2D and 3D Analysis of Microscopy Images

Master 2: Apprentissage et Algorithmes

Author: Lucas RINCON DE LA ROSA
Sorbonne University
Paris VI
lucas.rincondelarosa@icm-institute.org

Supervisor: Agusti ALENTORN
Institut du Cerveau (ICM)
Paris VI
agusti.alentorn@icm-institute.org

Abstract

This document collects the work during the second year master internship at the Brain Institute of Paris (ICM) associated to the M2 master studied at Sorbonne University. The main objective has been the study and analysis of 2D and 3D brain cell images. More precisely, the segmentation and tracking of this cells as it can give essential information about the migration and behaviour of tumors. Several models for two-classes semantic segmentation have been compared in 2D datasets and instance segmentation has been performed using Mask-RCNN. Future work aims at expand it to 3D datasets working with different object representation like point clouds or meshes instead of stack images and perform tracking on both 2D and 3D datasets.

Contents

1 Introduction

1.1 Scientific problem

Brain tumors represent one of the most fatal cancers and account for substantial morbidity and mortality [1]. The histo-molecular classification following the World Health Organization (WHO) criteria, has deeply evolved in the last years and it currently required the integration of histological features with molecular biomarkers to define a broad range of brain tumors [2]. However, the survival rates of high-grade gliomas has not improved in the last 20 years and there is an unmet need to provide novel therapeutic targets and improve the therapeutic options [1]. One of the most important molecular markers in gliomas is isocitrate dehydrogenase (IDH) mutations are frequently found in low and high-grade gliomas and have a profound impact in the cellular metabolism and tumor microenvironment, in close relationship with the production of an onco-metabolite produced by this mutation (D2-hydroxyglutarate, or D2-HG) [3, 4]. Furthermore, due to the central nervous system (CNS) localization, it is difficult to dynamically analyze the biology of the evolution of gliomas but recent efforts shed light in the molecular alterations that are produced during treatment, suggesting that these tumors are highly heterogeneous and with an important plasticity to develop resistance to different therapeutic approaches [5].

1.2 Current knowledge

With the advent of new imaging technologies, there are currently many ex vivo and in vitro approaches allowing to robustly mimic the gliomas and its microenviroment with the possibility to analyze the 3D morphology and the movement and their spatio-temporal relationship [6]-[10]. The vast majority of these methods allow correct analysis of the shape, morphology and its cross-talk with the microenvironment, although the 3D analysis is usually limited mostly due to computational cost to perform this analysis and therefore its applicability is still limited in biology and health sciences [11]. However, novel methods allowing to correctly, rapidly and robustly infer the shape analysis of objects are currently available [12], as well as methods to infer the 3D objects tracking (i.e. cells) [13] and to combine it with physical modeling to interpret the consequences of the interaction of tumors cells with their microenvironment [14].

In this project we will use 3 different but complementary approaches to thoroughly describe the shape morphology analysis and its correlation with their genomic background of different brain tumors types considering different exposure to treatment, the network relationship with different cell types or with the microenvironment and vasculature by using human brain organoids, in vitro 3D shape analysis of human immune cells of different glioma subtypes as well as the iDISCO visualization of different glioma mouse models. Brain organoids offer the possibility of real-time studies while recapitulating the human brain microenvironment. This type of model

has demonstrated an unparalleled ability to predict treatment response in cancer patients [15]-[17]. Recently, Linkous et al [18] demonstrated the feasibility of using brain organoids for high-throughput screening of therapeutic compounds against glioblastomas (GBM), the most frequent and aggressive high-grade glioma. Finally, others [19, 20] have shown the relevance of using this type of organoid to study invasion in GBM. The processes involved in the invasion of GBMs are regulated by molecular mechanisms that are still poorly understood, and rarely studied taking into account the heterogeneity of GBM cells, their plasticity in response to therapeutic stress, and their evolution in the brain microenvironment, considering the 3D shape analysis and its relationship with their genomic background, and this is what we propose to do here.

Similarly, single cell morphology is an emergent readout of the molecular underpinnings of a cell's functions and, thus, can be used as a method to define the functional state of an individual cell. A recent study on breast cancer found that breast cancer cells showed a distinct and heritable morphological traits associated with genomic and transcriptomic phenotypes [21]. In addition, the local signalling networks related with cell morphology (i.e. cell protusion, adhesion and tension) are well characterized in cancer cells [22] but further improvement in the mechanisms of the immune tumor microenvironment's shape and its relationship with cancer cell evolution or responsiveness to treatment is less well elucidated. We will analyze the impact of immune related single-cell morphology of different types of gliomas correlated with multi-omics data (transcriptome and methylation) and how both are correlated and we will generalize our results using public datasets like the the tumor cancer genome atlas (TCGA).

Finally, the iDISCO procedure a method to immunolabel and image large intact samples, including adult mouse brain [8]. This method has been used to analyze the 3D morphology of mouse brain vasculature and the relationship with different brain cells [9]. This approach has been used in mouse brain tumors to analyze the 3D distribution of tumors cells within brain vasculature [23]. However, this study used human glioblastoma cells that were orthotopically grafts to obtain xenograft model, and thus, it is an immunocompromised mouse model without the possibility to correctly analyze the immune tumor microenvironment of those tumors.

Robust representation of the shape of 3D objects is a major goal in the field of computer vision [24], and there has been rapid progress in this domain. Supervised and unsupervised methods of shape description have been built on large-scale labelled datasets of 3D models of everyday objects [25, 26]. A particularly challenging setting is extending the power of convolutional neural networks (CNNs) to learning directly on curved surfaces [24, 27]. Unlike volumetric [28] or point-based [29] approaches, surface-based methods exploit the connectivity of the surface representation to improve performance, and furthermore can be robust in the presence

of non-rigid deformations, making them a strong solution for many tasks such as deformable shape matching [30]. Furthermore, multi-object 3D tracking is a recently proposed approach to facilitate mobile robots to perform well-informed motion planning and navigation by localizing surrounding objects in 3D space and time, that allow to segment, classify and follow different objects at real-time [31]. Finally, these computer vision analytical approaches can be combined with different "blocks" or types of data to facilitate the integration of different sources of data [32]. However, the application of these methods for biological related areas is still limited [33, 34]. The aim of this project is to provide a novel framework to analyze 3D and spatio-temporal microscopy imaging data from using different models of brain tumors, considering the shape and morphology of tumors cells in different contexts or different treatments with the possibility to analyze the cross-talk with tumor micro-environment considering their genomic or multi-omics relationship. These approaches could be easily transferred to other diseases or tissues to further generalize our results.

2 Cell Tracking Challenge (CTC)

2.1 History

The main datasets used during the project in order to obtain results and study the behaviour of the models have been obtained from the Cell Tracking Challenge [35]. Also, two of the three datasts used for the semantic segmentation benchmark have been taken from this source. The challenge was launched in 2012, looking for newer and more robust methods of cell segmentation and tracking using microscopy images. In its ten years of existence, it has hold six editions. Even more, since 2017, online submissions are available and evaluated each month, posting the resulting ranking on their website.

Since their first edition, 2D and 3D time-lapse datasets of moving cells and nuclei were available to download. The first edition was held in the International Symposium on Biomedical Imaging (ISBI) [36] 2013 in San Francisco (CA, USA) and the report on the logistics, methods and experimental results were published in Bioinformatics [37]. After the success of the first edition, a second one was held in the ISBI 2014 in Beijing (China). As novelties, they extended the datasets, not only with real, but also computer-generated data by a cell simulator [38]. The third edition was held by the ISBI 2015 in Brooklyn (NY, USA), organized due to the increasing number of participants and submissions.
After this three first editions, a description on each of them and a compilation of 21 algorithms and 13 total datasets was published in Nature Methods in 2017 [39].

In 2018, three years after the last edition was held, they organized the fourth one, in the ISBI 2019 in Venice (Italy). Here they introduced a newer benchmark for the segmentation task and

the dataset repository was extended by adding light-sheet microscopy and real and computer-generated time-lapse sequences of single cells [40]. The main update introduced in the ISBI 2020, holding the fifth edition (virtually due to sanitary conditions), was the addition of the so called silver reference segmentation annotations. These were computer-generated annotations for nine datasets in order to facilitate some methods. In the sixth and last edition until now, organized by the ISBI 2021, the silver annotations for training video segmentation were extended to 13 datasets as well as new methods aiming to improve the generalizability to the rest of the datasets.

In June 2022, a report synthesized 89 algorithms used in the history of the challenge including a repository containing 20 datasets. The outcomes of three insightful studies about the relationship between the technical and biological performance of the benchmarked algorithms and the dataset and annotation properties, as well as the generalizability and the reusability of top-performing algorithms, was published in Nature Methods in 2023 [41].

2.2 Datasets

The dataset repository consists of 2D and 3D time-lapse sequences of fluorescent nuclei or cells moving in a substrate environment, along with 2D phase contrast, and differential interference contrast microscopy videos of cells moving on a flat substrate. As was mentioned above, in the fifth edition of the challenge 2D and 3D videos of computer-generated fluorescent cells and nuclei of different shapes and motion patterns were provided. Up to this day, the challenge repository contains 20 datasets (only 13 existed in the first edition).

Both the 2D and 3D dataset structure is similar. The images are encoded in *.tif* files containing only one channel (meaning black and white images). 3D datasets are encoded as a 2D-stack, so that there are multiple 2D images taken in different depth. In this case, all 3D datasets contain only 5 images in the z axis where the 2D images have sizes between 250×250 and 1024×1024 pixels. It is important to say that the resolution of the 2D-stack for some depth values is not astonishing and it makes them not optimal for certain tasks like segmentation, but this will be discuss in a different section.

Before diving into the main sections of the project, I will briefly talk about the datasets annotations. Each dataset, no matter if it is two or three dimensional, consist of two videos taken in different times for training and testing sets. This way, we end up with four videos. For segmentation and tracking, there exist the so called *Gold Truth (GT)* annotation. This one is made by multiple professionals and the final annotation consist of the average. Due to the highly cost and complication to obtain it, there are only a few frames from the videos that are annotated. This means, that inside the training sets containing the two videos, only some of the frames are annotated, which really is a limitation. For three dimensional datasets the

problem intensifies, since we do not only have frames in time, but also in depth, and only one out of the five images in the 2D-stack is annotated. In the sixth edition, *Silver Truth (ST)* annotations for segmentation were added. Since these annotation were computer generated using the methods and algorithms outputting the best results in the previous editions, they were easily obtained. This way, the datasets containing this annotations are much completed and all frames in the videos (and all images in the 2D-stack) are annotated.

3 Segmentation

The task of segmentation plays a fundamental role in computer vision. It consists in dividing and classifying every pixel of the image into different classes or groups. For doing so, the methods used in segmentation should be able, first to separate the pixels of the objects from the background and then group them in clusters taking into account similar properties and characteristics, like color, intensity, saturation...

It is important to differentiate between image classification and image segmentation: the former gives information about the whole image, taking into account all the pixels and giving them a label. The latter, first groups the pixels into different elements or objects and then performs an image classification for each of them. Inside image segmentation one can find different kinds:

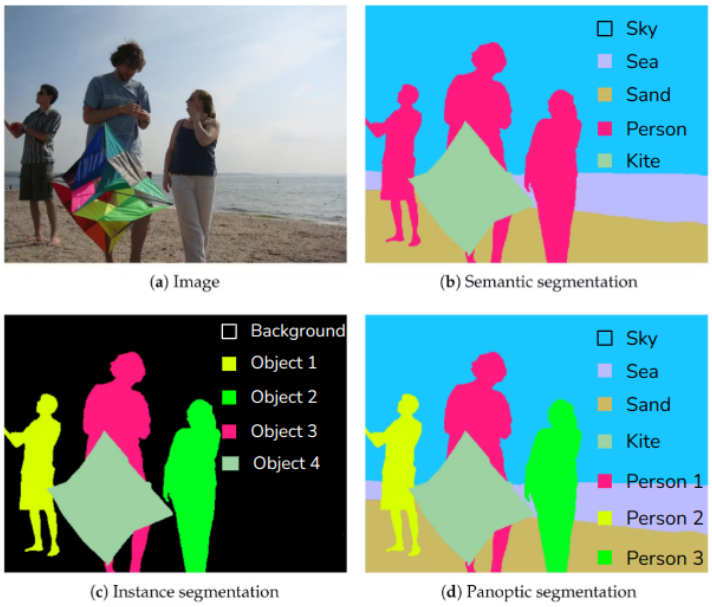
- Semantic segmentation: is the task of assigning a class label to every single pixel in an input image. In this case, the number of labels will depend on the training performed to the dataset and the method used for segmentation. Semantic segmentation is based on the principle that the content of an image can be divided into several semantic classes. These semantic classes can then be used to identify and track objects in the image. It can be used both in Supervised as well as Unsupervised learning.

