefore require less parameter setting and manual intervention. Deep-learning based tracking methods will be tested on DMG cells to assert their performance.

# 3
# Methods

The goal of this work was to segment and track cells forming a TM-network, in order to better understand their behavioral patterns. Patterns of TM can then be studied to determine their role in tumor cell behavior. This segmentation task consisted of two parts, as segmentation of the nucleus as well as the cell membrane (that forms the TM) were needed. Thereafter, each cell nucleus needed to be tracked through time and space. Additionally we presented a data protocol as the results of an investigation into how to obtain data that is compatible with the proposed method. The goal beyond the scope of this work is to gather information about events of cell-to-cell connections which is critical to get a full understanding of behavioral implications, as these events could be related to intra-cell communication.

## 3.1 Data

The data used in this research is confocal laser scanning fluorescence microscopy data. All image experiments were done on the ZEISS LSM 880 confocal microscope. The objective lenses used are 10x and 25x. For the 10x objective the numerical aperture was set to 0.45 and the voxel size to $x = 0.83$ µm, $y = 0.83$ µm and $z = 3.23$ µm for each of the spatial dimensions. For the 25x objective the numerical aperture was set to 0.8 and the voxel size to $x = 0.332$ µm, $y = 0.332$ µm and $z = 1.21$ µm. Time interval was set to 10 minutes in all cases. Images were captured in five dimensions. Three of those dimensions are the spatial dimensions of height, width and depth. The remaining dimensions are the number of channels captured and the temporal dimension, i.e. how many frames there were captured. The datatype is 16-bit unsigned integer, meaning we capture values in the range of $\mathbb{N} \cap [0, 65536]$. These datasets were created from the live imaging of mice. The procedure of obtaining tissue starts with plugging C57Bl/6 mice to allow for pup collection. Pregnancy was confirmed after the embryo was 10 days old (E10) and the birth of the litter was checked from E19-E21. At 9 days post-natal (P9) a small number of pups ($n = 1 \vee n = 2$) were collected from the litter for the organotypic brain slice (OBS) protocol. The mice were initially euthanized via $CO_2$ before being decapitated by scissors to ensure euthanasia. After decapitation brain samples were collected. Using a Vibratome with brains embedded in agarose, the brain was sliced into sagittal sections including the midbrain areas. The slices were 350 µm thick. The obtained slices were put in culture for 15 days whereafter a patient-derived tumor spheroid was implanted in each slice. These slice were then prepared for endogenous fluorescence in which fluorescent proteins are embedded into the cells. The tissue is then

ready for image acquisition on the microscope.
A visual representation of this protocol is illustrated in Figure 3.2. The time-lapse imaging that is generated in step 4 of this protocol was used in this work. One frame of this time-lapse, with cells exhibiting tumor microtubes, is shown in Figure 3.1. In this figure, the nuclei are shown in green and the cell membrane in red.

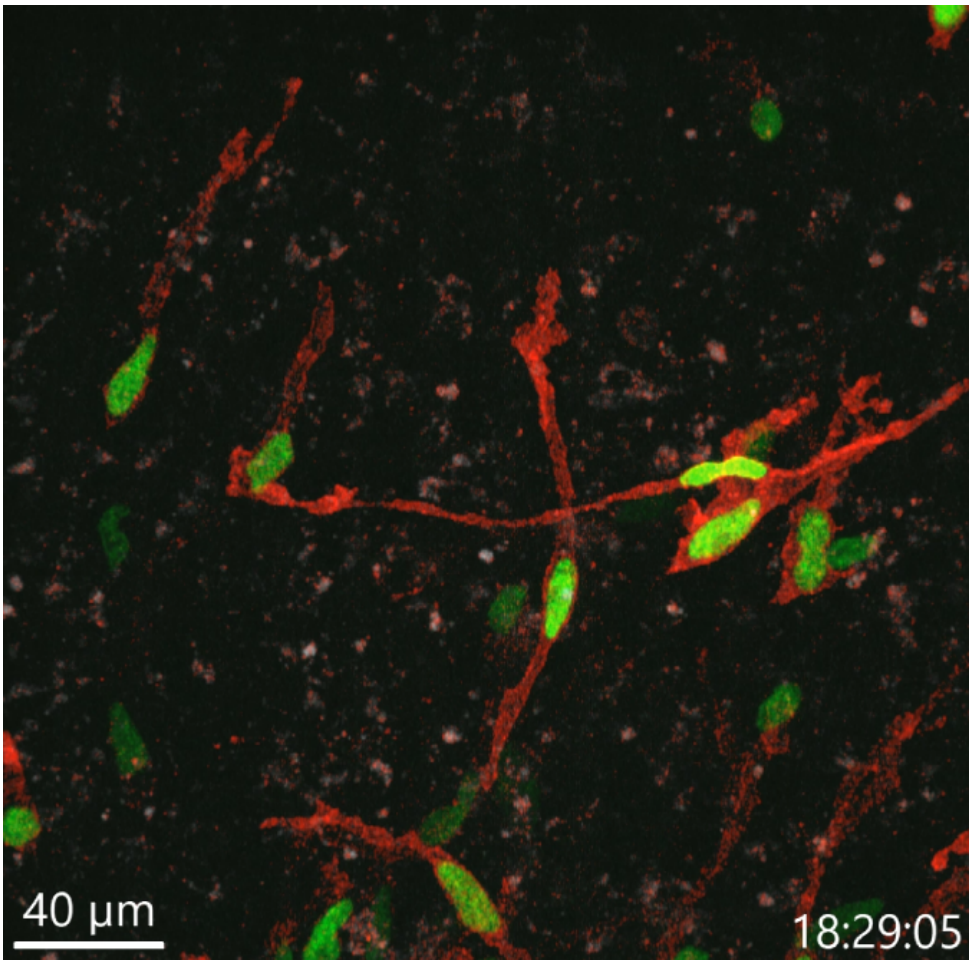



**Figure 3.1:** Image of brain cells exhibiting tumor microtubes in red and their nuclei in green. The fluorescent proteins used to color the nuclei and membranes are H2B-mNeonGreen and mScarletl-CAAX respectively. H2B is a histone protein and CAAX is a hydrophobic motif. The image is processed such that the contrast is adjusted (range unknown) and smoothed (smoothing factor unknown) for visual clarity.

## 3.2 Ground truth generation

To establish a ground truth set for the DMG cell membrane channel, a labeled dataset of microglia cells stained with an ionized calcium-binding adapter molecule 1 (IBA1) marker was used to train a 3D U-NET model. Leveraging the fact that microglia exhibit similar structures as DMG protrusions, the trained U-NET model

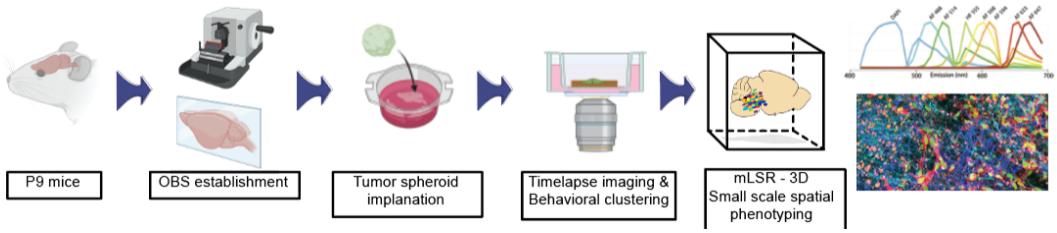
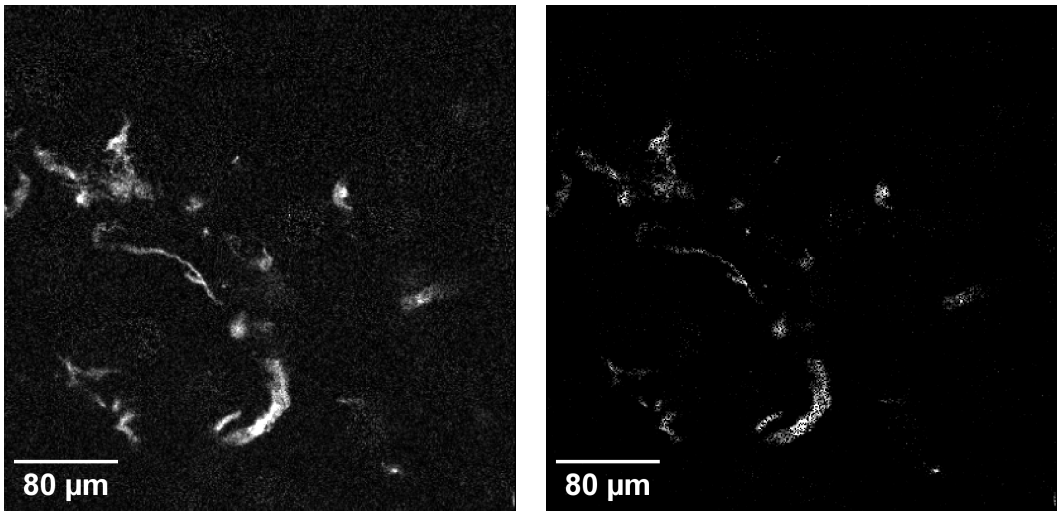



**Figure 3.2:** Protocol for going from P9 (9 days postnatal) mice to image data. Data for this proposed research is generated in step 4 of this pipeline.

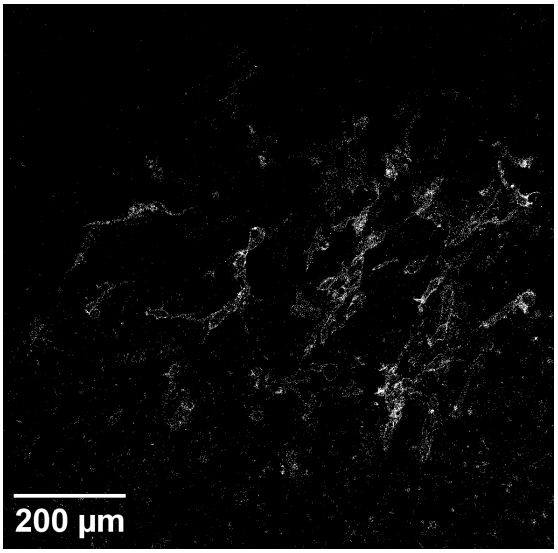
can be used to generate a mask. This mask can be manually corrected where needed to the end of obtaining a ground truth.
There are some innate differences between microglia cells and DMG cells as well as characteristics of the data sets. To bridge some of these differences, image augmentation methods were used. A combination of Gaussian noise ($\mu = 0.0$, $\sigma = 0.001$), brightness noise (scaling factor sampled from $\mathcal{U}[-0.4, 0.1]$) and pepper noise ($p = 0.05$) made the two datasets look visually comparable. Figure 3.3 shows the microglia cell data before and after augmentation. Notice that the level of noise, the brightness of the signal and overall pixel intensity match the DMG data in Figure 3.4a after augmentation. The model was then trained on the augmented microglia data.