  As it has been stated before, Image Classification and Image Segmentation are quite similar. If one compares them you can see that Image Segmentation is nothing but a classification of the pixels of the image within certain context. That is why Semantic Segmentation started as a slight modification of image classification. The first successful network model for image segmentation used a fully convolutional network (FCN) [44]. In the last years, new models with larger and smaller modifications have been proposed, but most of them run under a convolutional network like DeepLab, FastFCN, MaskRCNN or Transformer-based models.

- Instance Segmentation: it goes further in the task than semantic segmentation. In this case, one is not interested in grouping the pixels in different clusters based on a semantic label, but being able to differentiate each of these objects individually.

- Panoptic Segmentation: it is considered to be the most complete one. It doesn't only perform Instance Segmentation in the image (so identifying each individual object in the frame) but it also groups them based on a semantic label. It is nothing more than the mixture of Semantic and Instance Segmentation.

In Figure 1 is shown an example of the difference between these three types of image segmentation. The following subsections will cover the models used for the segmentation task, both for semantic and instance.



(a) Image (b) Semantic segmentation (c) Instance segmentation (d) Panoptic segmentation

Figure 1: Comparison of different types of image segmentation.

3.1 Semantic segmentation models

3.1.1 UNet

The 'UNet' architecture was introduced in 2015 by Ronneberger et al. [43] in relation to the ISBI challenge for medical image segmentation. By using this architecture trained on transmitted light microscopy images, they were able to finish top first in the 2015 Cell Tracking Challenge by a large margin.
As shown in their article, they construct the network upon the already known Fully Connected Network [44]. The main difference is that they achieve even better results in segmentation with fewer training data, so that it is possible to do a notable training with small datasets. This is achieved by substituting some of the downsampling operators by upsampling ones; the pooling operators that decrease the dimension of the image in a usual convolutional network are replaced by an operator that does exactly the opposite; augment the dimension. Part of

the features are cropped and combined with previous ones of the same dimension so that the new feature has doubled its size. This way, one gets a higher resolution feature and at the end, a better segmentation as output.

Network Architecture: I will dedicate a small section to talk about the network architecture, as I think this neural network has has an important role during the internship and we haven't view during the master courses.
The network architecture is illustrated in Figure 2. At first glance one can already understand where this network gets its name from. The left size of the "U" represents the downsampling part (contracting path) where the features dimensions gets smaller in each layer and the right size the upsampling one (expansive path) where the features dimension start growing again until it ends up with a size similar to the original image.
If one focuses on the contracting path, it can see that it has the general structure of a usual convolutional network consisting of convolutional and pooling layers. In this case, two different 3x3 convolutional layers plus a rectified linear unit (ReLU) are followed by a 2x2 pooling layer. In particular, the number of feature channels is doubled and the size of each of them is halved in each downsampling step. Up to this point, we are working with a slight modification of a usual CNN.

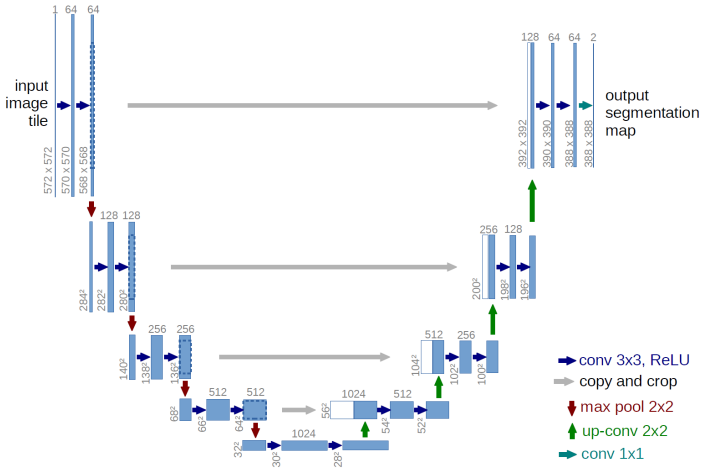


Figure 2: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

The big change introduced in [43] was the expansive path. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped

feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

An important detail needs to be kept in mind in order to make it work. The original input size should be one such that the consecutive layers in the contracting path (let side) end up in and even x- and y- size. This is due to the fact that in the expansive path (right side) the features are going to be halved in order to be cropped and concatenated, so an odd size would not work.

The other two models used were chosen from the PyTorch pre-trained models, so one only need to process the data to make them work but not create the model from zero.

3.1.2 FCN

The Fully Convolutional Network (FCN) was introduced in the paper [44] in 2015. FCN stands out by its fully convolutional nature, replacing traditional fully connected layers with convolutional layers. This design choice retains spatial information, enabling the network to process inputs of varying sizes and produce output segmentation maps of the same size. FCN also incorporates skip connections that link lower-level and higher-level feature maps, effectively capturing both fine-grained and coarse information. The network further employs transposed convolution (upsampling) layers to increase feature map resolution, aligning them with the input image size. Multiple output heads provide multiscale predictions, improving the model's ability to capture details. FCN is trained with labeled datasets, leveraging pixel-wise classification loss functions like softmax cross-entropy and optimization techniques such as stochastic gradient descent (SGD). It has found applications in various domains, including autonomous driving and medical image analysis.
What sets FCN apart from earlier architectures is its emphasis on spatial preservation, skip connections, and efficiency in handling diverse input sizes. Unlike traditional convolutional neural networks (CNNs) used for image classification, FCN maintains pixel-level information throughout the network, making it suitable for pixel-wise tasks like semantic segmentation. Its use of skip connections helps combine fine and coarse features, enhancing segmentation accuracy. FCN's adaptability to varying input sizes has made it practical for real-world applications, and its success has inspired the development of numerous improved variants, solidifying its place as a fundamental building block for detailed pixel-level analysis in computer vision.

Network Architecture: Starts with an input layer that takes an image of variable size. It then employs convolutional layers to extract low-level features, followed by additional layers that capture increasingly complex information. These intermediate layers are connected

to corresponding layers in the upsampling path through skip connections, preserving high-resolution spatial details.
In the upsampling part, transposed convolutional layers are used to increase the spatial resolution of the feature maps, aligning them with the original input size. The final convolutional layer produces dense pixel-wise predictions for each class. These predictions are usually processed through a softmax activation function to generate class probabilities for each pixel. FCNs are trained on labeled datasets, optimizing pixel-wise loss functions like softmax cross-entropy. The result is an efficient and versatile network capable of segmenting images into different classes while preserving spatial information. More advanced FCN variants incorporate additional features and optimizations to further improve segmentation accuracy.

Pretrained models on large-scale datasets in PyTorch typically use architectures based on well-known backbone networks, such as VGG or ResNetin order to maximize the feature extraction capabilities of these backbones and add decoder layers for pixel-wise predictions.

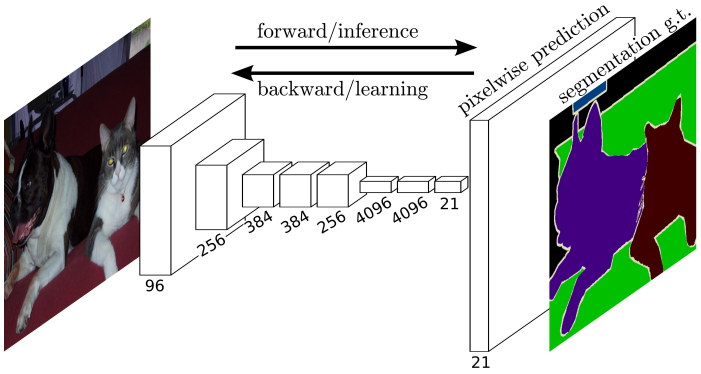


Figure 3: FCN example architecture.

3.1.3 DeepLab

This deep learning model was introduced in [45] in 2017. It is a highly influential model for semantic segmentation in computer vision. What sets DeepLab apart is its effective use of atrous convolutions (also known as dilated convolutions) and a unique combination of multi-scale features. Atrous convolutions allow to capture spatial context at different scales without significantly increasing computational complexity, unlike traditional CNNs. By carefully selecting dilation rates, DeepLab can control the receptive field of the network, enabling it to capture both fine-grained details and larger contextual information in the same layer.
Another distinguishing characteristic of DeepLab is its use of a "Spatial Pyramid Pooling" (SPP) module, which operates at multiple spatial scales to capture features at different levels of granularity.

This helps in handling objects of varying sizes within the same image. Additionally, DeepLab often incorporates skip connections from lower-level feature maps to higher-level ones, similar to U-Net, to further enhance segmentation accuracy by combining fine and coarse features.

One of the most significant advancements in DeepLab is the integration of "atrous spatial pyramid pooling" (ASPP), which extends the SPP concept by applying atrous convolutions at multiple scales. This ASPP module efficiently captures rich context information, making DeepLab highly effective in capturing fine object boundaries and small-scale details. Overall, DeepLab's architecture, with its emphasis on atrous convolutions, multi-scale features, and ASPP, has achieved state-of-the-art performance in semantic segmentation tasks, making it a preferred choice in applications such as autonomous driving, medical image analysis, and scene parsing.

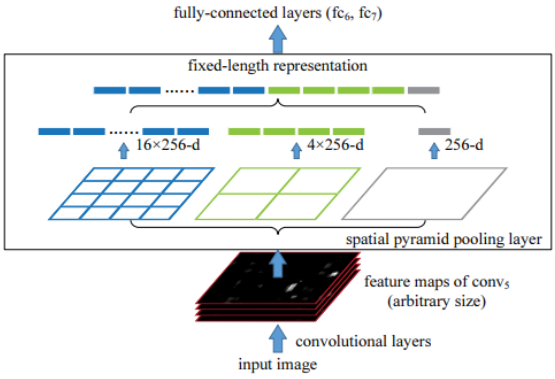


Figure 4: Network structure with a spatial pyramid pooling layer where 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

3.2 Instance segmentation models

3.2.1 Post processing methods

As was explained before, instance segmentation is nothing more than an extension of semantic segmentation. In this case one doesn't only want to classify each of the pixels of the image in a class or label, but also be able to distinguish between instances in the same class. This cannot be done directly using the deep learning models introduced in 3.1 but one can find a way to make them work.

During this section we will focus on two-classes segmentation, since it is the task that we will be dealing with, but a similar argument could be used with a multiclass problem. Let's suppose we have a model for semantic segmentation e.g. UNet architecture, already trained and working for our cell images. In this case all cells belong to the same kind (that is why it is a two-class segmentation problem). Without changing the model, one can try to post-process the outputs of the semantic segmentation model (so one binary mask per image) to extract the different features. Before going any deeper into the details, it is relevant to highlight the fact that this is not an easy task and it takes advantage of machine learning algorithms rather than deep learning models. The first important aspect would be to deeply analyse the dataset one

is working with and extract different features that could be helpful to detect the cells. It won't be the same detecting stem cells which are more rounded and well-delimited than neuron cells, that have not only the nuclei but long and fine extensions. Some of these features could be the shape, size, area of pixels that they fill, color gradient in the borders, perimeter...Of course, the less classes one has and the clearer the background is, the easiest it will be to detect the instances.

Once the primary features have been selected, one needs to be able to apply them to the images. This can be done using several python modules as "SkImage" which has plenty of different algorithms to find patterns and aspects in the images. In our case, the datasets from the CTC had really clear images with a flat background which helped a lot in the process.

3.2.2 Mask R-CNN

Mask R-CNN was developed by the Facebook AI Research Team [46]. This network is an extension of "Fast R-CNN" [47] which could only identify detection boxes on the objects but not segmentation perse. It is a region-based network using deep convolutional networks. At the moment is was a good improvement in object detection models with respect to the accuracy and training and testing speed.

Network Architecture: As was explained before, "Mask R-CNN" runs overs "Fast R-CNN", so the network architecture is the same except the new branch added to "Mask R-CNN" for the segmentation task. During training the network need both the images and targets, which are nothing more than the masks. In the case of "Fast R-CNN" the targets consist of the bounding boxes. The network can be divided into two blocks: the convolutional *backbone* and the network *head*. The first one is composed of convolutional and pooling layers in order to perform feature extraction over the image. The network *head* consists on the concatenation of the branch from "Fast R-CNN" for bounding box detection and a third branch added specifically for mask prediction, allowing instance segmentation. An example of this architecture is shown in Figure 5.

3.2.3 DiffusionNet

Both "UNet" and "Mask R-CNN" work with 2D images, so in case one wants to apply them to three-dimensional data, changes should be made. This last neural network that is presented is called "DiffusionNet" [48] and is the only one that operates directly in 3D data, both point-cloud or meshes. DiffusionNet is based on the idea of using spatial diffusion as the main network operation. This means that the network learns to propagate information across the surface of a shape, rather than just between neighboring points. This makes DiffusionNet more robust to discretization artifacts, and allows it to learn more complex relationships between

the features of a shape.