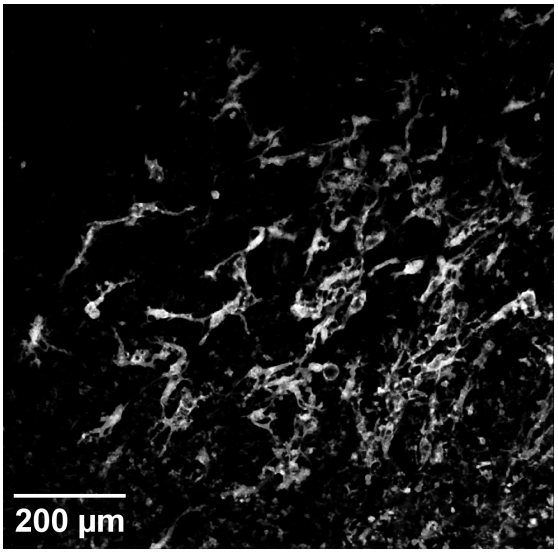

**(a)** Microglia **(b)** Augmented image of microglia

**Figure 3.3:** Unaugmented and augmented 2D slices of microglia data. For visual clarity the flipping and rotation augmentations were not applied.

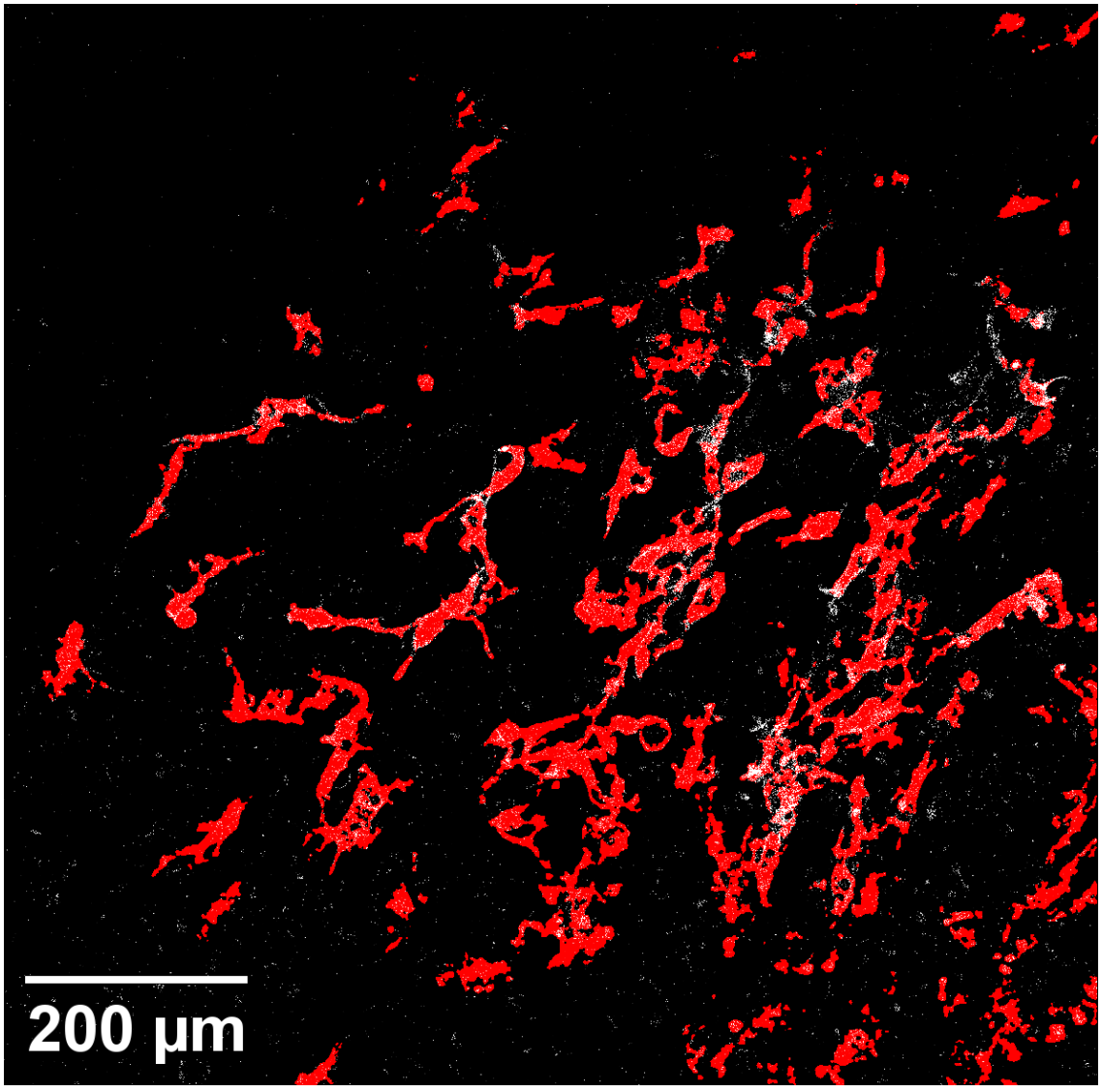
To be able to use the model for inference on the DMG data, the data needed to be sliced up in tiles to fit the available memory of the GPU (NVIDIA Quadro RTX 6000 24GB). Tiles of size $320 \times 320 \times 32$ in the dimensions of $x, y$ and $z$ were sliced. To avoid artifacts that could occur near the borders of the input images, an overlap of 10 pixels in the $x$ and $y$ dimensions and 2 in the $z$ dimension was added. The resulting tiles were stitched back together minus the overlap.




(a) Original data




(b) Pixel-wise probability map




(c) Thresholded probability map in red overlain on the original data

**Figure 3.4:** Results the ground truth generation where in **(a)** we see an example of a 2D slice of the data. In **(b)** we see the probability map obtained from the model. Figure **(c)** combines figure **(a)** with a thresholded probability map show in red. Threshold value = 0.225.

After having obtained the predictions on the DMG data, post-processing was applied. As the result of the prediction is an image with pixel-wise probabilities (normalised to values between 0 and 1), a threshold needs to be set to obtain a binary mask. Visual inspection of the pixel-wise probabilities on several regions of the image indicated that the threshold value would lie between 0.2 and 0.3. Masks obtained with threshold values of 0.2, 0.225, 0.25, 0.275 and 0.3 were overlain on the original image, of which the value of 0.225 was the best trade-off between capturing all of the signal and capturing the least amount of noise. In addition to thresholding, a minimum size filter was applied to remove some of the noise that ended up in the mask due to the relatively low threshold value. The filter size was set to filter any volume smaller than 1000 pixels. After obtaining the masks for all of the tiles for all the timepoints, the overlapping pixels were removed and all tiles were stitched back together. A 2D slice of the result can be seen in Figure 3.4. Because the mask contained mistakes, we cherry picked regions where the mask matched the data and used them for training.

## 3.3 Preprocessing

After obtaining a ground truth, the data was prepared for training. Data augmentation operations were implemented to increase the number of training data points as well as ensure robustness of the model against variance with respect to contrast, brightness, resolution and signal-to-noise ratios (SNR). The described robustness was attained with the data augmentation operations of flipping, cropping, translation and several types of noise. Flipping was implemented with a probability of $p = 0.25$ over any of the three spatial dimensions. The cropping operation was always performed as cropping manages our memory requirement by ensuring each datapoint fits in memory. Additionally it trained the model on structures that might be partially cropped out. As proposed in [13] and [23], slight elastic deformations were implemented to make the model invariant against biologically plausible deformations without having them well-represented in the training data. Implementation of these deformation was done via Elasticdeform [24], a python repository to deform N-dimensional images. Deformation grids of 3, 4 and 5 points were used, with a measure of deformation $\lambda \in \mathbb{N} \cap [3, 10]$. The probability of deformation was set to $p = 0.5$.

In addition to augmentation we applied normalization to the data, specifically percentile min-max normalization which scales all values to the range of $[0, 1]$. Regular min-max normalization is highly sensitive to outliers as you divide every datapoint by the maximal value in the dataset. In our case, using 16-bit unsigned integers, that maximal value is $2^{16} - 1 = 65535$. However, these maximal values are exclusively observed in noise. We circumvented normalizing on noise by using the $99^{\text{th}}$ percentile as our max value. However, dividing our data by the $99^{\text{th}}$ percentile would result in values larger than one. We brought our values into the desired range by clipping all values larger than 1 to 1. Since our data consisted of unsigned integers we did not need to check for values below 0. Clipping values above 1 changes the mean and variance of the data, which is not a problem since we assumed no Gaus-

sian distribution in our data.

Let $N$ be the ordered list of pixel values $X$. Let $P$ be the percentile. Then, using Equation 3.1a, we calculated the $P^{\text{th}}$ percentile index $n$. We ensured $n \in \mathbb{N}$ by utilizing the floor function, indicated by the square brackets in Equation 3.1a. Let $N_n$ be the value in $N$ corresponding to index $n$. Now having obtained the $P^{\text{th}}$ percentile value, for every $x_i \in X$, we used Equation 3.1b to rescale the value. After having rescaled every value in $X$, we clipped all values above 1 to 1 using Equation 3.1c.

OrdinalRank(N, P)

$$n = [\frac{P}{100} \times N] \tag{3.1a}$$

Normalize($N, n, X$)

$$f(x_i) = \frac{x_i}{N_n} \tag{3.1b}$$

Clip($x_i$)

$$f(x_i) = \begin{cases} 1 & \text{if } x_i > 1 \\ x_i & \text{otherwise} \end{cases} \tag{3.1c}$$

## 3.4 Model architecture

The SOTA cell segmentation model is currently 3D U-NET [23], depicted in Figure 3.5. 3D U-NET can be defined in two paths. The first path is a contractive (encoder) path, the second being a expansive (decoder) path. In the contractive path, the network starts with a series of convolutional blocks using two $3 \times 3 \times 3$ convolutional layers, batch normalisation and ReLU (Rectified Linear Units) activations. These blocks effectively downsample the input, while doubling the number of feature channels. Between every convolutional block, a $2 \times 2 \times 2$ max pooling operation is inserted with a stride of two in every dimension. In the expansive path, blocks consist of a $2 \times 2 \times 2$ up-convolution with a stride of two in every dimension followed by the same double convolution as the contractive path employs. These up-convolutions upsample the feature maps back to the original input dimensions. Additionally, a skip connection is implemented between layers in the contractive and expansive paths of equal resolution. The final layer consists of a $1 \times 1 \times 1$ convolution that reduces the output to the number of channels.

To establish a baseline performance, a 3D U-NET was trained and the resulting model evaluated. Implementation was done in Tensorflow's Keras [25]. The implementation was done as per the authors specification. The model was trained for as many iterations as training data points, which equated to 27500 iterations in our case. Computation time was just under 60 hours, or 2.5 days.

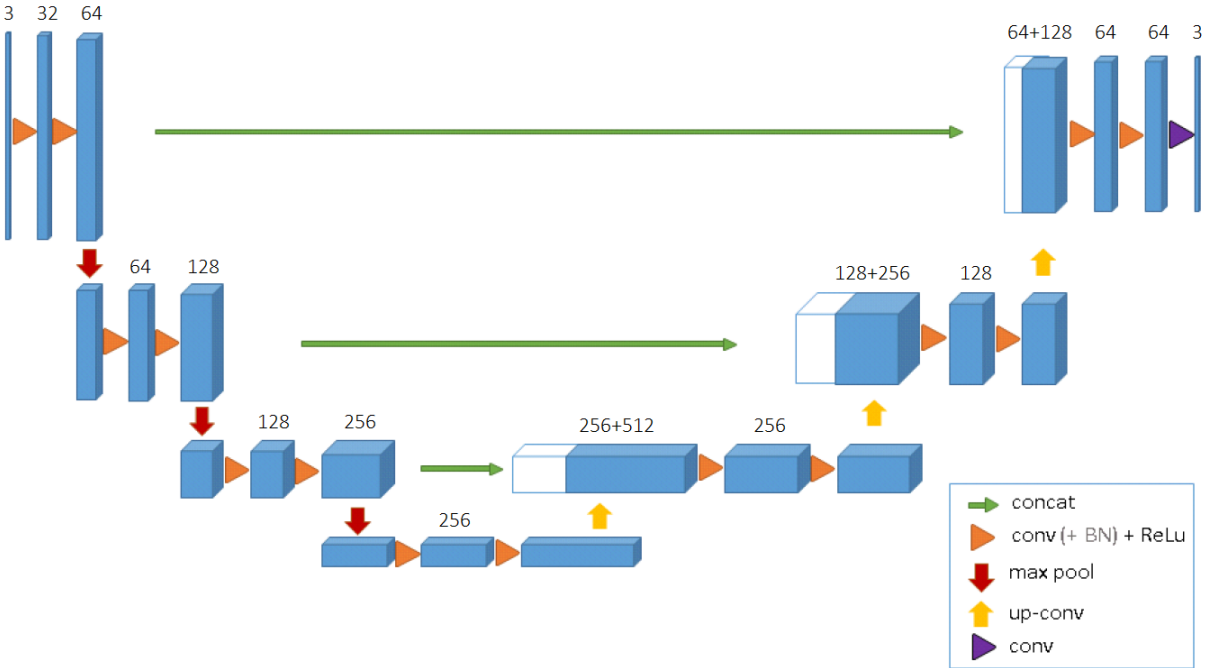



**Figure 3.5:** The 3D U-NET architecture. The blue boxes are the feature maps in every layer. Above every feature map is the number of channels. Adopted from [23].

## 3.5 Hyperparameter optimization

After training the 3D U-NET model, an effort was made to see whether any performance could be gained by hyperparameter optimization (HO). The most common HO algorithms are grid search and random search [26]. These strategies are, however, naive approaches. The grid search takes a grid of parameters and values and tries every possible combination, resulting in an exponentially increasing number of evaluations. Therefore, grid search becomes too computationally demanding in higher dimension parameter spaces. The other naive approach, random search, randomly samples possible combinations. Because these combinations are sampled from the entire parameter range, in stead of a grid, more distinct values are sampled. This increases the chance of finding an optimal value.

Bayesian optimization methods [27, 28, 29] have been developed in an effort to increase search efficiency with respect to naive methods. This family of methods focuses on identifying good configurations faster than naive methods by adaptively selecting configurations from the parameter space. Although Bayesian optimization methods outperform random sampling, *Li et al.* [30] found the increase in performance to be marginal. In their work, they also propose a novel algorithm called Hyperband. The idea behind Hyperband is to select a resource, epochs for example. The resource is then divided and allocated to randomly sampled configurations. Every configuration is used in training a model, where early stopping stops poorly performing training cycles, and more resources are allocated to well performing training cycles.

To be able to employ the Hyperband algorithm, a hyperparameter space needs to

**Table 3.1:** Hyperparameter optimization search space. Note that the activation function does not pertain to the output layer's activation function.

| Parameter | Search space |
|---|---|
| activation function | ELU, LeakyReLU, ReLU |
| kernel initializer | He normal, He uniform, zeros, ones |
| learning rate | [1e-5, 1e-2] |
| optimizer | Adam, Stochastic Gradient Descent |
| - Adam: $\beta_1$ | [1e-2, 0.9999] |
| - Adam: $\beta_2$ | [1e-2, 0.9999] |
| - SGD: Momentum | [1e-2, 0.9] |
| - SGD: Nesterov | True, False |
| pooling operation | Max pooling, average pooling |

be defined. As with any HO algorithm, the hyperparameters in this space need to carefully chosen by estimating their influence on model performance. Table 4.1 depicts the chosen hyperparameters and their search space. The following sections state why these hyperparameters were chosen to be tuned. Computation time for the hyperparameter tuning including the training of the model with the optimal parameters was 178 hours, or 7.4 days.

### 3.5.1 Activation functions

In the original 3D U-NET, the activation functions of the hidden layers are all ReLU [31] activation functions. While ReLU is the most popular activation function [32], it does have limitations. One such shortcoming is the dying ReLU problem where a very large gradient updates the weights (or the bias) such that the neuron becomes inactive and outputs zero for virtually every input. If large numbers of neurons output zero, backpropagation fails to update weights and the network can no longer learn.
To circumvent the dying ReLU problem, a small positive gradient can be added to prevent neurons from dying. ReLU with an additional positive gradient is called LeakyReLU, as seen in Equation 3.2b. LeakyReLU attempts to solve the dying ReLU problem by having a small slope for negative inputs. Another alternative to ReLU is Exponential Linear Unit (ELU, Equation 3.2c) in which the derivative of the negative part of the function approaches a 0-asymptote. Therefore, there is always a gradient larger than 0, albeit a small one.

ReLU($x$)

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.2a}$$

LeakyReLU($x$)

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{otherwise} \end{cases} \tag{3.2b}$$

ELU($x, \alpha$)

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases} \tag{3.2c}$$

### 3.5.2 Kernel initializers

Performance of weight initialization strategies in neural networks is governed by the activation function of the neurons. To be more concise, the type of non-linearity present in the activation function affects the choice of weight initialization [33].
Firstly, we want the weight initialization strategy to be of a non-constant nature. Consider a network initialized with constant weights $\alpha$. During the forward pass all neurons contribute equally to the cost because the output of all neurons will be $f(\alpha x_1 + \alpha x_2 + \cdots + \alpha x_n)$ where $f$ is some activation function. Therefore, every neuron will have the same gradient and update the same way.
There are many non-constant weight initialization strategies. Most common are random normal (Equation 3.3a), random uniform (Equation 3.3b), Xavier normal (Equation 3.3c), Xavier uniform (Equation 3.3d, [34]), He normal (Equation 3.3e, [33]) and He uniform initialization (Equation 3.3f). Random normal initialization is randomly sampling a normal distribution with a zero mean and small ($\approx 0.05$) standard deviation. Random uniform initialization is randomly sampling a uniform distribution where the min and max value are set to have a zero mean, conventionally -0.05 and 0.05.
Xavier uniform initialization is not applicable, since it assumes linearity of the activation function. This initialization strategy is proposed to work best for sigmoid and tanh activation functions, as these functions are approximately linear for small, zero-centered, inputs. ReLU and its derived activation functions do not meet this criteria. *He et al.* [33] proposed a small change to the Xavier initialization to make it suitable for activation functions such as ReLU and LeakyReLU as they are non-differentiable at 0 and non-linear around 0.
Let $W$ be the weights, $W_j$ be the weights at layer $j$ and $n_j$ be the number of input units in the weight tensor at layer $j$. Per mathematical convention $\mathcal{N}$ and $\mathcal{U}$ denote Gaussian normal and uniform distributions, respectively. Then, the distributions can be defined as follows.

Normal($\mu, \sigma$)

$$W \sim \mathcal{N}(\mu, \sigma^2) \tag{3.3a}$$

Uniform($\alpha$)

$$W \sim \mathcal{U}[-\alpha, \alpha] \tag{3.3b}$$

XavierNormal($n_j$)

$$W_j \sim \mathcal{N}(0, \sqrt{\frac{2}{n_j + n_{j+1}}}) \tag{3.3c}$$

XavierUniform($n_j$)

$$W_j \sim \mathcal{U}[-\sqrt{\frac{6}{n_j + n_{j+1}}}, \sqrt{\frac{6}{n_j + n_{j+1}}}] \tag{3.3d}$$

HeNormal($n_j$)

$$W_j \sim \mathcal{N}(0, \sqrt{\frac{2}{n_j}}) \tag{3.3e}$$

HeUniform($n_j$)

$$W_j \sim \mathcal{U}[-\sqrt{\frac{6}{n_j}}, \sqrt{\frac{6}{n_j}}] \tag{3.3f}$$

These distributions were coined to initialize the weights of fully connected dense layers containing neurons. Coincidentally, we can use these distributions to initialize the weights of the kernels in convolutional layers. The weight tensor will be shape of the kernel size, the number of input units will be the feature map or the input image in the previous layer, depending on which occurs in the previous layer, and the number of output units will be the feature map in the next layer. Consequently we were able to sample one of the distributions above to fill the weight tensor.

### 3.5.3 Learning rate

The learning rate of a network is of direct influence on time to convergence and therefore of significant importance. Setting the learning rate too low will increase time to convergence and increase the chance of getting stuck in saddle points (local minima). Setting the learning rate too high can result in perpetually overshooting the global minimum.

> "Determining a good learning rate [...] becomes more of an art than science for many problems." *Lu.* [35]

Each optimizer needed to be tuned separately, as the learning rate for one optimizer might not be optimal for another. Some optimizers require an initial learning rate, which will then be decayed or mutated in some way following a set protocol. Other optimizers use a constant learning rate. To this end, a somewhat wide range of learning rates was chosen. As per convention, the learning rate space follows a logarithmic scale: 1e-2, 1e-3, 1e-4, 1e-5.

**Table 3.2:** Update rules per optimization algorithm, adapted from [36]

| SGD($H_t, \eta_t$) |
|---|
| $\theta_{t+1} = \theta_t - \eta_t \nabla \ell(\theta t)$ |
| **Momentum($H_t, \eta_t, \gamma$)** |
| $v_0 = 0$ |
| $v_{t+1} = \gamma v_t + \nabla \ell(\theta t)$ |
| $\theta_{t+1} = \theta_t - \eta_t v_{t+1}$ |
| **NAG($H_t, \eta_t, \gamma$)** |
| $v_0 = 0$ |
| $v_{t+1} = \gamma v_t + \nabla \ell(\theta t)$ |
| $\theta_{t+1} = \theta_t - \eta_t(\gamma v_{t+1} + \nabla \ell(\theta t))$ |
| **Adam($H_t, \alpha_t, \beta_1, \beta_2, \epsilon$)** |
| $m_0 = 0, v_0 = 0$ |
| $m_{t+1} = \beta_1 m_t + (1 - \beta_1)\nabla \ell(\theta_t)$ |
| $v_{t+1} = \beta_2 v_t + (1 - \beta_2)\nabla \ell(\theta_t)^2$ |
| $b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}$ |
| $\theta_{t+1} = \theta_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}}+\epsilon} b_{t+1}$ |

### 3.5.4 Optimizer

Choosing the right optimizer determines the training speed and the final performance of the network [36]. Because there is no theory or protocol in finding the right optimizer, empirical studies were referenced [37] and hyperparameter tuning strategies were employed.

The optimizer algorithms chosen were Stochastic Gradient Descent (SGD), SGD with Momentum (Momentum), SGD with Nesterov Accelerated Gradient (NAG) and Adam. Optimization algorithms can be defined by the update rule which they employ [36]. Out of all optimization algorithms used for training neural networks, SGD employs the simplest update rule. The momentum algorithm is an adaptation of the SGD algorithm, which simulates momentum through the incorporation of the previous parameter update in the update rule. Table 3.2 states the update rules per algorithm, with $H_t$ being the history of iterates including the function and gradient values such that $H_t = \{\theta_s, \nabla \ell(\theta_s), \ell(\theta_s)\}_{s=0}^{t}$ where $\ell$ is a differentiable loss function, $\nabla \ell(\theta_s)$ is the vector of first partial derivatives and $\theta$ represents the vector of model parameters. Each optimizer has their own hyperparameters. In SGD, there is a learning rate scheduler $n$. Momentum and NAG have a learning rate scheduler $n$ and a momentum parameter $\gamma$, only differing in their implementation. Adam's hyperparameters are stepsize $\alpha$, exponential decay rate for 1st and second moment estimates $\beta_1$ and $\beta_2$ and a parameter for numerical stability $\epsilon$.