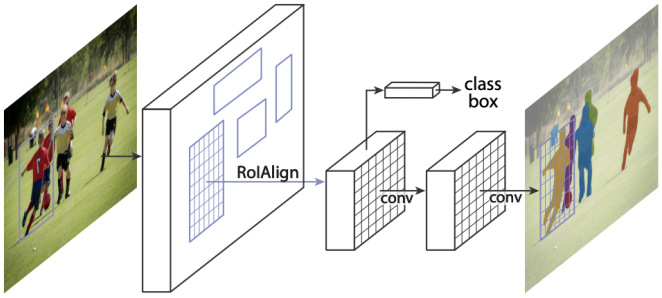


Figure 5: Mask R-CNN architecture example. The first part is the architecture of Fast R-CNN and then a branch containing two convolutional layers is added for the mask prediction.

Even though it is still a relatively new architecture, it has the potential to revolutionize the way we learn on surfaces. It is more robust to discretization artifacts than previous methods, and it can learn more complex relationships between the features of a shape. This makes DiffusionNet a powerful tool for a variety of tasks in computer graphics, computer vision, and robotics.

Network Architecture: in Figure 6 is shown a scheme of the architecture of DiffusionNet. As one can see, it is composed of several *DiffusionNet blocks*. Each of this constructing blocks is the result of combining the three main building blocks of the model: multi-layer perceptrons (MLPs) applied at each point to model pointwise scalar functions of feature channels, a learned diffusion operation for propagating information across the domain, and local spatial gradient features to expand the network's filter space beyond radiallly-symmetric filters.

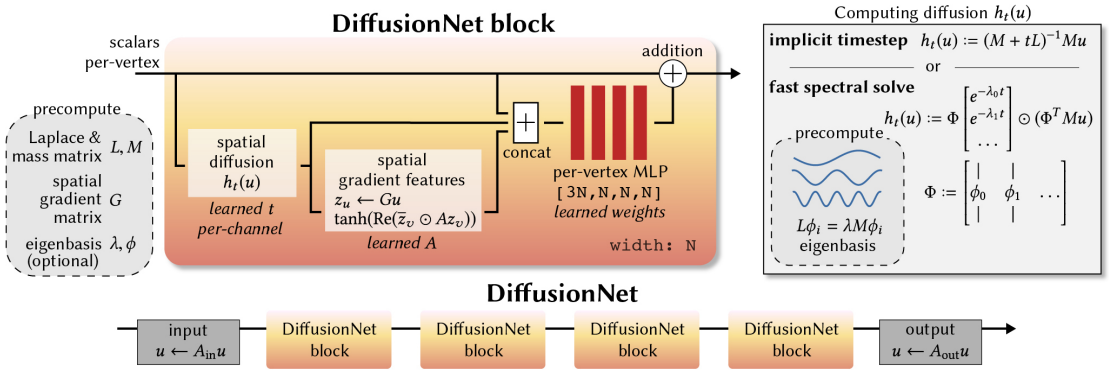


Figure 6: DiffusionNet architecture. It is composed of successive identical DiffusionNet blocks. Each block diffuses every feature for a learned time scale, forms spatial gradient features, and applies a spatially shared pointwise MLP at each vertex in a mesh/point cloud/etc.

4 Tracking

Object tracking is a computer vision task that aims at detecting and following the movement of an object in a sequence of images or video frames. The goal of object tracking is to maintain a consistent association between the object and its representation across different frames, despite changes in position, scale, orientation, and appearance. Is is a challenging problem and its difficulty depends in great measure on the dataset used. Some problems that one may encountered when performing object tracking are: changing the appearance of the object due to lighting or deformation, occlusion of other objects since the view can be blocked, scale changes depending on the zooming of the camera, blur in the picture due to the movement...

To perform object tracking in a sequence of video frames, one can think of two possible approaches: using machine learning algorithms or deep learning models. Each has its strengths and weaknesses. Depending on the dataset that one is using and the type of tracking that he wants to perform, a different approach should be recommended. Also, the accuracy and deepness of object tracking should be considered.

4.1 Machine Learning Algorithms

Machine Learning learn from data using statistical methods as well as algorithms trained to make classifications or predictions. This way, it works with algorithms and mathematical methods that can be precisely described and studied, giving the user a better understanding of what is going on under the computer. When data is clearly organized with a fixed structure, machine learning algorithms are widely suggested as they can give astonishing results.

With respect to this project, we want to focus on machine learning algorithms applied to object tracking. Due to the order followed in the project, object tracking was proposed after object segmentation, or more precisely, it was thought to be done after the segmentation task. That is why we could think of ways of performing tracking with the masks of the real images (since we could obtain the mask of any video frame by applying the trained segmentation model first). Since I didn't do any tracking course or project during the master, we thought it could be a good first approach to try to mimic and understand how these machine learning algorithms work. This way, we didn't focused on one specifically but rather tried to create one by just understanding the problem we were solving and the data we had.

In this case we were dealing with a set of frame video images (with fixed order) and its respective masks. Just to be clear, this masks are of the same size of the original images but consist of multi-label, in other words, the background pixels correspond to zero and the rest to the objects detected in the image. Pixels of the same object have the same label (1, 2, 3, ...) and

the maximum label corresponds to the total number of objects in the image. The problem is that when segmentation in done, the labels are assigned arbitrarily in order of detection of the instances, so that one object can have a different label in successive frames. With this data, the goal is to assign or propagate the label of the objects to the consecutive frames, so that each object has the same label during the whole video.

To solve this first task, we though on assign to each object in a frame the label of the object in the previous frame such that the Intersection over Union (IOU) [49] evaluation metric was the highest. In Appendix 7 is added an explanation of the different metrics used for theses algorithms as well as during the training process for segmentation.

This was done by creating a function that calculated the IOU between two binary masks (so only background and one object per mask). The second auxiliary function took as entries two consecutive frames and for every object in the two masks, performed the IOU using the previous function. As output it assigns to each object in the current frame the label of the object in the previous one that had the best correspondence. With these two functions, we could create a main loop that iterates through every frame and propagated the labels from one frame to the next.

This first step corresponds to a very simple tracking algorithm and could work relatively fine in some cases. But as it was said before, the result of an algorithm depends a lot on the dataset used, and one should understand the behaviour of the data in order to create a better algorithm. In this case, the datasets we were using had some characteristics that required more complex algorithms in order to obtain good results. Some of this features are:

- Cells entering and exiting: from one frame to the other, some of the cells could disappear from the frame as well as some new cells could enter our observation area.
- Mitosis: whenever cells are reproducing, the end up dividing themselves into two smaller cells. In global, for each cell divided one ends up with one more cell in the frame.
- Cells merging: even though the cells do not merge to create a new one, they may get close enough that they touch, so that when object segmentation is performed, they are considered as one.

All these aspects should be considered if one wants to create a decent algorithm. A way to include these features into the algorithm is similar to what was done before with the IOU. First, one has to consider the feature that he thinks may make a difference and then create a lost functions associated to this feature. For example, in addition to IOU on could take the centroid of each object detected and compare the distance between centroids from one

frame to the next one or the area of each object (although this would not work in the case of cells that move too quickly or change abruptly its morphology). And this can be done for pretty much every feature, as long as one finds a way to quantize the changed between frames.

It is clear that object tracking is not an easy task and for complex datasets, the creation of a decent algorithm can be really challenging. For this reason deep learning algorithms for object tracking can be really handy.

4.2 Deep Learning Algorithms

As was just explained, machine learning algorithms do have its use in object tracking, but one should know how and when to use them, and be aware of its limits. In order to deal with those limitations due to data complexity, deep learning algorithms can be of more interest in this subject. The main reason is that they are able to extract more complex features and relations between objects and frames.

For example, using a Convolutional Neural Network (CNN) one can not only detect and segment the cells but also extract features about their morphology, location, movement, migration, transformation...in order to relate ones with others. There are also some methods that use a specific type or neural network that tries to predict the movement of the detected objects and compare them with the next frame. This methods are much more complex than the previous algorithms and that is why they are able to give better results, above all when the data in relatively complex. Nevertheless, the computational costs can be severely increased.

In the last years, Graph Neural Networks (GNN) have proved to be very powerful and effective for a variety of tasks, and they are becoming increasingly popular in a variety of fields, such as natural language processing, computer vision, and biology. As its name says, these are a type of neural network that can be use when the data is organized or structured in the form of a graph. In that case, GNN update the graph by propagating messages (information) between the nodes. The real question here is how can we turn our data consisting of images (or masks) of cells into a graph. At a first glance, we can just imagine that the detected objects in the frame are the nodes of the graph and the edges, the relation between each of them. Those relations, associations or features are the ones extracted by the GNN.

One of the methods that obtained a better result in the tracking task in the Cell Tracking Challenge is presented in [50]. They decided to use a GNN, as many other people may have, but what makes it work better than others is the core of the GNN: the way the nodes features and edges associations are updated in each layer.

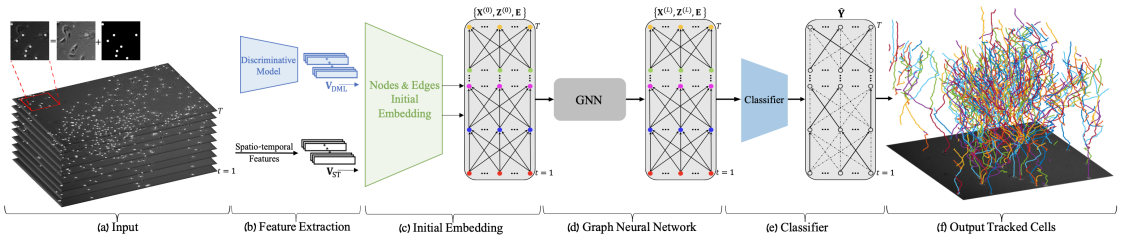


Figure 7: General representation of the architecture of the GNN presented by [50]. (a) The input is composed of a live cell microscopy sequence of length T and the corresponding sequence of label maps. (b) Each cell instance in the sequence is represented by a feature vector which includes DML and spatio-temporal features. (c) The entire microscopy sequence is encoded as a direct graph where the cell instances are represented by its nodes and their associations are represented by the graph edges. Each node and edge in the graph has its own embedded feature vector. (d) These feature vectors are encoded and updated using Graph Neural Network (GNN). The GNN (which is illustrated in Fig. 2(a)) is composed of L message passing blocks which enable an update of edge and node features by their L-th order neighbors (i.e., cell instances which are up to L frames apart). (e) The GNN's edge feature output is the input for an edge classifier network which classifies the edges into active (solid lines) and non-active (dashed lines). During training, the predicted classification $\hat{Y}$ is compared to the GT classification Y for the loss computation. Since all the framework components are connected in an end-to-end manner the loss backpropogates throughout the entire network. (f) At inference time, cell tracks are constructed by concatenating sequences of active edges that connect cells in consecutive frames.

5 Experiments

5.1 Semantic segmentation

As was introduced before, the segmentation experiments will be divided in two: those containing the results of semantic segmentation and the ones focusing on instance segmentation. For semantic segmentation three models have been chosen: UNet, FCN and DeepLab. All of them have been already introduced and a deeper insight about them can be found in the Appendix. The UNet model consists of a basic architecture trained from random initialized weights whereas the FCN and DeepLab models have been taken form the PyTorch module with pretrained weights. For the datasets we have used three different ones: 'Fluo-N2DH-GOWT1', 'Fluo-C2DL-MSC' and 'Warwick QU'. The first two are available in the CTC. The last one is probably the most challenging one due to its different nature. In Table 1 one can find three subtables, each of them containing the results of the models being applied to a particular dataset, going from the least to the most challenging. For each model, it has been chosen the epoch where the validation loss is the lowest. In general, it also corresponds to the highest accuracy metric results, although this may vary a bit.