As stated by *Choi et al.* [36] an optimizer algorithm can be fully defined by its

update rule and hyperparameters $\phi$. Conventionally, only subsets of optimizer hyperparameters are tuned, which could lead to false conclusions regarding optimal hyperparameter settings or optimizer algorithms. Using the definition of *Choi et al.*, we can state that $\text{Momentum}(H_t, n_t, \gamma) \neq \text{Momentum}(H_t, n_t, 0.8)$ because the former has a two free hyperparameters and the latter has one. For this reason, all optimizer hyperparameters were passed to the Hyperband algorithm to be tuned.

### 3.5.5 Pooling

The idea of feature pooling is, as many operations in neural networks, inspired by biology [38]. There are a few pooling operations that are used in deep learning, mostly average pooling and max pooling. Pooling can happen in one, two or three dimensions. The goal of pooling is to achieve invariance with respect to, amongst others, spatial position, noise and brightness. *Boureau et al.* [39] concluded that the best performance of either max pooling of average pooling largely depends on the data and features. As per these findings, max pooling and average pooling were added to the search space of the Hyperband algorithm. To keep the size of the hyperparameter space within reason, all layers will were given the same pooling operation. By doing this, we increased the hyperparameter space by a factor of $2^1$ in stead of $2^3$ since we apply the pooling operation three times.

## 3.6 Loss function

The loss function cannot be tuned by the Hyperband algorithm, as this is the metric that determines the fitness of every candidate model. Tuning the loss function would be the equivalent of selecting the loss function that generally outputs the lowest values. There are several loss functions used in image segmentation, of which a few are suitable to binary classification problems [40]. The abundance of loss functions meant a selection of loss functions needed to be made. Evaluation of every loss function that is applicable to binary image segmentation would have been too time-intensive. Instead, findings of other authors in the medical image segmentation domain were used to select promising loss functions. The resulting selection consisted of four candidate loss functions (Equation 3.4).
Binary cross-entropy is derived from cross-entropy, where the target is either 1 or 0. To obtain the range of values needed for binary cross-entropy, a sigmoid activation function was employed in the final layer of the network. This activation function outputs values between 0 and 1 which ensures that both $log(\hat{y})$ and $log(1-\hat{y})$ are defined, as $log(x) \forall x \in [0, -\infty]$ and $log(1-x) \forall x \in [1, \infty]$ are undefined. The Dice Loss is adapted from the Dice coefficient, which is a metric of similarity commonly used in computer vision. It is defined by subtracting the Dice coefficient from 1. The Tversky loss is adapted from the Tversky index, similar to the Dice loss. The Tversky index is a generalization of the Dice coefficient. Through introducing a $\beta$ coefficient, the Tversky loss weighs the false positives and false negatives heavier or lighter. By weighing false positives more, the loss function attempts to minimize the occurrence of false positives. This is analogous for false negatives. The Focal Tversky loss takes the Tversky index and scales it by a factor $\gamma$. The result is a

better learning of hard examples (if $\gamma > 1$), for example with very imbalanced classes where the occurrence of a certain class is minimal. The value of this factor should lie in the range of $\gamma \in [1, 3]$ [41] where if $\gamma = 1$ the equation simplifies to the Tversky loss.
Having a degree of control over the penalization of false negatives and false positives gives us an indirect influence on the recall vs precision trade-off. This is a great benefit to have since we wanted to account for false negatives by having high recall. Predicting no cancer cell where there is cancer should occur as little as possible. The Tversky loss allowed us to do this. In stead of using the FocalTverskyLoss to focus on hard examples, we assign each class a weight. This eliminated the need to tune yet another parameter while achieving the same concept of focusing more on signal than noise.

BinaryCrossEntropy$(y, \hat{y})$

$$\ell_{BCE} = -(ylog(\hat{y}) + (1 - y)log(1 - \hat{y})) \tag{3.4a}$$

DiceLoss$(y, \hat{y}, \epsilon)$

$$\ell_{DL} = 1 - \frac{2y\hat{y} + \epsilon}{y + \hat{y} + \epsilon} \tag{3.4b}$$

TverskyLoss$(y, \hat{y}, \beta, \epsilon)$

$$\ell_{TL} = 1 - \frac{y\hat{y} + \epsilon}{y\hat{y} + \beta(1 - y)\hat{y} + (1 - \beta)(1 - \hat{y})y + \epsilon} \tag{3.4c}$$

FocalTverskyLoss$(y, \hat{y}, \beta, \epsilon, \gamma)$

$$\ell_{FTL} = \ell_{TL}(y, \hat{y}, \beta, \epsilon)^{\gamma} \tag{3.4d}$$

## 3.7 Performance metrics

To be able to compare different models we needed metrics to evaluate every model on. For this work the Tversky index (Equation 3.5a) and the Intersection over Union (IoU, Equation 3.5b) were used. The Tversky index parameters $\beta$ and $\epsilon$ were set to 0.3 and 0.7, respectively. These values emphasize higher recall. The Tversky index and IoU were chosen as they are commonly used in computer vision in addition to being blindly selected as most representative of the quality of the prediction by the researcher who imaged the data, R. Collot.

$TverskyIndex(y, \hat{y}, \beta, \epsilon)$

$$TI = \frac{y\hat{y} + \epsilon}{y\hat{y} + \beta(1 - y)\hat{y} + (1 - \beta)(1 - \hat{y})y + \epsilon} \tag{3.5a}$$

$IoU(y, \hat{y})$

$$IoU = \frac{y\hat{y}}{y\hat{y} + (1 - y)\hat{y} + (1 - \hat{y})y} \tag{3.5b}$$

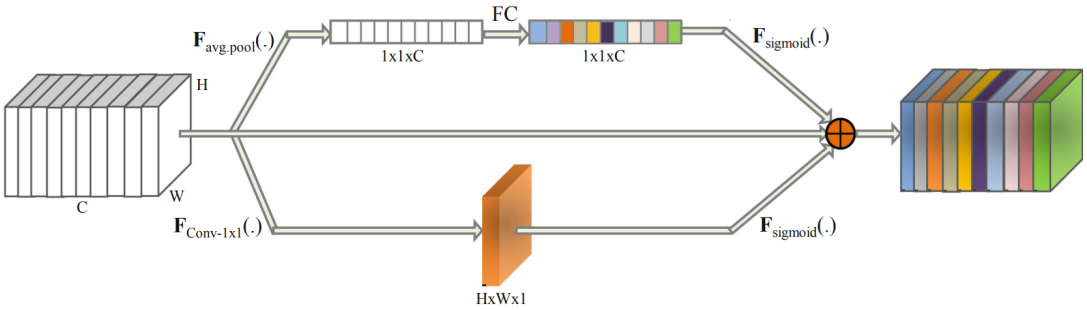



**Figure 3.6:** 3D attention module displaying the spatial and channel attention with skip connection. Adopted from [43].

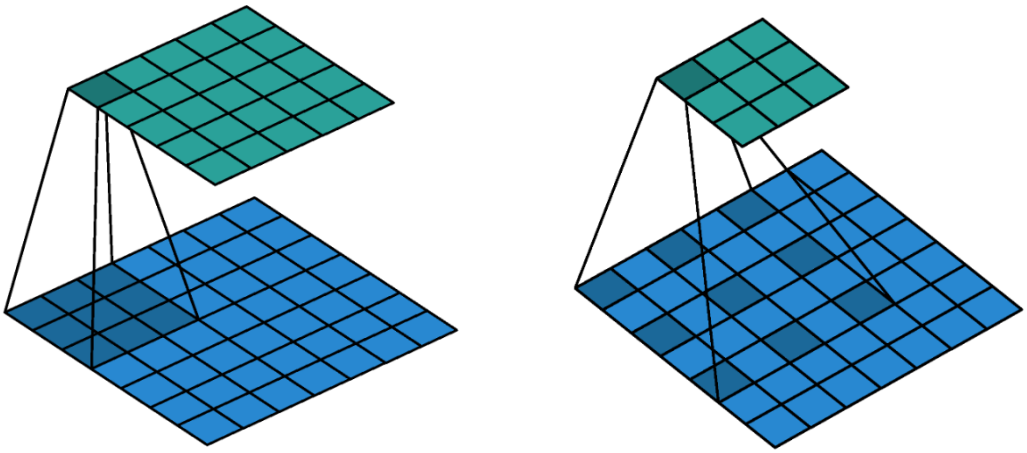
## 3.8 Model architecture variations

In addition to the 3D U-NET architecture, different architectures were evaluated. In response to the 3D U-NET paper, many works attempted to improve upon the performance by substituting the regular convolutional blocks with more advanced blocks. Even though these variations were published as improvements, (3D) U-NET (based on citations and GitHub statistics) is still the most-used semantic segmentation model in the biomedical imaging domain. Outside of the biomedical imaging domain, the DeepLabv3+ [42] method is popular method for performing semantic image segmentation. Like U-NET, a 3D version is available.

### 3.8.1 3D Attention U-NET

3D Attention U-NET [43] is a variation on 3D U-NET in which attention gates are placed in each layer of the expansive part of the 3D U-NET architecture. Attention gates generate spatial and channel attention by evaluating feature relationships between channels and spatial dimensions. The result is the trimming of features that are insignificant. This ensures that only information-rich output is passed on to the next layer.

To obtain the 3D attention map a $1 \times 1 \times C$ convolution is performed, where C is the number of channels. This convolution aggregates all spatial feature correlations to a dimension of $H \times W \times 1$ where $H$ and $W$ indicate height and width of the feature map. At the same time, average pooling is performed and the result is fed to a neural network to get the channel correlations. Additionally a skip-connection is made next to these parallel excitations. The resulting information of these three operations is summed to obtain the 3D attention map. The entire 3D attention module is visualized in Figure 3.6. The addition of 3D attention modules in the 3D U-NET architecture required no additional changes to hyperparameters, training data or others.



**(a)** $l = 1$ **(b)** $l = 2$

**Figure 3.7:** (a) Standard convolution with kernel size $3 \times 3$, stride 2, padding 1 and atrous rate $l = 1$. (b) Atrous convolution with kernel size $3 \times 3$, stride 1, no padding and atrous rate $l = 2$. Adopted from [45].

### 3.8.2 DeepLabV3+

DeepLab is a method for semantic segmentation developed by Google. The newest iteration, DeepLabV3+ [42] was made for 2D image segmentation. However, 3D adaptations exist [44].

DeepLabV3+ employs an encoder-decoder type architecture, just like 3D U-NET. In the encoder they encode multi-scale contextual information by applying atrous convolutions. In an atrous convolution, the convolution filter (or kernel) is not simply overlaid on top of the input feature map. In stead, there is an extra parameter called the atrous rate $l$ which dictates the dilation of the kernel applied on the input feature map. The atrous rate can be interpreted as sampling the input feature map every $l$ pixels. Therefore, with $l = 1$ the atrous convolution simplifies to a standard convolution (Figure 3.7).

Atrous convolution is applied at multiple scales, meaning different atrous rates are employed. This simulates sampling at different resolutions. They use the rates of 6, 12, and 18 with a kernel size of $3 \times 3$. In addition to atrous convolution, a $1 \times 1$ standard convolution and an image pooling operation are concatenated and fed to another $1 \times 1$ standard convolution. The output of this convolution then goes through the decoder part of DeepLabV3+. The decoder starts by upsampling the encoder output and concatenating the result with the low-level features that come through a skip-connection from the atrous convolutional layers in the encoder part. This concatenation is then passed through a $3 \times 3$ convolution and upsampled to the original image dimensions.