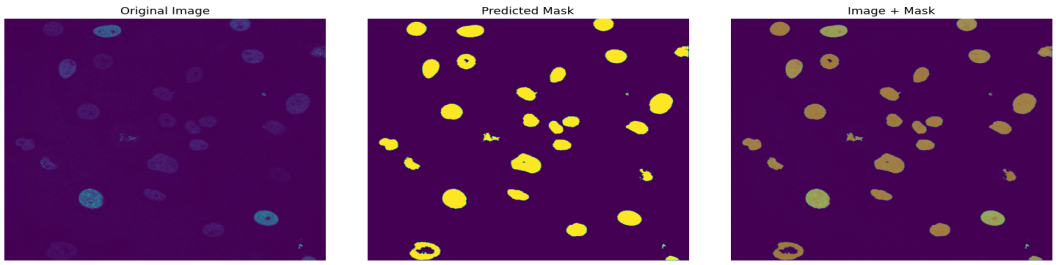
Models	Epoch	Train loss	Val loss	Dice	IoU
Fluo-N2DH-GOWT1					
DeepLab	95	0.0065	0.0077	0.9774	0.9558
FCN	100	0.0168	0.0204	0.9390	0.8851
UNet	98	0.0091	0.0087	0.9780	0.9570
UNet not	94	0.0087	0.0074	0.9772	0.9555
Fluo-C2DL-MSC					
DeepLab	80	0.0158	0.0293	0.8919	0.8049
FCN	92	0.0225	0.0360	0.8558	0.7479
UNet	97	0.0354	0.0335	0.8891	0.8003
UNet not	92	0.0340	0.0455	0.8580	0.7513
Warwick QU					
DeepLab	15	0.0954	0.2349	0.8929	0.8066
FCN	19	0.0975	0.1892	0.9129	0.8397
UNet	95	0.2898	0.2389	0.8802	0.7860
UNet not	91	0.2955	0.2398	0.8863	0.7958

Table 1: Quantitative evaluation on three 2D datasets for semantics segmentation. For each dataset, we compare results of multiple baselines (rows). For each model is shown in what epoch it obtains the best results (with respect to the minimum validation loss achieved). Best result for each dataset are indicated in bold.

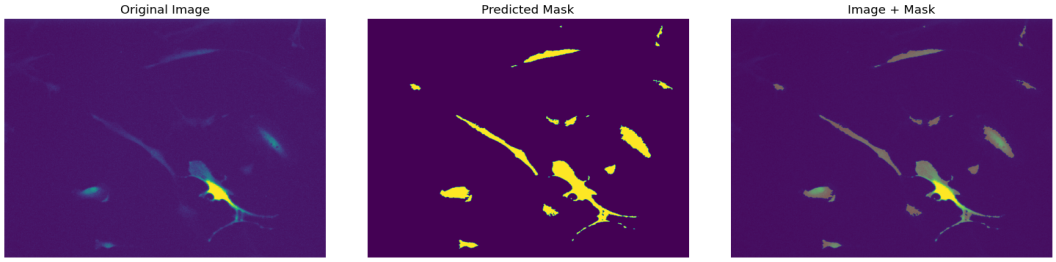
Taking a look at the table one can see that in the first two datasets, the results from each of the models is quite similar. The number of epochs needed to achieved the best results are close to the end of the experiment and the best metrics for Dice and IoU coefficients are in the same range. DeepLab seems to output the best general results while the pretrained model from FCN is behind the basic UNet architecture, with and without normalizing the inputs. In the last dataset the results differ . The validation loss in all the models is quite behind the values obtained in the other datasets and in this case, is the FCN model the one with the best results. As a general overview, the results from the UNet model with and without normalizing the input values are almost the same, so it makes small difference. Also, the first dataset achieves high Dice and IoU values, above 0.95 in most cases. This is due to the fact that the dataset is really homogeneous and the images are all taken in the same way. More about this characteristics can be found in the Appendix. In Figures 9 and 10 are represented the losses and metrics of all the experiments in Table 1 over 100 epochs and it can be noted that the first dataset outputs really smooth results, as the training and validation losses decrease in the same manner independently of the model. In the other hand, analysing the last dataset one can sees that the

pretrained models stop learning quite fast, in the first 20-30 epochs or so, whereas the UNet architecture (independently of normalized or not inputs) is still learning even at the end of the first 100 epochs. This is clearly visible in the training loss curve.

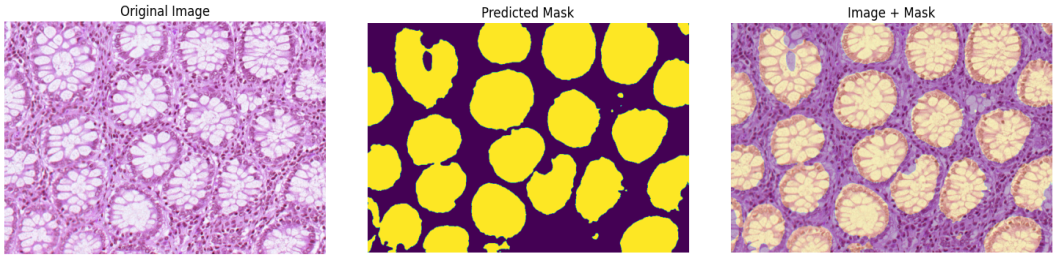
Finally, in Figure 8 are represented the original test images along with the mask predicted by the UNet model without normalizing the values. On can sees that this basic model is able to detect most cells but it has problems filling the holes in them and ignoring small contrast values that don't correspond to any instance. It is in the Fluo-C2DL-MSC dataset that this errors are more clearly visible, as the shape and size of the cells vary more than the others.



(a) Fluo-N2DH-GOWT1



(b) Fluo-C2DL-MSC



(c) Warwick QU

Figure 8: Example of the original image, the predicted mask and their overlap for the UNet architecture in the three datasets used for semantics segmentation.

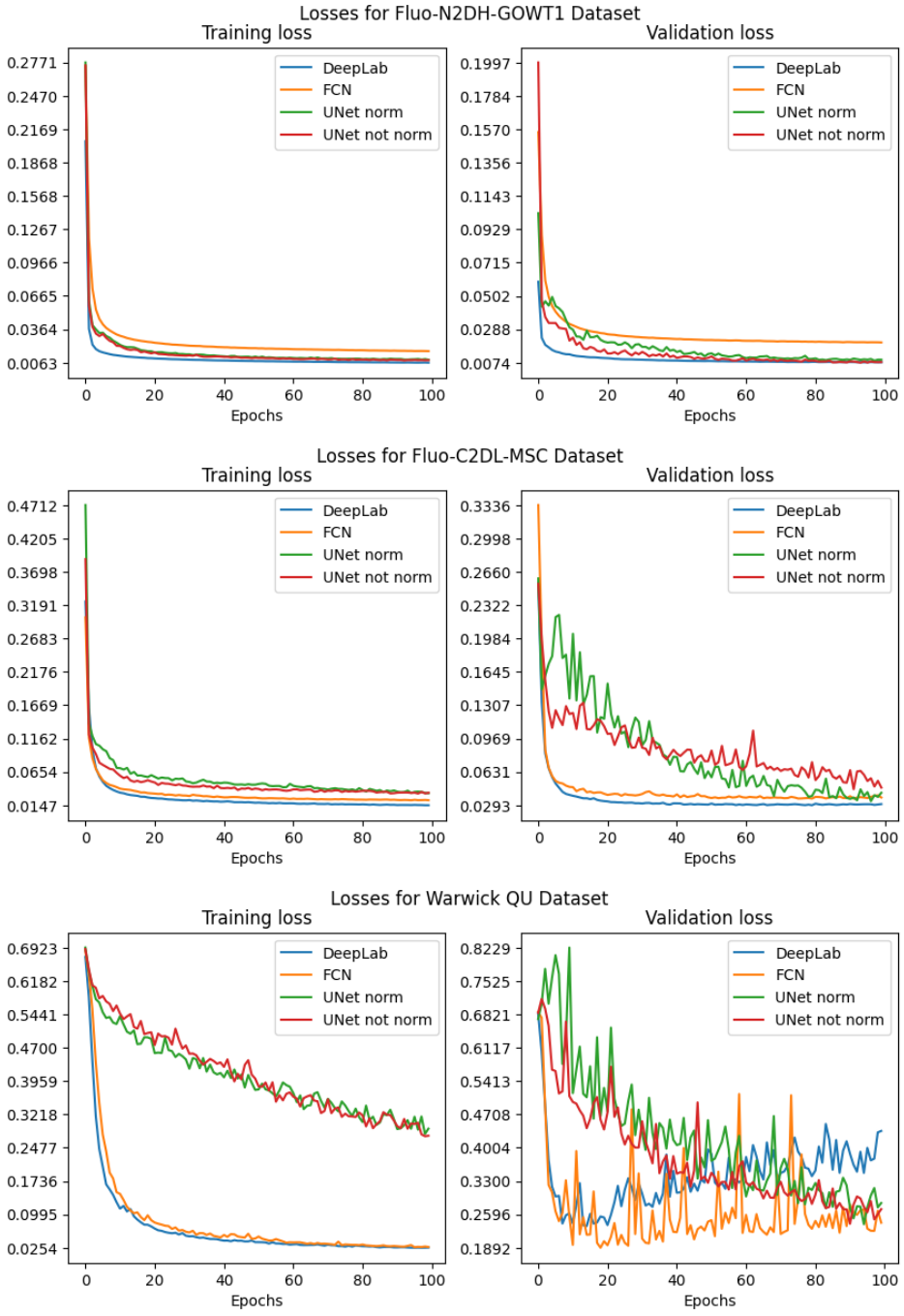


Figure 9: Comparison of training and validation losses for the 'Fluo-N2DH-GOWT1', 'Fluo-C2DL-MSC' and 'Warwick QU' datasets in semantic segmentation experiments. All models have been run over 100 epochs with a learning rate of 0.001.

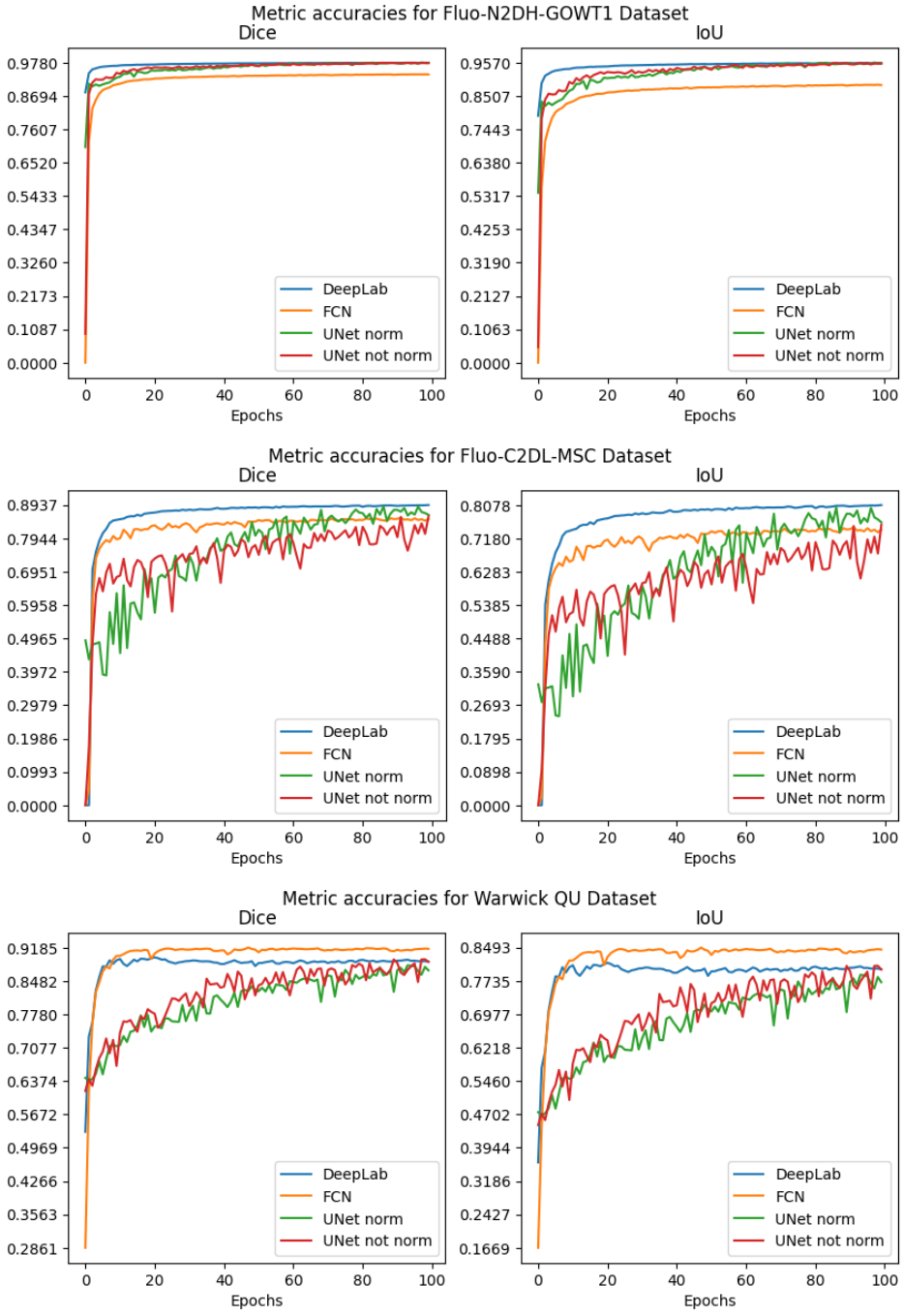


Figure 10: Comparison of Dice and IoU metrics for the 'Fluo-N2DH-GOWT1', 'Fluo-C2DL-MSC' and 'Warwick QU' datasets in semantic segmentation experiments. All models have been run over 100 epochs with a learning rate of 0.001.