## 3.9 Nucleus segmentation

To the end of nucleus segmentation, multiple methods are available. The preferred method of the Rios lab is StarDist [46, 47]. One of the known limitations of StarDist is the poor performance on elongated or deformed nuclei shapes with respect to rounder shapes, which are prevalent in DMG brain cells. This leads to a phenomenon called over-segmentation in which one nucleus is segmented as two or more, like in Figure 4.4e.
StarDist's performance on DMG nuclei can be party explained by the method relying on star-convexity of the objects to be segmented. Star-convex shapes are shapes where there exists a point $x_0$ in the object $X$ such that a line between point $x_0$ and every other point $x_n \in X$ lies fully within the object. Some DMG nuclei are not star-convex as they tend to fold themselves into a U shape when changing directions.
The software Ilastik [14] is able to learn from sparsely annotated nuclei. This eliminates the need for ground truth data, only requiring a few sparsely annotated examples (Figure 4.4a). The reason we cannot directly employ 3D U-NET or other segmentation models here is because for tracking we need instance segmentation, while the aforementioned models perform semantic segmentation.

## 3.10 Tracking

This section lists the different tracking methods that are compared as well as their reasons for consideration. The tracking methods are an established traditional method of tracking and a machine learning algorithm. Comparing a traditional tracking algorithm with a machine learning tracking algorithm presents a good opportunity to highlight the strengths and weaknesses of each method as well as showing whether the task of tracking is better suited to a machine learning solution.

### 3.10.1 Trackmate

TrackMate [20, 48] is an open source Fiji [49] plugin that can be used for automated, semi-automated, and manual tracking of single-particles. While it incorporates segmentation pipelines, only the tracker module of the software was used for this work. The tracker module has the ability to load a labeled image and perform tracking. TrackMate offers several tracking methods for labeled images. Not every tracking method is applicable to our data, as some methods rely on overlap between timeframes or constant velocity in the objects to track. TrackMate's implementation of the Linear Assignment Problem (LAP) tracker, based on the LAP framework proposed by *Jaqaman et al.* [50], is designed for single particle tracking. It assigns a single pixel in the center of each label as a particle, after which linking costs based on the squared distance are calculated. These costs can be modulated through setting features such as shape matching, intensity matching, etc. TrackMate offers a feature to tune these feature values.
While the quality of the tracking results is most important, TrackMate is also highly integrable in any pipeline by providing incorporation of most common segmentation tools like Ilastik [14], StarDist [46, 47], Cellpose [51] and Weka [52] directly in the

graphical user interface (GUI). It is also extensible through programming languages Python and Java. TrackMate is therefore highly versatile and easy to integrate regardless of the users workflow, which is partly why it was considered for this work.

### 3.10.2 3DeeCellTracker

3DeeCellTracker [22] is a method developed to automate the entire pipeline of segmentation and tracking in 3D microscopy timelapse data. It attempts to do so by utilizing deep learning to optimize the parameters that would normally have to be re-optimized when, for instance, lighting conditions change.
3DeeCellTracker takes in 3D timelapse images and feeds them to a 3D U-Net with a watershed transformation to obtain the object identities. These object identities need to be manually corrected in timepoint 1. It then uses another transformation function (a combination of a feed forward neural network and a rigid point registration method called PR-GLS [53]) to correct all subsequent timepoints based on the manual adjustments and track the cells.
3DeeCellTracker is, to our knowledge, the only free, openly available 3D tracking method that fully relies on deep learning. This comes with a few caveats. The first being that manual correction of one intermediary step is required. The second caveat is that 3DeeCellTracker has a few tracking parameters that need to be tuned. Lastly, the authors recommend altering the architecture of the 3D U-NET inspired network that is used for their segmentation based on the imaging conditions. This requires in-depth knowledge of neural networks and the influence of the architecture on performance and decreases generalizability of the method.

# 4
# Results

Our results can be divided in semantic protrusion segmentation, nuclei instance segmentation, nuclei tracking and the data protocol.

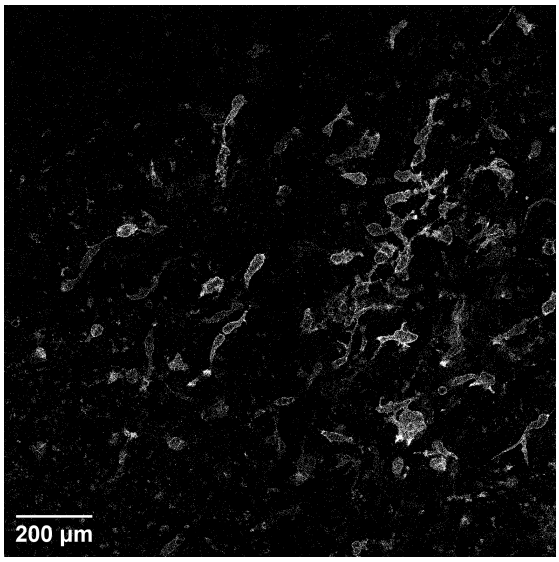
## 4.1 Protrusion segmentation

Every model proved to be usable for the task of protrusion segmentation. Differences between models pertained mostly to errors in very thin protrusions and sensitivity to noise. Performance was measured using the metrics described in Section 3.7, using a test set that was not used for training or validation. The evaluation of each model is shown in Table 4.2, where we can see the best value per metric highlighted in bold. We observe the best Tversky index and binary IoU in the tuned 3D-UNET model. We observe the worst metric scores on the DeepLabV3+ model.

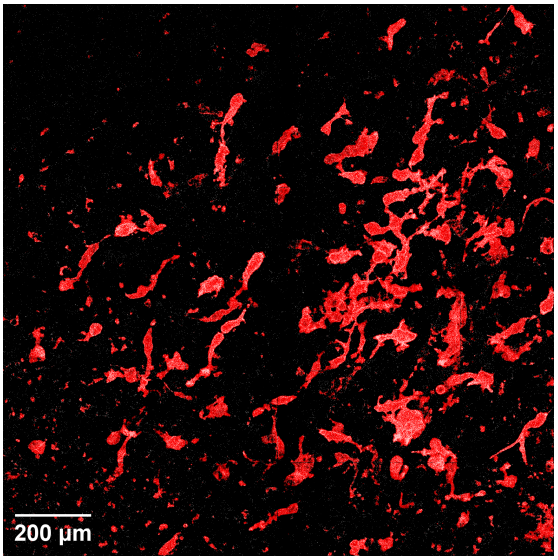
Predictions of each model can be seen in Figure 4.1. The small blobs that are most prominent in the bottom left of the figures are considered signal as they are biological material that was excreted from the cell membranes.
The HO algorithm Hyperband, used to tune the 3D-UNET model, ran 145 trials until settling on the approximate optimal values as shown in Table 4.1. Note that the optimizer parameter values and the pooling operation significantly differ from their suggested default values. Where in Adam the $\beta_1$ and $\beta_2$ are suggested to be set on 0.9 and 0.999 respectively, we observe significantly lower values. Where *Çiçek et al.* proposed maximum pooling operations, Hyperband finds average pooling to be more optimal.

**Table 4.1:** Hyperparameter optimization results. Note that the activation function does not pertain to the output layer's activation function.

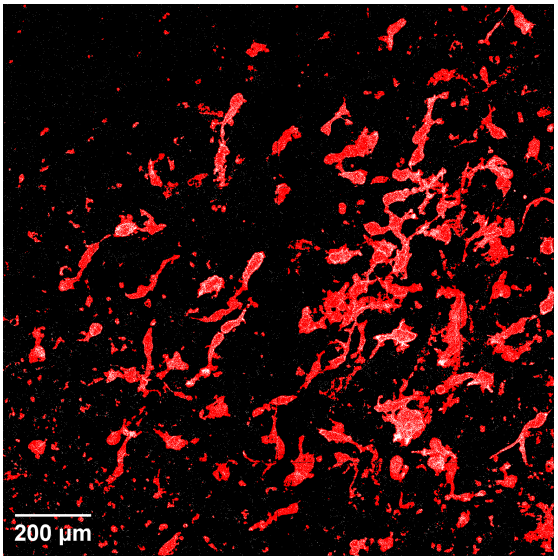
| **Parameter** | **Optimal value** |
|---|---|
| activation function | ELU |
| kernel initializer | He normal |
| learning rate | 7.29e-4 |
| optimizer | Adam |
| - Adam: $\beta_1$ | 0.317 |
| - Adam: $\beta_2$ | 0.208 |
| pooling operation | Average pooling |



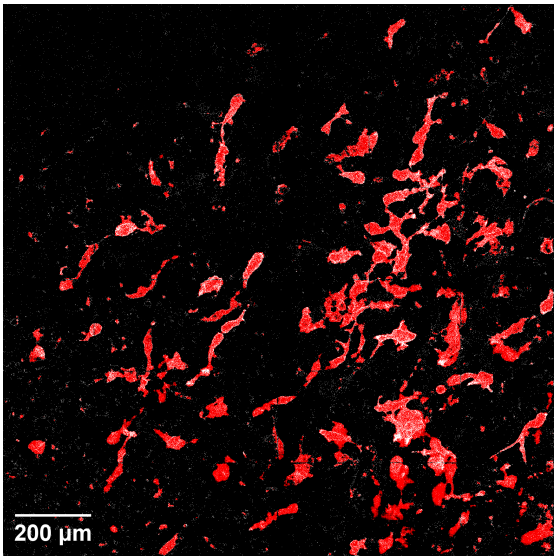

(a) Original image



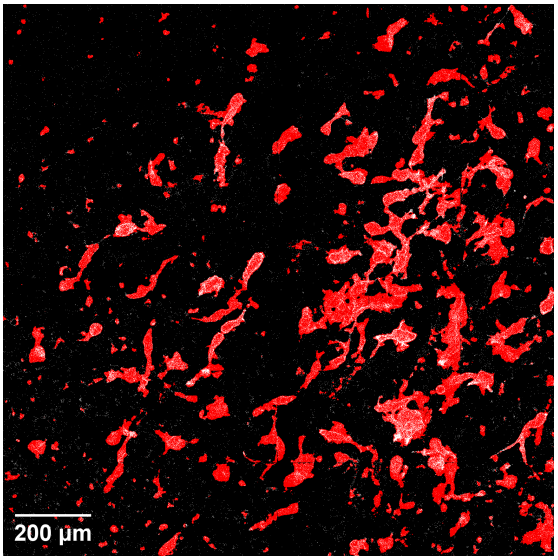

(b) 3D-UNET, threshold 0.8




(c) Hyperband, threshold 0.95




(d) DeepLabV3+, threshold 0.75




(e) 3D Attention U-NET, threshold 0.8

**Figure 4.1:** Protrusion segmentation results shown as a 2D slice of the XY-plane. In figure **(a)** we see the original image, contrast adjusted for visual clarity. The 3D-UNET model's predictions are shown in **(b)**. **(c)** contains the predictions of the Hyperband 3D-UNET model. In **(d)** we show the prediction of the DeepLabV3+ model. Lastly **(e)** depicts the 3D Attention U-NET.

**Table 4.2:** Metrics of protrusion segmentation performance per model architecture. Best score in each metric is highlighted in bold.

| Model | Tversky index | Binary IoU |
|---|---|---|
| 3D U-NET | 0.683 | 0.638 |
| 3D U-NET Hyperband | **0.821** | **0.836** |
| 3D Attention U-NET | 0.763 | 0.702 |
| DeepLabV3+ | 0.641 | 0.623 |

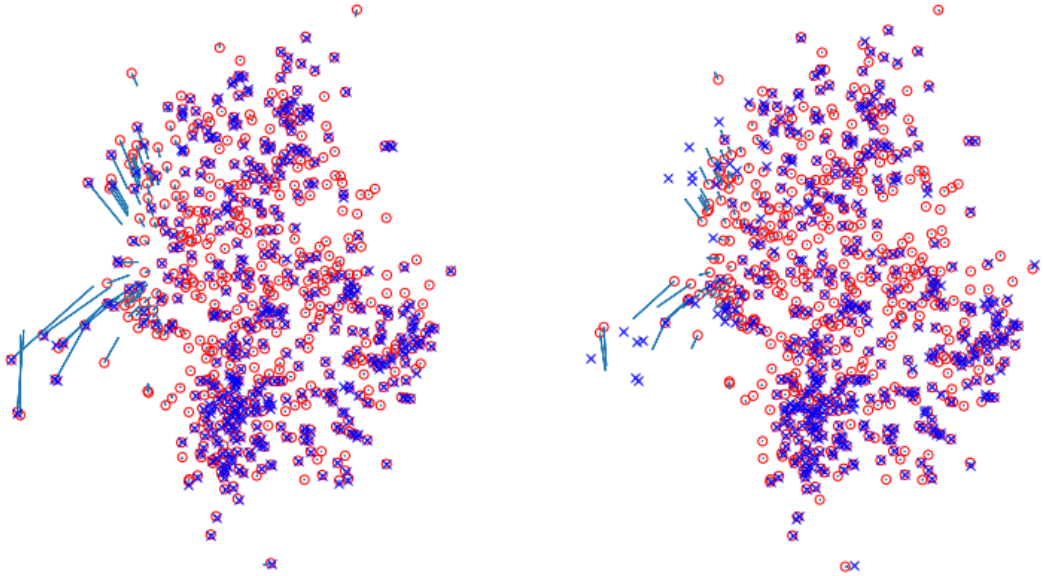
## 4.2 Nuclei segmentation

Ilastik's GUI makes annotating cells and training an object classifier relatively simple such that no coding expertise is required, nor any extensive domain knowledge. Nuclei segmentation in Ilastik required annotating a total of 40 cells over 3 timepoints. The timepoints used where the first, middle and last timepoint. The process was done iteratively, starting with the annotation a few cells and correcting mistakes in segmentation by annotating them. The results are shown in Figure 4.4. Several cells are under or over-segmented. The over-segmentation errors highlighted in Figure 4.4e are most likely caused by really dim signal in the thin center of the nuclei. The dim signal is mistakenly labeled as background and the two remaining halves of the nucleus are labeled as different instances. Under-segmentation occurs mostly in adjacent nuclei that have no hyperplane of background to separate them. By drawing a line of background between touching cells in the annotation phase, the model would start to over-segment cells in other regions as it would learn to predict background in signal if the intensity was lower at some location in the nuclei. The 2D slice depicted in Figure 4.4 was found to be representable of the whole dataset in terms of number of errors. As this method is unsupervised, no performance metrics can be computed.

## 4.3 Tracking

Tracking could then be performed using the segmented nuclei. The 3DeeCellTracker method requires a manual correction of its predicted segmentation in the first timepoint, for which we used the segmentation results from Ilastik. As seen in Figure 4.2 quite a lot of cell positions were adjusted. Using our data, the 3DCellTracker method often predicts cells from a certain area to co-migrate towards a specific spot. Adjusting the parameters $\beta$ and $\lambda$ which control coherence/independence of cell positions resulted in less severe convergence of cell positions. Additionally the number of tracking iterations was increased to 10 to potentially increase the tracking accuracy.

After optimizing the parameters, the results from Figure 4.2 were obtained. In this figure we see the convergence problem is still present. In addition to wrongly predicting this co-migration, the distance that it is predicting is too large. As 3DeeCellTracker has no parameter for setting a maximum distance between two timepoints, we were unable to correct this issue. On the left side of Figure 4.2a we notice a lot



**(a)** Timepoint 0. Elapsed time: 00:00:00 **(b)** Timepoint 1. Elapsed time: 10:00:00

**Figure 4.2:** 2D slice of the tracking results of the 3DeeCellTracker method in which red circles depict nuclei, blue crosses indicate the cell position was adjusted based on the manual correction of the segmentation, blue lines indicate the predicted track with respect to the next timepoint.

of blue crosses but no accompanying red, indicating some cells were no longer found.

Tracking with TrackMate required inputting the segmentation obtained from Ilastik. Results of the TrackMate method can be found in Figure 4.3. We observe less obvious tracking mistakes when compared to 3DeeCellTracker. Mistakes in TrackMate mostly pertain to segmentation mistakes, namely under-segmentation causing tracks to merge. TrackMate was set to a maximum migration distance of 50 µm, using the LAP tracker. A larger size image of the final timepoint can be found in Appendix A.1

Note that TrackMate was significantly faster in computation time when compared to 3DeeCellTracker. Where 3DeeCellTracker took around 15 minutes per timepoint, TrackMate took 1 second per timepoints.

## 4.4 Data protocol

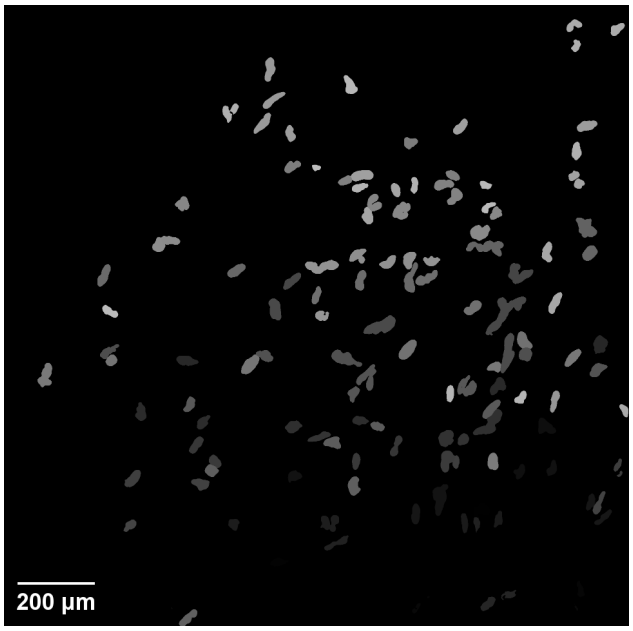
To ensure usability of the proposed method, a data protocol was established. Not all microscopy methods and imaging experiments are compatible with the proposed method. This chapter aims to describe the degree of generalizability over different imaging settings when doing imaging experiments.

### 4.4.1 Data collection

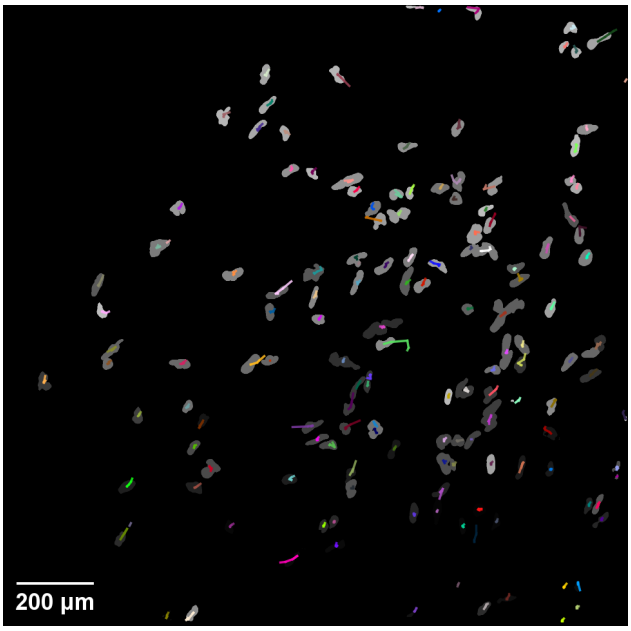
For imaging experiments, guarantee of usable results can only be given on laser scanning confocal microscopes. Use an objective of 10x or 25x. For the 10x objective the numerical aperture was 0.45 and the voxel size to $x = 0.83$ µm, $y = 0.83$ µm and $z = 3.23$ µm. For the 25x objective the numerical aperture was 0.8 and the voxel size to $x = 0.332$ µm, $y = 0.332$ µm and $z = 1.21$ µm. Deviating from these objectives might result in degraded performance. Imaging data used in this method was 16-bit. In theory, 8-bit and 32-bit data can be used as the preprocessing dictates normalization to the range of $[0, 1]$. However, going 8-bit means imaging at a lower quality, which could impact performance. Because of the augmentation steps implemented during the training phase of the protrusion segmentation, SNRs do not influence performance, given noise levels are within reason. A good concept to remember when judging fitness of the produced data for this method is "garbage in, garbage out", which is often reiterated in the ML community.
Tracking performance was calculated on datasets with timesteps of 10 to 15 minutes apart, which dictates a certain disparity of position between frames. Using smaller windows between frames is less influential on tracking performance than using larger windows.
The method for obtaining mouse tissue containing DMG cells is described in chapter 3.1.

### 4.4.2 Data preprocessing

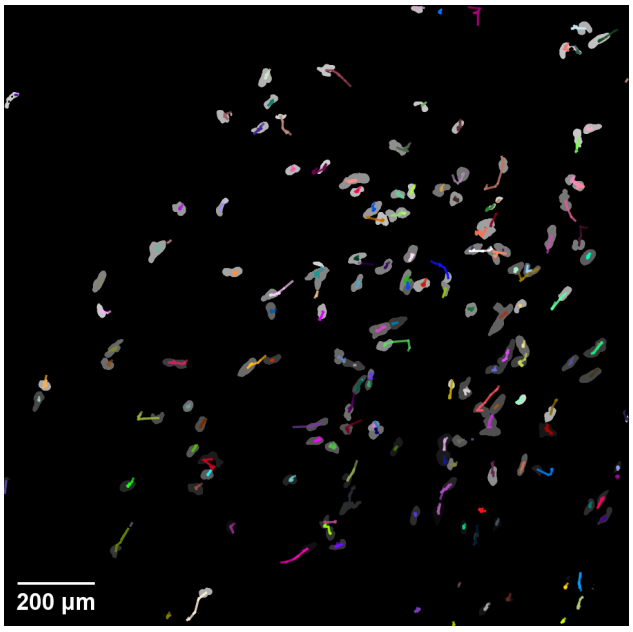
Most preprocessing steps are done automatically because of the implementation of the method into the STAPL3D software. Keep in mind that $99^{\text{th}}$ percentile min-max normalization might not be suitable to all data. One can usually find a suitable percentile experimentally by looking at the histogram of observed values.