5.2 Instance segmentation

For the experiments during the internship, the semantic segmentation task was the one where we devoted more time, as it settles the bases for the rest of the work. In this case only Mask R-CNN architecture was well implemented in order to get comparison results. CTC datasets were also used for the purpose since most of them already had instance segmentation task which is not easy to obtain. By default Mask R-CNN model is prepared to be used with 3 channel images (colored images) which doesn't correspond to the CTC datasets. One of the easiest ways to deal with this kind of technical problems is to convert 1 channel images to 3 channels by just copying the unique grayscale channel 3 times. This is an easy approach although it may be not the best one in all cases, as it requires 3 times more memory per image without increasing the useful information. Nevertheless, in this case it was preferred this approach in contrast to changing the network architecture to make it compatible with 1 channel images. In Table 2 one can find the same metric values as the ones shown for the semantic segmentation experiments. In this case the comparison is done for one unique model and different datasets (in contrast with semantic segmentation): 'Fluo-N2DH-GOWT1' and 'Fluo-C2DL-MSC'. Since the second one has different shape values for the images corresponding to the first and second videos sequences (see Appendix 7 for more details) and to avoid confrontations with the model, it was decided to train them as different datasets, named with the extension '0T' with $T = \{1, 2\}$ indicating the video sequence.

Datasets	Epoch	Train loss	Val loss	Dice	IoU
		MaskRCNN			
Fluo-N2DH-GOWT1	88	0.0713	0.1173	0.8554	0.7473
Fluo-C2DL-MSC-01	29	0.1686	0.4343	0.3327	0.1995
Fluo-C2DL-MSC-02	92	0.0993	0.2836	0.4936	0.3276

Table 2: Quantitative evaluation on three 2D datasets for instance segmentation. Results are shown for MaskRCNN model giving the epoch at which it obtains the best results (with respect to the minimum validation loss achieved). Best result for each dataset are indicated in bold.

Although the training loss may look similar in all cases, one can see a big difference in the validation loss and the metric accuracy values. Clearly, the 'Fluo-N2DH-GOWT1' dataset outputs the best results, achieving a validation loss twice as low as the second dataset in the list. With respect to the accuracy values, the differences are even greater. What is more shocking is the fact that for the same dataset corresponding to different video sequence ('Fluo-C2DL-MSC-01' and 'Fluo-C2DL-MSC-02'), the metric accuracies are quite different. This can be seen with more clarity in Figure 11 where is shown the comparison of the 4 columns of Table 2 over the first 100 epochs. One can see that the 'Fluo-N2DH-GOWT1' has a really good training curve, giving smooth results for the validation loss and the metric coefficients.

In contrast, although the general beaviour of the other dataset is similar, the values it achieves are much worse and less smooth, having a not ignorable error.

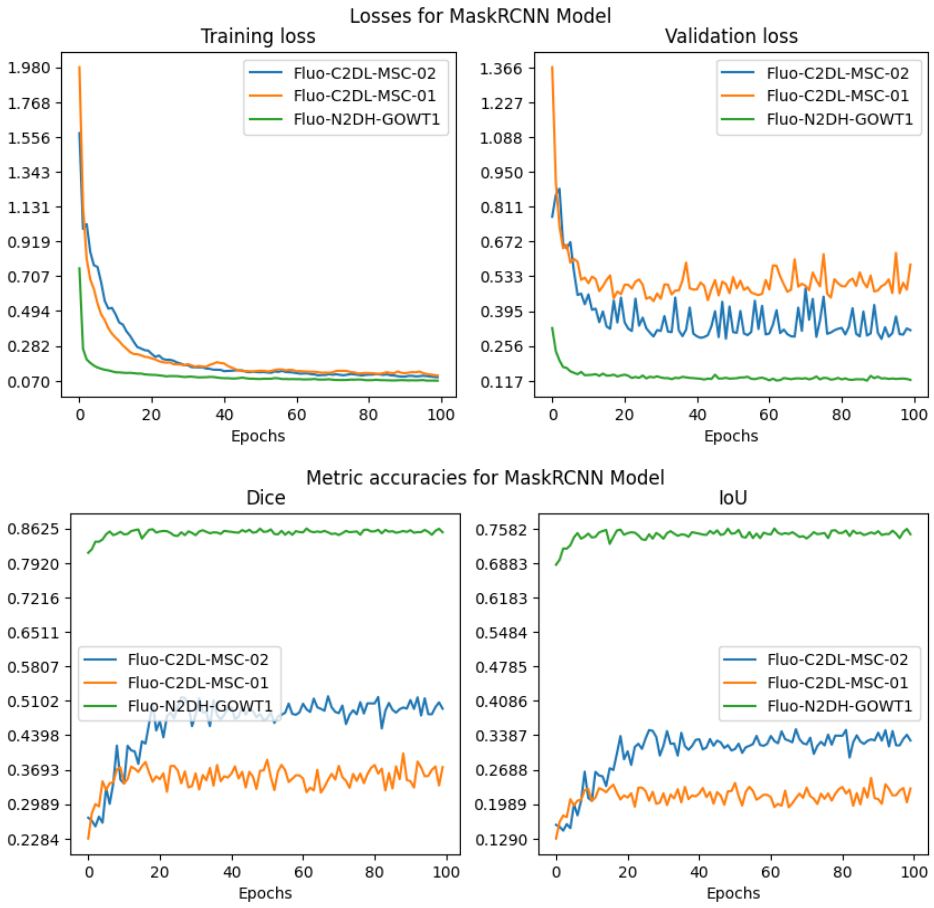
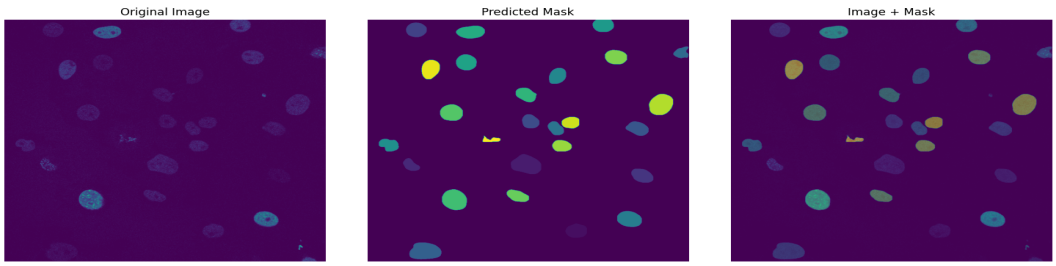


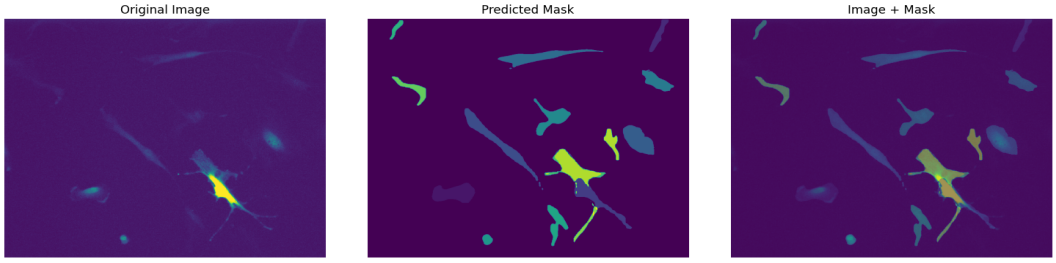
Figure 11: Comparison of training and validation losses in the first row and Dice and IoU metric coefficients in the second one using Mask-RCNN model and 3 different 2D datasets from the CTC. All models have been run over 100 epochs with a learning rate of 0.005

About technical details, it is important to note that instance segmentation should normally output worse results than the semantic task using the same model. This is clearly visible is one compares the curves in Figure 11 to those in Figures 9 and 10. For each of the datasets, the losses and accuracies achieved in the segmentation task are much better. Without going too deep, one can see that the best dice and IoU coefficient obtained in the instance task for the second dataset is the value at which the other models start their training. This gives us a perspective of how hard the simple task of identifying instances once you know to segment can get. The explanation of this change of behaviour between models models is obvious, as instance segmentation models must do the same semantic segmentation with the complexity of also being able to identify individual objects. In this case this comparison cannot be shown

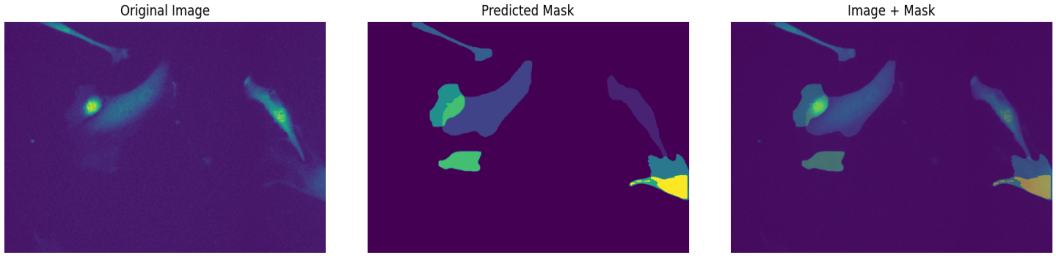
due to the lack of time to compute all the experiments. Nevertheless, by looking at the masks output by the trained model, one can see that although the segmentation is quite similar to that of the previous models used in section 5.1, individual instances are not clearly identified, and some of the cells appear to be merged as a unique one in the case that they are touching. In Figure 12 are shown an original image along with the prediction output by the trained Mask-RCNN model for each of the datasets. One can see once again that the mask output for the first dataset is quite good, while the ones for the other datasets lack some precision. This is the same behaviour than the one obtained in the semantic task, although here it is easier to see. This may be due to the fact that the first dataset has smoother values and constant shapes for the cells, having a good separation between them in most cases. In the other hand, the 'Fluo-C3DL-MSC' dataset has brighter values in the core of the cells, with variable shpaes adn overlapping instances in some frames, what makes the segmentation and detection task more difficult to the model.



(a) Fluo-N2DH-GOWT1



(b) Fluo-C2DL-MSC-01



(c) Fluo-C2DL-MSC-02

Figure 12: Example of the original image, the predicted mask and their overlap for the Mask-RCNN architecture in the three datasets used for instance segmentation.

6 Limitations and Impacts

In this last section, I will do a reflection on the work, its limitations due to lack of data, noise, weak models, as well as the impacts this path of investigation can lead to.
In the first place, if I had to point out the biggest problem we encountered when doing the project and that limited us the most, it wouldn't be the models or the mathematics under them, but the datasets. About them, I could list plenty of different characteristics that, in this kind of projects, lead to complex problems and not astonishing results. Whenever working on a Deep Learning project, it doesn't matter if it's about Natural Language Processing (NLP), object detection, instance segmentation, image classification, etc., one already knows that you will need a lot of data for training, since the more and various data you have, the better and more complex features the model will learn. If you want to create a model to distinguish between dogs and cats, you may be good with a hundred images, but if the classification is oriented to object detection for an autonomous car, for example, then you will need images of persons, children, traffic lights, cars, bikes, skates, dogs, roads, crosswalks... and taken from different angles, perspectives, and coloration. This makes the number of images needed increase exponentially. Not to forget that each of them (or most of them) should be paired with a mask that has to be hand-made.

Even with these inconveniences, computer vision applied to autonomous cars or face recognition has the advantage that the data needed is quite common in society and, even more, is accessible and easy to produce. Not to mention that you yourself could take a camera and take your own pictures for a project. This is a great advantage that doesn't apply to the medical field. One of the problems is the difficulty in obtaining the data, since they come from living beings, even humans, which makes the process slow. Also, it is not as easy as taking pictures in the street or portraits of people, but images taken from a microscope. Even more, if you want your images to be of high quality, the process is more complex. Following this point, the data is really heavy: for being of high quality, like Whole Slide Imaging (WSI) or for being in 3D.

Another problem one can encounter is that making a general model that is able to classify or segment different cells is quite complicated for the points treated before. Having high-quality data of one type of cells is costly, but having enough of different types for training a general model is even more. This happened to us in our project, since we found some open databases of different cells, but each of them was really different from the others, or there weren't enough images of each, or the masks were not great. Linked to this point, making masks is a hard task since one has to make them themselves and it's not so clear how to do so. Images of cells do not have precise borders or differences between the background and the cells. Usually, for taking the images, one has to treat the cells with some type of marker in order to make them visible or have a bigger contrast with others. The problem is that these substances usually affect more

the nucleus or the cytoplasm of the cells, so you end up with a bright center and blurry contours.