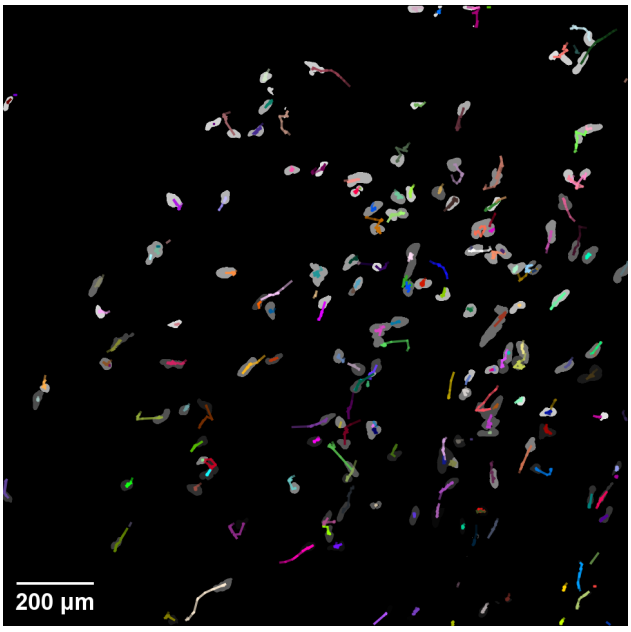

(a) Timepoint 0. Elapsed time: 00:00:00




(b) Timepoint 4. Elapsed time: 00:40:00

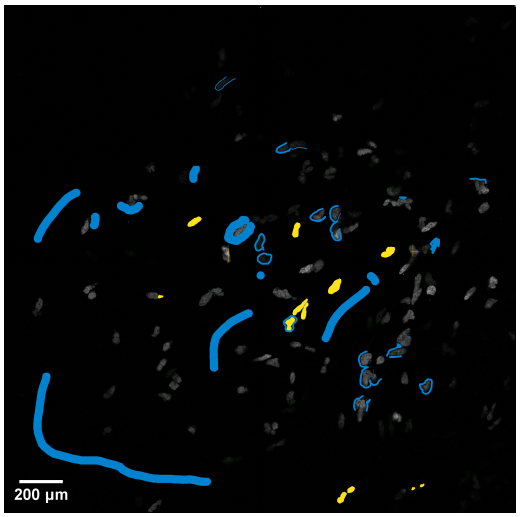



(c) Timepoint 9. Elapsed time: 01:30:00

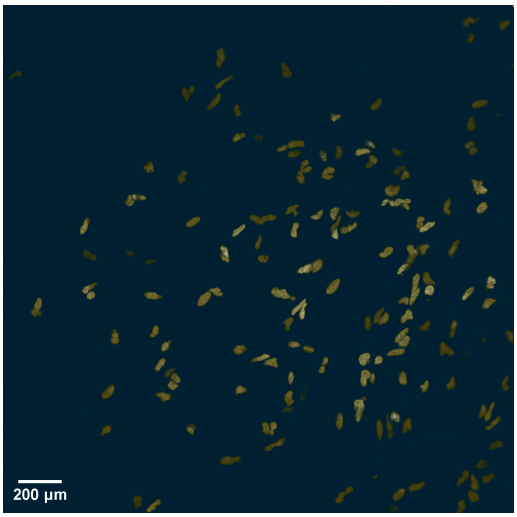



(d) Timepoint 14. Elapsed time: 02:20:00

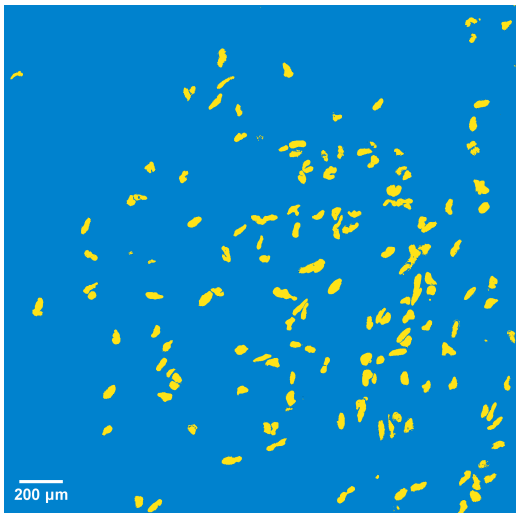
**Figure 4.3:** Tracking results obtained from TrackMate at four different timepoints, shown as 2D slices. Each color represents the tracking of a nucleus. Note that while nuclei might disappear because they change z-plane, tracks are still drawn.



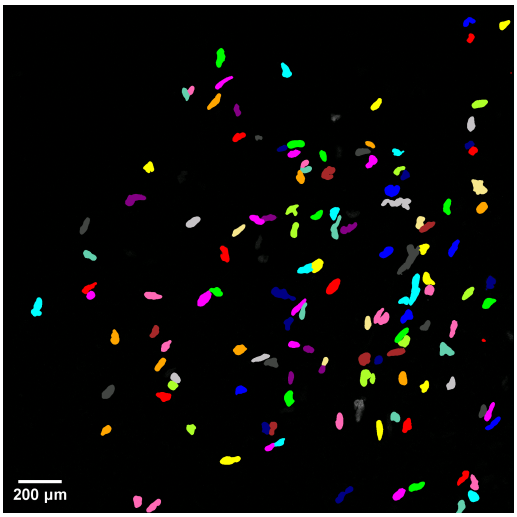

(a) Sparsely annotated cells




(b) Class-wise probability




(c) Foreground-background segmentation




(d) Instance segmentation


(e) Examples of over and under-segmentation

**Figure 4.4:** Nuclei segmentation results, shown as a 2D slice of the XY-plane. In **(a)** we see the sparsely annotated cells in yellow and background in blue. Figure **(b)** shows the computed probability map based on the annotation. This probability is then thresholded to obtain mask **(c)**. Using hysteresis thresholding, the final instance segmentation results **(d)** are obtained. Examples of mistakes are highlighted with arrows in **(e)**.

# 5
# Discussion

The results indicate that 3D U-NET with optimized hyperparameters performs the best in the task of semantic segmentation of DMG protrusions. This is in line with the relevant literature [13, 23, 30] discussed in the related work chapter. Due to time constraints HO could only be performed on one model, which was chosen to be 3D U-NET. Therefore, one could argue that other models which did not have their hyperparameters optimized might perform better when optimized with Hyperband. If we make the naive assumption that all models would benefit equally from hyperparameter tuning, 3D Attention U-NET would outperform the tuned 3D U-NET by 4% on the Tversky index. We speculate that we observe the lowest scores with the DeepLabV3+ model because it was designed to perform well on the semantic segmentation benchmarks. In contrast, U-NET derived architectures were designed to work with sparse data and a low number of classes specifically within the medical imaging domain. Applying a domain specific model could be the cause of the difference in performance on this specific task.

The results of the Hyperband algorithm indicate that average pooling outperforms max pooling, while max pooling is suggested by the authors of 3D U-NET. The authors of the 3D Attention U-NET also suggest changing the pooling operation, possibly for also having observed an increase in performance. Additionally, optimal optimizer hyperparameters were found to be significantly different than the values proposed in their work. This validates the claim by *Choi et al.* [36] that optimizers are defined by their free parameters and all parameters should be tuned for optimal results.

A choice was made to not use the same neural network for both membrane and nucleus segmentation as this would limit the experimental models to ones that are designed for multi-label predictions. Additionally the including of the nucleus segmentation could hinder the performance of the membrane segmentation as the classification problem becomes more complex. On the contrary, including nucleus segmentation in the same model as protrusion segmentation would have the benefits of reducing the number components to the end of simplicity of the method. The different models that were compared in this work all support multi-class labels, however at the time of creating the experimental set-up it was not clear which models would be compared.

The results we obtained using 3DeeCellTracker conflict with its authors' findings. In their work, they successfully segment and track a 3D tumor spheroid, with an accu-

racy of 97% over 894 cells. In our implementation of their work, such performance could not be replicated. While implementing their work, several bugs in the codebase were found and fixed, pertaining to the reading and processing of input data. Possibly, some bugs remained unnoticed that hindered the performance of their method. If this proved to be true, the comparison between a SOTA DL algorithm and a traditional algorithm would be based on the quality of the implementation. For this reason, it is not possible to conclude whether deep learning-based or traditional tracking methods are better suited to the task of tracking elongated. We can only state that TrackMate outperforms 3DeeCellTracker.

The method proposed in this work is possibly limited to the microscopy method that was used to obtain the experimental data. While the Rios Lab uses confocal laser scanning microscopy to do live imaging of DMG cells, other research labs might not own this type of microscope. If the imaged data of other researchers is incompatible with the trained models and classifiers used in this work, new models need to be trained.

The real value of this work lies future work where this method is used to enable in-depth characterization of the structure and dynamic properties of TM-networks, allowing better understanding of the infiltrative capacities of DMG tumors. Additional future work could look into methods combining information from the nucleus and membrane channel. This would require instance segmentation of each protrusion next to linking each nucleus instance to a membrane instance. This enables characterization of protrusion behaviour, possibly uncovering novel information about the role of TM-networks in defining tumor cell behavioral patterns. Another continuation of this work could look into combining the segmentation task of both channels into one model, where an effort would have to be made to create ground truth data for both channels.

# 6
# Conclusion

In this work we have proposed a method to optimally segment DMG nuclei and membrane protrusion data taken from confocal laser scanning fluorescence microscopy by combining SOTA methods for each of the necessary intermediary steps. Protrusion segmentation was achieved through combining a 3D-UNET neural network with the bandit based HO algorithm Hyperband to outperform some of the SOTA segmentation models. We observe a significant improvement of almost 20% through HO when compared to the proposed hyperparameters of 3D-UNET. Using the Hyperband algorithm, a bigger parameter space could exhausted when compared to naive or Bayesian optimization approaches.

Nucleus segmentation was accomplished with Ilastik's pixel classifier and object detector networks, outperforming the go-to method in the Rios Lab, StarDist, while being considerably faster in both computation time and set-up time. Another perk is that no coding expertise or extensive domain knowledge is needed to train or use Ilastik's classifier, eliminating the need for biologists to delegate these tasks to bioinformaticians.

The optimal tracking method is TrackMate, which provides excellent integration for the Ilastik pixel classifier as well as performance that trumps machine learning approaches. TrackMate shares Ilastik's perk of providing a simple GUI to enable people without coding expertise to use the method. The DL-based method proved to be inadequate for the tracking of DMG nuclei all while being significantly slower.

By having drafted a data protocol, researchers can use said protocol to prepare or create new data to the end of protrusion segmentation and nucleus tracking. The data protocol contains information about the microscope settings, data type and size and the required preprocessing steps. Additionally, the method was implemented in the software package managed by the Rios lab, STAPL-3D.

To our knowledge, this work is the first to combine the SOTA hyperparameter tuning algorithms and 3D-UNET. We show that there is significant performance to be gained when applied to the task of protrusion segmentation. No other work has demonstrated the ability to segment and track elongated nuclei in 3D in addition to segmenting their cell membranes exhibiting tumor microtubes. This work can serve as a guide to enable research into in-depth characterization of the now incurable disease, to the end of targeting the invasive nature of DMG as a means of treatment.

# Bibliography


[1] H. S. Venkatesh, W. Morishita, A. C. Geraghty, D. Silverbush, S. M. Gillespie, M. Arzt, L. T. Tam, C. Espenel, A. Ponnuswami, L. Ni, *et al.*, "Electrical and synaptic integration of glioma into neural circuits," *Nature*, vol. 573, no. 7775, pp. 539–545, 2019.

[2] S. Gillespie and M. Monje, "An active role for neurons in glioma progression: making sense of scherer's structures," *Neuro-oncology*, vol. 20, no. 10, pp. 1292–1299, 2018.

[3] N. J. Jeon, H. Na, E. H. Jung, T.-Y. Yang, Y. G. Lee, G. Kim, H.-W. Shin, S. I. Seok, J. Lee, and J. Seo, "A fluorene-terminated hole-transporting material for highly efficient and stable perovskite solar cells," *Nature Energy*, vol. 3, no. 8, pp. 682–689, 2018.

[4] E. R. Gerstner, "Mri and pet: Noninvasive tools to probe the biology of diffuse intrinsic pontine glioma," *Journal of Nuclear Medicine*, vol. 58, no. 8, pp. 1262–1263, 2017.