Once the limitations due to the data used in the medical field have been discussed, I will talk about the impact these kinds of projects or investigations may have in society. This I will divide into direct impacts on society and environmental impact. Focusing on the first part, it is clear that Computer Science (in general, like Machine and Deep Learning algorithms and methods) applied to medical subjects do have a good purpose. The single goal in these cases is to obtain better methods and utilities that can help our health. In relation to our project, the applications in the field are various. Cells segmentation and tracking are spread throughout medical imaging. This is due to the fact that cell migration is really important to take care of diseases or illnesses since scientists can see how the cells (in this case, tumors) are affected when using one or another treatment and how they react. Also, cell relations and interactions with each other and their environment can help detect mutations and know more about the tumors.

Lastly, I will briefly talk about the environmental impact this or similar projects may have. In general, deep learning projects like image analysis or NLP models are quite known for the high computational resources they need. Especially large deep learning models require significant power to train and run the models, so the energy consumption cannot be ignored. If, instead of using a local machine for the runs, the project is based in a data center of operation, this cost may be higher due to the fact that energy is also needed to keep the facilities working, the use of powerful utilities like GPUs, or the energy necessary to maintain the servers and cooling systems at a low temperature. This energy, although primarily electricity, may come from fossil fuels, which increase the carbon footprint. Another important point is the fact that the intensive use of high technological equipment needed for the tasks and its rapid development may decrease their life and increase the obsolescence time of the hardware. This is why it should be carefully treated and managed to ensure it is recycled. Last but not least, one of the biggest wastes of energy I think one can find in these large models is the inefficiency of algorithms. Once one is working with powerful equipment, it tends to think less about the efficiency and time-consuming runs and executes the same algorithm multiple times, just making some changes in the hyperparameters and using more epochs than needed. In fact, this is something that occurred to me the first times I worked with better machinery, and after a few weeks, I realized that I stopped thinking about better ways to improve and apply my algorithms and started executing the programs multiple times, just making some changes in the hyperparameters and using more epochs than needed. In fact, this is something that I find should be done at the end of each project: whenever you think it is as efficient as you can make it, you can try and start fine-tuning it to give it that little push it needs to give the best results.

7 Conclusion

This document summarizes all that has been done during the 5-month internship at the Paris Brain Institute. Since the beginning, most things I have done were new to me, since I was not formally trained in computer vision. This way, I have learned technical aspects about image representation in NumPy arrays and Torch tensors, as well as the different data types. But most importantly, I had the opportunity to work with various deep learning models and architectures, mostly for image segmentation. This has helped me gain confidence and be able to adapt each of the datasets with its own characteristics to the architectures of the models. This is probably the hardest part when doing a deep learning project, since the modifications for the training process and the calculation of losses and accuracies are negligible in most cases once you are able to read the dataset with the model. For this, it is necessary to spend enough time debugging and studying the dataset you are given, including the different shapes and aspect ratios of the images, color tonalities, and the range of values. Afterward, one should study the model architecture that is going to be used and see what inputs and outputs it needs in order to create a dataset that responds well to it. In particular, for this project, the hardest model to use has been Mask-RCNN, since it requires not only the images and masks but also bounding boxes for detection and the mask of each individual instance.

Regarding the results obtained in the experiments, one can learn that image segmentation is a challenging problem with long and resource-intensive training processes, which depend mostly on the data one is using. The difference it makes when using one dataset over another can be significant, as shown in the learning curves. In particular for this project, when working with cell segmentation, the complexity of the task will depend a lot on the markings used to photograph the cells as well as the type of cells. Brain cells and, in particular, gliomas are known for having long and thin ramifications extending from the nuclei, which makes segmentation harder. Additionally, these ramifications are used to connect and interact with other cells. For this reason, the task of instance segmentation becomes more complex, as the model has to learn to separate axons from different cells or merged cells from one another.

Lastly, there are many aspects that were intended to be added to the internship, but in the end, it was not possible due to the lack of time and probably due to being too ambitious in the goals (which is not bad at all). The task of segmentation, both semantic and instance, took more of our time than we thought, and that is why the tracking and analysis of 3D images could not be finished. A lot of work has already been done but not enough to provide comparative results, which is why we decided not to include it in the results section, although an appendix has been added for this purpose.

Acknowledgments

The author would like to thank, first of all, his mentor and supervisor PH Agusti Alentorn for all his help and dedication during the internship project. He has dedicated a lot of his professional and personal time to answer all my doubts and send me references that could be helpful for the project. Pr Maks Ovjanikov from École Polytechnique in Paris for his suggestions in the technical sections and the provision of the DiffusionNet model fro 3D instance segmentation. The Cell Tracking Challenge project for having in their disposal the public datasets extremely useful for the development of the project as well as the reviews of the work of the different teams that took part in the challenge. To SIRIC CURASMUS association as this internship could not have taken place by any means without their financing. And last but not least, to the Paris Brain Institute (ICM) for letting me take part in their scientific research projects and having use of their facilities and private datasets.

References

[1] Miller KD, Ostrom QT, Kruchko C, Patil N, Tihan T, Cioffi G, et al. Brain and other central nervous system tumor statistics, 2021. CA Cancer J Clin. 2021;71:381–406.

[2] Louis DN, Perry A, Wesseling P, Brat DJ, Cree IA, Figarella-Branger D, et al. The 2021 WHO Classification of Tumors of the Central Nervous System: a summary. Neuro-Oncology. 2021;23:1231–51.

[3] Yan H, Parsons DW, Jin G, McLendon R, Rasheed BA, Yuan W, et al. IDH1 and IDH2 mutations in gliomas. N Engl J Med. 2009;360:765–73.

[4] Kohanbash G, Carrera DA, Shrivastav S, Ahn BJ, Jahan N, Mazor T, et al. Isocitrate dehydrogenase mutations suppress STAT1 and CD8+ T cell accumulation in gliomas. J Clin Invest. 2017;127:1425–37.

[5] Varn FS, Johnson KC, Martinek J, Huse JT, Nasrallah MP, Wesseling P, et al. Glioma progression is shaped by genetic evolution and microenvironment interactions. Cell. 2022;185:2184-2199.e16.

[6] Dekkers JF, Alieva M, Wellens LM, Ariese HCR, Jamieson PR, Vonk AM, et al. High-resolution 3D imaging of fixed and cleared organoids. Nat Protoc. 2019;14:1756–71.

[7] Kirst C, Skriabine S, Vieites-Prado A, Topilko T, Bertin P, Gerschenfeld G, et al. Mapping the Fine-Scale Organization and Plasticity of the Brain Vasculature. Cell. 2020;180:780-795.e25.

[8] Renier N, Wu Z, Simon DJ, Yang J, Ariel P, Tessier-Lavigne M. iDISCO: a simple, rapid method to immunolabel large tissue samples for volume imaging. Cell. 2014;159:896–910.

[9] Renner H, Grabos M, Becker KJ, Kagermeier TE, Wu J, Otto M, et al. A fully automated high-throughput workflow for 3D-based chemical screening in human midbrain organoids. Elife. 2020;9:e52904.

[10] Tan H-Y, Cho H, Lee LP. Human mini-brain models. Nat Biomed Eng. 2021;5:11–25.

[11] Popenda M, Zok T, Sarzynska J, Korpeta A, Adamiak RW, Antczak M, et al. Entanglements of structure elements revealed in RNA 3D models. Nucleic Acids Research. 2021;49:9625–32.

[12] Sharp N, Attaiki S, Crane K, Ovsjanikov M. DiffusionNet: Discretization Agnostic Learning on Surfaces. ACM Trans Graph. 2022;41:27:1-27:16.

[13] Efficient Visual Tracking with Exemplar Transformers [Internet]. DeepAI. 2021 [cited 2023 Jan 9]. Available from: `https://deepai.org/publication/efficient-visual-tracking-with-exemplar-transformers`

[14] Pradelli F, Minervini G, Tosatto SCE. Mocafe: a comprehensive Python library for simulating cancer development with Phase Field Models. Bioinformatics. 2022;38:4440–1.

[15] Nagle PW, Plukker JTM, Muijs CT, van Luijk P, Coppes RP. Patient-derived tumor organoids for prediction of cancer treatment response. Semin Cancer Biol. 2018;53:258–64.

[16] Jabs J, Zickgraf FM, Park J, Wagner S, Jiang X, Jechow K, et al. Screening drug effects in patient-derived cancer cells links organoid responses to genome alterations. Mol Syst Biol. 2017;13:955.

[17] Vlachogiannis G, Hedayat S, Vatsiou A, Jamin Y, Fernández-Mateos J, Khan K, et al. Patient-derived organoids model treatment response of metastatic gastrointestinal cancers. Science. 2018;359:920–6.

[18] Linkous A, Balamatsias D, Snuderl M, Edwards L, Miyaguchi K, Milner T, et al. Modeling Patient-Derived Glioblastoma with Cerebral Organoids. Cell Rep. 2019;26:3203-3211.e5.

[19] Krieger TG, Tirier SM, Park J, Jechow K, Eisemann T, Peterziel H, et al. Modeling glioblastoma invasion using human brain organoids and single-cell transcriptomics. Neuro Oncol. 2020;22:1138–49.

[20] Ogawa J, Pao GM, Shokhirev MN, Verma IM. Glioblastoma Model Using Human Cerebral Organoids. Cell Rep. 2018;23:1220–9.

[21] Wu P-H, Gilkes DM, Phillip JM, Narkar A, Cheng TW-T, Marchand J, et al. Single-cell morphology encodes metastatic potential. Science Advances. American Association for the Advancement of Science; 2020;6:eaaw6938.

[22] Bakal C, Aach J, Church G, Perrimon N. Quantitative Morphological Signatures Define Local Signaling Networks Regulating Cell Morphology. Science. American Association for the Advancement of Science; 2007;316:1753–6.

[23] Lagerweij T, Dusoswa SA, Negrean A, Hendrikx EML, de Vries HE, Kole J, et al. Optical clearing and fluorescence deep-tissue imaging for 3D quantitative analysis of the brain tumor microenvironment. Angiogenesis. 2017;20:533–46.

[24] Geometric Deep Learning: Going beyond Euclidean data | IEEE Journals & Magazine | IEEE Xplore [Internet]. [cited 2023 Jan 11]. Available from: https://ieeexplore.ieee.org/document/7974879

[25] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, et al. 3D ShapeNets: A Deep Representation for Volumetric Shapes. 2015 [cited 2023 Jan 11]. p. 1912–20. Available from: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Wu_3D_ShapeNets_A_2015_CVPR_paper.html

[26] Uy MA, Pham Q-H, Hua B-S, Nguyen T, Yeung S-K. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. 2019 [cited 2023 Jan 11]. p. 1588–97. Available from: https://openaccess.thecvf.com/content_ICCV_2019/html/Uy_Revisiting_Point_Cloud_Classification_A_New_Benchmark_Dataset_and_Classification_ICCV_2019_paper.html

[27] Poulenard A, Ovsjanikov M. Multi-directional geodesic neural networks via equivariant convolution. ACM Trans Graph. 2018;37:236:1-236:14.

[28] VoxNet: A 3D Convolutional Neural Network for real-time object recognition | IEEE Conference Publication | IEEE Xplore [Internet]. [cited 2023 Jan 11]. Available from: https://ieeexplore.ieee.org/document/7353481

[29] Charles RQ, Su H, Kaichun M, Guibas LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. IEEE Computer Society; 2017 [cited 2023 Jan 11]. p. 77–85. Available from: https://www.computer.org/csdl/proceedings-article/cvpr/2017/0457a077/12OmNAJDBye

[30] Boscaini D, Masci J, Rodolà E, Bronstein M. Learning shape correspondence with anisotropic convolutional neural networks. Advances in Neural Information Processing Systems [Internet]. Curran Associates, Inc.; 2016 [cited

2023 Jan 11]. Available from: https://papers.nips.cc/paper/2016/hash/228499b55310264a8ea0e27b6e7c6ab6-Abstract.html

[31] Kim A, Ošep A, Leal-Taixé L. EagerMOT: 3D Multi-Object Tracking via Sensor Fusion. 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021. p. 11315–21.

[32] Kang M, Ko E, Mersha TB. A roadmap for multi-omics data integration using deep learning. Briefings in Bioinformatics. 2022;23:bbab454.

[33] Sugawara K, Çevrim Ç, Averof M. Tracking cell lineages in 3D by incremental deep learning. Elife. 2022;11:e69380.

[34] Pachitariu M, Stringer C. Cellpose 2.0: how to train your own model. Nat Methods. 2022;19:1634–41.

[35] Cell Tracking Challenge. http://celltrackingchallenge.net/

[36] International Symposium on Biomedical Imaging. https://archives.embs.org/wp-content/archives/ISBI/2013/

[37] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. W. Martin Maška and others, A benchmark for comparison of cell tracking algorithms, Bioinformatics, Volume 30, Issue 11, June 2014, Pages 1609–1617, https://doi.org/10.1093/bioinformatics/btu080

[38] D. Svoboda and V. Ulman, MitoGen: A Framework for Generating 3D Synthetic Time-Lapse Sequences of Cell Populations in Fluorescence Microscopy, in IEEE Transactions on Medical Imaging, vol. 36, no. 1, pp. 310-321, Jan. 2017, doi: 10.1109/TMI.2016.2606545.

[39] Ulman, V., Maška, M., Magnusson, K. et al. An objective comparison of cell-tracking algorithms. Nat Methods 14, 1141–1152 (2017). https://doi.org/10.1038/nmeth.4473

[40] D. V. Sorokin et al., FiloGen: A Model-Based Generator of Synthetic 3-D Time-Lapse Sequences of Single Motile Cells With Growing and Branching Filopodia, in IEEE Transactions on Medical Imaging, vol. 37, no. 12, pp. 2630-2641, Dec. 2018, doi: 10.1109/TMI.2018.2845884.

[41] Maška, M., Ulman, V., Delgado-Rodriguez, P. et al. The Cell Tracking Challenge: 10 years of objective benchmarking. Nat Methods 20, 1010–1020 (2023). https://doi.org/10.1038/s41592-023-01879-y

[42] J. Long, E. Shelhamer and T. Darrell, Fully convolutional networks for semantic segmentation, in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015 pp. 3431-3440. doi: 10.1109/CVPR.2015.7298965

[43] O. Ronneberger, P. Fischer and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015, Part III, LNCS 9351, pp. 234–241, 2015. DOI: 10.1007/978-3-319-24574-4_28

[44] J. Long, E. Shelhamer and T. Darrel. Fully Convolutional Networks for Semantic Segmentation. arXiv:1411.4038v2.

[45] L. C. Chen, G. Papandreou, F. Schroff and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. arXiv:1706.05587v3.

[46] K. He, G. Gkioxari, P. Dollár, R. Girshick. Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969.

[47] R. Girshick. Fast R-CNN. arXiv:1504.08083.

[48] N. Sharp, S. Attaiki, K. Crane, M. Ovsjanikov. DiffusionNet: Discretization Agnostic Learning on Surfaces. arXiv:2012.00888v3.

[49] V. S. Subramanyam. IOU (Intersection over Union). Medium 2021. `https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef`

[50] T. Ben-Haim, T. Riklin Raviv. Graph Neural Network for Cell Tracking in Microscopy Videos. arXiv:2202.04731v2.

Appendix A: experiments

The code with all the experiments and functions related for the semantic and instance segmentation is available at `https://github.com/lucas-rdlr/segmentation-and-tracking`. Notebooks and functions for tracking and 3D analysis are also available in the same link although, as commented earlier, they are not finished and any experiment have been done concerning those aspects. All the experiments have been tested in the same local machine with a GPU NVIDIA Corporation TU104GLM [Quadro RTX 5000 Mobile / Max-Q]. In particular, all experiments for semantic segmentation have been done using the Cross Entropy Loss and the Stochastic Gradient Descent optimizer from PyTorch with constant parameters: $learning\ rate = 0.005$, $momentum = 0.95$ and $weight\ decay = 0.0001$. No transformations have been done to the data except those experiments with normalized data (explicitly indicated) and the resize of the images for two reasons: in case of using the UNet architecture, as it was already said in Section 3.1.1, the height and width of the images must be divisible by 16 (or other factor in case the architecture is modified) in order to be able to crop and paste the arrays from the downsampling path to the upsampling one. Even though other model is used, the images from certain datasets must be resized in order to feed the dataloader with a fixed data shape. 'Fluo-C2DK-MSC' and 'Warwick QU' datasets have images of different sizes and all batches in a dataloader must be the same size. Also, there are some cases where the images have equal size and the model doesn't require an specific shape (like UNet) must resize it chosen to reduce the memory usage. In Table 3 are recompiled total run time over 100 epochs and the GPU memory of the models for the different datasets.

Models	Time (m)	Memory (MB)	Params	Shape
Fluo-N2DH-GOWT1				
DeepLab	170	5488.87	45,773,252	(1,640,640)
FCN	140	5522.32	54,298,327	(1,640,640)
UNet	145	2043.26	28,953,474	(1,640,640)
Fluo-C2DL-MSC				
DeepLab	68	4160.30	45,773,252	(1,480,640)
FCN	60	4193.52	54,298,327	(1,480,640)
UNet	87	1560.06	28,953,474	(1,480,640)
Warwick QU				
DeepLab	32	2511.00	45,779,524	(3,346,512)
FCN	26	2543.93	54,304,599	(3,346,512)
UNet	80	1056.64	28,954,626	(3,368,544)

Table 3: Comparison of run time and memory usage for the semantic segmentation experiments over 100 epochs.

Appendix B: datasets

- **Fluo-N2DH-GOWT1**: the dataset consists of mouse stem cells images. It has two video sequences of 92 frames each of them taken with an elapse of 5 minutes between frames and a resolution of 0.240×0.240 microns per pixel. The images have 1 unique channel (grayscale images) kept in *.tif* files of 1024×1024 pixels for both video sequences encoded in *uint16* values.
  The masks are also *.tif* files (of the same dimension respectively) consisting of instance segmentation masks. In this case all the different instances masks are kept in the same image and each instance is represented with a different value. In case one want to perform semantic segmentation with this dataset (as is our case) one can just convert it to a unique binary mask by taking boolean values (0 for background and 1 for instances).

  This is the most homogeneous dataset we have work with and so the one that has output the best results for all models. This is due to the fact that the cells in them have a rounded and quite detectable size and the background in completely flat, which makes it easy to isolate the instances, even using only watershed or threshold methods. Also, since the dataset actually corresponds to a video sequence, all images are very similar which makes it easier for the models to get trained.

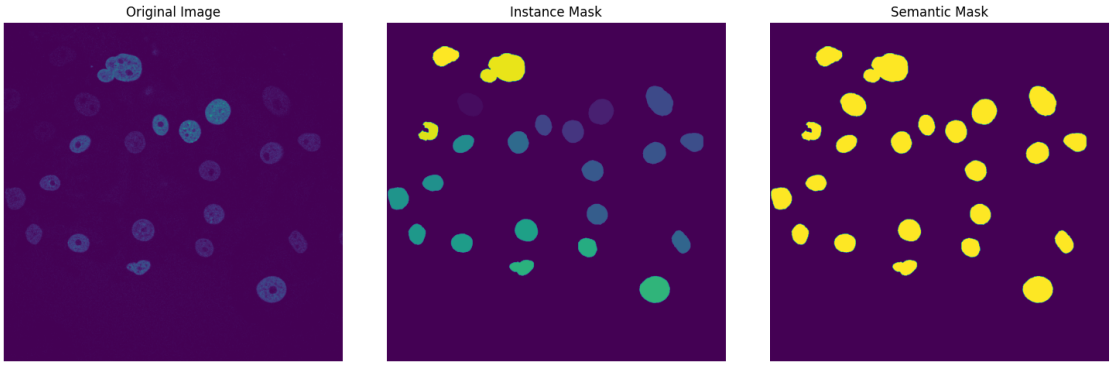


Figure 13: Example of a frame for the Fluo-N2DH-GOWT1 showing the original frame with its associated instance mask as well as the semantic mask.

- **Fluo-C2DK-MSC**: the dataset consists of rat mesenchymal stem cells images. It has two video sequences of 48 frames each of them taken with an elapse of 20 minutes between frames and a resolution of 0.3977×0.3977 microns per pixel. As the previous CTC dataset, the images have 1 channel kept in *.tif* files of 832×992 pixels for the first video and 782×1200 pixels for the second one, encoded in *uint16* values.
  The same characteristics as described before apply for the masks.

  In this case, although the dataset comes also from the CTC, the images are harder to

segment. They also correspond to a video sequence but the main problem may come from the fact that the cells themselves have more different shapes between them: ones are more elongate than others that may appear more rounded. This discrepancy between the instances and the fact that they are more blurred with the background makes the training process harder than the first dataset.

Figure 14: Example of a frame for the Fluo-C2DK-MSC showing the original frame with its associated instance mask as well as the semantic mask.

- **Warwick QU**: the data has been taken from the Gland Segmentation Challenge consisting of a variety of histologic images of glands in a colorectal cancer. The images are taken using the Hematoxylin and Eosin (H&E) stain technique widely used in the field of pathology and biology. It is commonly used to stain tissue samples for microscopic examination and provides detailed information about the cellular structure and composition of tissues. The dataset contains 85 training images and 80 for test. The resolution is of 0.62005×0.62005 microns per pixel with a non-fixed size for the input images. Most of them (79 in the case of training images) have a size of 522×775 pixels, being those the biggest ones. This dataset doesn't correspond to a video sequence, which means that each image is not as similar to the others as with the previous datasets. Also the sizes of the images vary more and they are not all taken in the same way, which added to the fact that here we don't work with a flat background makes the learning process quite harder.

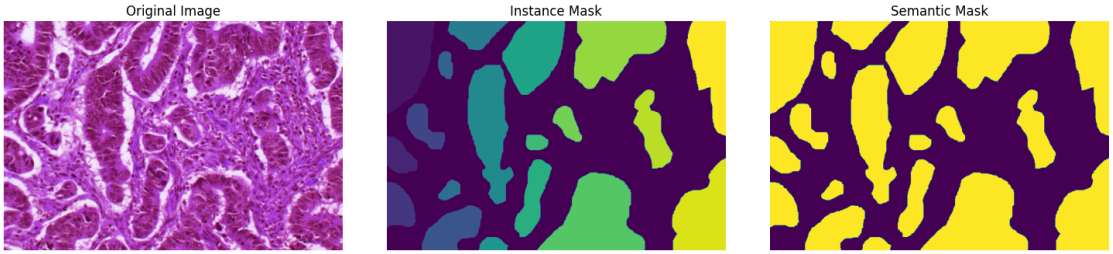


Figure 15: Example taken from the Warwick QU dataset showing the original frame with its associated instance mask as well as the semantic mask.

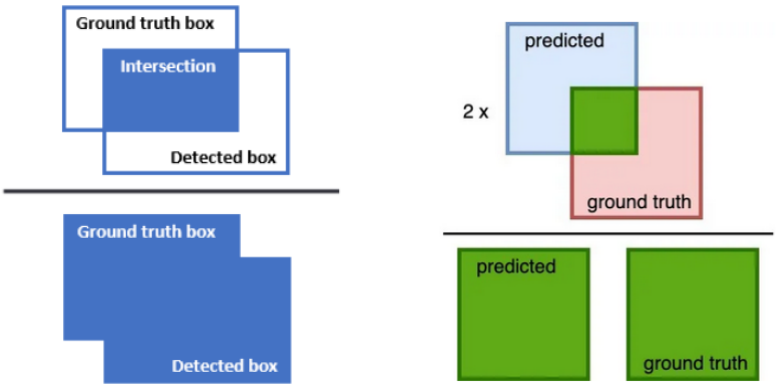
Appendix C: metrics

- **Jaccard Index or Intersection over Union (IoU)**: it is a metric used to evaluate the accuracy of object detection and image segmentation tasks. It measures the degree of overlap between two sets, often used to compare the predicted bounding boxes or regions with the ground truth or the actual objects or regions of interest in an image. In Figure 16 (a) is represented the way this metric is calculated. The values are in the range 0-1, so it can be thought as the probability of the two objects to be the same. As one can see, the greater the IoU is, the more intersection the two objects have so the probability is higher.

$$IOU = \frac{A \cap B}{A \cup B}$$

- **Dice coefficient**: this metric is used in the same way as the IoU explained above. They differ in the way that for IoU one divides by the union of the two instances whereas in the dice coefficient one divides by the total area (so the intersection area in counted twice as one can sees in Figure 16 (b)). Also, the intersection or area of overlap is counted twice in the numerator. Although it may seem like they should be the same, there exists an inequality showing that the Dice coefficient is always greater than the IoU.

$$Dice = \frac{2 * (A \cap B)}{A + B}$$



(a) (b)

Figure 16: Diagrammatic representation of the formula to calculate the IoU (left) and the Dice coefficient (right). In both cases, higher intersection between objects corresponds to higher metric values.