[5] R. L. van Ineveld, M. Kleinnijenhuis, M. Alieva, S. de Blank, M. B. Roman, E. J. van Vliet, C. M. Mir, H. R. Johnson, F. L. Bos, R. Heukers, *et al.*, "Revealing the spatio-phenotypic patterning of cells in healthy and tumor tissues with mlsr-3d and stapl-3d," *Nature Biotechnology*, pp. 1–7, 2021.

[6] T. Cooney, A. Lane, U. Bartels, E. Bouffet, S. Goldman, S. E. Leary, N. K. Foreman, R. J. Packer, A. Broniscer, J. E. Minturn, *et al.*, "Contemporary survival endpoints: an international diffuse intrinsic pontine glioma registry study," *Neuro-oncology*, vol. 19, no. 9, pp. 1279–1280, 2017.

[7] D. Hargrave, U. Bartels, and E. Bouffet, "Diffuse brainstem glioma in children: critical review of clinical trials," *The lancet oncology*, vol. 7, no. 3, pp. 241–248, 2006.

[8] W.-T. Wu, W.-Y. Lin, Y.-W. Chen, C.-F. Lin, H.-H. Wang, S.-H. Wu, and Y.-Y. Lee, "New era of immunotherapy in pediatric brain tumors: Chimeric antigen receptor t-cell therapy," *International Journal of Molecular Sciences*, vol. 22, no. 5, p. 2404, 2021.

[9] R. Majzner, S. Ramakrishna, A. Mochizuki, S. Patel, H. Chinnasamy, K. Yeom, L. Schultz, R. Richards, C. Campen, A. Reschke, *et al.*, "Epct-14. gd2 car t-cells mediate clinical activity and manageable toxicity in children and young adults with h3k27m-mutated dipg and spinal cord dmg," *Neuro-Oncology*, vol. 23, no. Supplement_1, pp. i49–i50, 2021.

[10] J. Theruvath, E. Sotillo, C. W. Mount, C. M. Graef, A. Delaidelli, S. Heitzeneder, L. Labanieh, S. Dhingra, A. Leruste, R. G. Majzner, *et al.*, "Locoregionally administered b7-h3-targeted car t cells for treatment of atyp-

ical teratoid/rhabdoid tumors," *Nature medicine*, vol. 26, no. 5, pp. 712–719, 2020.

[11] G. L. Lin, S. Nagaraja, M. G. Filbin, M. L. Suvà, H. Vogel, and M. Monje, "Non-inflammatory tumor microenvironment of diffuse intrinsic pontine glioma," *Acta neuropathologica communications*, vol. 6, no. 1, pp. 1–12, 2018.

[12] D. Benitez-Ribas, R. Cabezón, G. Flórez-Grau, M. C. Molero, P. Puerta, A. Guillen, E. A. González-Navarro, S. Paco, A. M. Carcaboso, V. Santa-Maria Lopez, *et al.*, "Immune response generated with the administration of autologous dendritic cells pulsed with an allogenic tumoral cell-lines lysate in patients with newly diagnosed diffuse intrinsic pontine glioma," *Frontiers in oncology*, vol. 8, p. 127, 2018.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[14] C. Sommer, C. Straehle, U. Köthe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 230–233, 2011.

[15] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, pp. 357–361, IEEE, 1994.

[16] N. Rusk, "Deep learning," *Nature Methods*, vol. 13, no. 1, pp. 35–35, 2016.

[17] S. Webb, "Deep learning for biology.," *Nature*, vol. 554, no. 7690, pp. 555–558, 2018.

[18] T. Kudo, S. Jeknić, D. N. Macklin, S. Akhter, J. J. Hughey, S. Regot, and M. W. Covert, "Live-cell measurements of kinase activity in single cells using translocation reporters," *Nature protocols*, vol. 13, no. 1, pp. 155–169, 2018.

[19] K. E. Magnusson, J. Jaldén, P. M. Gilbert, and H. M. Blau, "Global linking of cell tracks using the viterbi algorithm," *IEEE transactions on medical imaging*, vol. 34, no. 4, pp. 911–929, 2014.

[20] D. Ershov, M.-S. Phan, J. W. Pylvänäinen, S. U. Rigaud, L. Le Blanc, A. Charles-Orszag, J. R. Conway, R. F. Laine, N. H. Roy, D. Bonazzi, *et al.*, "Bringing trackmate in the era of machine-learning and deep-learning.," *Biorxiv*, 2021.

[21] C. Wen, T. Miura, Y. Fujie, T. Teramoto, T. Ishihara, and K. D. Kimura, "Deep-learning-based flexible pipeline for segmenting and tracking cells in 3d image time series for whole brain imaging," *bioRxiv*, p. 385567, 2018.

[22] C. Wen, T. Miura, V. Voleti, K. Yamaguchi, M. Tsutsumi, K. Yamamoto, K. Otomo, Y. Fujie, T. Teramoto, T. Ishihara, *et al.*, "3deecelltracker, a deep learning-based pipeline for segmenting and tracking cells in 3d time lapse images," *Elife*, vol. 10, p. e59187, 2021.

[23] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, pp. 424–432, Springer, 2016.

[24] G. van Tulder, "elasticdeform: Elastic deformations for N-dimensional images," Mar. 2021.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[26] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, pp. 281–305, 03 2012.

[27] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[28] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*, pp. 507–523, Springer, 2011.

[29] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[32] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[35] J. Lu, "Adasmooth: An adaptive learning rate method based on effective ratio," 2022.

[36] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *arXiv preprint arXiv:1910.05446*, 2019.

[37] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," *Advances in neural information processing systems*, vol. 30, 2017.

[38] D. Kato, M. Baba, K. S. Sasaki, and I. Ohzawa, "Effects of generalized pooling on binocular disparity selectivity of neurons in the early visual cortex," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 371, no. 1697, p. 20150266, 2016.

[39] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

[40] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–7, IEEE, 2020.

[41] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," in *2019 IEEE 16th international symposium on biomedical imaging (ISBI 2019)*, pp. 683–687, IEEE, 2019.

[42] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

[43] M. Islam, V. Vibashan, V. Jose, N. Wijethilake, U. Utkarsh, and H. Ren, "Brain tumor segmentation and survival prediction using 3d attention unet," in *International MICCAI Brainlesion Workshop*, pp. 262–272, Springer, 2019.

[44] D. Choi, "Pytorch deeplabv3+ 3d," 2019.

[45] C. Yu, Z. Hu, B. Han, P. Wang, Y. Zhao, and H. Wu, "Intelligent measurement of morphological characteristics of fish using improved u-net," *Electronics*, vol. 10, no. 12, p. 1426, 2021.

[46] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, pp. 265–273, 2018.

[47] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3d object detection and segmentation in microscopy," in *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[48] J.-Y. Tinevez, N. Perry, J. Schindelin, G. M. Hoopes, G. D. Reynolds, E. Laplantine, S. Y. Bednarek, S. L. Shorte, and K. W. Eliceiri, "Trackmate: An open and extensible platform for single-particle tracking," *Methods*, vol. 115, pp. 80–90, 2017.

[49] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nature methods*, vol. 9, no. 7, pp. 676–682, 2012.

[50] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser, "Robust single-particle tracking in live-cell time-lapse sequences," *Nature methods*, vol. 5, no. 8, pp. 695–702, 2008.

[51] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature methods*, vol. 18, no. 1, pp. 100–106, 2021.

[52] S. R. Garner *et al.*, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*, vol. 1995, pp. 57–64, 1995.

[53] J. Ma, J. Zhao, and A. L. Yuille, "Non-rigid point set registration by preserving global and local structures," *IEEE Transactions on image Processing*, vol. 25, no. 1, pp. 53–64, 2015.

# A

# Appendix 1

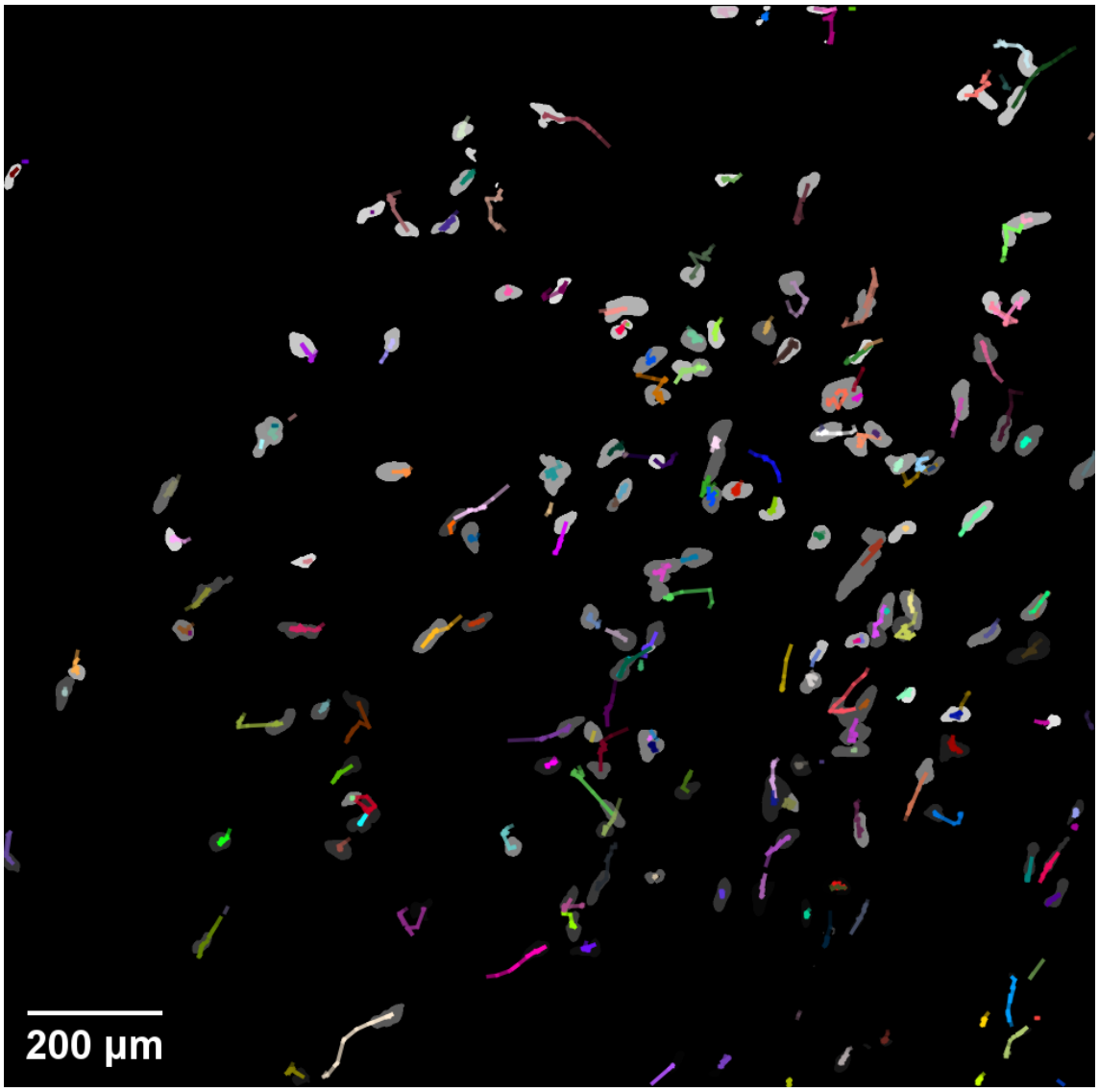



**Figure A.1**

**GRADUATE SCHOOL OF NATURAL SCIENCES**
**UTRECHT UNIVERSITY**
Utrecht, the Netherlands
www.uu.nl

Utrecht University