Appendix D: future work

As was introduced in the conclusions of the project, not all goals that were set at the beginning have ended up seeing light. It is easy to make the error of being too ambitious, specially in research project were the speed at which one progresses is not at all homogeneous and it is easy to encounter problems or barriers that one didn't consider at first. In this case, our final goal was to be able to use deep learning models for tracking of 3D cell images. It is clear that if that is your only and ultimate goal, you can just look for good 2D or 3D instance segmentation masks and try to find the best model. Nevertheless, our purpose was to come up with comparison methods for our own data taken in the laboratory. In this case we only had the raw images and the masks for training had to be done by hand. Since this takes time, we thought that the best way to approach the project would be to start from the bottom. The most obvious route guide to follow and the one that we chose was to start doing the equivalent task using 2D data and then extend the results to the 3D problem. Before doing instance segmentation, we focused on semantic segmentation as it can help identify barriers or limitations in your data that could be helpful for the next step. We derived too much time in these two steps, and that is why the tracking and 3D analysis couldn't be finished at time. This is not something that bothered us since this project will be continued as a PhD thesis, and it is preferable to take smart and sensible steps before going any further.

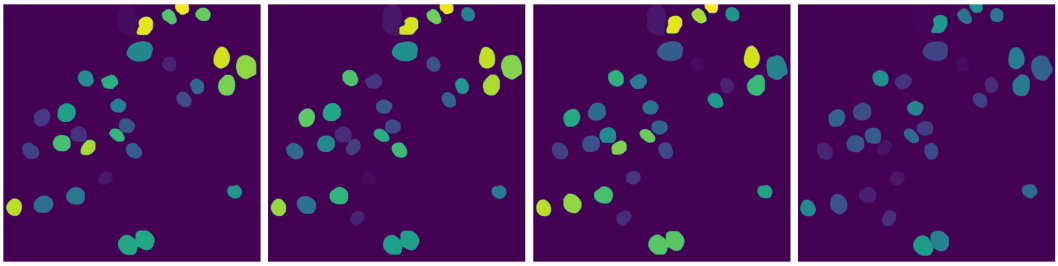
In this appendix will be exposed the advances that had been done related to the tracking and 3D analysis that has not been shown in the experimental results.

Appendix D.1: tracking

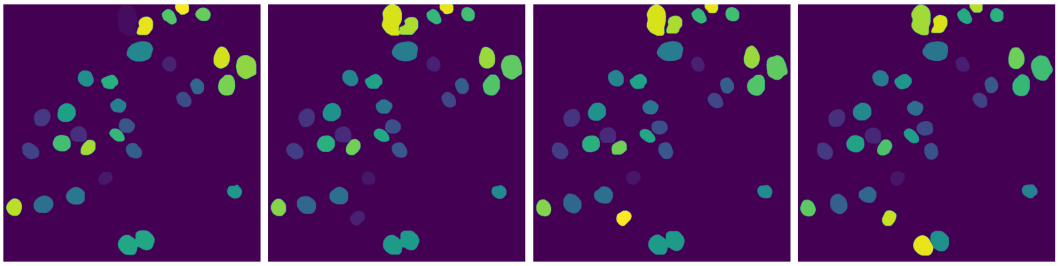
Earlier in Section 4.1 it was introduced that the tracking of cells could be approached via machine learning algorithms. This was the first way I tried to do it since it would give me a good perspective of how direct tracking could be done without using neural networks. For this purpose, I started creating really simple algorithms to track instances from one frame to the other, and with each result I came up with an upgrade that would give better and better results, until I could have something close to tracking cells. All this technical upgrades have a lot in common with the feature extraction that was introduced in Section 3.2.1.

As one can see in Figure 17 (a), without doing any processing to the frames, the predicted images from the Mask-RCNN model assigns different labels to the cells in consecutive cells. To fix this one can try to associate in a very simple way, cells from one frame to the other. Algorithm 1 takes as input two instance masks of consecutive frames and assigns each cell in the current mask with a cell in the previous one in order to minimize the IoU between the two instances. As the pseudo-code shows, this is done by looping over the cells of a frame and for

each of them, looping over all the cells in the previous one and calculating the IoU (with a function previously design) between their two binary masks.



(a) Frames without processing the predictions



(b) Frames after preprocessing the predictions using Algorithms 1 and 2

Figure 17: Comparative results of the first 4 predicted frames using the model Mask-RCNN for instance segmentation applied to the first video sequence of the test dataset of 'Fluo-N2DH-GOWT1' (a) without applytin any correction and (b) using Algorithms 1 and 2 to assign cells and propagate the label.

With just this code we can check which cell in a frame is more likely to be the same as one in the previous frame. Nevertheless, this doesn't take count of the cells that may enter or exit the frames, or a cell that divides in two. To deal with this, Algorithm 2 was created. It loops over two frames at a time and computes the Algorithm 1 to assign the cells and change the label to that of the cell in the previous frame. After, it checks if in the new frame there are more instances than the previous one, and it that is the case, it changes the label to those instances to a new one. Of course, this new label cannot be one previously used, as it is supposed to be a completely new cell.

After applying this function to the predicted instance masks, we get the result in Figure 17 (b). This way we can see that same cells in consecutive frames have the same label, which then propagates to the rest of the frames one by one.
More improvements can be made using the same idea after analysing the dataset one is using and identifying its most important features. The next step we will focus on regarding cell tracking will be to implement the Graph Neural Network (GTT) model but also to update

Algorithm 1 Assign Cells

```
function TrackCells(previous_masks, current_masks)
    unique_previous ← length of unique values in previous_masks - 1
    unique_current ← length of unique values in current_masks - 1
    P(C) ← length of unique_previous -1 (unique_current
    cost_matrix ← zeros matrix of size P × C
    for i, pr in Enumerate(unique_previous) do
        for j, cr in Enumerate(num_current) do
            previous ← previous_masks × (previous_masks == pr)
            current ← current_masks × (current_masks == cr)
            cost_matrix[i, j] ← 1 - IOU(previous, current)
        end for
    end for
    assignment ← LinearSumAssignment(cost_matrix)
    return assignments
end function
```

Algorithm 2 Propagate Label

```
function TrackFrames(predictions)
    current_frame ← prediction[0]
    current_labels ← unique values in current_frame
    L ← length of current_labels
    previous_frame ← change labels of current_frame to {1,2,...,L}
    for current_frame in predictions[1:] do
        assignments ← TrackCells(previous_frame, current_frame)
        current_frame ← change labels of current_frame regarding assignments
        if length of current_frame > length of previous_frame then
            current_frame ← add new labels to the cells not assigned
        end if
    end for
    return assignments
end function
```

this algorithms so that they are capable of track the datasets from our laboratory. Of course, this way of tracking has its limitations, as it is really specialized in one type of data, so it is hard to generalize it to other. Nevertheless, since we want to come up with a way to track particular brain cells, we think this won't be a problem, but rather an advantage.

Appendix D.2: 3D analysis

In this last section I will include some results or advances that we have made for 3D datasets. Before giving some examples, I will briefly explain the technical aspects this kind of data presents and how to work with them. In general, one can find three different types of 3D data:

- **Stack-3D**: it is the simplest and easiest 3D data type that one can find. This is nothing more than 2D images of a plane of the object one on top of the other along the last axis. For this one needs to be able to take images of the object at different heights or depth, which in the case of cells can be done by adjusting the zoom of the lens of the microscope in order to focus on different depths. Stacks-3D are used in biomedical imaging as it is the easiest way to obtain a 3D representation of cells.

- **Point cloud**: it is a collection of data points in a three dimensional coordinate system. Each data point represents a specific position in space and can be defined by its x, y, and z coordinates. They are commonly used in various fields such as computer graphics, computer vision, remote sensing, geospatial mapping, and 3D modeling. A way to obtain them is by using multiple sensors or cameras around the object and merging them to get the 3D perspective. That is way it is not easy to obtain a point cloud directly of cells, but one can do it indirectly as is shown later in Algorithm 3.

- **Meshes**: this last representation is probably the most complete one. It has its base in the point cloud but gives a solution to the problem of having empty spaces between coordinates. Meshes are normally point clouds where each of the coordinates represents a vertex of a polygon, usually triangles for their simplicity. Of course there exists multiple ways to convert a point cloud to a mesh by applying regression algorithms, but the most common one in the *marching cubes algorithm*.

To implement 3D segmentation one needs to understand the data type one is working on, as not all models are capable to deal with the same data structure. In the case of Stacks-3D a simple approach is to perform segmentation in each of the layers or stack of the image using 2D segmentation models, like UNet for semantic segmentation or Mask-RCNN for instance. For 2-classes semantic segmentation (cell and background like our case) there is not a big problem, but when dealing with instance segmentation, one should come up with a way (similar to that of tracking along frames) to assign the same label to cells in different layers. Even more, if one is doing 3D tracking using this type of data structure, labels must be propagated through

time and space (along frames and layers of the image). I the case that point clouds or meshes are chosen as object representation, other models must be used. In our case we though of DiffusionNet, as it was developed by one of our collaborators in Ècole Polytechnique and it can deal with both data types.

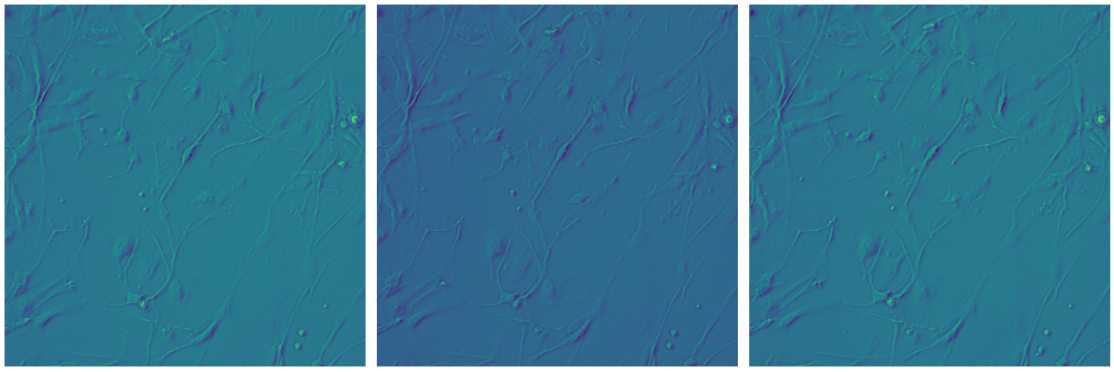


Figure 18: Example of 3D microglia image taken from our own laboratory as a stack-3D of 3 layers.

In Figure 18 is shown an example of a Stack-3D, in this case of 3 layers. The ones from the CTC consisted of 5 layers and there are more complex datasets with high resolution that can have even 50 layers. OF course, the more layers, the less space between them and so more information about the 3D structure. It is clear that this type of dataset is not easy to segment, as the cells in it have different shapes and sizes, with rounded nuclei and long ramifications. Also, the background and the instances are not so easy distinguish, only by contrasts and shadows. By applying Algorithm 3, one can transform this 3D structure to a point cloud by taking as coordinates the pixels where we consider that there is and instance and by the z coordinate the layer from the stack-3D that we took that coordinate. In this case we got the point cloud directly by segmenting individually the instances of the image. This can be done using multiple GUI's to delimit the borders of the objects. In Figure 19 we show the point cloud representation of one of this instances.
Related to this topic, we would like to go deeper into the subject and be able to properly segment this type of dataset using point cloud or mesh representation. We already know that segmentation in 2D via de stack images is possible but it is probably not the best approach, as there is a lot of information missing between stacks, and a way to deal with it is by converting it to mesh and filling those gaps with regression. Also, DiffusionNet has great results when doing image classification and instance segmentation in RNA datasets, what gives us hope when applying it to cell images. There is still a lot of work in process and multiple methods and different approaches, but during the months of the internship we have learned a lot about 3D representation and the limitations or barriers that we may encounter, which is also part of the learning process.

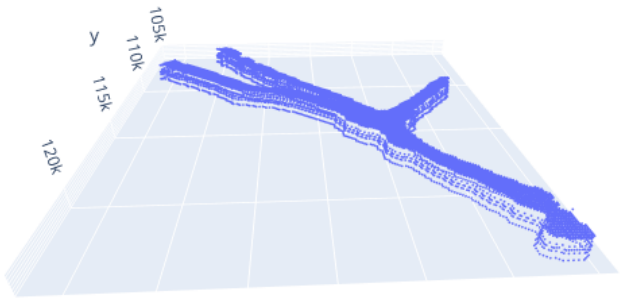


Figure 19: Example of 3D microglia instance taken from our own laboratory as point cloud representation.

Algorithm 3 Stack to Point Cloud

```
function STACK_TO_POINT_CLOUD(stacked_image)
    point_cloud ← empty array
    values ← empty array
    for i ← 0 to LENGTH(stacked_image) do
        slice ← stacked_image[i]
        (rows, cols) ← SHAPE(slice)
        z ← i                                   ▷ Z-coordinate is the index of the slice
        for x ← 0 to rows do
            for y ← 0 to cols do
                if slice[x, y] > threshold then          ▷ Threshold to consider values
                    APPEND(point_cloud, [x, y, z])
                    APPEND(values, slice[x, y])
                end if
            end for
        end for
    end for
    return point_cloud, values
end function
```